Region Segmentation Using LiDAR and Camera

M. Hossein Daraei[1], Anh Vu[2] and Roberto Manduchi[3]

***Abstract*—Inspired by the ideas behind superpixels, which segment an image into homogenous regions to accelerate subsequent processing steps (e.g. tracking), we present a sensor-fusion-based segmentation approach that generates dense depth regions referred to as supersurfaces. This method aggregates both a point cloud from a LiDAR and an image from a camera to provide an over-segmentation of the three-dimensional scene into piece-wise planar surfaces by utilizing a multi-label Markov Random Field (MRF). A comparison between this method that generates supersurfaces, image-based superpixels, and RGBD-based segments using a subset of KITTI dataset is provided in the experimental results. We observed that supersurfaces are less redundant and more accurate in terms of average boundary recall for a fixed number of segments.**

***Index Terms*—superpixel, sensor fusion, lidar, segmentation**

I. Introduction

ENVIRONMENT understanding provides vital information for autonomous driving systems mainly through the use of feature-based sensors (e.g. camera, LiDAR, and RADAR) and prior information (e.g. map). Different from recognizing object classes, modeling the 3D structure of the surroundings plays an important role in navigation and situation awareness. In order to understand the geometric structure of the scene, sensor measurements are usually segmented into those belonging to the ground plane and those belonging to a set of independent objects. Subsequently, a 3D model is assigned to each segment. Over-segmenting a 2D image, i.e. superpixel segmentation, is a preprocessing technique used in many computer vision applications such as image segmentation [1] and 3D reconstruction [2]. The idea is to provide a concise image representation by grouping pixels into perceptually meaningful compact patches that adhere well to object boundaries and do not overlap with multiple objects [3]. It dramatically reduces the complexity of the subsequent task, e.g. object segmentation. For many such applications, it is far easier to merge superpixels than to split them [4].

In this work, we introduce supersurfaces which are computed by tightly-coupled processing of sparse LiDAR and dense Camera measurements. In addition to a spatial over-segmentation of the scene, supersurfaces include 3D region surfaces as well. Unlike superpixels, the goal of supersurface segmentation is to represent possibly large surface areas of 3D objects using a parametric model. Such surfaces may be merged or tracked in the next stages, thus while preferred to be large they should not cross object boundaries. Homogeneous regions and gradients of images provide clues on object boundaries, but without depth data we are unable to reconstruct surfaces. It is possible to measure semi-dense depth in a stereoscopic vision [5] system and employ it for superpixel segmentation, but this approach requires extra processing and lacks accuracy especially in texture-less or distant areas. LiDARs as active sensors, on the other hand, give accurate range measurements at a sparse grid of azimuth-elevation angles, providing rich information about scene structure and discontinuities. Due to their sparsity, they alone cannot reliably estimate objects true boundaries. In several cases (e.g. black objects or windows in Fig. 1) there is remarkable missing data in LiDAR measurements which results in neglecting some object parts in segmentation. In order to combine the complementary characteristics of these sensors, we suggest to process them in a single joint Markov Random Field (MRF) energy minimization. This is achieved using sensor calibration parameters as well as estimated 3D surfaces that help relating sensor spaces. Fig. 1 summarizes our MRF-based approach for supersurface segmentation. The energy function is optimized via max-flow/min-cut and a few move-making iterations (α-expansion [6]) algorithms. The results demonstrate the adherence of extracted supersurfaces to both image and point cloud discontinuities.

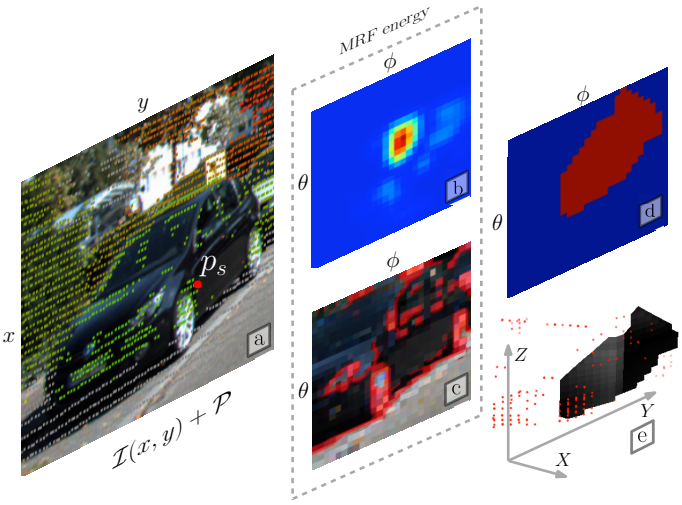


Fig. 1: Given (a) an image $\mathcal{I}$ and a point cloud $\mathcal{P}$ and a point p_s, we compute (b) MRF unary potentials based on LiDAR points, and (c) MRF pairwise weights from image gradients on a 2D angular grid $\Omega_\epsilon = (\theta, \phi)$. The MRF solution is (d) a segment covering LiDAR points while aware of image edges. The final output is (e) a three-dimensional surface segment (or a supersurface).

The rest of this paper is organized as follows. In Sec. II earlier work on scene segmentation is studied. In Sec. III we formally state the problem. Sec. IV covers our ground re-

[1,3] M.Hossein Daraei and Roberto Manduchi are with Baskin School of Engineering, University of California Santa Cruz, 1156 High St, Santa Cruz, CA 95064 USA {daraei,manduchi}@soe.ucsc.edu

[2]Anh Vu is with Electronics Research Laboratory, Volkswagen Group of America, 500 Clipper Dr, Belmont, CA 94002, USA anh.vu@vw.com

moving, surface reconstruction, and supersurface segmentation algorithms. Sec. V provides experimental results, and Sec. VI concludes the paper.

II. Prior Work

As an early technique for 2D image over-segmentation, FH [7] estimates superpixels through finding minimum spanning trees of the global graph of pixels and rule-based merging of adjacent superpixels. But it does not consider compactness, size, and shape regularity of superpixels. Normalized cut [8] implicitly takes compactness into consideration, but it has limited applicability due to its high computational complexity. Turbopixel generates uniform and compact superpixels by iterative dilation of seeds that are distributed evenly. Moore et al. [9] proposed an algorithm to preserve the topology of a regular lattice during superpixel segmentation. Liu et al. [10] suggests an objective function based on the entropy rate of a random walk and a regularization encouraging superpixels of the same size. Bergh et al. present SEEDS in [11] by encouraging color homogeneity and shape regularity in their objective function. Achanta et al. [12] proposed a simple and fast algorithm referred to as linear clustering based algorithm (SLIC). It generates superpixels by performing iterative K-means clustering in a five dimensional space combining color and spatial information. Li and Chen [3] extend SLIC to capture global as well as local similarity information. In their Linear Spectral Clustering (LSC) each pixel is mapped to a ten dimensional space in which weighted k-means approximates the spectral clustering solution in linear time. Veksler et al [13] reformulated superpixel generation as an energy minimization problem which is solved using Graph Cuts [6]. They cover the input image with overlapping square patches of fixed size, which is equal to the maximum superpixel size. Then they search for the optical assignment of pixels to patches (labels) so that the global energy is minimized. Our method is similar in spirit to their work. We also employ an energy-based approach with overlapping patches as labels, but our patches are defined as physical regions in three-dimensional space.

Superpixels have in recent years been employed for 3D scene reconstruction in stereoscopic systems [5], [14], [15]. Güney et al [5] propose using object-class specific CAD models for disparity proposal (displets) and aggregate them in a superpixel-based Conditional Random Field (CRF). Yamaguchi et al, [14] use a piece-wise planar model and segment the scene using superpixels. They use a Markov Random Field (MRF) with a static scene assumption, and manage to optimize segments, ego-motion, and plane parameters. Vogel et al, [15] find segment plane parameters in a similar fashion, but they also consider a pairwise term between nearby planes. In order to deal with dynamic scenes, they assume a rigid motion for each segment. Menze et al, [16] group segments into objects and find motion parameters for each group, resulting in a dramatic reduction in cost.

There is another class of over-segmentation techniques that make use of RGB-D data [17], [18]. Depth Adaptive Superpixels (DASP), introduced by Weikersdorfer et al [18], performs spectral graph clustering on a spatio-temporal graph of RGBD data combining position and normal information with color data. Pappon et al, [17] introduce Voxel Connectivity Cloud Superpixels (VCCS) which generates supervoxels in 3D by processing the connectivity of RGB-D depth channel. Despite their promising results on indoor RGB-D dataset, they fail to handle LiDAR point clouds of outdoor scenes especially in regions with low point density. In a probabilistic framework, Held et al, [19] improve LiDAR point cloud segmentation by adding temporal and semantic information to the conventional spatial approach. Their algorithm requires object tracking, does not take advantage of dense image data, and outputs point clusters rather than object masks.

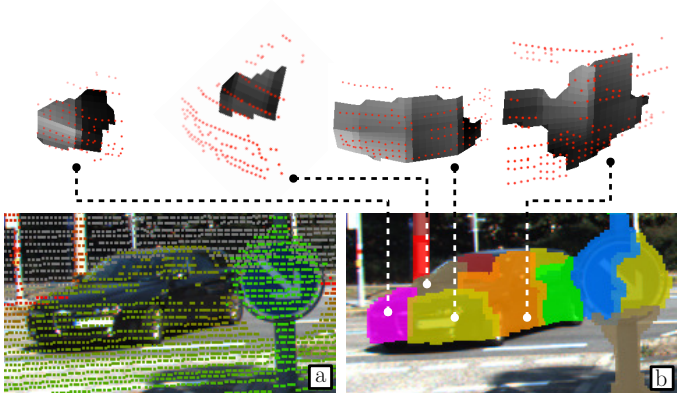


Fig. 2: (a) shows LiDAR-image data for a black car suffering from missing data, and (b) demonstrates non-ground supersurfaces along with their 3D models.

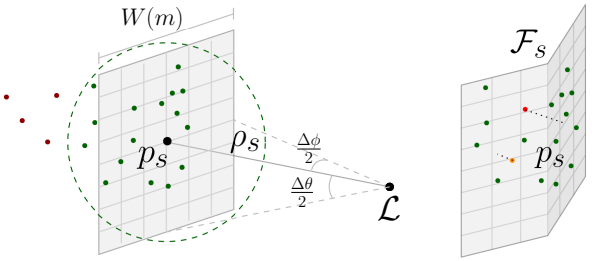
III. Problem Statement

Let $\mathcal{I}(x, y)$ and $\mathcal{P} = \{p_1, p_2, \ldots, p_M\}$ denote respectively the Camera image grid and the set of LiDAR measurements. For simplicity we assume calibration parameters are given for both sensors and all measurements are synchronized to the same time reference. Note that LiDAR points of a single scan are essentially range measurements at discrete azimuth-elevation angles. This is an important point that allows us to model them simply as a sparse 2D depth map rather than a 3D point cloud. We accordingly define $\Omega_\epsilon(\theta, \phi)$ as a two-dimensional and discrete azimuth-elevation grid of cell size ϵ°. Accordingly LiDAR points are represented in spherical coordinates as $p_i = (\theta\ \phi\ \rho)$ and mapped to Ω_ϵ. Now let $\mathcal{G} \subseteq \mathcal{P}$ represent the set of LiDAR reflections off the ground. We are interested in finding an over-segmentation $\mathcal{S} = \{s_1, \cdots, s_N\}$ that covers all non-ground points in Ω_ϵ. Each segment is a label mask spanning a subset of $\theta - \phi$ cells in Ω_ϵ. It is desired for segment borders to align with depth discontinuities as well as image boundaries. For each segment s is also required a surface (depth map) $\mathcal{F}_s(\theta, \phi)$ in order to capture the 3D structure of the scene.

IV. Proposed Method

In this section we describe different components of the proposed supersurface segmentation method. The first step is to select a cell size ϵ, according to which the angular grid Ω_ϵ is defined. While smaller ϵ results in finer segment resolution,

it will be less computationally efficient. In what follows we describe ground point removing, segment initialization, surface modeling for each segment, and finally first- and second-order potentials for MRF to specify object masks. Ground points are removed in order to ensure segment masks do not overlap with the ground surface. Removing them also eliminates the connectivity of independent objects through ground. Estimating an efficient and dense depth map for each segment enables the algorithm to map surface points in Ω_ϵ to unique image pixels. This helps with connecting sensor spaces, especially for regions with missing LiDAR measurements. Fig. 2 shows supersurface segments computed for a car with missing data due to its color.



(a) $\mathcal{B}_s$ in LiDAR space (b) Estimated $\mathcal{F}_s$

Fig. 3: Each segment s has (a) a set of assigned points (in green) and a square box $\mathcal{B}_s$ centered on its anchor p_s, then (b) a 3D surface $\mathcal{F}_s$ is fit to assigned points.

A. Preprocessing

As preprocessing we identify ground points $\mathcal{G}$ (along with outliers) and remove them from the point cloud. A small subset of ground points are first identified by constructing a coarse 2.5D grid for the point cloud [20]. This is followed by fitting a parametric surface model $Z = \mathcal{F}_\mathcal{G}(X, Y)$ in vehicle reference system. Then all points in $\mathcal{P}$ within a fixed threshold (10cm) of $\mathcal{F}_\mathcal{G}$ are marked as ground points. In order to handle slight curvatures, the ground surface is parametrized using the model in Sec. IV-C but defined over XY plane.

B. Segment initialization

The role of this step is to initialize a set of overlapping segments that cover all non-ground points. For each segment s we set a rectangle $\mathcal{B}_\int$ on the polar domain of Ω_ϵ which limits the segment area. A set of points $\mathcal{P}_s \subset \mathcal{P}$ is also assigned to each segment. Let $\mathcal{U}$ denote the dynamic set of non-ground points not assigned to any segments. The initialization process for a new segment s is to randomly choose an anchor point $p_s \in \mathcal{U}$, set $\mathcal{B}_s$ in Ω_ϵ around p_s, and assign to s the points in $\mathcal{P}$ that are within a Euclidean distance τ from p_s (Fig. 3a). Assigned points are added to segment point set $\mathcal{P}_s$ and removed from $\mathcal{U}$. This process is repeated until $\mathcal{U} = \emptyset$. Note that assignment to multiple segments is possible for points, as we do not remove them from $\mathcal{P}$. The associated rectangle $\mathcal{B}_s$ (as shown in Fig. 3a) is in Cartesian space a 3D rectangle centered on p_s and normal to the line connecting LiDAR origin $\mathcal{L}$ to p_s. We constrain them to be a square of size W meters, and compute the corresponding size on Ω_ϵ as,

$$\Delta\phi = \Delta\theta = 2tan^{-1}\Big(\frac{W}{2\rho_s}\Big) \tag{1}$$

where ρ_s is the range (depth) for p_s. Fig. 4b-c represents as an example the estimated $\mathcal{B}$s for all segments extracted from a sample scene. Note that W has a physical meaning and can be set based on expected object size and regardless of object position. This is in contrast to the method of Veksler et al [13] where maximum segment size is defined in terms of pixels. If available, semantic information for p_s can be employed to specify W.

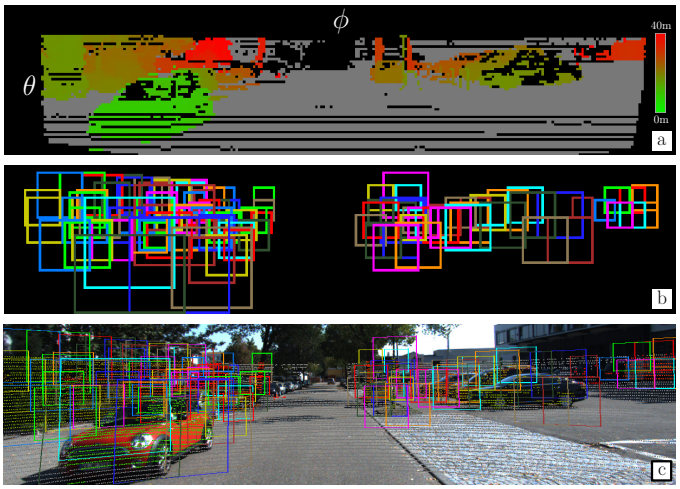


Fig. 4: (a) LiDAR points on $\Omega_\epsilon = (\theta, \phi)$ grid with gray representing ground (or far) points, (b) square boxes $\mathcal{B}$ for all segments in Ω_ϵ, and (c) projection of points and boxes on image.

C. Surface reconstruction

In order to associate each (θ, ϕ) cell in $\mathcal{B}_s$ to its corresponding image pixel, we need to have an estimated depth for that particular cell. Since points in $\mathcal{P}_s$ are sparse (especially in regions with missing LiDAR data) and do not cover all cells we need to fit a dense surface and interpolate depth. Segment surface $\mathcal{F}_s$, defined as a depth map over $\mathcal{B}_s$ grid, is estimated through minimizing an energy term consisting of a data and a regularization term. The data term encourages $\mathcal{F}_s$ to align with points in $\mathcal{P}_s$. We also add a non-quadratic (Huber [21]) penalizer to handle outliers. Similar to [14], [15], we also add a piecewise planar prior term. The prior term helps with depth interpolation in regions where LiDAR measurements are not available, e.g. the black car in Fig. 2. Inspired by the regularization term in [22], [23], we minimize the second-order derivatives of $\mathcal{F}$ to encourage smooth surfaces. In addition, we want the surface to potentially fold along sparse line patterns to prevent over-smoothing. To accomplish this, we add a spatial Huber penalizer to the regularization term as well [24]. Assuming $p_i = (\theta_i \ \phi_i \ \rho_i)$, surface energy term is defined as,

$$E_{\mathcal{F}} = \sum_{p_i \in \mathcal{P}_s} \rho\Big(\mathcal{F}(\theta_i, \phi_i) - \rho_i)\Big) + \gamma \sum_{\theta,\phi \in \mathcal{B}_s} \rho\Big(\|\mathcal{H}\mathcal{F}(\theta,\phi)\|_F\Big) \tag{2}$$

where γ is a constant, $\rho(.)$ denotes Huber penalizer [21], $\|.\|_F$ is the Frobenius norm, and $\mathcal{H}$ represents Hessian matrix. The energy term in (2) is minimized using Iterative Reweighted Least Squares (IRLS). For more efficiency, we represent $\mathcal{F}_s$ as a linear combination of N_B basis functions,

$$\mathcal{F}(\theta,\, \phi) = \sum_{j=1}^{N_B} d_j \phi_j(\theta,\, \phi) \tag{3}$$

Basis functions employed in our algorithm are introduced in Sec. V-A. At each IRLS iteration, the data term will be a quadratic function of $\mathbf{d} = (d_1, \cdots, d_{N_B})$. As shown in Klowsky et al [22], the regularization term can also be reformulated as a quadratic function of $\mathbf{d}$. Huber weights for data term decrease during re-weighting iterations for outliers (encoded with color in Fig. 3b). In the regularization term, most (θ, ϕ) cells will be assigned large Huber weights, resulting in penalization of second-order derivatives and smooth surface. But these weights will decrease for a sparse pattern of cells, which leads to less regularization and possibly surface bending.

D. MRF-based mask optimization

Through reconstruction approach of Sec. IV-C we have a dense and piece-wise smooth surface computed over maximum surface rectangle $\mathcal{B}_s$. The purpose of this section is to find the optimal mask on $\mathcal{B}_s$ that cuts segment surface $\mathcal{F}_s$ against other segments as well as ground. Finding the optimal supersurface mask is defined as an energy minimization on an MRF over Ω_ϵ. An MRF formulation helps to aggregate LiDAR-based and image-based information about segment mask. Since we have multiple segments with overlapping $\mathcal{B}_s$, a multi-label MRF is employed. We find the optimal labels ℓ for (θ, ϕ) cells that minimize the MRF energy. Let $\mathcal{G}_s = (\mathcal{V}_s, \mathcal{E}_s)$ define a graph for each segment. $\mathcal{V}_s$ is the set of (θ, ϕ) cells on $\mathcal{B}_s$ and $\mathcal{E}_s$ is the set of 8-neighborhood edges in $\mathcal{B}_s$ connecting nearby cells. We define MRF unary (first order) potentials $D(i)$ for each vertex v_i based on geometric features in LiDAR space, and compliment it with pairwise (second-order) potentials $V(i,j)$ for each edge $\{v_i, v_j\}$ based on image gradients. The unary potential describes the likelihood of a particular (θ, ϕ) cell to be occupied by segment s. While the pairwise term enforces shape smoothness and encourages boundaries to coincide with sharp image edges. Let ℓ be the global labeling for all cells in all segments. The overall energy for segment s given ℓ is,

$$E_s(\ell) = \sum_{v_i in \mathcal{V}_s} [\ell_i = s] D(i) + \sum_{\{v_i, v_j\} \in \mathcal{E}_s} [\ell_i = s, \ell_j \neq s] V(i,j) \tag{4}$$

where $[\cdots]$ is Iverson bracket, and ℓ_i denotes the segment label for v_i. The global MRF energy is defined simply as,

$$E(\ell) = \sum_{s \in \mathcal{S}} E_s(\ell) \tag{5}$$

where $\mathcal{S}$ is the set of all segments including an extra ground segment. As we have a multi-label energy we must use move-making approximation algorithms [13], e.g. $\alpha\beta$-swap or α-expansion. In most cases it is sufficient to perform 3 iterations of α-expansion [6] to converge to a locally optimal solution. In what follows we describe in detail the computation of $D(i)$ and $V(i,j)$.

E. First-order potentials

The unary term for segment s is the combination of a prior D_p and a data term D_d. The prior assigns larger potentials to cells that are closer to the anchor $p_s = (\theta_s\, \phi_s\, \rho_s)$. And the data term is defined in terms of the projection of segment assigned points $\mathcal{P}_s$ onto $\mathcal{B}_s$. Represented as a spatial grid, the overall unary term is computed as,

$$D(\theta,\phi) = G_s(\theta,\phi) + \alpha \sum_{p_i \in \mathcal{P}_s} w_i\, K_i(\theta,\phi) \tag{6}$$

where G_s and K_s are 2D Gaussian kernels, α is a constant, and w_i is the weight assigned to p_i. G_s has mean (θ_s, ϕ_s) and standard deviation $\Delta\theta/6 = \Delta\phi/6$, and covers the whole box. K_i is centered on (θ_i, ϕ_i) and has a variance equal to LiDAR's characteristic angular error variance. Point weights w_i are also computed as,

$$w_i = \exp\Big(\frac{-d(p_i, p_s)^2}{2\tau^2} + \frac{-(\rho_i - \mathcal{F}(\theta_i,\phi_i))^2}{2\sigma_F^2}\Big) \tag{7}$$

where $d(.,.)$ denotes Euclidean distnace, τ is the assignment threshold from Sec. IV-B, and $\sigma_F = 0.3m$. Note the weighting function in (7) gives less weight to surface outliers or points that are far from the anchor.

F. Pairwise potentials

Let $\Gamma(v)$ denote the projection of a vertex v onto the image. For each $\{v_i, v_j\} \in \mathcal{E}_s$, the idea behind pairwise potentials is to see if the line segment connecting $\Gamma(v_i)$ and $\Gamma(v_j)$ crosses any strong gradients. This approach has been successfully employed as an affinity metric in contour-based segmentation [25], [26]. Similar to [25] we define the affinity distance $d_e(v_i, v_j)$ as the maximum edge map value along the connecting line segment. The pairwise energy between v_i and v_j is then defined as,

$$V(i,j) = \xi \exp\Big(\frac{-d_e(v_i, v_j)^2}{2\sigma_e^2}\Big) \tag{8}$$

where σ_e and ξ are constants. Note if the line segment crosses an edge the pairwise potential $V(i,j)$ will be close to zero. This encourages v_i and v_j to have different labels. We also project the vertex 3D point $\mathbf{X} = (\theta\ \phi\ \mathcal{F}_s(\theta,\phi))$ as,

$$\tilde{\mathbf{x}} = \mathbf{K}_c\, [\,\mathbf{R}\,|\,\mathbf{t}\,]\, \mathbf{X} \tag{9}$$

where $\tilde{\mathbf{x}}$ is the projected point in homogeneous coordinates, $\mathbf{K}_c$ is camera projection matrix, and $[\,\mathbf{R}\,|\,\mathbf{t}\,]$ is the extrinsic parameters converting LiDAR to camera reference system. We simply use Canny edge detector to quickly compute image edge map.

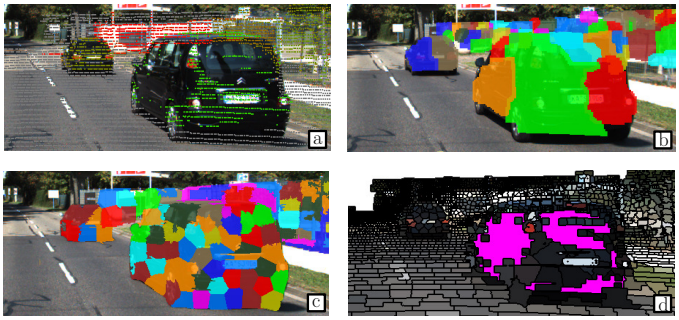


Fig. 5: Superpixels generated for (a) a sample scene using different methods, (b) proposed method, (c) SLIC [12], (d) DASP [18]. No superpixel is generated for the pink region.

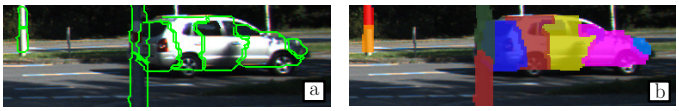


Fig. 6: Transferring supersurfaces from Ω_ϵ to image plane is susceptible to parallax, as the pole occludes the car in (a). This can be solved through (b) depth-ordering of surface segments.

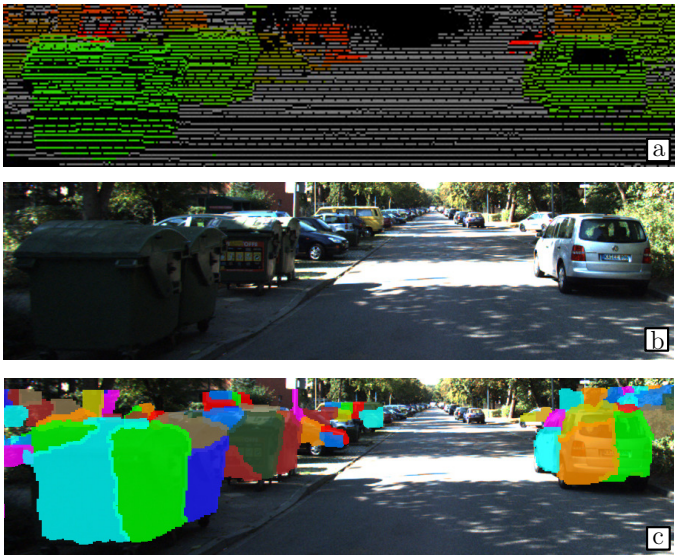


Fig. 7: A sample scene from KITTI containing obstacles of different classes, with (a) LiDAR measurements and (b) image data. Supersurface segments are extracted for non-ground points in (c).

V. Experimental Results

In order to evaluate our method, we test it on 30 LiDAR-image samples from KITTI raw dataset [27]. LiDAR point clouds have 64 vertical layers and are synchronized with image data. Using Interactive Segmentation Tool [1], object contours are annotated on the image plane for this subset. Estimated supersurfaces are projected onto the image plane (Sec. IV-D) and their boundaries are compared with ground truth. We use average boundary recall (Avg BR) to evaluate the results of various methods. BR measures average fraction of ground truth edges falling within at least $2\epsilon^\circ$ of a segment boundary. High BR shows that the superpixels are well aligned with ground truth object boundaries. Since we are interested in efficient representation of the scene, there is a preference for fewer segments for a fixed BR. Therefore, we tune the segment size parameter for each method so they produce relatively the same number of segments to cover non-ground regions.

A. Implementation

We set maximum supersurface size W to 2.0 meters and angular cell size ϵ to 0.2° for our experiments. Model parameters are selected based on LiDAR range and angular accuracy, and the constants are tuned heuristically. Depending on ϵ we need different sets of nested basis functions to cover the angular grid Ω_ϵ. Let $L^2(\mathbb{R})$ denote the set of all functions $f : \mathcal{R} \to \mathcal{R}$ for which $\|f\|^2$ is integrable over $\mathcal{R}$. A multi-resolution analysis of L^2 is a nested set of vector spaces $V^0 \subset V^1 \subset V^2 \subset \dots$ where $clos_{L^2}(\cup_{k\in\mathbb{Z}} V^k) = L^2(\mathbb{R})$. The basis functions for the spaces V^j are called scaling functions $\{\phi_i^j\}$. There are many possibilities for the choice of multi-resolution basis functions, but we choose square B-spline wavelets [23] due to their simplicity and continuous first and second order derivatives. Note these derivatives (used to compute $\mathcal{H}$ in Eq. 2) are precomputed and cached. Currently the method is implemented in Python and uses `pymaxflow`[2] to implement segment graphs and to perform max-flow/min-cut. With this implementation and selected parameters it takes 5.5 seconds on average to process one frame on a single-threaded Intel Core i7 CPU.

B. Discussion

We compare our supersurfaces with the results of SLIC [3] [12] and Veksler et al [4] [13] as two image-based superpixel segmentation techniques. We also experiment with DASP [5] [18] as an RGB-D based method. Since DASP requires a depth channel with the same size as the input image, we project our sparse point clouds onto the image plane and generate semi-dense 16-bit `pgm` files. Fig. 5 shows the results different methods on a scenario consisting of two cars at different distances. Depth-aware techniques (supersurface and DASP) produce segments with approximately similar physical size. While image-based methods (SLIC) produces segments that are of the same spatial size. As shown in Fig. 5c, this results in significantly more segments for near object while they have a fixed physical size. This is a disadvantage for 3D scene processing, as we prefer efficient object representations for tasks such as tracking. There is also noticeable missing data in LiDAR measurements of Fig. 5a due to vehicles dark color. As a result, DASP fails to generate segments for these regions (marked as pink in Fig. 5d). Table I summarizes the quantitative results for various methods in terms of average boundary recall (Avg BR) for a relatively fixed average number of segments covering non-ground regions. Superpixel segmentation techniques that are purely image-based (SLIC

[1] https://www2.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/resources.html

[2] https://github.com/pmneila/PyMaxflow

[3] http://ivrl.epfl.ch/research/superpixels

[4] http://www.csd.uwo.ca/~olga/Projects/superpixels.html

[5] https://github.com/Danvil/asp

	SLIC [12]	Veksler [13]	DASP [18]	Proposed
Avg Seg	73.6	77.5	71.1	70.2
Avg BR	0.66	0.62	0.75	0.83

TABLE I: Average boundary recall (Avg BR) and average number of segments for non-ground regions (Avg Seg) for two image-based (SLIC, Veksler), one RGB-based (DASP), and the proposed method.

and Veksler) have poor boundary recall when lower number of segments are allowed (or equivalently segment size is large). This is primarily because image boundaries do not in general coincide with object boundaries, and with larger segments a smaller fraction of image boundaries are covered. DASP performs better compared to SLIC and Veksler, but has less boundary recall compared to the proposed method. This is partly due to its failure to estimate surface normals from simulated `pgm` depth map.

C. Parallax and class-independence

Since LiDAR and camera are mounted at different positions on ego-vehicle, they have different viewpoints and observe different things. This is referred to as parallax, and results in overlapping segments when they are transferred from Ω_ϵ to the image plane. Fig. 6a represents such case where pole segments are occluding the vehicle. As shown in Fig. 6b, this can be resolved by depth-ordering of layers and checking their estimated surface $\mathcal{F}_s$.

Fig. 7a-b represents a scene with irregular obstacles from different classes with ground LiDAR points marked in gray. Segmented supersurfaces shown in Fig. 7c are computed for non-ground regions, and work equally well for all objects regardless of their class.

VI. Conclusion

An efficient surface-based over-segmentation approach was introduced in this paper, combining accurate but sparse range measurements from a LiDAR with dense image data. As shown in the results, compared to image-based superpixels our supersurfaces are more efficient when structural information are available for the scene. There are a few possible directions for future work. We can couple supersurfaces with a tracking module and refine them through accumulating temporal data. As mentioned in Sec. IV-B, semantic information could be incorporated for further improvement of segments. Supersurface merging can also be investigated in order to form objects. Finally a multi-processor version of the overall work will be implemented, as many processes in the proposed method could run in parallel.

References

[1] B. Fulkerson, A. Vedaldi, and S. Soatto, "Class segmentation and object localization with superpixel neighborhoods," in *Computer Vision, 2009 IEEE 12th International Conference on*. IEEE, 2009, pp. 670–677.

[2] A. Bódis-Szomorú, H. Riemenschneider, and L. Van Gool, "Superpixel meshes for fast edge-preserving surface reconstruction," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 2011–2020.

[3] Z. Li and J. Chen, "Superpixel segmentation using linear spectral clustering," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 1356–1363.

[4] A. Levinshtein, A. Stere, K. N. Kutulakos, D. J. Fleet, S. J. Dickinson, and K. Siddiqi, "Turbopixels: Fast superpixels using geometric flows," *IEEE transactions on pattern analysis and machine intelligence*, vol. 31, no. 12, pp. 2290–2297, 2009.

[5] F. Guney and A. Geiger, "Displets: Resolving stereo ambiguities using object knowledge," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 4165–4175.

[6] Y. Boykov, O. Veksler, and R. Zabih, "Fast approximate energy minimization via graph cuts," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 23, no. 11, pp. 1222–1239, 2001.

[7] P. F. Felzenszwalb and D. P. Huttenlocher, "Efficient graph-based image segmentation," *International journal of computer vision*, vol. 59, no. 2, pp. 167–181, 2004.

[8] J. Shi and J. Malik, "Normalized cuts and image segmentation," *IEEE Transactions on pattern analysis and machine intelligence*, vol. 22, no. 8, pp. 888–905, 2000.

[9] A. P. Moore, S. J. Prince, and J. Warrell, "lattice cut-constructing superpixels using layer constraints," in *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*. IEEE, 2010, pp. 2117–2124.

[10] M.-Y. Liu, O. Tuzel, S. Ramalingam, and R. Chellappa, "Entropy rate superpixel segmentation," in *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*. IEEE, 2011, pp. 2097–2104.

[11] M. Van den Bergh, X. Boix, G. Roig, B. de Capitani, and L. Van Gool, "Seeds: Superpixels extracted via energy-driven sampling," in *European conference on computer vision*. Springer, 2012, pp. 13–26.

[12] R. Achanta, A. Shaji, K. Smith, A. Lucchi, P. Fua, and S. Süsstrunk, "Slic superpixels compared to state-of-the-art superpixel methods," *IEEE transactions on pattern analysis and machine intelligence*, vol. 34, no. 11, pp. 2274–2282, 2012.

[13] O. Veksler, Y. Boykov, and P. Mehrani, "Superpixels and supervoxels in an energy optimization framework," *Computer Vision–ECCV 2010*, pp. 211–224, 2010.

[14] K. Yamaguchi, D. McAllester, and R. Urtasun, "Efficient joint segmentation, occlusion labeling, stereo and flow estimation," in *Computer Vision–ECCV 2014*. Springer, 2014, pp. 756–771.

[15] C. Vogel, K. Schindler, and S. Roth, "Piecewise rigid scene flow," in *Proceedings of the IEEE International Conference on Computer Vision*, 2013, pp. 1377–1384.

[16] M. Menze and A. Geiger, "Object scene flow for autonomous vehicles," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2015, pp. 3061–3070.

[17] J. Papon, A. Abramov, M. Schoeler, and F. Worgotter, "Voxel cloud connectivity segmen/tation-supervoxels for point clouds," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, 2013, pp. 2027–2034.

[18] D. Weikersdorfer, D. Gossow, and M. Beetz, "Depth-adaptive superpixels," in *Pattern Recognition (ICPR), 2012 21st International Conference on*. IEEE, 2012, pp. 2087–2090.

[19] D. Held, D. Guillory, B. Rebsamen, S. Thrun, and S. Savarese, "A probabilistic framework for real-time 3d segmentation using spatial, temporal, and semantic cues," in *Proceedings of Robotics: Science and Systems*, 2016.

[20] M. Montemerlo, J. Becker, S. Bhat, H. Dahlkamp, D. Dolgov, S. Ettinger, D. Haehnel, T. Hilden, G. Hoffmann, B. Huhnke, *et al.*, "Junior: The stanford entry in the urban challenge," *Journal of field Robotics*, vol. 25, no. 9, pp. 569–597, 2008.

[21] P. J. Huber, *Robust statistics*. Springer, 2011.

[22] R. Klowsky and M. Goesele, "Wavelet-based surface reconstruction from multi-scale sample points," 2014.

[23] F. Calakli and G. Taubin, "Ssd: Smooth signed distance surface reconstruction," in *Computer Graphics Forum*, vol. 30, no. 7. Wiley Online Library, 2011, pp. 1993–2002.

[24] M. H. Daraei, A. Vu, and R. Manduchi, "Velocity and shape from tightly-coupled lidar and camera," in *2017 IEEE Intelligent Vehicles Symposium*, 06/2017 2017. [Online]. Available: http://escholarship.org/uc/item/0vs6q8m4

[25] P. Arbelaez, M. Maire, C. Fowlkes, and J. Malik, "Contour detection and hierarchical image segmentation," *IEEE transactions on pattern analysis and machine intelligence*, vol. 33, no. 5, pp. 898–916, 2011.

[26] ——, "From contours to regions: An empirical evaluation," in *Computer Vision and Pattern Recognition, 2009. CVPR 2009. IEEE Conference on*. IEEE, 2009, pp. 2294–2301.

[27] A. Geiger, P. Lenz, C. Stiller, and R. Urtasun, "Vision meets robotics: The kitti dataset," *International Journal of Robotics Research (IJRR)*, 2013.