Extracting roads from dense point clouds in large scale urban environment

Aleksey Boyko, Thomas Funkhouser

Department of Computer Science
Princeton University
35 Olden Street, Princeton, NJ 08540-5233

Abstract

This paper describes a method for extracting roads from a large scale unstructured 3D point cloud of an urban environment consisting of many superimposed scans taken at different times. Given a road map and a point cloud, our system automatically separates road surfaces from the rest of the point cloud. Starting with an approximate map of the road network given in the form of 2D intersection locations connected by polylines, we first produce a 3D representation of the map by optimizing Cardinal splines to minimize the distances to points of the cloud under continuity constraints. We then divide the road network into independent patches, making it feasible to process a large point cloud with a small in-memory working set. For each patch, we fit a 2D active contour to an attractor function with peaks at small vertical discontinuities to predict the locations of curbs. Finally, we output a set of labeled points, where points lying within the active contour are tagged as "road" and the others are not. During experiments with a LIDAR point set containing almost a billion points spread over six square kilometers of a city center, our method provides 86% correctness and 94% completeness.

Keywords: road extraction, urban environments, LIDAR

1. Introduction

Constructing semantically-tagged 3D models of urban environments is a long-standing problem with applications in navigation, planning, social engineering, and virtual tourism. In particular, semantic tagging and modeling of roads is crucial to understanding the complete structure of a city, since roads provide a continuous surface spanning an entire city, segment the city into blocks, and provide

contextual cues for recognizing smaller objects (e.g., fire hydrants are usually a fixed distance from a roadside). As such, accurate extraction of roads is an important problem in GIS analysis, scene segmentation, and object recognition.

Due to its importance, researchers have developed methods to extract roads from several types of input data, including satellite and aerial imagery, aerial LIDAR data, and terrestrial LIDAR scans.

For road extraction from satellite and aerial images, a variety of methods have been proposed, using cues based mainly on color, monochromatic intensity and texture patterns (Fortier et al., 1999; Mena, 2003). For example, common algorithmic strategies include region growing (Amo et al., 2006; Bicego et al., 2003; Hu et al., 2007; Mena and Malpica, 2005; Tesser and Pavlidis, 2000), segmentation and clustering (Ferchichi and Wang, 2005; Wan et al., 2007), machine learning (Butenuth et al., 2003; Yager and Sowmya, 2003), multi-scale extraction and refinement (Baumgartner and Hinz, 2000; Heipke et al., 1995; Mayer et al., 1998; Steger, 1998), and active contours (Laptev et al., 2000; Peng et al., 2008). These methods tend to work well in rural environments, where color and intensity is relatively distinctive and consistent within roads, and in urban environments when assumptions can be made about the structure of roads (e.g., a semi-regular grid pattern (e.g., Hu et al., 2004; Youn and Bethel, 2004)) and/or a knowledge base and carefully tuned parameters can be provided (e.g., Hinz, 2004). However, good performance has not been achieved in general urban environments (Mayer et al., 2006).

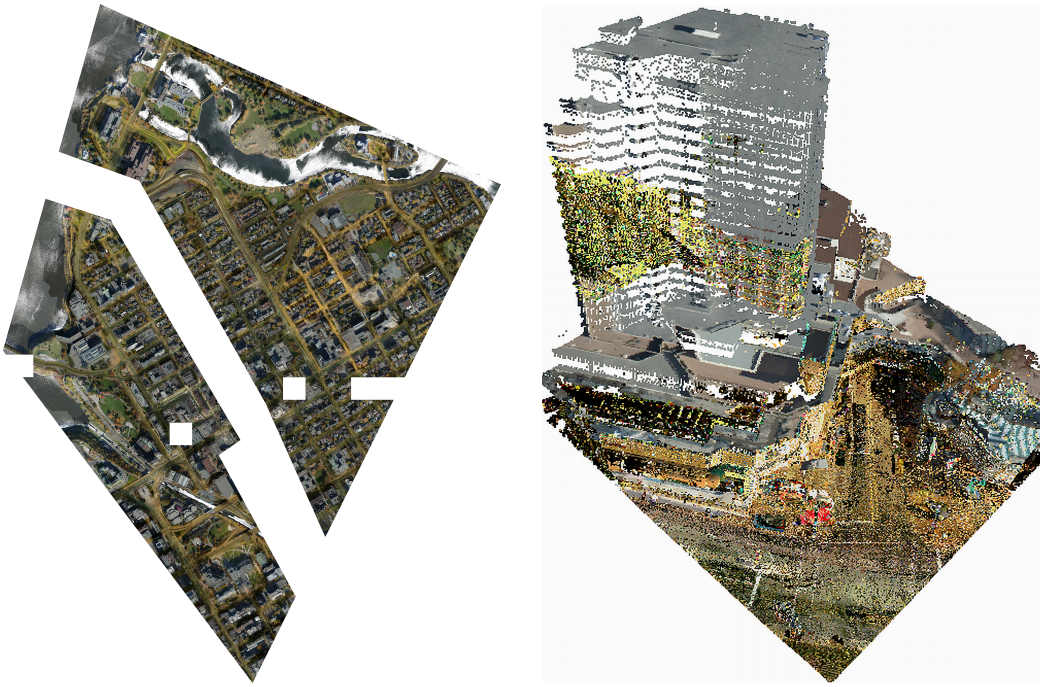
A number of papers have addressed the problem of road extraction from aerial LIDAR scans (ALS) (Alharthy and Bethel, 2003; Choi et al., 2007; Clode et al., 2007). In (Alharthy and Bethel, 2003) the points were filtered by their intensity, proximity to a digital terrain model(DTM) and then the network was extracted by finding connected components. (Clode et al., 2007) use a similar approach to select candidate road points. However, instead of using a connected component filter they extract the road area by employing a phase coded disk operator (Clode et al., 2004) and follow with joining and intersecting the extracted roads. In (Choi et al., 2007), road points were extracted by a series of circle buffers clustering points by elevation and reflectance and merging clusters based on the maximum possible slope of the road. This work has been extended to a parallel algorithm by (Li et al., 2008). (Hatger and Brenner, 2003) offer a method which does not rely on the intensity at all - it extracts roads by planar approximations along the lines of roads in an existing GIS map. (Oude Elberink and Vosselman, 2006) fit a polygonal annotated map to an ALS to assign proper third dimension to the objects, modeling discontinuities in the elevation, followed by surface reconstruction.

Other research groups have considered point clouds from terrestrial LIDAR scans (TLS) acquired with a sensor mounted on a vehicle (e.g., Chen et al., 2009; Goulette et al., 2006; Jaakkola et al., 2008; Yu et al., 2007). These methods achieve good results in some urban settings by leveraging temporal and spatial structuredness of the data and knowledge of the position and the angle of the scanner when it scans each point. Unfortunately, many point clouds, such as the one used in our study, are provided without such structure or information. Also, these methods provide limited means to deal with frequent and variably large interruptions in curb lines that occur in urban environments. Finally, road extraction proceeds by considering local windows in the direction of the scanning vehicle's movement, providing little global consistency in the case when a description of an entire road network of a city is necessary.

The goal of our work is to derive a method for extraction of roads from a large scale dense point cloud merged from multiple aerial and terrestrial source scans of an urban environment. For example, consider the point cloud shown in Fig. 1: it was collected by Neptec with one airborne scanner (78 scans/second, full waveform) and four car-mounted TITAN scanners (50 scans/second, first return only), facing left, right, forward-up, and forward-down as the car moved along all roads in a 6 kilometer^2 region of downtown Ottawa, Canada (Neptec, 2009). Scans were merged at the time of collection and output as a single point cloud containing 951 million points, each with a UTM position, intensity, color and an identifier of the source scanner. The total size of the data set is approximately 35GB. The reported error in alignments between airborne and car-mounted scans is 0.05 meters, and the reported vertical accuracy is 0.04 meters.

Such a large, dense point cloud provides unique opportunities for road extraction. Since data is included from aerial scanners, the point cloud provides large-scale coverage of all roads in an entire city, making it possible to employ algorithms that enforce global consistency of extracted roads. Since data is also included from terrestrial scanners, it provides fine-scale sampling density and precision suitable for detection of curbs and other small features of roads, and it provides coverage of all roads, including multilevel intersections, overpasses, bridges, and tunnels. This combination makes it possible to couple local and global road extraction criteria to achieve good results in difficult urban settings.

However, there are many challenges. The points from different scanners may be captured at different times, in different weather conditions, under different illumination, and with different sampling densities. The scanline-order of points acquired from each scanner may be discarded during the merging process (as is the case for our data set). Misalignments and/or ghosting of moving objects (e.g.



(a) 2D projection of the entire point cloud.

(b) Screen shot of the point cloud.

Figure 1: Point cloud acquired with aerial and terrestrial LIDAR scanners covering a six kilometer^2 region of downtown Ottawa. (a) The image on the left shows a top-down view of the scanned area, while (b) the image on the right shows a perspective view of one region (a road going under the building forming a tunnel)

.

pedestrians and cars) may appear as point clouds are merged; and, there may be large occlusions in the data set due to trees, cars, and other urban objects. Moreover, the sheer size of the data makes it impossible to store all points in memory simultaneously. These challenges make it difficult to employ traditional algorithms directly.

In this paper, we describe a method for high precision (up to a curb) road extraction from a large scale dense merged point cloud of an urban environment. The key idea is to leverage both global properties of roads (topology and smoothness) and local road features (curb boundaries) to extract roads accurately and consistently. In contrast to the previous methods, our approach deals with very large point clouds that are not expected to offer any information about the location and orientation of the scanner, nor to be organized into scanlines or grids, nor to be consistent in its color and intensity values or sampling rate (as long as sampling is dense enough to notice objects as small as a curb). Furthermore, our approach treats any multilevel structure, including long tunnels, as any other case, processing them in the same pipeline. Our method separates the stage of a global 3D fitting from accurate local 2D fitting, each of the stages working on a small subset of the larger dataset. This enables us to keep entire relevant subsets in memory while processing an extremely large point cloud, as well as to parallelize extraction routines for independent streets. Within each subset, we employ ribbon snakes to extract smooth road boundaries aligned with curbs. We have implemented a prototype system and present qualitative and quantitative results of using active contours with various curb detectors within the framework of our approach on a large multisource point cloud of an urban environment.

2. Method

Our method takes as input a point cloud and an approximate 2D map of a road network (e.g., as provided by (OpenStreetMap, 2010)), and it produces a model of the road boundaries along with a tag for every point in the cloud indicating whether it is part of a road or not. It is intended to work for road networks that can be represented as a connected set of smooth continuous surfaces of slowly changing width and elevation delimited partially by elevation discontinuities (e.g., curbs), as are commonly found in urban environments.

The pipeline of processing steps is depicted in Fig. 2. We first project the given 2D map of the road network onto the given 3D point cloud in a manner that preserves the road network's topology and ensures geometric continuity - creating a *map spline*. We then split the map spline into independent parts, *road patches*,

and extract a subcloud of relevant points for each road patch. These subclouds are small enough and can be processed independently with an out-of-core framework that requires small working sets. For each subcloud, we build a 2D *attractor map* that estimates the locations of elevation discontinuities. Then, we compute a *ribbon snake* for each road patch by optimizing an active contour that aims to maintain smoothness of its boundary while fitting its boundary to likely predictions in the attractor map. Finally, the points in the respective subcloud that fall inside the active contour are labeled as road points. The following subsections describe the motivation and implementation for each of these steps in detail.

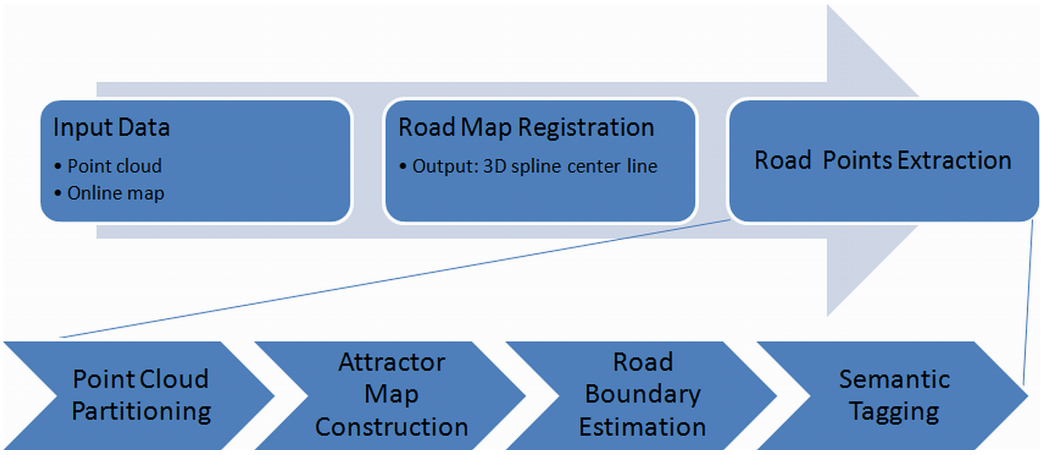


Figure 2: Processing pipeline.

2.1. Road Map Registration

The first step of our processing pipeline is to register the input 2D map of the road network with the input 3D point cloud. Typical maps acquired on-line (e.g., OpenStreetMap, 2010) are provided as a graph, with nodes representing *road intersections* (given as 2D locations) and edges representing *road patches* (given as 2D polylines), but without 3D elevations (Fig. 3). Since later stages of our processing pipeline require 3D locations for points on street maps, our goal in this step is to estimate elevations for every 2D point in the map by projection onto the 3D point cloud.

While simply projecting 2D map points onto 3D LIDAR points may seem simple at first glance, it is difficult in cases where the LIDAR points are poorly sampled, where roads cross over one another (e.g., at bridges and tunnels), where

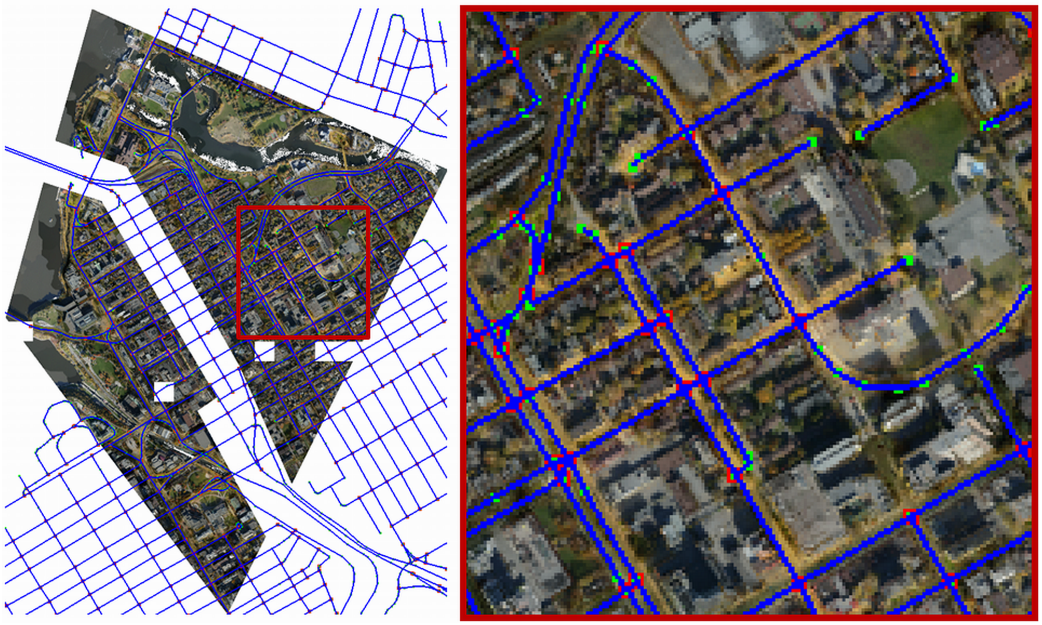


Figure 3: Open Street Map data provided as input to our method (OpenStreetMap, 2010). Zoomed view of area highlighted with red rectangle appears on right. Red points are intersections. Green points are polyline vertices. Blue lines are roads.

there are obstacles over the road (e.g., trees and cars), and/or where the map is inaccurate (e.g., goes outside the road boundary). In these cases, a naive projection onto the closest LIDAR points will yield road maps with sharp elevation discontinuities. Instead, we need a globally consistent and smooth set of road elevations throughout the entire city.

Our approach is to optimize elevations of Cardinal spline control vertices arranged in a network with 2D locations and topology of the road network given in the input map. We place a spline control vertex at the 2D position of every intersection in the map and at 2D positions sampled regularly at 15 meter intervals along every polyline of the connecting intersections in the input map (Fig. 4). We then solve for the elevations (Z coordinates) for these control vertices V to minimize an error function $E(V)$ providing the weighted sum of squared distances between the elevations of points s on the spline curve S and nearby LIDAR points $N(s)$:

$$E(V) = \sum_{s \in S(V)} (w(s) \sum_{p \in N(s)} (s_z - p_z)^2)$$

where V is the set of control vertices, s is a set of points sampled at 1 meter intervals along the Cardinal spline S defined by V, s_z is the elevation of s, p is a LIDAR point in the set of points $N(s)$ within 15 cm of s in 2D, $w(s)$ is a weight computed as the inverse of the variance of the elevations within $N(s)$, and p_z is the elevation of p (Fig. 4).

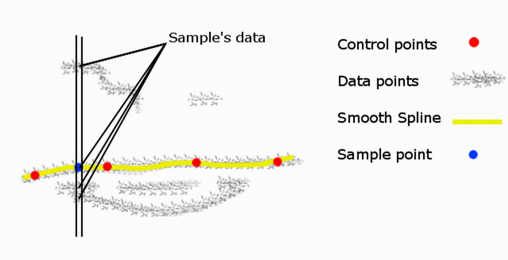


Figure 4: Projecting a 2D map onto the 3D point cloud.

This approach fits a smooth spline curve to the LIDAR data, weighted to favor points with small elevation variance, and thus draws the curve to 3D LIDAR points that are most clearly on the road. The resulting elevations are guaranteed to be smooth, since the spline curve is guaranteed to be C^1 continuous at intersections and C^2 continuous everywhere else, with the expected curvature determined by the spacing of control vertices (15 meters). Since the control network is con-

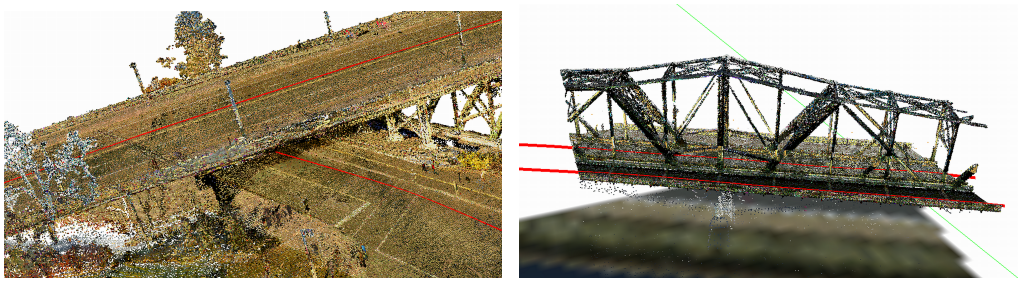
nected throughout the entire city (control vertices at intersections are shared and therefore connect splines representing adjacent roads), and the elevations for all roads are solved simultaneously, elevations are predicted correctly even in areas of occlusions and/or missing data, as large-scale smoothness constraints guide the solution to correct results in regions where data is locally imperfect.

After solving for the control vertices V, we snap the elevation for each sample point s to that of the LIDAR point closest to the spline curve position at s. Finally, to correct minor discontinuities introduced in this step, we finish by visiting each sample point s in order and assigning its elevation to be that of the preceding sample if its incline exceeds a conservative threshold (35% - the incline of the world's steepest road (NZ, 2005)).

In the end, we have a set of connected polylines with 3D points lying on the road surface. Fig. 5 shows some results. To evaluate the quality of results of this step, we look at how often a smooth projection onto the point cloud was not possible, and how much of an incline anomaly throughout the city these failure cases produced. To address the first question, we compare how many map samples could not be smoothly projected onto the point cloud within the maximum incline window. For a manually created map of Ottawa (described in Section 3.4), there are only 11 samples that do not fit the Ottawa point cloud smoothly, out of over 4×10^4 sample points in the input map. To evaluate the smoothness of the final 3D map spline, we look at the average and largest absolute incline of the resulting map splines. Average inclines are very close to zero, which shows that overall fitting of the map to the point cloud of a rather flat region is successful. The maximum incline is 0.4, which means that among the 11 samples that do not fit smoothly, the biggest jump is 40 cm vertically per 1 meter on a plane. We believe that this minor and infrequent issue can be explained by the map passing through an object on the road, e.g. a car. Visual observations also indicate good quality of fitting, e.g. Fig 5, where map splines fit to correct road surfaces of the overpass, and to both roadways of a dual roadway elevated bridge, with one roadway densely covered from above.

2.2. Point Cloud Partitioning

The second step is to partition the point cloud into "subclouds" that contain parts of the input data that can be processed independently. This step is necessary because the entire point cloud is far larger than fits into memory on most computers, and thus must be partitioned into small "working sets" that can be processed efficiently.



(a) Overpassing roads. (b) Bridge map fitting.

Figure 5: Map fitting and extraction results. Map splines projected onto the cloud are shown as red lines. These images illustrate a correct fitting of the map spline to the point cloud in a case of roads crossing in 2D but not in 3D, and on a partially covered elevated dual roadway bridge.

There are many ways in which a point cloud can be partitioned into working sets. For example, previous works break point clouds into spatial tiles arranged in a regular grid and process each tile independently. However, that approach requires processing challenging road topologies within each tile (since a tile can contain many disconnected parts of the road network), and it can introduce noticeable seams at tile boundaries if tiles are processed independently. Instead, we aim for a partitioning that yields simple road topologies within each working set and whose results can be stitched together with few seams.

Our approach is to partition the point cloud into independent working sets based on the topology and geometry of the input map. Specifically, for spline curve S representing a road patch connecting two intersections of the given road network, we extract a *subcloud* for S with the following steps:

1. For each sample s in S (spaced at 1 meter intervals), we estimate a local support plane $P(s)$ by fitting a plane to LIDAR points within a 4 meter radius of s.
2. For each spline sample s, we extract a working set of points $WS(s)$ from the LIDAR point cloud that lie within a distance of 0.5 meters from the support plane $P(s)$ and within a 22 meter radius of s (where 22 meters is 6 times the standard lane width in Canada: 3.66 meters (RTAC, 1986)).
3. We merge working sets of all the spline samples into one set of LIDAR points to form a subcloud for S.

This approach was chosen for several reasons. First, roads exhibit rapid changes in their structure (width, curvature) at intersections, and thus intersections are

natural places to partition road networks into independent patches with large coherence within each patch and manageable seams at patch boundaries. Second, LIDAR points on roads that cross over one another (bridges, tunnels, etc.) are separated by this process into separate subclouds, preventing them from interfering with each other. Finally, the subcloud for any given road patch is generally small, since it contains points only near the road elevation within a small radius of the road patch centerline (empirically, the size of each road patch is on the order of $\times 10^3$ to $\times 10^5$ times smaller than the size of the entire point cloud), and thus it provides an appropriate granularity for memory management during road extraction.

Our method for finding points in the subcloud is accelerated by an out-of-core spatial indexing structure that stores points on disk in contiguous blocks based on cells of a regular 2D grid. This structure enables rapid retrieval of points within a given 2D region of the point cloud. Even with this indexing, for the Ottawa data set, the subcloud extraction step takes several hours, most of which is spent reading data from disk.

An example result of the subcloud extraction process is shown in Fig. 6. Points within the subcloud for one road patch are shown in white and blue, while others are shown in black – note that obstacles sitting above the road are excluded from the subcloud.

2.3. Attractor Map Construction

The third step is to analyze the LIDAR points within the subcloud for each road patch and construct an *attractor map*, $a(x, y)$, to be used for fitting an active contour surrounding the road area within the patch in the next step. The ideal attractor map has a large value at road boundaries and small values everywhere else, so that the active contour will snap to road boundaries precisely.

In urban environments, roads usually have constraints on their widths and have boundaries recognizable by elevation discontinuities (e.g., curbs). Thus, our attractor map has two terms, an extent constraint, $e(x, y)$, and a curb detector, $c(x, y)$:

$$a(x, y) = \alpha_{attr} \cdot e(x, y) + \beta_{attr} \cdot c(x, y)$$

The extent constraint, $e(x, y)$, penalizes road boundaries outside the expected range of distances from the input map spline. In our current implementation, we set $e(x, y)$ to zero if (x, y) is less than one lane width (3.66 meters in Canada (RTAC, 1986)) or greater than 6 lane widths from the closest point on the map road spline, and 1 otherwise. This term will penalize road predictions that are too thin or wide.

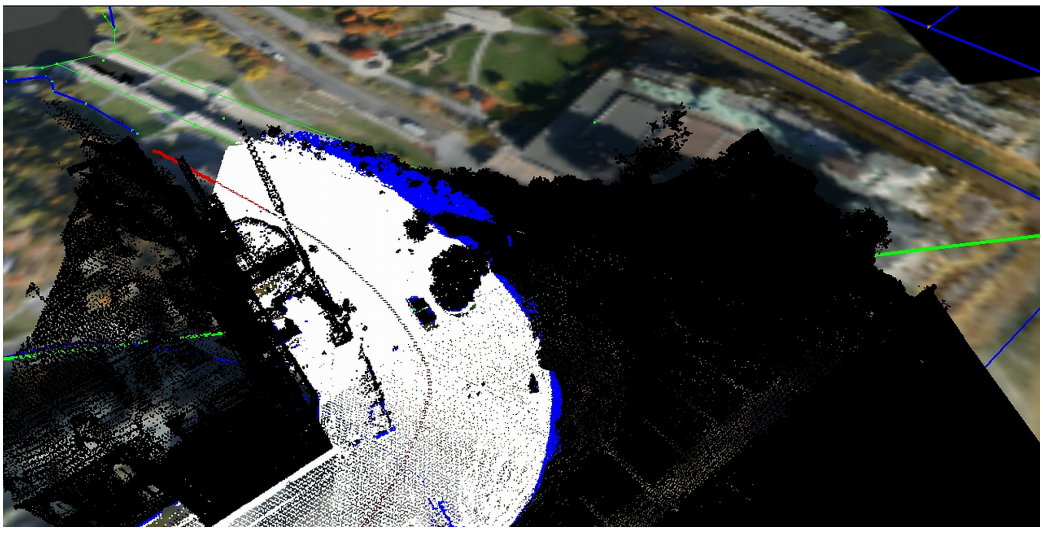


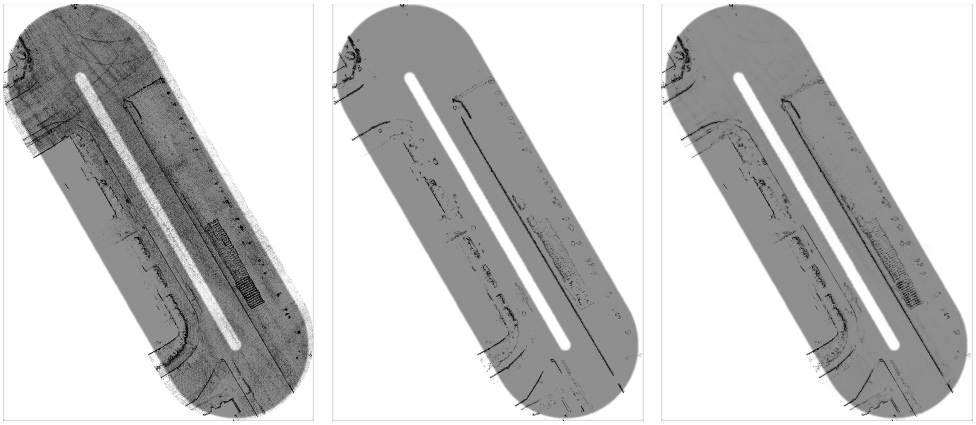
Figure 6: Subcloud extracted for a road patch. The patch centerline is shown as a red line. Points belonging to the subcloud are shown in white and blue, other points are shown in black for contrast. Low resolution 2D projection of the cloud in original color is provided for context.

The curb detector, $c(x, y)$, provides an estimate for probability that a curb can be found at (x, y). To compute it, we compute a curb detector $c(p)$ at every point p in the subcloud and then aggregate them by averaging within regularly sampled 2D grid cells. Specifically, for every point p in the road patch subcloud, we estimate a value $c(p)$ by analyzing the normal direction and elevation variation within a local neighborhood $N(p)$ of p as follows:

$$c(p) = f(\Delta h) \cdot (1 - (\vec{n}_p \cdot \vec{n}_s^2) \tag{1}$$

where Δh is a function of the local elevation variance of points in $N(p)$, $\vec{n}_p$ is a normal estimated by at the point p based on principal component analysis of $N(p)$, $\vec{n}_s$ is a normal of the support plane estimated at the nearest road patch spline point, and $N(p)$ is a set of at least 20 points closest to p.

In this work, we consider three possible definitions of $f(\Delta h)$. The first, f_n, is simply a constant function (elevation variation is not considered in the attractor map). The second, f_{nStep}, is a step function, with value 1 if the elevation range lies within preset bounds determined by road construction standards (4 to 8 inches in (Seattle, 2010)). The third, f_{nExp}, is a continuous function that is largest when $\Delta h = \Delta h_{avg}$ and smaller elsewhere, with $\sigma_{\Delta h_{max}}$ defining how narrow the



(a) a_n (b) a_{nStep} (c) a_{nExp}

Figure 7: Visualization of attractor maps for one example road segment. Darker pixels indicate larger values. The non-white region represents area where $e(x, y) \neq 0$.

window for the elevation variation should be before the $c(p)$ becomes negligibly small. Following the same road construction standards, we use a 15 cm average and a 5 cm deviation to ensure that the standard-compliant curbs fall within the one standard deviation.

$$f_n(\Delta h) = 1 \tag{2}$$

$$f_{nStep}(\Delta h) = \begin{cases} 1, \Delta h_{min} \leq \Delta h \leq \Delta h_{max} \\ 0, otherwise \end{cases} \tag{3}$$

$$f_{nExp}(\Delta h) = exp\left(-\frac{(\Delta h - \Delta h_{avg})^2}{2\sigma^2_{\Delta h_{max}}}\right) \tag{4}$$

Figure 7 shows visualizations of the attractor maps computed with each of these three $f(\Delta h)$ functions (a_n uses f_n, and so on). a_n identifies curb lines on the map. However, it fires on any remotely vertical group of points and thus generates a lot of noise in the attractor map. The a_{nStep} measure filters out much of the noise, mostly firing on curbs and similar small objects. However, it uses a step threshold and any road delimiter that is too small or too tall disappears from the map. This results in increased gaps in the curb line. a_{nExp} offers a trade-off between the former two by still capturing but downweighting deviations in the curb's height.

2.4. Road Boundary Estimation

The third step is to estimate the boundaries of the road from the 3D curve representing the road patch centerline (Sec. 2.1) and the attractor map representing the expected locations of curbs (Sec. 2.3).

This problem is challenging for several reasons. First, the point cloud may have noisy and missing data (e.g., due to occlusions). Second, the given road patch "centerline" may not be located at or even near the center of the road (e.g., due to inaccuracies in the input map). Third, there can be numerous obstacles (e.g., cars) and other urban features (e.g., barriers) on the road that could provide high values in the attractor map and thus be confused with road boundaries. Finally, there are many places where road boundaries are not distinguishable in the attractor map (e.g., at driveways or when the shoulder is gravel), which makes finding the road boundary difficult without simultaneous analysis of large regions. As a result, simple region growing algorithms that threshold the attractor map and "flood fill" will produce erroneous results in all but the simplest cases.

Our approach is to extract road boundaries by optimization of a "ribbon snake" (Fua, 1997), a parametric contour, $\mathbf{v}(s) = (\mathbf{x}(s), w(s))$, described by the position of its centerline $\mathbf{x}(s)$ and its width $w(s)$ as a function of arc length s. The ribbon snake is fit to a potential by optimizing its centerline position and width to minimize an energy functional with internal smoothness and external data approximation terms.

This ribbon snake representation was chosen because it provides a parameterization of the road boundary that allows efficient formulation of an error functional measuring how well the predicted road boundary fits to the given input data. Specifically, our error functional, E_{snake}, includes an internal term favoring smoothness of the road boundary and an external term favoring alignment of the road boundary with high values in the attractor map:

$$E_{snake} = \int_0^1 \kappa_{snake} E_{internal}(\mathbf{v}(s)) + \gamma_{snake} E_{external}(\mathbf{v}(s)) ds \tag{5}$$

The internal smoothness energy term, $E_{internal}$, measures how similar the shape of the ribbon centerline is to the input road patch centerline. To facilitate computation of this term, the ribbon centerline is represented based on $t(s)$, a function of arc-length s representing the distance from ribbon snake centerline to the input road patch curve in the direction locally perpendicular to the input road patch curve ($\vec{n}_t(s)$). This formulation enables us to enforce the snake's internal energy to depend on the initial road patch curve rather than penalizing

the initial curvedness, if any. It also facilitates modeling global shifts of the ribbon centerline, which is important because the input road patch centerline might be significantly off-center (due to inaccuracies in the input map). To enable the snake to translate globally without penalty we introduce t_{avg} that we compute as an average of $t(s)$ over the whole snake, and we actually represent the snake as $\mathbf{v}(s) = (u(s), w(s))$, where $u(s) = (t(s) - t_{avg})$. We compute the internal energy term as:

$$E_{internal}(\mathbf{v}(s)) = (\alpha_{snake}|\mathbf{v}_s(s)|^2 + \beta_{snake}|\mathbf{v}_{ss}(s)|^2)/2 \tag{6}$$

The external data approximation energy term, $E_{external}$, measures how well the ribbon snake boundary aligns with the attractor map:

$$E_{external}(\mathbf{v}(s)) = -a(x(\mathbf{v}(s)), y(\mathbf{v}(s)))$$

In order to enforce the smoothness of the road's width and center, without encouraging shrinkage of its area, we set the rigidity parameter of the snake (β_{snake}) well above other parameters, and keep the elasticity parameter (α_{snake}) very small. Parameters γ_{snake} and κ_{snake}, which control the trade-off between smoothness and approximation terms of the snake's energy function, were chosen experimentally ($\gamma_{snake} = 10$ and $\kappa_{snake} = 1$).

As in the original snake paper (Kass et al., 1988), we optimize the ribbon snake by iteratively refining components of $\mathbf{v}(s)$ using the following matrix equation:

$$\mathbf{v}_t = (A + \gamma_{snake}\mathbf{I})^{-1}(\gamma\mathbf{v}_{t-1}(s) - \mathbf{f}_{t-1}(s))$$

where, t and $(t-1)$ indices refer to respective values at current and previous iterations of the solution, $A = \alpha_{snake}\mathbf{v}_{ss}(s) + \beta_{snake}\mathbf{v}_{ssss}(s)$ is a diagonal matrix representation of the derivative of the $E_{internal}(\mathbf{v}_{t-1}(s))$, and $\mathbf{f}(s)$ is derived from $E_{external}$. Namely, for our snake representation $\mathbf{f}(s) = ((\vec{f} \cdot \vec{n}_t(s))_l + (\vec{f} \cdot \vec{n}_t(s))_r, (\vec{f} \cdot \vec{n}_t(s))_l - (\vec{f} \cdot \vec{n}_t(s))_r)$, where $\vec{f} = \vec{f}(x, y)$ is a "force" vector that attracts the snake towards the minima of $E_{external}$ taken at left and right points of the ribbon snake (indices l and r, respectively). To compute $\vec{f}(x, y)$ and avoid the necessity of initializing the snake near the curb lines, similarly to (Marikhu et al., 2007), we use an improved approach proposed in (Xu and Prince, 1997) where the authors introduced the concept of the gradient vector flow (GVF) to enable attractors to propagate their influence further into the image and increase the capture range of an active contour. To do this $\vec{f}(x, y)$ is represented as $(u(x, y), v(x, y))$ and its components are computed by minimizing the following error function:

$$E_{GVF} = \int\int \mu(u_x^2 + u_y^2 + v_x^2 + v_y^2) + |\nabla P|^2|\mathbf{f} - \nabla P|^2 dxdy \tag{7}$$

where $P(x, y) = E_{external}(x, y)$. Computation of GVF for each attractor map needs to be done only once - before fitting the snake.

In Fig. 8 we illustrate the process on an example of the road patch located in the point cloud shown in original color in Fig. 8a and by elevation encoded in color in Fig. 8b. Fig. 8c shows the attractor map $a(x, y)$ built from the subcloud of the given road patch. The GVF field obtained from $a(x, y)$ is shown in Fig. 8d with the direction of $\vec{f}(x, y)$ at every point encoded by hue: red meaning $\vec{f}(x, y)$ pointing right, blue - down, cyan - left, and green - up. Fig. 8e shows the initial position of the snake along the map spline position for the road patch. Fig. 8f shows the final result of fitting the snake, which smoothly and accurately fits the curb lines where they exist with only occasional and very small smooth protrusions outside where there is no curb.

2.5. Semantic Tagging

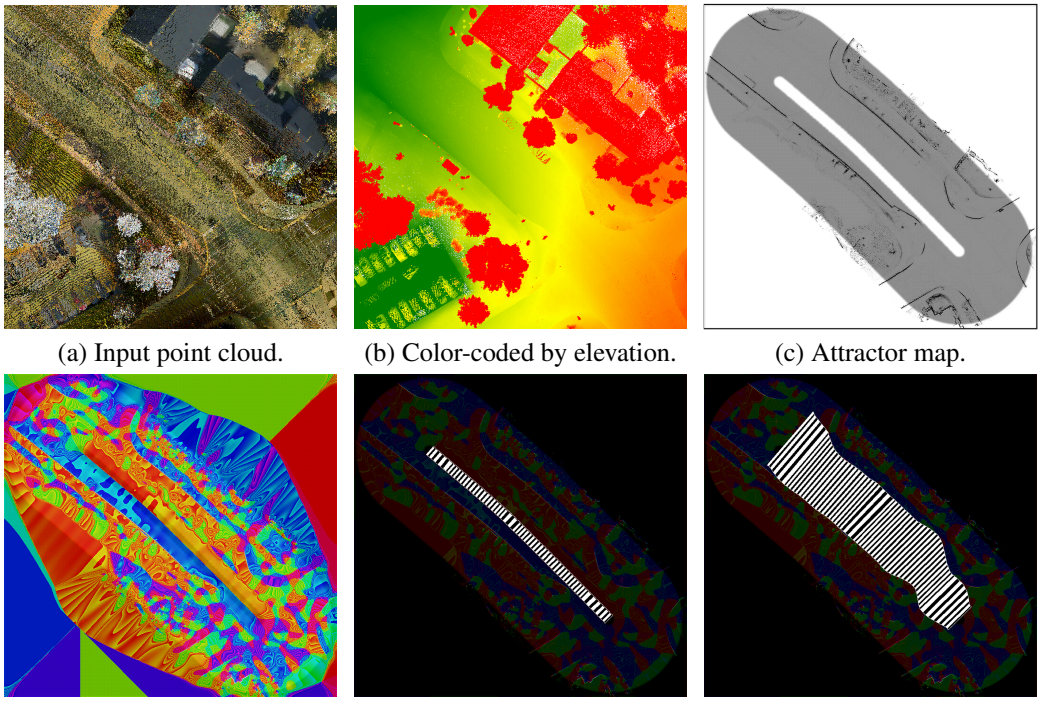
The final step of our pipeline is to produce a set of points to be tagged as being on a road. This step is quite easy once the ribbon snake for every road patch has been computed. For each road patch, we simply identify the set of points within the patch's subcloud whose projection onto the XY plane falls within the region delimited by the ribbon snake for that patch. We then unite the sets of points found for all road patches to produce the final output.

Since road patches are processed independently, this process could potentially miss points and/or introduce discontinuities at road intersections. However, in practice, since patch subclouds extend into the middle of intersections, the points tagged as road for adjacent patches almost always overlap, and so points are rarely missed. Also, since patches overlap at intersections, the precise placement of the road boundary for each patch at each intersection is not as important as the boundary of their union, and thus the final output almost always provides continuous road boundaries, even at intersections.

3. Results and Discussion

In order to test how well the proposed method detects roads in a large LIDAR point cloud, we ran a series of experiments with the methods described in the previous sections on the Ottawa data set described in the introduction. Again, this data set comprises 951 million LIDAR points covering several hundred road patches in a downtown area.

In addition to describing accuracy and timing results for our method, we report the results of experiments with different input data (road maps) and different



(a) Input point cloud. (b) Color-coded by elevation. (c) Attractor map.

(d) Hue-coded GVF map. (e) Snake initialization. (f) Final snake.

Figure 8: Snake fitting of a single road patch. (a) - road patch (view from above) in the original color, (b) - same view with points color-coded by their elevations: from green (smallest) to red (highest), (c) - attractor map formed from the subcloud around the road patch (darker regions are stronger attractors), (d) - GVF map with the vector direction encoded with the hue, (e) - snake is initialized around the map spline with a minimal width of 1 lane, (f) - final result of fitting the snake to the attractor map.

Input road map	Manual
Attractor map	a_{nExp}
3D spline sampling rate	every 1 m
Control vertex sampling rate	every 15 m
Fitting data support cylinder radius	15 cm
Maximum road incline window	$\pm 35\%$
Maximum expected road width	10 lanes
Minimum expected lane width	1 lane
Subcloud extraction vertical window	1 m
Subcloud extraction support radius	22 m
α_{attr}	0.2
β_{attr}	1
Δh_{avg}	15 cm
$\sigma_{\Delta h_{max}}$	5 cm
Δh_{min}	10 cm
Δh_{max}	20 cm
α_{snake}	0.01
β_{snake}	100
γ_{snake}	10
κ_{snake}	1

Table 1: Summary of the parameters and their values used in our experiments.

algorithmic design decisions (curb detectors). These experiments provide an indication of the sensitivity of our method to the varying parameters. Unless otherwise specified, parameters were set to the values shown in Tab. 1 chosen either to match standards of road construction or empirically.

On average, each experiment (beginning to end computation for the entire point cloud) took approximately 24 hours using a cluster machine of 20 2.2GHz Opteron Dual-Core processors with 8GB of memory and 11 2.3GHz Opteron Quad-Core processors with 16GB of memory.

3.1. Ground Truth and Evaluation Metrics

To evaluate the accuracy of road detection results in each experiment, we compare them to a manually created *ground truth* representation of the roads, gt. This ground truth is represented as a 2D grid specifies for every 0.5 x 0.5 meter cell a value of "1" if any road in the city overlaps the cell (shown in green and red in

Fig. 9), and "0" otherwise. It was created by manual marking of an image overlaid on the point cloud using an interactive tool.

To evaluate how well our road prediction matches this ground truth data set, we produce a *predicted grid*, $pred$, by projecting all LIDAR points in the road prediction output into a 2D grid and then setting to "1" the value of every grid cell containing at least one point from the predicted set and all other grid cells to "0." Then, we compute the following accuracy metrics by comparing the two grids:

- *Correctness (precision)*: $p = TP/(TP + FP)$
- *Completeness (recall)*: $r = TP/(TP + FN)$
- *Quality*: $q = TP/(TP + FP + FN)$
- *Average spill size*: $s = \sum_{\|roadside_{pred}\|} d(roadside_{gt}, roadside_{pred})/\|roadside_{gt}\|$
- *Prevailing spill direction*: $d = (FP - FN)/(FP + FN)$

where TP is the number of true positives (gt(x,y)=pred(x,y)=1), TN is the number of true negatives (gt(x,y)=pred(x,y)=0), FP is the number of false positives (gt(x,y)=0, pred(x,y)=1), FN is the number of false negatives (gt(x,y)=1, pred(x,y)=0), and $roadside_{gt}$ and $roadside_{pred}$ are pixels in the ground truth and prediction, respectively, that are on the boundary of the road areas. The nominator of s is a sum of unsigned distances from each pixel in $roadside_{pred}$ to the closest pixel in $roadside_{gt}$, and the denominator is the size of the $roadside_{gt}$.

The first three metrics (p, r, and q) are standard for evaluation of the road extraction algorithms (e.g., Clode et al., 2007; Harvey, 1999; Heipke et al., 1997; Mayer et al., 2006; Wiedemann, 2003). The latter two are new in this paper. First, s, is the average distance of the roadside in the predicted extraction from the roadside in the ground truth, indicating how far, on average, from the actual curb line the predicted road area ended. Simply put, it describes how deep of a spill we can expect per 1 meter of the reference road, allowing us to make conclusions about the shape of the spills: given same TP, FP, FN, and TN whether the spills are long and shallow or deep and short. Second, d, shows which direction of the spills is prevailing - inside the ground truth area or outside. The direction of spills is indicative of the susceptibility of the method to terminating the road growth too early or too late, depending on the ability of our method to accurately locate the curbs. If d is negative, snakes are prone to early growth termination due to spurious snake attractors, and vice versa - when d is positive, curb detector

does not create strong enough attractors to hold the snakes from growing outside the curb line.

These evaluation methods leverage marked 2D grids rather than marked 3D points for reasons of practicality (manually labeling 3D points is much more difficult). Although they do not precisely characterize predictions of 3D structures (bridges, tunnels, and overpasses) and they introduce discretization errors (due to the grid), we find that they provide a good overall evaluation of the methods and are quite representative of results observed qualitatively through visual observation.

3.2. Results

Evaluation of the results of our method on the Ottawa LIDAR point cloud are presented visually in Fig. 9. The color coding depicts true positives in green, false negatives in red, false positives in magenta, and true negatives in cyan. Most of the area is cyan and green indicating that correct classification prevails. Detailed observation shows that our method succeeds in most cases, making minor errors in cases where curbs are not well-defined. In particular, it succeeds in many difficult cases where roads have gaps in the curb line. For example, Fig. 9c shows a very complicated intersection, where all 15 corners are level with the ground for crosswalks. It also succeeds in cases where roads are part of a multilevel environment. For example, Fig. 9b shows a properly extracted open single-roadway bridge, and Fig. 9a shows a long tunnel underneath a building extracted from the area shown in Fig. 1. These examples would require special treatment using previous methods.

While our algorithm performs well for most roads throughout the city, there are cases where it fails. In particular, areas where several snakes meet are often jagged and abrupt because there is no inter-snake interaction in our model. Although snakes that meet at an intersection cover most of it, each of their area of interest ends midway into the intersection, where the lack of curbs enables snakes to behave in a manner inconsistent with one another. Another source of error with respect to the reference is spilling through gaps in curbs. Although we demand high rigidity from the snake, if we have a long interruption in the curb line, a nearby curb of another road can provide enough attraction for a snake to smoothly bend outside the road area or even to spill completely over a thin roadway separator. Unless there is a much longer consistent curb line around the gap to support the line, these spills can become large, in extreme cases of closely located gaps, even allowing the snake contour to bypass entire curb segments between the gaps. An example of poor extraction with these cases is in Fig. 9d. In Fig. 9e you can

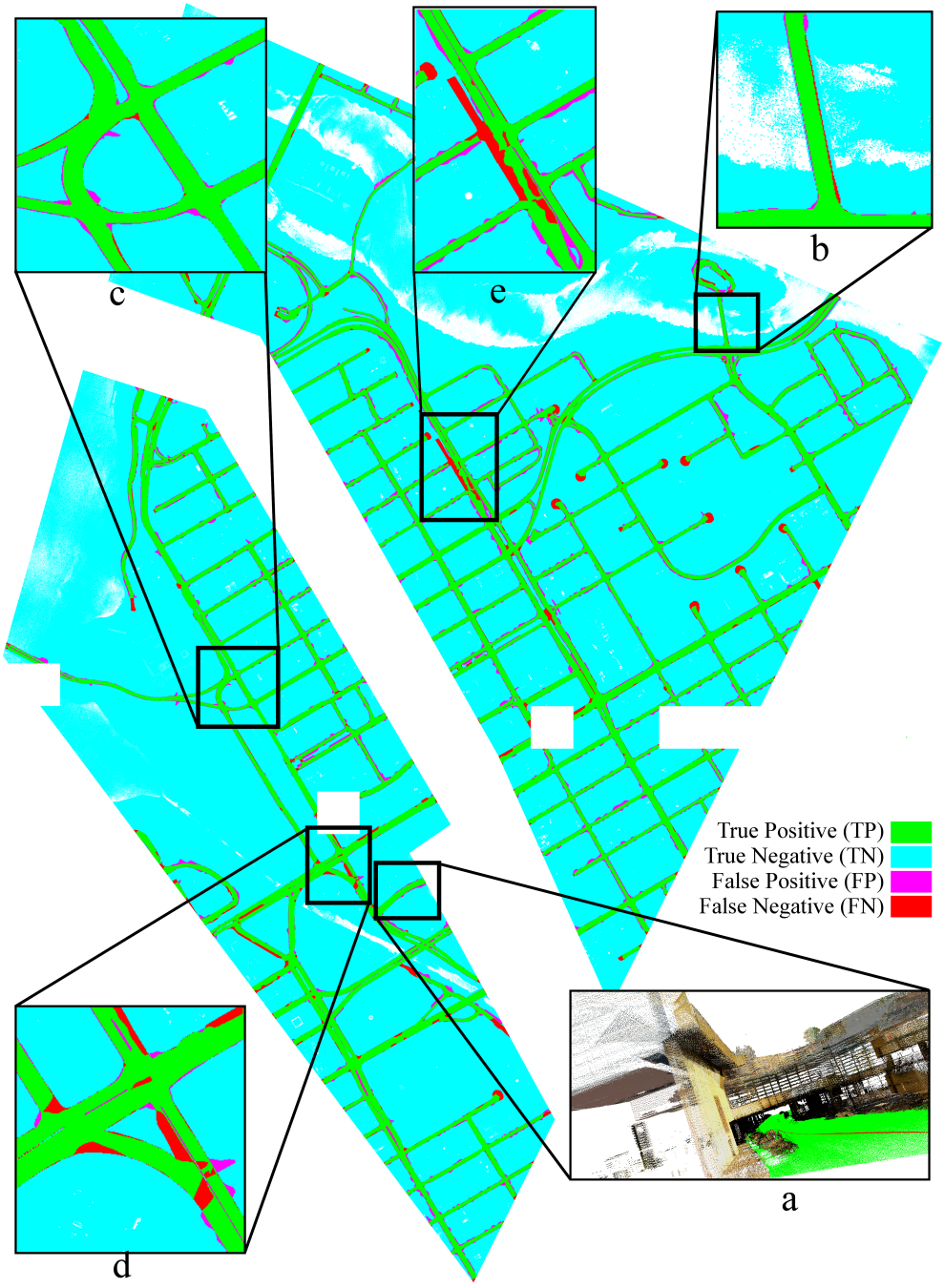


Figure 9: Classification error map.

see another failure case - here the road is being reconstructed and moved to follow an alternate geometry; hence, the features of the road are poorly defined and inconsistent throughout this area causing an incorrect solution.

Overall, our algorithm achieves 86.3% precision, 94.0% recall, 81.8% quality, 2.02 average spill size, and 42.2 prevailing spill direction for the Ottawa data set. We believe that this performance is suitable for many applications that leverage detected roads as contextual cues for object recognition and urban planning.

3.3. Attractor Map Comparison

In this subsection, we study how the construction of the attractor map affects the overall road detection results. In this study, we ran three experiments, keeping all parameters the same (as in Tab. 1) except the attractor map, for which we tested a_n (Eq. 2), a_{nStep} (Eq. 3), and a_{nExp} (Eq. 4).

A comparison of the results is shown in Tab. 2. a_n provides the highest precision, yet the lowest recall and quality, and negative d, suggesting that snake fitting often terminates before reaching the actual curb. The average spill size for a_n is also the largest. The combination of a small negative d, lowest quality, and a large s describe the predominantly inward deep spills. a_{nStep} has the opposite features - lowest precision, highest recall, high positive spill direction. This proves the qualitative observation that a_{nStep} filters out too much data, providing large windows for the snake to leak outside of the road's area. Smaller s with a much larger absolute value of d suggests that the spills are more undulate. This also makes sense from the perspective of the snake formalism: smaller values on the attractor map create weaker input from the approximation term of the snake's energy and the smoothness enforcement dominates. When using a_{nExp}, we can see that although its precision and recall are marginally smaller than the others, it has the highest quality, suggesting that it finds the best balance between early and late terminations. Positive d suggests that the spills are predominantly outward, and the smallest of three s together with the highest quality shows that the spills are shallow and the predicted curb lines in this case are closer to the ground truth. These observations suggest that a_{nExp} is the most robust curb detector for this data set.

3.4. Input Map Comparison

In this last subsection, we study how the choice of input map affects the road detection results. In this study, we ran two experiments, each using one of the following two road maps as input:

Metric	a_n	a_{nStep}	a_{nExp}
p (%)	**88.7**	83.2	86.3
r (%)	86.2	**95.7**	94.0
q (%)	77.7	80.2	**81.8**
s	2.62	2.26	**2.02**
d (%)	**-11.6**	63.3	42.2

Table 2: Results of using different curb detectors. Best values are given in bold.

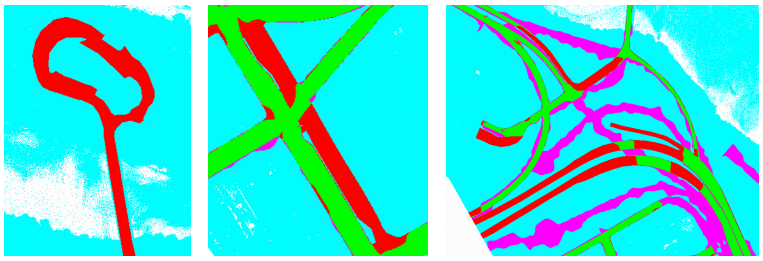
- **Manual:** a map created specifically for the Ottawa point cloud using an interactive tool to select approximate 2D positions of intersections and 2D polylines near the centerlines of road patches. The map yields 383 road patches ranging in length from 5.5 to 627 meters, averaging at 107 meters.
- **OpenStreetMap:** a map downloaded from www.openstreetmap.org. This map is freely available on-line (Fig. 3). This map yields 688 road patches ranging in length from 1.3 to 1563 meters, averaging at 132 meters.

Table 3 shows the evaluation measures for road predictions produced by our method with these two maps. The results achieved with OpenStreetMap are lower than the ones with the Manual map by 9% in correctness, 13% in completeness, 19% in quality, and more than twice as large in average spill size. The reasons for these differences are due mainly to extra, missing, and misplaced roads in the OpenStreetMap data. For example in Fig. 10a, an entire street is marked with red (FN), indicating that the street is not present in the map at all. Also some curb-separated dual roadway roads are only half-extracted due to the map indicating only one of the roadways (Fig. 10b). This results in the drop in completeness. The drop in correctness comes from the parts of the map that do not follow the path of the roads in the point cloud: the road in the subcloud has been extracted regardless and it greatly contributed to both FP and FN areas in the classification error map - see Fig. 10c for an example.

These results indicate that our current implementation is sensitive to the quality of the map used. However, we believe it would be possible to extend our approach to overcome errors in the map by detecting roads automatically and/or optimizing their positions directly from the point cloud. This is a topic for future work.

Metric	Manual Map	OpenStreetMap
p (%)	86.3	78.4
r (%)	94.0	81.3
q (%)	81.8	66.4
s	2.02	4.64
d (%)	42.2	8.2

Table 3: Results with manually created and OpenStreetMap maps.



(a) Road is not in the map at all.

(b) Dual roadway presented as a single line in the map.

(c) Map describes a discrepant road network.

Figure 10: Failure cases when using map from OpenStreetMap. *Green* is TP, *Red* is FN, *Magenta* is FP, *Cyan* is TN, *Black* is no points.

4. Conclusion

We have described an approach for a map guided extraction of roads from a large dense merged point cloud made of real life urban environment scans. By partitioning the problem into patches, using a curb detector to encode elevation discontinuities, and fitting ribbon snakes to model road boundaries, we combine both small-scale (curb detectors) and large-scale (snake smoothness) cues to extract roads more robustly than either alone. Our method is able to deal with all kinds of roads, including bridges, tunnels, and multilevel intersections. It is parallelizable, does not rely on point cloud's color, texture or intensity properties, and demands only for the cloud to be sampled densely enough and the magnitude of noise to be low enough for the curb-like features to be noticeable. Results of experiments on a point cloud with almost a billion points suggest that the system is scalable and provides fairly accurate results for difficult cases encountered in urban environments.

This method has several limitation that suggest for the following directions of future work. Among the main directions we see the need to decrease the sensitivity of our approach to the incorrect map data or to completely extract the road network directly from the point cloud. A set of verification steps will benefit the robustness of our approach, e.g. preventing road extraction at a poorly fitted 3D spline area. Also, striving to achieve scalability by treating each road patch independently, our method does not explicitly deal with road intersections where subclouds overlap. Special treatment is necessary for the intersection areas extraction due to their possible topological complexity, especially in urban environments. This can be achieved in the future by allowing the snakes to interact at the overlapping areas, among the possible solutions we see using ziplock snakes (Neuenschwander et al., 1997) in the overlap areas or explicit modeling of the intersections in the manner it is done in (Mayer et al., 1998, Sec.4.3). Global fitting of a network of snakes to the entire point cloud may yield better results, but may also be extremely computationally expensive. Another future work direction is the cross-data portability of our approach. The applicability of our method to data collected from other cities and countries as well as possibility of extension to rural environments is an interesting direction for future investigation. Finally, the running time to process the entire city can be decreased by optimizing the implementation.

Acknowledgments

This work would not be possible without data, funding, and support from several people. First, we thank Neptec, John Gilmore, and Wright State University for access to the 3D LIDAR data set. Second, we thank Alex Golovinskiy for source code and guidance. We also thank our reviewers for their valuable suggestions. Finally, we acknowledge NSF (IIS-0612231, CCF-0702672, CNS-0831374, and CCF-0937139), Intel, and Google for funding to support this project.

References

Alharthy, A., Bethel, J., 2003. Automated road extraction from lidar data, in: Proc. ASPRS Annual Conference, Anchorage, Alaska. Unpaginated CD-ROM.

Amo, M., Martinez, F., Torre, M., 2006. Road extraction from aerial images using a region competition algorithm. IEEE Transactions on Image Processing 15 (5), 1192 – 1201.

Baumgartner, A., Hinz, S., 2000. Multi-scale road extraction using local and global grouping criteria. International Archives of Photogrammetry and Remote Sensing 33 (Part B3/1), 58–65.

Bicego, M., Dalfini, S., Vernazza, G., Murino, P., 2003. Automatic road extraction from aerial images by probabilistic contour tracking, in: International Conference on Image Processing, pp. 585–588.

Butenuth, M., Straub, B.M., Heipke, C., Willrich, F., 2003. Tree supported road extraction from arial images using global and local context knowledge, in: Proc. 3rd International Conference on Computer Vision Systems, Berlin, Heidelberg. pp. 162–171.

Chen, X., Kohlmeyer, B., Stroila, M., Alwar, N., Wang, R., Bach, J., 2009. Next generation map making: geo-referenced ground-level lidar point clouds for automatic retro-reflective road feature extraction, in: Proc. of the 17th ACM SIGSPATIAL International Conference on Advances in Geographic Information Systems (GIS '09), ACM, New York, NY, USA. pp. 488–491.

Choi, Y.W., Jang, Y.W., Lee, H.J., Cho, G.S., 2007. Heuristic road extraction, in: International Symposium on Information Technology Convergence, IEEE Computer Society, Los Alamitos, CA, USA. pp. 338–342.

Clode, S., Rottensteiner, F., Kootsookos, P., Zelniker, E., 2007. Detection and vectorization of roads from lidar data. Photogrammetric Engineering and Remote Sensing 73(5), 517–536.

Clode, S., Zelniker, E., Kootsookos, P., Clarkson, V., 2004. A phase coded disk approach to thick curvilinear line detection, in: Proc. 17th European Signal Processing Conference, EUSIPCO. pp. 1147–1150.

Ferchichi, S., Wang, S., 2005. Optimization of cluster coverage for road centreline extraction in high resolution satellite images, in: International Conference on Image Processing, pp. 201–204.

Fortier, M.F.A., Ziou, D., Armenakis, C., Wang, S., 1999. Survey of Work on Road Extraction in Aerial and Satellite Images. Technical Report 241. Universite de Sherbrooke, Quebec, Canada.

Fua, P., 1997. Model-based optimization: An approach to fast, accurate, and consistent site modeling from imagery, in: Firschein, O., Strat, T. (Eds.), RADIUS: Image Understanding for Intelligence Imagery, Morgan Kaufmann. pp. 903–908.

Goulette, F., Nashashibi, F., Abuhadrous, I., Ammoun, S., Laurgeau, C., 2006. An integrated on-board laser range sensing system for on-the-way city and road modelling. International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences 34 (Part A), 3–5.

Harvey, W., 1999. Performance evaluation for road extraction, in: Proc. International Workshop on 3D Geospatial Data Production: Meeting Application requirements, ISPRS WG II/6, Paris, France. pp. 175–185.

Hatger, C., Brenner, C., 2003. Extraction of road geometry parameters from laser scanning and existing databases. International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences 34 (Part 3/W13), 225–230.

Heipke, C., Mayer, H., Wiedemann, C., Jamet, O., 1997. Evaluation of automatic road extraction. International Archives of Photogrammetry and Remote Sensing 32 (3-2W3), 47–56.

Heipke, C., Steger, C., Multhammer, R., 1995. A hierarchical approach to automatic road extraction from aerial imagery, in: Integrating Photogrammetric Techniques with Scene Analysis and Machine Vision II, pp. 222–231.

Hinz, S., 2004. Automatic road extraction in urban scenes and beyond. International Archives of Photogrammetry and Remote Sensing 35 (Part B3), 349–354.

Hu, J., Razdan, A., Femiani, J., Wonka, P., Cui, M., 2007. Fourier shape descriptors of pixel footprints for road extraction from satellite images, in: Intenational Conference on Image Processing, pp. 49–52.

Hu, X., Tao, C.V., Hu, Y., 2004. Automatic road extraction from dense urban area by integrated processing of high resolution imagery and lidar data. International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences 35 (Part B3), 288–292.

Jaakkola, A., Hyyppa, J., Hyyppa, H., Kukko, A., 2008. Retrieval algorithms for road surface modelling using laser-based mobile mapping. Sensors 8, 5238–5249.

Kass, M., Witkin, A., Terzopoulos, D., 1988. Snakes: Active contour models. International Journal of Computer Vision 1(4), 321–331.

Laptev, I., Mayer, H., Lindeberg, T., Eckstein, W., Steger, C., Baumgartner, A., 2000. Automatic extraction of roads from aerial images based on scale space and snakes. Machine Vision and Applications 12 (1), 23–31.

Li, J., Lee, H.J., Cho, G.S., 2008. Parallel algorithm for road points extraction from massive lidar data, in: Proc. IEEE International Symposium on Parallel and Distributed Processing with Applications, IEEE Computer Society, Washington, DC, USA. pp. 308–315.

Marikhu, R., Dailey, M.N., Makhanov, S., Honda, K., 2007. A family of quadratic snakes for road extraction, in: Proc. 8th Asian conference on Computer vision, Springer-Verlag, Berlin, Heidelberg. pp. 85–94.

Mayer, H., Hinz, S., Bacher, U., Baltsavias, E., 2006. A test of automatic road extraction approaches. The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences 36 (Part 3), 209–214.

Mayer, H., Laptev, I., Baumgartner, A., 1998. Multi-scale and snakes for automatic road extraction, in: Proc. 5th European Conference on Computer Vision, Freiburg, Germany. pp. 720–733.

Mena, J., Malpica, J., 2005. An automatic method for road extraction in rural and semi-urban areas starting from high resolution satellite imagery. Pattern Recognition Letters 26 (9), 1201–1220.

Mena, J.B., 2003. State of the art on automatic road extraction for gis update: a novel classification. Pattern Recognition Letters 24 (16), 3037–3058.

Neptec, 2009. Wright state 100. www.daytaohio.com/Wright\%5FState100.php. (Accessed October, 2010).

Neuenschwander, W., Fua, P., Iverson, L., Szkely, G., Kbler, O., 1997. Ziplock snakes. International Journal of Computer Vision 25(3), 191–201.

NZ, 2005. Baldwin street. New Zealand on the Web. www.dunedin.nz.com/baldwin-street.aspx. (Accessed June, 2011).

OpenStreetMap, 2010. Open street map - the free wiki world map. www.openstreetmap.org. (Accessed October, 2010).

Oude Elberink, S., Vosselman, G., 2006. Adding the third dimension to a topographic database using airborne laser scanner data. International Archives of Photogrammetry, Remote Sensing and Spatial Information Sciences 36 (Part 3), 92–97.

Peng, T., Jermyn, I.H., Prinet, V., Zerubia, J., 2008. An extended phase field higher-order active contour model for networks and its application to road network extraction from vhr satellite images, in: Proc. 10th European Conference on Computer Vision, Springer-Verlag, Berlin, Heidelberg. pp. 509–520.

RTAC, 1986. Manual of geometric design standards for canadian roads. www.worldcat.org/isbn/0919098479. (Accessed October, 2010).

Seattle, 2010. Seattle right-of-way improvement manual. www.cityofseattle.net/transportation/rowmanual/. (Accessed October, 2010).

Steger, C., 1998. An unbiased detector of curvilinear structures. IEEE Transactions on Pattern Analysis and Machine Intelligence 20 (2), 113–125.

Tesser, H., Pavlidis, T., 2000. Roadfinder front end: an automated road extraction system, in: Proc. 15th International Conference on Pattern Recognition, Barcelona , Spain. pp. 338–341.

Wan, Y., Shen, S., Song, Y., Liu, S., 2007. A road extraction approach based on fuzzy logic for high-resolution multispectral data, in: Fourth International Conference on Fuzzy Systems and Knowledge Discovery, IEEE Computer Society, Los Alamitos, CA, USA. pp. 203–207.

Wiedemann, C., 2003. External evaluation of road networks. International Archives Of Photogrammetry, Remote Sensing And Spatial Information Sciences 34 (Part 3/W8), 93–98.

Xu, C., Prince, J., 1997. Gradient vector flow: A new external force for snakes, in: Computer Vision and Pattern Recognition, San Juan , Puerto Rico. pp. 66–71.

Yager, N., Sowmya, A., 2003. Support vector machines for road extraction from remotely sensed images, in: Petkov, N., Westenberg, M. (Eds.), Computer Analysis of Images and Patterns. Springer Berlin / Heidelberg. volume 2756 of *Lecture Notes in Computer Science*, pp. 285–292.

Youn, J., Bethel, J.S., 2004. Adaptive snakes for urban road extraction. International Archives of Photogrammetry and Remote Sensing 35 (Part B3), 465–470.

Yu, S., Sukumar, S.R., Koschan, A.F., Abidi, M.A., 2007. 3d reconstruction of road surfaces using an integrated multi-sensory. Optics and Lasers in Engineering 45 (7), 808–818.