Strip Histogram Grid for Efficient LIDAR Segmentation from Urban Environments

Thommen Korah
Nokia Research Center, Hollywood
{thommen.korah}@nokia.com

Swarup Medasani Yuri Owechko
HRL Labs, Malibu
{smedasani,yowechko}@hrl.com

Abstract

As part of a large-scale 3D recognition system for LIDAR data from urban scenes, we describe an approach for segmenting millions of points into coherent regions that ideally belong to a single real-world object. Segmentation is crucial because it allows further tasks such as recognition, navigation, and data compression to exploit contextual information. A key contribution is our novel Strip Histogram Grid representation that encodes the scene as a grid of vertical 3D population histograms rising up from the locally detected ground. This scheme captures the nature of the real world, thereby making segmentation tasks intuitive and efficient. Our algorithms work across a large spectrum of urban objects ranging from buildings and forested areas to cars and other small street side objects. The methods have been applied to areas spanning several kilometers in multiple cities with data collected from both aerial and ground sensors exhibiting different properties. We processed almost a billion points spanning an area of 3.3 km^2 in less than an hour on a regular desktop.

1. Introduction

This work describes an approach for segmenting 3D objects from high-resolution scans of complex urban environments. Advances in sensor technology have enabled such colorized point clouds to be routinely collected using both ground-based and airborne LIDAR platforms. The push towards location-based services has increased demand for highly accurate digital maps of urban environments. The input 3D data contains millions of data points $\mathbf{p} = (\mathbf{x}, \mathbf{y}, \mathbf{z})$ that store the spatial coordinates and possibly RGB color information. Segmentation can provide valuable contextual information to subsequent recognition or scene understanding modules, making these tasks more efficient. Millions of 3D points need to be reduced to perceptually "meaningful" groupings. To be effective for target recognition, simulation, or disaster planning, processing must scale sub-

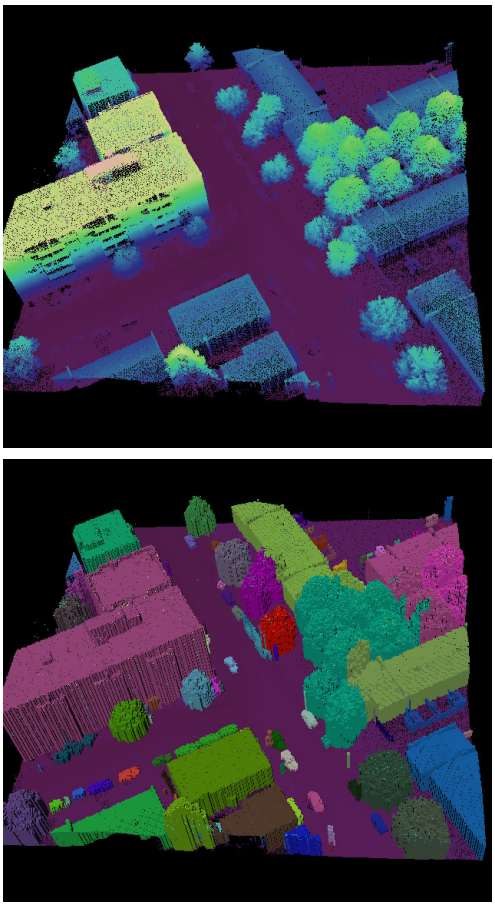


Figure 1: Top image is an input pointcloud for a 100x100 square meter tile color-mapped by height. Bottom shows the result of segmentation. Each colored region ideally corresponds to a physical object. This tile has over 3 million points.

linearly with the number of points. As a key part of our 3D recognition system that demonstrated over 60% accuracy on 40 classes, segmentation took less than an hour on a regular PC to process a collection of nearly 1 billion points.

Detailed geometric data at city-scales has not been pos-

sible to collect from real urban environments until recently [15]. Figure 1 shows an input tile and a sample result where each color boundary encapsulates a segmented 3D object. Region boundaries in image segmentation [4] may only correspond to differences in 2D perceptual cues such as color or texture, whereas LIDAR segmentation recovers individual geometric structures from the scene. Typical urban environments are cluttered; several objects touch or overlap with others making it difficult to delineate their physical extents. The nature of data acquisition results in irregular sampling without any inherent knowledge of connectivity or topology. Robustness to partial objects and non-uniform points is therefore important. Prior knowledge of the world enables humans to quickly decompose the scene into a few semantically plausible ones. Similarly, LIDAR processing results must adhere to the regularity found in the world and the way we perceive it.

The key insight behind our 3D algorithm is a new representation scheme called the *Strip Histogram Grid* (SHG) that efficiently captures such regularity eg., objects rest on the ground and appear connected as coherent objects. Any recognition system needs to draw upon a rich, compact, yet meaningful representation of the primitive 3D points. Popular methods include range images sampled on a regular 2D lattice [8], polygon meshes built from triangulation [10], and voxels [19] which divide space into a regular grid storing statistics of each cell. Our scheme is derived from voxel grids and represents the scene as a regular lattice of vertical 3D strips, where each strip is a histogram of population. This retains the computational efficiency of voxels while encapsulating a larger region to compute more global statistics. The strip grid effectively captures the underlying scene structure where objects rise vertically up from the lateral ground.

In the next section, we describe some related work. We then describe our SHG representation. This is followed by methods that segment the scene into separate physical objects for recognition. Due to the wide range of objects in an urban scene, our segmentation framework is exercised multiple times with different grouping constraints tailored to specific object classes. Each stage gradually refines our knowledge of the scene and makes use of information from the preceding steps as context. Finally, various results are shown for a large, modern city.

2. Previous Work

Efficient interpretation of large-scale, unstructured point clouds requires a solution to three sub-problems: segmentation, classification, and contextual reasoning. Various methods have been proposed for shape matching and classification of isolated rigid objects by comparing them with stored models [18, 12]. 3D shape descriptors such as spherical harmonics [11], shape contexts [2], or spin images [9] are typically employed for retrieval. The normalized cuts framework of [20] and tensor voting methods of [14] attempt to first fit local surfaces to 3D points, but will not scale well to large outdoor scenes.

3D segmentation and contextual reasoning in outdoor urban environments has been less studied; data resolution was often insufficient to capture ground objects such as poles, fences, etc. Nevertheless, specialized techniques have been proposed for building segmentation, forest detection, and other remote sensing applications [13, 1, 5, 7]. Since our data was acquired from very high resolution aerial and ground sensors, we attempt to extract a wide range of object types ranging from skyscrapers and forests to power lines, highway signs, and fire hydrants. We also target processing speeds that are at least an order of magnitude faster than those reported by recent methods [13, 3, 6].

Computational efficiency and the target application usually govern the choice of representing 3D geometric data. Level sets [17] and other continuous approximations like B-splines or Bezier curves have a low memory footprint, but are not amenable to segmenting out physical objects. Mesh-based representations [10] require non-trivial processing to construct and cannot easily handle dynamic updates to the input data. Other implicit geometry representations such as voxels allow efficient processing, but can be sensitive to missing information and empty cells since they only store local statistics. Unlike our strip grid, the geometric units in the above methods cannot easily incorporate scene-level context which is important for good segmentation.

Recently, Carlberg [3] and Golovinskiy [6] also reported results on very high density pointclouds. The segmentation of [3] requires reconstructing surfaces by triangulation, which can be slow and sensitive to noise. In contrast, we take the raw points and efficiently populate the strip grid cells. They merge results of algorithms that work separately on aerial and ground collections, with very strict assumptions on the ordering of acquired 3D scans. Our segmentation method is not dependent on sensor type or ordering and is therefore generally applicable to any outdoor geometry. The min-cut segmentation of Golovinskiy [6] assigns foreground or background labels to each point by iteratively operating at 3 scales, using heuristics to pick the best one. They only attempt to identify small objects with radius less than 5 meters. Our objects of interest encompass all sizes and shapes, and we avoid labeling a billion points by organizing them into larger semantic units that capture important global cues.

3. Strip Representation

The 3D scene is represented as a grid of vertical 3D strips called the Strip Histogram Grid, where each strip S is a population histogram of points that fall within it. More global statistics can be computed across cells in the entire strip.

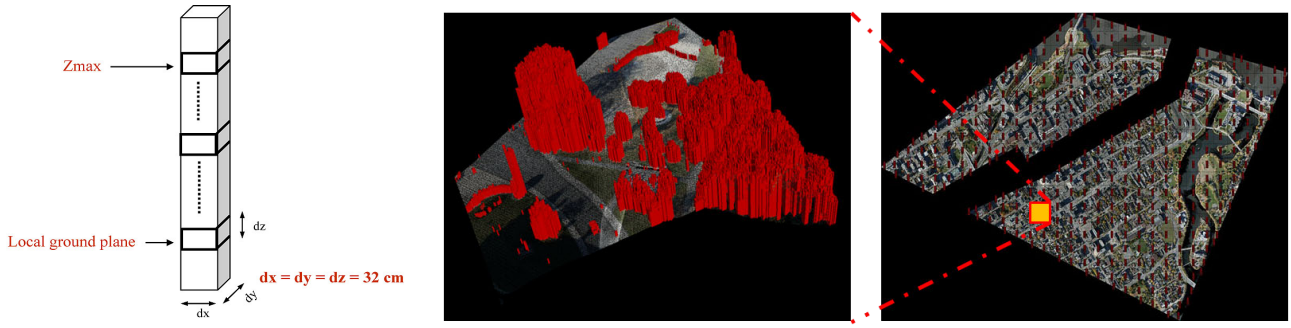


Figure 2: (a) A strip is a vertical histogram of population with cell dimensions dx, dy, and dz and has properties associated with it. (b) Populated SHG strips plotted in red for an example tile; strips belonging to 3D objects such as trees, fences, and poles "pop out" of the scene; (c) The entire data consists of 376 tiles spanning an area of 3.3 km^2 from downtown Ottawa.

The lateral dimensions of the strip are dx and dy, while dz is the width of the vertical histogram bins. All experiments presented here use $dx = dy = dz = 32cm$ and have yielded good results on data from a variety of sensors. The SHG representation maintains the efficiency of voxels, but more naturally encodes the underlying structure of the real world. Objects in the real world rise up from the ground, implying that grouping adjacent strips together will encapsulate the whole object. Smooth height distribution is another strong cue for identifying objects, and strips make it easy to compare these properties. The strength of this representation is that it makes many of the segmentation algorithms intuitive and also allows borrowing techniques from 2D image processing. An illustration of a strip and the set of populated strips (points above ground plane) for a tile is shown in Fig. 2.

Initializing the strip cells result in an $N_x \times N_y$ grid of strips, where $N_x = \lceil \frac{\max(\mathbf{x}) - \min(\mathbf{x})}{dx} \rceil$. Given a data point $\mathbf{p}_c = (x, y, z)$ in UTM coordinates, the corresponding strip S_i in which the point falls is first computed. The histogram bin $S_i(k)$ within the strip is then incremented where $k = \lfloor \frac{z - \min(\mathbf{z})}{dz} \rfloor$. Once populated, we compute a set of features or attributes for each strip based on neighborhood statistics as described below.

When dealing with outdoor 3D data, an estimate of the ground plane is a very important contextual cue. A single global ground plane is computed by histogramming the z-values of all points and estimating the lowest level with enough supporting points. This would suffice if the terrain is mostly flat. However, our data spans a large area with several topographical variations like undulating hills, steep cliffs, water-bodies, etc. Each strip therefore stores an estimate of the local ground plane S_i^{gnd}. Every cell in strip S_i has a ground support variable $G_i(k)$ that is initialized to 0. When a point is added to cell $S_i(k)$, it increments the value (support) $G_j(k)$ of all strips j that fall within the neighborhood of S_i. The neighborhood is defined as a circular region of 1 meter radius centered on S_i. The local ground plane S_i^{gnd} for strip i is then estimated as the lowest z-level with G_i greater than threshold τ_{gnd}. This simple voting method is robust to any outlier sensor points below the ground – a common problem with our ground sensor. It also avoids the need for spatial indexing to make neighborhood searches between points.

Other contextual features stored for each strip are strip height S^H relative to the local ground, extent of z-value within the strip S^{zmin} and S^{zmax}, boolean flag S^{POP} denoting whether a strip is populated above the ground, number of points above the local ground-level S^{pts}, number of continuously populated cells above ground S^C, and the local terrain slope S^{zslope} computed on a 3×3 strip neighborhood.

4. SHG Segmentation

Algorithm 1 SEGMENTSTRIPS$(S, V, \mathbf{P})$

```
curlabel ← 0
for each strip s ∈ S do
    s^label ← NIL
end for
for each strip s ∈ S do
    if V(s) = true and s^label = NIL then
        LABELREACHABLESTRIPS(s, curlabel, P)
    end if
    curlabel ← curlabel + 1
end for
```

Segmentation involves grouping strips that have similar attributes. We adopt a graph-based approach to segmentation that partition nodes into components such that each region C ideally corresponds to a 3D object in the world. The set of strips $\mathbf{S}$ constitute the nodes in the graph with edges to its 8-connected neighbors. Edge weights between strips should be lower if they belong to the same object, and

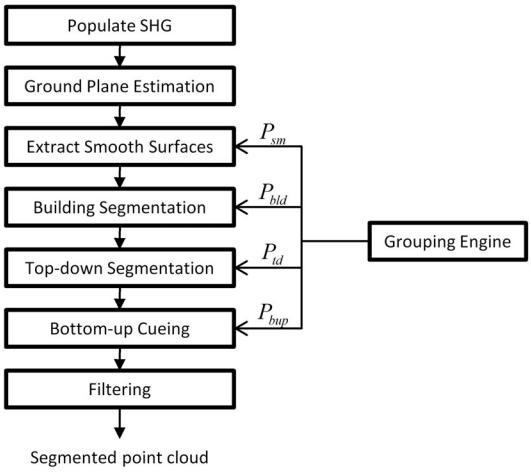


Figure 3: SHG segmentation pipeline.

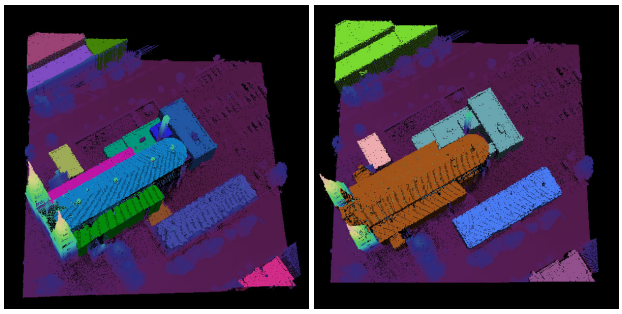
higher otherwise. These weights are derived from a function $\mathbf{P}$ that measures the compatibility between two neighboring strips S_i and S_j. This framework resembles popular graph-based approaches to image segmentation [4, 16] and can therefore borrow techniques from that domain.

Figure 3 shows the pipeline of stages in the segmentation procedure. Initialization for an input point cloud consists of building the strip grid and computing the attributes described in Section 3. A grouping engine performs segmentation in multiple phases. Global information gleaned from previous stages can be used as additional context. Smooth surfaces are extracted first – a good indication of man-made structures. The second phase grows smooth regions outward to precisely segment out building areas. Top-down segmentation then exploits height constraints to identify boundaries between non-building objects. Shorter objects that fall under overhanging structures like trees and power lines can be missed by height-based approaches alone. A bottom-up method attempts to cue objects that project out from the ground plane. A final phase filters out small and inconsistent regions with minimal points or inconsistent geometry.

Each phase invokes algorithm SEGMENTSTRIPS on strips $\mathbf{S}$ with the two predicate functions $V(s)$ and $\mathbf{P}(u, v)$ as parameters. $V(s)$ determines whether grouping for a particular component may start at strip s; all reachable strips from s (based on compatibility $\mathbf{P}$) are then extracted and labeled by a connected components algorithm LABELREACHABLESTRIPS. Though simple and efficient, these compatibility constraints are able to produce very natural segment boundaries.

4.1. Smooth Surface Extraction

Smoothness is a good indicator of large man-made structures such as buildings. We further assume that regularity in height alone is a sufficient signature for making this identification. A significant amount of processing time is thus avoided because there is no need for independent point classification by surface estimation or other local neighborhood methods as in [3, 13]. A 3×3 kernel is used to first compute partial derivatives of strip height along the $\mathbf{X}$ and $\mathbf{Y}$ axes, following which the local gradient for each strip S^{zslope} is estimated. A strip is labeled as smooth $S^{sm} = true$ if the range of slope within the neighborhood is less than a threshold $\tau_z = 10$. For robustness, we only require that at least 5 out of 8 neighboring strips pass this constraint.



(a) (b)

Figure 4: (a) Smooth surfaces extracted for a given tile; different levels of the roofs are broken up and edge effects are prominent. (b) Results of building segmentation by growing the smooth surfaces.

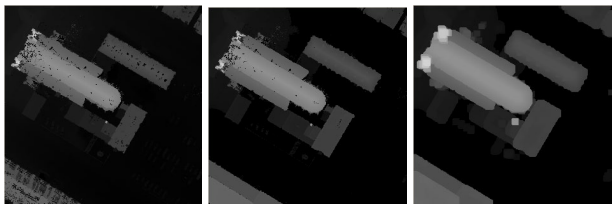
For two adjacent strips s_i and s_j, the grouping engine uses the following predicate to extract out smooth surfaces:

$$P_{sm}(s_i, s_j) = \begin{cases} true & \text{iff} \quad \begin{array}{l} (s_i^{sm} \; and \; s_j^{sm} \; and \\ \quad |s_i^{zmax} - s_j^{zmax}| < 10) \\ \text{OR} \\ ((s_i^{sm} \; or \; s_j^{sm}) \; and \\ \quad |s_i^{zslope} - s_j^{zslope}| < 10) \end{array} \end{cases} \quad (1)$$

The first condition is valid if two smooth strips have similar height (within 10 cm) and the less stringent second condition is good at capturing strips that belong to smooth, inclined surfaces. These regions are typical of flat or gabled roofs. Though split up at every roof indentation, they can be very useful as seeds for identifying potential building structures. A post-process filtering operation removes very small smooth regions (less than 20 square meters) and components that have an excessive percentage of tall strips. Typical roofs should only have tall strips along the edges rising up from the the local ground plane. Figure 4(a) shows extracted smooth regions for a tile.

4.2. Building Segmentation

Buildings are very large and have unique geometric signatures that are best extracted in a separate phase. Recognizing building regions early also provides valuable context information for urban scenes. The smooth surfaces provide seeds that can be grown and merged with other surfaces to



(a) (b) (c)

Figure 5: (a) Support-plane map G_M computed by taking the point with minimum z-value; while effective at determining ground points, spurious points within buildings make roofs noisy. (b) Map B'_M formed by fusing (a) with smoothness map. (c) Map B_M obtained after median filtering and dilating the image (b).

encompass the whole building. A key challenge in urban scenes is to geometrically separate out roofs from trees that closely hug the building facade. While stronger constraints can prevent growing smooth regions, it may exclude roof strips around building edges that can exhibit many interesting elevation patterns.

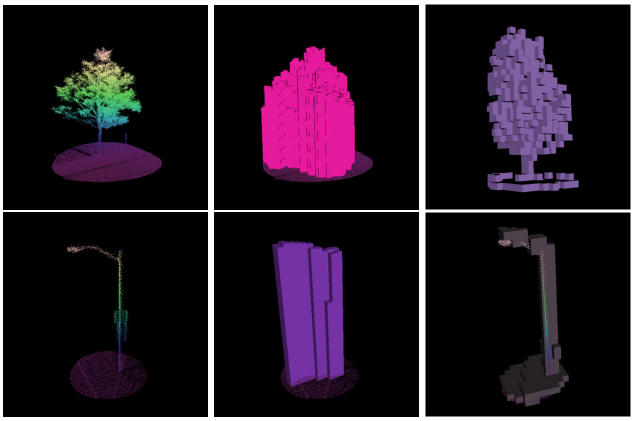
To resolve these two competing goals, we fuse information from the smoothness map and a support-plane map G_M to identify definite non-building areas. The intuition here is that unlike roofs, the ground points are visible under trees and vegetation. We calculate an image G_M of dimensions $N_x \times N_y$, where each pixel contains the minimum height from the ground plane of all points in that strip. Let S_M be another image that stores the height of all smooth strips. We produce a provisional map $B'_M = \max(G_M, S_M)$ by doing a pixel-wise max operation. B'_M is then median filtered and dilated for three iterations to produce a building map B_M. Finally, all strips or pixels in B_M with height S^{BM} less than 3 meters is set to 0. The maps constructed for the tile in Fig. 4 are shown in Fig. 5. Darker intensities in this image correspond to "brake zones" where non-smooth strips should not be merged with a building region.

The support plane map is a good indicator of supporting surfaces that could include ground as well as roofs. The $\max$ operation compensates for this by replacing pixels in smooth regions with the height of the roof. The dilation on the fused map serves to increase the margin around buildings so that broken edges along the facade may be combined. The net effect of these operations is that the buildings become prominent while the ground map suppresses the effect of adjacent trees preventing segmentation from spiraling out of control.

The predicate function for building segmentation is defined as:

$$P_{bld}(s_i, s_j) = \begin{cases} true & \text{iff} \quad (s_i^{sm} \; and \; s_j^{sm} \; and \; R_H \leq 1.4) \\ & \quad \text{OR} \\ & \quad (|s_i^{BM} - s_j^{BM}| < 300 \; and \\ & \quad R_H \leq 1.2). \end{cases} \tag{2}$$

The ratio of heights between two strips $R_H = \frac{\max(s_i^H, s_j^H)}{\min(s_i^H, s_j^H)}$ is used to identify discontinuities. The first condition thus allows two smooth strips possibly belonging to different roofs to merge. The second condition connects loose strips at the edges of the facade while using B_M to avoid capturing non-building strips. Adjacent strips with a difference in B_M of 3 meters or more are not allowed to merge. Figure 4(b) shows the buildings obtained by growing the seed regions.



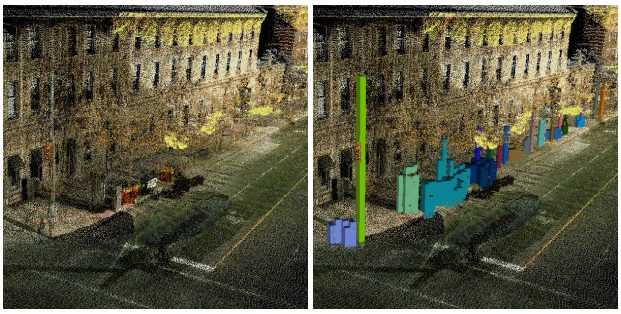
I II III

Figure 6: Top-down segmentation: (I) Two sample point clouds of a tree and light standard. (II) Labeled strips after top-down segmentation. (III) Segmented points can be excised to perform more sophisticated analysis.

4.3. Top-Down Segmentation

We exploit the nature of height distribution in the real world to determine object connectivity. Top-down segmentation assumes that all strip points above the ground belong to the same object. This is equivalent to a nadir view of the scene where only objects that are not blanketed by other overhanging structures are visible. Small street-side entities like trashcans or fire-hydrants that fall under surrounding foliage may be "hidden". Nevertheless, over 70% of the objects – especially the larger structures in an urban setting – can be retrieved in this scheme. Top-down view of the scene combined with the strip representation allows extremely efficient processing. Where the points are all captured from an aerial sensor, this scheme alone is sufficient for segmentation as the sides of objects are seldom visible without ground data.

The predicate used for grouping is only applied to strips



(a) (b)

Figure 7: Bottom-up cueing: (a) Several street objects blanketed by overhanging trees that top-down cueing will not retrieve. (b) Cued poles and other street side objects.

that have not already been classified as a building. It uses the ratio of heights between adjacent populated strips for grouping:

$$P_{td}(s_i, s_j) = \{true \quad \text{iff} \quad (s_i^{POP} \; and \; s_j^{POP} \; and \; R_H \leq 1.4). \tag{3}$$

Column II of Fig. 6 shows the result of top-down segmentation on two isolated pointclouds of a tree and light standard. The resulting volume is a tight bound around the object. Once excised from a larger pointcloud, we can perform more intensive shape analysis like voxel-based segmentation in column III.

4.4. Bottom-up Cueing

Terrestrial LIDAR data will include street-side objects that may not be visible to an aerial sensor due to occlusion or lack of resolution. Bottom-up cueing (Fig. 7) identifies highly populated strips projecting up from the ground and then proceeds to group them based on bottom-up height. Rather than taking the maximum $z-$value of points in the strip as its height S^H, the distribution of points within the strip cells is exploited. The longest array of consecutively populated cells from the ground is identified to compute a bottom-up height S^C. Since ground data is more prone to noise and clutter, we use a larger neighborhood to make computation of S^C robust. To be considered populated, the vertical Gaussian weighted population at a cell should exceed threshold $\tau_c = 5$ points. The predicate function $\mathbf{P}_{bup}$ for bottom-up cueing is similar to $\mathbf{P}_{td}$ (3), except that all height and ratio computations are based on S^C instead of the top-down strip height.

5. Results

We processed our algorithms on collections from 3 major cities – Ottawa, Budapest, and Boston. In addition to spanning multiple city blocks, the terrain contained woods, highways, rivers, cliffs, and bridges. The sensor characteristics were different for all these cities with some containing purely aerial data. We present results only from Ottawa, which was the most challenging dataset [15], with an average spatial density of 300 $points/m^2$ and reported sensor accuracy of 4 cm. Due to inconsistent intensity information across sensors, we rely on 3D geometric data only. The data is processed tile-by-tile, where each tile has 100m sides with a total of 376 tiles. We present both qualitative and quantitative results on a set of ground truthed data.

Figure 8 shows a visualization of segmentation results on different tiles. Strips belonging to different segments are drawn in a unique color. We evaluate segmentation results on a set of 5 tiles that were manually marked and labeled. Each ground truth object is tested against segmented regions to measure (a) precision or how many of the segmented regions were relevant and (b) recall or how many of the relevant regions were segmented correctly. Precision and recall are combined using the harmonic mean to compute the F-Measure as $F = 2 * \frac{precision \times recall}{precision + recall}$. Two factors may affect this metric adversely. First, the provided ground truth was not exhaustive (covered only about 80% of the objects) and only contained labels for objects that were to be recognized. Since our segmentation algorithm is designed to extract all objects in the scene, it may lower precision values. Secondly, medium-sized objects like trees and cars are labeled as point objects in the ground truth with a center and radius. Segmentation extracts tight boundaries around such asymmetrically shaped objects leading to reduced recall. We therefore complement the F-Measure with a qualitative evaluation; each ground truth object is presented with overlapping segmentations overlaid on it in an automatically generated form. A user can then mark on the form whether the object was correctly segmented or not.

Table 1 gives the segmentation performance based on analyst feedback as well as the F-Measure. Qualitatively, our technique correctly extracts over 90% of objects from the 3D scene. Even with skewing, we obtain strong F-Measures for the segmentation. Figure 9 shows ground truth areas marked in red, segmented regions in blue, and the intersection of the two in green for 4 adjacent tiles. The large blue regions are in fact correctly segmented buildings that were not present in the ground truth. The figure also illustrates regions that show up as missed detections (red) due to trees and cars being simplistically represented as point objects. Detected segments reveal the structure of the city-block containing buildings, street side objects, trees, and parked cars.

Effective segmentation is inextricably linked to recognition performance on such large datasets. On a sequestered set of more than 25 exhaustively ground-truthed tiles, we have achieved over 60% recognition rate across 42 different classes. To reflect real-time operation, we first generate training examples by exercising strip segmentation on ground truthed point clouds. We learn classifiers for the ex-

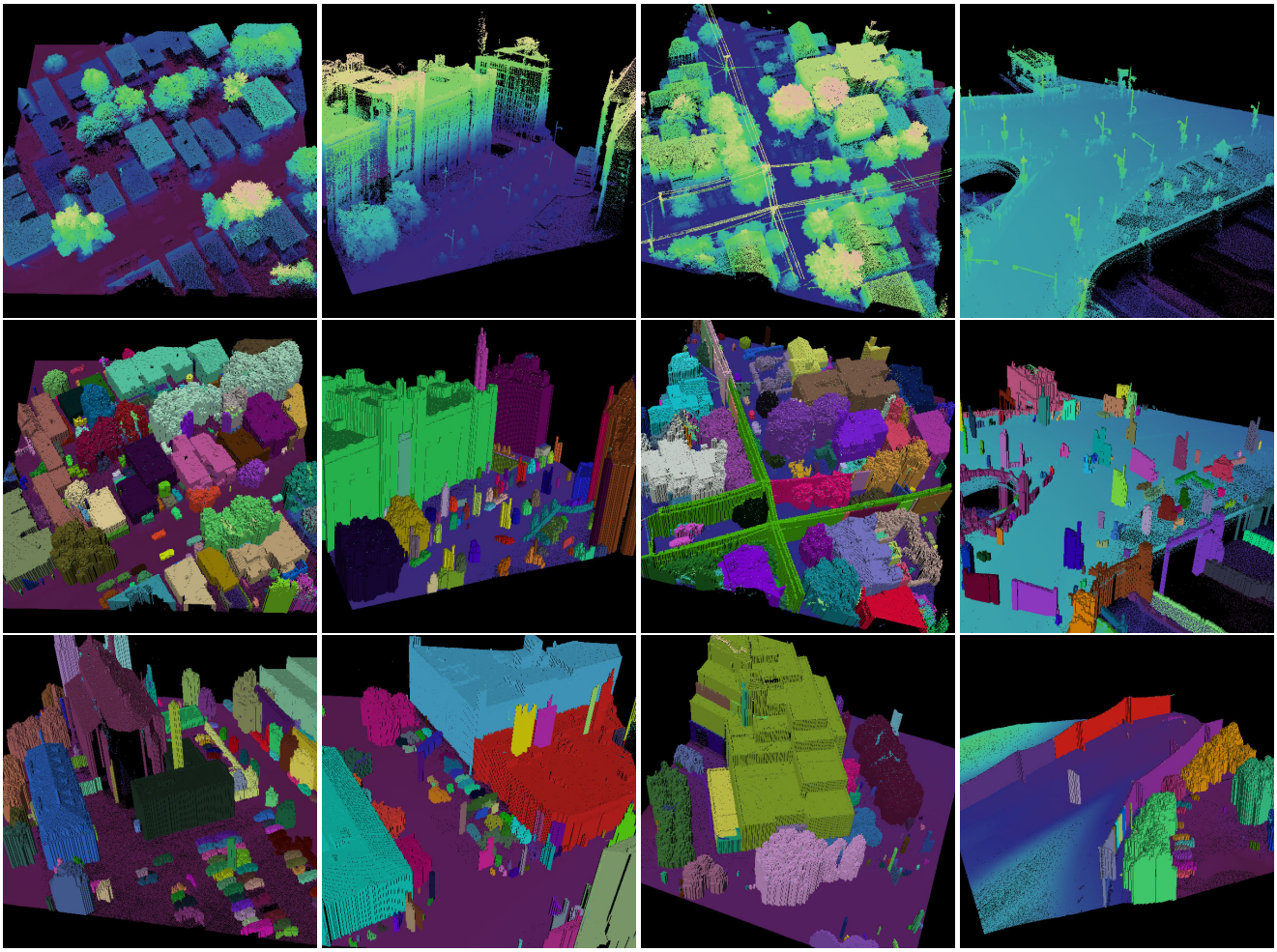


Figure 8: Results after segmentation for various tiles. Strips grouped together into a connected component are drawn with a random color. Building strips detected in Sec. 4.2 are also assigned to a unique segment. Top row contains the input tiles for the segmentation results displayed in the middle row. Third row shows additional tile results. Buildings, trees, power lines, street lights, cars, and many other objects are segmented out. The reader is encouraged to zoom in.

cised point clouds using a hierarchical set of geometric and shape feature descriptors. The probability of identification for a few select classes is shown in Table 1. Segmentation is good at pruning out background clutter resulting in more discriminative classifiers. The recognition numbers of buildings and trees are based on an area-based scoring to reflect segmentation accuracy i.e., 84% of strips within ground truthed trees were correctly recognized. Detection does not always require 100% segmentation accuracy.

Efficiency is another key ingredient when processing large pointclouds. Table 1 also shows processing details of our segmentation system. We segmented nearly billion points from 280 tiles in less than an hour on a regular PC. Computation is dominated by the strip initialization phase that estimates the local ground plane using points in the neighborhood. The average initialization time per tile is 8.3 seconds while grouping takes only 1.0 second per tile. Our processing speed of 270,000 points per second is almost two orders of magnitude faster than the most recent results from Golovinskiy [6] on the same dataset at 5690 points per second.

6. Conclusion

We described a 3D segmentation algorithm for large-scale LIDAR data collected from multiple city blocks. We define a new Strip Histogram Grid representation that allows primitive data points to be organized into larger semantic units with contextual properties associated with it. The representation is used to extract out a variety of objects from buildings to powerlines and fire hydrants. We believe that our LIDAR segmentation is more comprehensive and efficient than recently reported work. A key design issue for speed was to avoid point-wise processing of nearly 1 billion LIDAR points. Our algorithms were robust to data

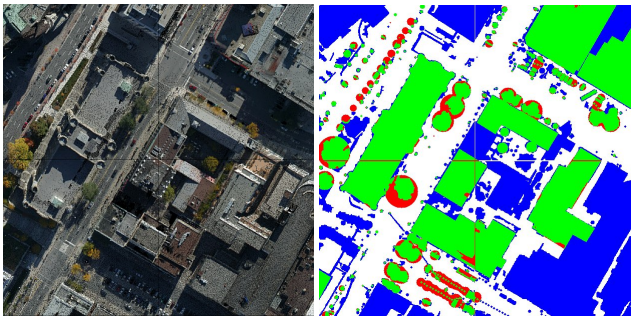


Figure 9: 4 sample tiles (left) and segmented regions (blue) overlaid on ground truth (red). Intersecting regions are plotted in green. Precision is $\frac{Green}{Green+Blue}$ and recall is $\frac{Green}{Green+Red}$.

Segmentation Performance (I)

Class	Count	Segmented	F-Measure
Point objects	488	89%	0.55
Buildings	18	100%	0.90
Trees	107	92%	0.61

Recognition (II)

Class	Count	PID
Buildings	183	0.82
Trees	47	0.84
Car	103	0.90
Street lights	108	0.86
Poles	90	0.85
Sign	62	0.73

Processing (III)

Points	0.94 billion
Area	3.3 km^2
System	3.0 GHz PC
Segmentation	59 min
Average time	9.3 sec/tile
Disk I/O	40 min

Table 1: Results: (I)Table showing qualitative and quantitative accuracy of segmentation. (II) Per-class recognition results from more than 25 sequestered tiles. (III) Processing speed and platform.

collected from different cities and sensors. We showed results on a public dataset collected in Ottawa [15].

The lack of exhaustive ground truth lowered our quantitative numbers. Steep cliff-sides may be confused for buildings because we process tiles separately. Using surrounding tiles for context to estimate a bare-earth model would resolve this. We can also combine 3D geometry with large photo collections to create high-fidelity textured models.

References

[1] P. Axelsson. Processing of laser scanner data -algorithms and applications. *ISPRS Journal of Photogrammetry & Remote Sensing*, 54(2-3):138–147, 1999.

[2] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *IEEE Trans. Pattern Anal. Mach. Intell.*, 24:509–522, 2002.

[3] M. Carlberg, J. Andrews, P. Gao, and A. Zakhor. Fast surface reconstruction and segmentation with ground-based and airborne lidar range data. In *Proc. of 3DPVT08*, 2008.

[4] P. F. Felzenszwalb and D. P. Huttenlocher. Efficient graph-based image segmentation. *Int. Journal of Comput. Vision*, 59:167–181, 2004.

[5] S. Filin. Surface clustering from airborne laser scanning data. In *In ISPRS Commission III, Symposium*, 2002.

[6] A. Golovinskiy, V. G. Kim, and T. Funkhouser. Shape-based recognition of 3D point clouds in urban environments. *International Conference on Computer Vision (ICCV)*, 2009.

[7] N. Haala and C. Brenner. Extraction of buildings and trees in urban environments. *ISPRS Journal of Photogrammetry & Remote Sensing*, 54(2):130–137, 1999.

[8] G. Hetzel, B. Leibe, P. Levi, and B. Schiele. 3d object recognition from range images using local feature histograms. In *Computer Vision and Pattern Recognition*, 2001.

[9] A. Johnson and M. Hebert. Using spin images for efficient object recognition in cluttered 3d scenes. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 21(5):433 – 449, May 1999.

[10] T. R. Jones, F. Durand, and M. Desbrun. Non-iterative, feature-preserving mesh smoothing. In *SIGGRAPH '03*, pages 943–949, New York, NY, USA, 2003. ACM.

[11] M. Kazhdan, T. Funkhouser, and S. Rusinkiewicz. Rotation invariant spherical harmonic representation of 3d shape descriptors. In *Symposium on Geometry processing*, 2003.

[12] A. Makadia and K. Daniilidis. Spherical correlation of visual representations for 3d model retrieval. *Int. Journal of Comput. Vision*, Online, 2009.

[13] B. Matei, H. Sawhney, S. Samarasekera, J. Kim, and R. Kumar. Building segmentation for densely built urban regions using aerial lidar data. *IEEE Conf. on Comp. Vision and Pattern Recognition*, 2008.

[14] C. Min and G. Medioni. Tensor voting accelerated by graphics processing units (gpu). In *Proc. of the International Conference on Pattern Recognition (ICPR)*, 2006.

[15] Neptec. Wright state 100. http://www.daytaohio.com/Wright_State100.php.

[16] C. Rother, V. Kolmogorov, and A. Blake. Grabcut - interactive foreground extraction using iterated graph cuts. In *SIGGRAPH*, 2004.

[17] J. A. Sethian. *Level Set Methods and Fast Marching Methods: Evolving Interfaces in Computational Geometry, Fluid Mechanics, Computer Vision, and Materials Science*. Cambridge University Press, 1999.

[18] J. W. Tangelder and R. C. Veltkamp. A survey of content based 3d shape retrieval methods. *Shape Modeling International*, 00:145–156, 2004.

[19] N. Vandapel, D. Huber, A. Kapuria, and M. Hebert. Natural terrain classification using 3-d ladar data. In *IEEE International Conference on Robotics and Automation*, 2004.

[20] Y. Yu, A. Ferencz, and J. Malik. Extracting objects from range and radiance images. *IEEE Transactions on Visualization and Computer Graphics*, 7:351–364, 2001.