Building Accurate 3D Spatial Networks to Enable Next Generation Intelligent Transportation Systems

Manohar Kaul
Aarhus University
mkaul@cs.au.dk

Bin Yang
Aarhus University
byang@cs.au.dk

Christian S. Jensen
Aarhus University
csj@cs.au.dk

***Abstract*—The use of accurate 3D spatial network models can enable substantial improvements in vehicle routing. Notably, such models enable eco-routing, which reduces the environmental impact of transportation. We propose a novel filtering and lifting framework that augments a standard 2D spatial network model with elevation information extracted from massive aerial laser scan data and thus yields an accurate 3D model. We present a filtering technique that is capable of pruning irrelevant laser scan points in a single pass, but assumes that the 2D network fits in internal memory and that the points are appropriately sorted. We also provide an external-memory filtering technique that makes no such assumptions. During lifting, a triangulated irregular network (TIN) surface is constructed from the remaining points. The 2D network is projected onto the TIN, and a 3D network is constructed by means of interpolation. We report on a large-scale empirical study that offers insight into the accuracy, efficiency, and scalability properties of the framework.**

I. Introduction

While today's vehicle routing services rely on 2D spatial networks, future generations of such systems and *Advanced Driver Assistance Systems* (ADAS) require 3D models that accurately capture elevation and slope. Different applications pose different accuracy and resolution requirements to such 3D models.

Applications that target fuel savings and reduced greenhouse gas emissions, benefit from the availability of an accurate 3D map. A transportation study finds that eco-routing that uses a 3D spatial network model can yield fuel cost savings of 8–12%, when compared to standard routing based on a 2D model [2]. Another study reports that the use of a 3D model built from aerial laser scan data for vehicle routing will yield annual fuel savings of approximately USD 6 billion in USA [3]. This figure stems from TomTom, a worldwide leading manufacturer of navigation systems. A study of models of vehicular environmental impact shows how increased accuracy of road slopes yields more accurate estimations of fuel consumption and greenhouse gas (GHG) emissions from vehicles [1].

ADAS applications provide critical information to the driver about the vehicle's surroundings. Here, a 3D map can enhance information obtained from vehicle sensors and can also serve as a failsafe mechanism when sensors fail [7]. For example, *adaptive headlights* take into account the slopes of the road ahead and intelligently steer the headlights in order to offer maximum night-time visibility. Under adverse weather conditions such as excessive fog or rain, the sensors that are used under normal conditions can fail to operate optimally, and a 3D map can be used instead. ADAS specifications require a 3D road model with an accuracy of at least ± 2 meters.

Three well-known and accepted methods of 3D map generation exist. First, a vehicle fitted with differential GPS and an *Inertial Navigation System* (INS) is driven on roads to capture their 3D road geometries [6]. Although this approach is well tested, it is expensive and cumbersome because it entails driving all existing roads to form a comprehensive 3D spatial network. Second, some major map providers use *digital elevation models* (DEM) generated from aerial images collected by *interferometric synthetic aperture radar* (IfSAR) to generate 3D maps [18], [19]. The accuracy and generation time of such models are unknown. Third, crowd-sourced data from *Personal Navigation Devices* (PND)s and vehicular GPS traces are used for extraction of elevation information in order to generate 3D spatial networks. The accuracy of such a model is also unknown and has not been compared to models created using the previous two methods. Also, the method relies on the availability of data that covers an entire transportation network, which is problematic.

Motivated by these observations, we propose two methods that use aerial laser scan data (LiDAR), illustrated in Figure 1, to augment 2D maps with elevation and slope information. *External Memory Filtering* (EMF) is an external memory

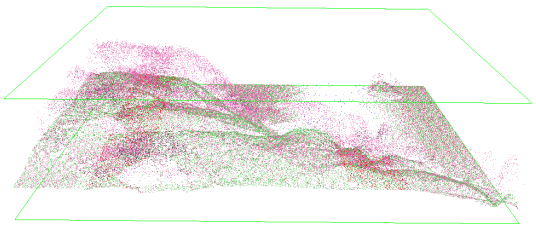


Fig. 1. Example Laser Scan Point Cloud (LiDAR)

algorithm that produces a 3D model of very high accuracy and resolution, and *One Pass Filtering* (OPF) is a streaming in-memory based solution that sacrifices some of the accuracy to generate a lower-resolution model, while reducing the map processing load.

In our experiments on a spatial network that spans *North Jutland*, Denmark, covering a region of 185km $\times$ 130km, we find that EMF can generate a 3D model with an accuracy of

± 20 cm in 21 hours with a memory use of just 230 MB, while OPF can generate the same model with an accuracy of ± 2 meters in approximately 2 hours, which is an order of magnitude faster than EMF, using 2.4 GB of main memory.

The two methods can be combined such that a very large 3D spatial network with relatively low accuracy can be generated quickly using OPF, upon which EMF can be used to *zoom in* on regions of interest, generating high-accuracy maps for these.

A key challenge is to achieve a suitable balance between the accuracy of a model, as dictated by the intended applications, and the storage space required by the model. This paper addresses that challenge by proposing techniques that are capable of using commodity computing hardware for generating accurate and compact 3D spatial network models from massive laser scan point clouds and a 2D spatial network.

To the best of the authors' knowledge, this is the first work that investigates the application of aerial laser scan data to the *lifting* of a 2D spatial network in a fast and efficient manner, to obtain a 3D spatial network. Specifically, our contributions are four-fold.

- We propose a novel filtering and lifting framework that uses an aerial laser scan point cloud for lifting a spatial network.
- Two alternative filtering techniques, a *one pass filter* and an *external memory based filter*, are proposed for obtaining the particular points from an aerial laser scan data set that are needed for the lifting.
- Techniques for spatial lifting, consisting of triangulation and interpolation, are proposed to augment the 2D spatial network with elevation information using the remaining laser points.
- We present a comprehensive and large-scale empirical study that offers insight into the accuracy, efficiency, and scalability properties of the framework.

The remainder of this paper is organized as follows. In Section II, we survey related work. Section III presents a formal problem definition, including the proposed filtering and lifting framework. Section IV details the filtering and lifting phases. Section V covers the empirical analyses of the proposed techniques. Finally, Section VI concludes the paper.

II. Related Work

Currently there are three main methods for constructing 3D road network models. First, some map vendors use vehicles instrumented with differential GPS and INS to capture 3D road geometries [6]. Using this method, NAVTEQ and TeleAtlas have spent several years on covering the major roads in nearly 50 major cities across U.S.A, Asia and Western Europe, but not secondary and tertiary roads [7]. Second, the remote sensing community has explored the creation of 3D models using elevation data from IfSAR and aerial images [18], [19]. Third, Waze[1] and TomTom's MapShare[2] allow users to upload GPS locations with altitude information whose accuracy is ± 3 meters [5].

This paper adopts a different approach and, to the best of the authors' knowledge, is the first to provide a method for obtaining a 3D spatial network by lifting a 2D spatial network using aerial laser scan data in a fast, efficient, and accurate fashion.

Tavares et al. [2] study eco-routing based on a 3D network and find that eco-routes are 1.8% longer than the corresponding shortest routes and that fuel cost savings are in the range 8–12%. Their 3D spatial network is developed based on contour lines, which are of much lower resolution than is the laser scan data we use. The *EcoMark* [1] framework for the evaluation of models of vehicular environmental impact illustrates how the use of an accurate 3D model yields better estimations of fuel and greenhouse gas emissions from transportation.

Several methods have been proposed that use statistical or machine learning techniques to classify aerial laser scan points into different categories [9], [10]. We only consider a particular category of points, namely *ground points*, which capture the actual land surface, and we exclude other points representing, e.g., vegetation, water, buildings, and noise. In the remainder of the paper, when mentioning laser scan points, we refer only to ground points.

III. Preliminaries

We introduce definitions that underly the proposed problem, formalize the spatial network lifting problem, and provide an overview of the filtering and lifting framework. An overview of the notation used in the paper is provided in Table I.

TABLE I
Notation

Notation	Description
G_{2D}	A 2D spatial network.
G_{3D}	A 3D spatial network.
P_c	A 3D laser scan point cloud.
p_i	A 3D laser scan point in P_c.
$prj^{\top}(p_i)$	The projection of a 3D point p_i onto the 2D plane.
$prj^{\perp}(g, \triangle)$	The projection of a 2D model element g onto a TIN $\triangle$.
$\epsilon N(g)$	The ϵ-neighborhood of a 2D model element g.

A. Data Modeling

A spatial network captures both the topology and the embedding into geographical space of a transportation network. We define 2D and 3D spatial networks next.

Definition 1: A **2D spatial network** is modeled as an undirected graph $G_{2D} = (\mathbb{V}, \mathbb{E}, F_{2D}, H_{2D})$, where $\mathbb{V}$ and $\mathbb{E}$ is the vertex set and edge set, respectively, and F_{2D} and H_{2D} are the functions recording the embedding of vertices and edges into the 2D plane, respectively.

A vertex $v_i \in \mathbb{V}$ indicates either a road intersection or the end of a road. An edge $e_k \in \mathbb{E} \subset 2^{\mathbb{V}}$ is defined as a set of two vertices, and represents a road segment connecting the two vertices. For example, edge $e_k = \{v_i, v_j\}$ represents a road

[1]http://www.waze.com
[2]http://www.tomtom.com/en_gb/maps/map-share/

segment that connects vertex v_i and vertex v_j. Function F_{2D} : $\mathbb{V} \rightarrow \mathbb{R}^2$ takes as input a vertex and returns its coordinates in the 2D plane. Function $H_{2D} : \mathbb{E} \rightarrow \mathbb{R}^2 \times \ldots \times \mathbb{R}^2$ takes as input an edge e, and outputs a 2D polyline represented by a sequence of 2D points. For example, edge $H_{2D}(\{v_i, v_j\}) = (v_i, a_1, a_2, a_3, a_4, v_j)$, as shown in Figure 2.

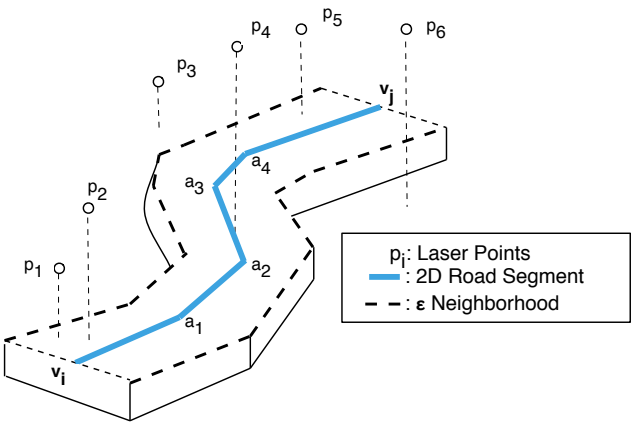


Fig. 2. A 2D Road Segment With its ϵ-Neighborhood

Definition 2: A **3D spatial network** is also modeled as an undirected graph $G_{3D} = (V', E', F_{3D}, H_{3D})$. The definitions of V' and E' in a 3D spatial network are identical to the counterparts in a 2D spatial network. Note that for higher resolutions, our method introduces more vertices and edges into V' and E' respectively. Functions F_{3D} and H_{3D} record the geometric information of vertices and edges in 3D space. Function $F_{3D} : V' \rightarrow \mathbb{R}^3$ takes as input a vertex and returns its coordinates in 3D space; and function $H_{3D} : E' \rightarrow \mathbb{R}^3 \times \ldots \times \mathbb{R}^3$ takes as input an edge and outputs a 3D polyline, represented as a sequence of 3D points. Figure 3 shows the 3D polyline of a road segment. Calculating slopes of each edge E' in G_{3D} becomes a trivial exercise.

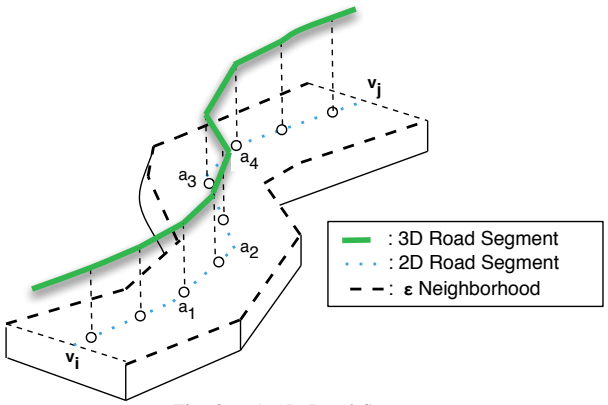


Fig. 3. A 3D Road Segment

Definition 3: A **Laser Scan Point Cloud** (P_c) is a set of laser points p_i, which is formalized as

$$P_c = \{\, p_i = (x_i, y_i, z_i) \in \mathbb{R}^3 \;\mid\; 1 \leq i \leq N\},$$

where N is the total number of laser points in the point cloud.

B. Problem Formulation

Spatial network lifting augments a 2D spatial network G_{2D} with elevation information extracted from a laser scan point cloud P_c and returns G_{2D}'s corresponding 3D representation G_{3D} as the result.

Intuitively, to lift a 2*D* model element (e.g., either a 2D polyline $H_{2D}(e_k)$ indicating a road segment or a 2D point $F_{2D}(v_i)$ indicating a road intersection), the 3D laser points locating nearby the 2D model element are more useful than the laser points that are further away. We take into account the laser points belonging to the **ϵ-Neighborhoods** of the 2D elements while lifting them.

Definition 4: Given a point cloud P_c, the **ϵ-Neighborhood** of a 2D model element g, denoted as $\epsilon N(g)$, is a set of laser points in P_c that satisfy a given spatial predicate ϵ.

$$\epsilon N(g) = \{p_i \in P_c \mid \epsilon(p_i, g)\},$$

where $\epsilon(p_i, g)$ denotes a spatial predicate defined on a laser point p_i and a 2D model element g.

Since a 2D point can be regarded as a special case of a 2D polyline, the following discussion focuses on the lifting of road segments (i.e., 2D polylines) instead of the lifting of road intersections (i.e., 2D points).

An example of an ϵ-Neighborhood of a road segment is shown in Figure 2, where the solid polyline in the center is a road segment $\{v_i, v_j\}$. The projection of a laser point $p_i \in P_c$ onto the 2D plane is defined as $prj^{\top}(p_i) = (x_i, y_i, 0)$. We let the spatial predicate ϵ be defined as $dist(prj^{\top}(p_i), H_{2D}(e)) \leqslant d$, which means that if a laser point satisfies the predicate, the shortest distance between its 2D projection $prj^{\top}(p_i)$ to the 2D polyline of the edge should not exceed d. The two dotted lines indicate the boundary of points that satisfy the predicate. Since the projected points $prj^{\top}(p_1)$, $prj^{\top}(p_2)$, $prj^{\top}(p_4)$ and $prj^{\top}(p_5)$ lie in the range bounded by the two dotted lines, $\epsilon N(H_{2D}(\{v_i, v_j\})) = \{p_1, p_2, p_4, p_5\}$.

The laser points in ϵ-Neighborhood $\epsilon N(g)$ can be transformed into a Triangulated Irregular Network (TIN), denoted as $\triangle(g)$, to approximate the surface around the road segment g. Projecting the 2D model element g onto its corresponding TIN $\triangle(g)$, its 3D polyline representation becomes available. For example, the 3D representation of road segment $\{v_i, v_j\}$ is shown in Figure 3. Assuming $prj^{\perp}(g, \triangle)$ indicates the projection of a 2D model element g onto a TIN surface $\triangle$, spatial network lifting is formalize as follows.

Definition 5: **Spatial network lifting** takes as input a 2D spatial network G_{2D} and a laser scan point cloud P_c, and it returns the corresponding 3D spatial network G_{3D} in which $F_{3D}(v) = prj^{\perp}(F_{2D}(v), \triangle(v))$ for every $v \in \mathbb{V}$, and $H_{3D}(e) = prj^{\perp}(H_{2D}(e), \triangle(e))$ for every $e \in \mathbb{E}$.

C. Framework Overview

Figure 4 depicts the overview of spatial network lifting, which consists of two major phases: *filtering* and *lifting*.

The filtering phase takes as input a 2D spatial network and a massive laser scan point cloud, and prunes irrelevant laser points in the point cloud in order to obtain an appropriate

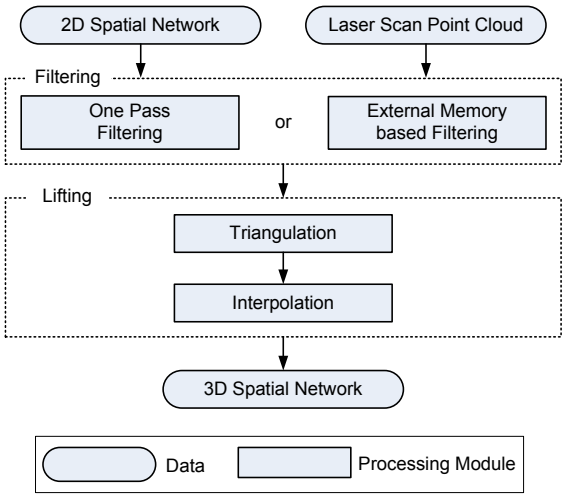


Fig. 4. Framework Overview

ϵ-Neighborhood of every road segment and road intersection in the 2D spatial network. Two alternative filtering methods, *one pass filtering* and *external memory based filtering*, are provided in order to exploit situations where the 2D networks fits in main memory and to also provide a general solution that does not make this assumption.

The lifting phase consists of two steps, where *interpolation* follows *triangulation*. In the triangulation step, laser points in ϵ-Neighborhoods are transformed into TINs. After projecting the 2D spatial network onto the TINs, the interpolation step computes the corresponding elevation information, thus providing a 3D spatial network as the final output.

IV. Spatial Network Lifting

Spatial network lifting focuses on generating an accurate and compact 3D spatial network in an efficient and scalable manner.

A. Filtering

1) Overview: The main task of the filtering phase is to obtain ϵ-neighborhoods for the 2D model elements in G_{2D}. Instead of proposing yet another index to filter the laser points, we explore the opportunities of applying existing indexing techniques. A priori knowledge of the laser points being almost uniformly distributed with a guaranteed minimum resolution (e.g., at least one laser point per square meter), significantly influences our decision to choose a space-driven indexing technique, in particular, the grid index, instead of a data-driven indexing technique, e.g., a tree-based index.

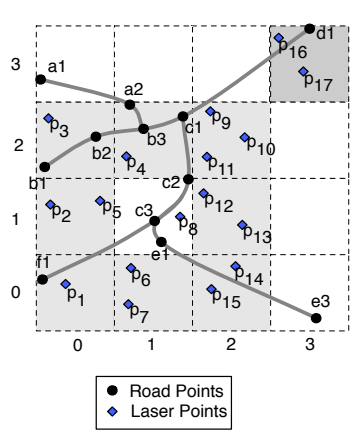
As an aside, we chose not to use an approch where we first build spatial indexes on the point cloud P_c and on the 2D spatial network (e.g., using R-trees) and then join the two by synchronized traversing the two indices [11]–[13]. This is because P_c is massive in size (for Denmark, on the order of terabytes) and is collected rarely. Thus, the join operation is not carried out repeatedly, and the filtering is merely an intermediate step in solving the lifting problem.

By utilizing grid based indices, the filtering becomes parallelizable and can be processed easily on powerful machines with large main memories. However, we also consider the setting where the available main memory is limited, as this renders the paper's proposal applicable to commodity hardware.

We provide two filtering approaches that differ primarily in how they manage the smaller data set, i.e., the 2D spatial network G_{2D}.

1) One pass filtering, described in Section IV-A2, assumes that there is enough internal memory to accommodate a grid index on G_{2D} and filters the point cloud P_c in a single pass by checking whether a laser point belongs to the ϵ-Neighborhoods of all road segments in G_{2D};
2) External memory based filtering, described in Section IV-A3, works without any assumption on main memory size, so neither G_{2D} nor P_c are assumed to fit in internal memory. Two different traversal strategies, row-major and z-curve order, are used for loading disk blocks into memory for filtering.

2) One Pass Filtering: One pass filtering (or *OPF* for simplicity) utilizes a uniform grid to index both the 2D spatial network and the laser points. Cells in the grid are squares, and the width of a cell is governed by a user specified parameter δ that needs to be given before generating the grid. Figure 5 shows an example of how grid cells map to both road points (points contained in 2D polylines) and laser points. For ease of illustration, only a small portion of laser scan points mapped to cells are shown, while in reality, the cells have much more laser scan points due to the high density of the laser scan point cloud.



OPF

Cell	H_g	H_p
(3,3)	{d1,c1}	p_{16}, p_{17}

EMF

Cell	Road Blk	Laser Blk
(0,0)	{c3,f1}	p_1
(0,1)		p_6, p_7
(0,2)		p_{14}, p_{15}
(1,0)		p_2, p_5
(1,1)	**{c2, c3}, {c3,f1}, {c3, e1}, {e1,e3}**	**p_8**
(1,2)		p_{12}, p_{13}
(2,0)	{b1,b2},...	p_3
(2,1)	{a2,b3},...	p_4
(2,2)		p_9, p_{10},p_{11}

Fig. 5. Grid Partitioning Index

The spatial predicate used in *OPF*, denoted as $\epsilon_{OPF}(g, p_i)$, returns true if a 2D model element g and the 2D projection of a laser point p_i, i.e., $prj^{\top}(p_i)$, fall into the same grid cell. Given the same 2D model element g, the bigger the cell width δ is, the more laser points are contained in its ϵ-Neighborhood $\epsilon N(g)$.

OPF assumes that the grid index on the 2D spatial network, i.e., the mapping H_g given below, can fit fully into internal memory.

$$H_g : C \rightarrow \bigcup_{e \in \mathbb{E}} H_{2D}(e).$$

For each cell c in grid cells set C, mapping H_g maintains a set, denoted as $H_g(c)$, containing all the 2D model elements that intersect with the cell.

Upon creation, H_g acts as a *seed* for generating the grid index on the point cloud P_c. *OPF* sequentially scans P_c only once to generate another mapping H_p from seed cells (those cells having 2D model elements in H_g) to laser points whose 2D projections are within the cells. H_p is formally defined as follows.

$$H_p : SC \rightarrow P_c; \text{ where } H_g(c) \neq \emptyset \text{ if } c \in SC \subseteq C.$$

Note that $H_p(c)$ records the ϵ-Neighborhoods of the 2D model elements in $H_g(c)$.

Isenburg and Lindstrom [14] observe that laser points are inherently *topologically-coherent*, which implies that the laser points are stored in an order that is an artifact of how they were collected by the planes flying over the covered land surfaces. In other words, the laser points in the point cloud are not stored randomly. Rather, laser points that are geographically close are also stored close to each other and hence the point cloud is stored in a manner that to some extent is *locality preserving*. *OPF* exploits this property and thus avoids performing several passes to sort the point cloud P_c.

Based on the above observation, the mapping H_p does not need to be maintained for every cell in memory at all times. Once $H_p(c)$ contains a sufficient number of laser points for cell c, *artificial 2D road points* are inserted at all intersections of the road edges and grid cell boundaries. Finally, the 2D model elements in $H_g(c)$ are lifted. Recall that the laser point cloud we use guarantees one laser point per square meter, meaning that a cell with width δ is expected to contain $\delta \cdot \delta$ laser points. As *OPF* scans the point cloud, when $H_p(c)$ contains more than $\alpha \cdot \delta \cdot \delta$ laser points, $H_p(c)$ can be passed to the lifting phase immediately. Here, the fill factor $\alpha \in (0, 1]$) is a parameter that represents a trade off between efficiency and accuracy: the higher the fill factor is, the more laser points must be contained in the ϵ-Neighborhoods, thus making the final 3D spatial network more accurate.

Algorithm 1 describes *OPF*. Function $GetIntersectedCells(ls,\ \delta)$ (in line 3) returns the cells that intersect with line segment ls according to cell width δ, and function $GetContainedCells(p_i,\ \delta)$ (in line 6) returns the cell that contains the 2D projection of laser point p_i according to cell width δ.

The advantages of one pass filtering are: (i) *OPF* does not incur any pre-processing cost of sorting or indexing the massive point cloud P_c; and it only scans the point cloud once; (ii) *OPF* is able to output parts of the resulting 3D spatial network with different accuracy requirements (by configuring α) as the laser points stream in. (iii) *OPF* can easily be parallelized to take advantage of either new hardware architectures like GPUs or cloud infrastructures like MapReduce.

Algorithm 1: OnePassFilter

```
   Input : 2D spatial network G_2D, grid cell width δ,
           point cloud P_c, fill factor α.
   /* Initialize mapping H_g                          */
1  for each edge e ∈ G_2D.𝔼 do
2      for each line segment ls ∈ H_2D(e) do
3          for each cell c ∈ GetIntersectedCells(ls, δ) do
4              H_g(c) ← H_g(c) ∪ ls;
   /* One pass scan on the point cloud P_c            */
5  for each laser point p_i ∈ P_c do
6      Cell c ← GetContainedCell(p_i, δ)
       /* Use H_g as a seed                           */
7      if H_g(c) ≠ ∅ then
8          H_p(c) ← H_p(c) ∪ p_i;
9          if |H_p(c)| ⩾ α · δ · δ then
               /* Lift the 2D model elements
                  in cell c                            */
10             Lifting(H_g(c), H_p(c));
11             Release H_g(c) and H_p(c);
```

3) External Memory Based Filtering: In contrast to *OPF*, external memory based filtering (abbreviated as *EMF*) works even if neither G_{2D} nor P_c is able to fit in internal memory. The basic goal of *EMF* is to efficiently filter large data sets of arbitrary sizes given a limited and fixed main memory budget.

In order to achieve an accurate TIN, *EMF* employs a spatial predicate $\epsilon_{EMF}(g, p_i)$ that returns true if the cell containing $prj^{\top}(p_i)$ is the cell that contains g or is one of the eight neighboring cells of the cell containing g (also called Moore neighbor cells of g). In the following discussion, we use *9-cell* to indicate a cell and its Moore neighbor cells. For example, the 9-cell of cell $(1, 1)$ is shown in the bottom left of Figure 5.

EMF scans both G_{2D} and P_c, organizing them into road blocks (lines 1–4 in Algorithm 2) and laser blocks (lines 5–7 in Algorithm 2), where each road (laser) block contains the road segments that intersect with (laser points that are in) a cell. The size of a laser block is typically decided by the cell width δ. Assuming each laser point takes 20 bytes (two doubles and one float), a laser block needs $20 \cdot \delta \cdot \delta$ bytes. (Table III in Section V-A details the block sizes.)

A road block typically takes up much less space than a laser block because it is uncommon to have very dense road segments (e.g., with a point for every one meter). We therefore assume that a road block has at most the same size as a laser block. After block reorganization, *EMF* reads road blocks into main memory according to a locality preserving space filling curve. After reading a new road block, *EMF* reads in its corresponding 9-cell laser blocks and overwrites laser blocks using the least recently used (LRU) [15] policy. A road block

Algorithm 2: ExternalMemoryFilter

Input : 2D spatial network G_{2D}, grid cell size δ, point cloud P_c, Curve Tag TAG; Memory Budget B.

/* Scan and sort G_{2D} into road blocks */

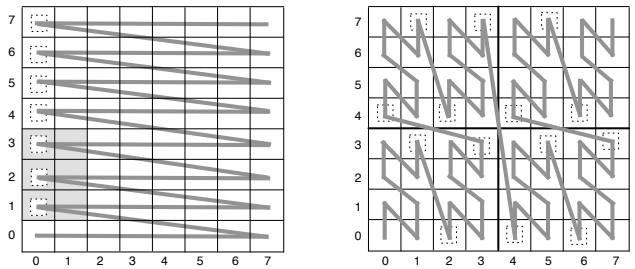
1 **for** *each edge* $e \in G_{2D}.\mathbb{E}$ **do**
2 **for** *each line segment* $ls \in H_{2D}(e)$ **do**
3 **for** *each cell* $c \in GetIntersectedCells(ls,\ \delta)$ **do**
4 $writeBlock(c,\ ls,\ FILE_G)$;

/* Scan and sort P_c into laser blocks */

5 **for** *each point* $p_i \in P_c$ **do**
6 Cell $c \leftarrow GetContainedCell(p_i,\ \delta)$;
7 $writeBlock(c,\ p_i,\ FILE_P)$;
8 Buffer $A \leftarrow \emptyset$;
9 **while** *Decide next cell* c *according to* TAG *curve* **do**
10 $A \leftarrow readBlock(c,\ FILE_G)$;
11 **for** *each cell* $c' \in MooreNeighbour(c) \cup c$ **do**
12 $B \leftarrow LRU(B,\ readBlock(c',\ FILE_P))$;
 /* Lift the 2D model elements in cell c */
13 $Lifting(A,\ B)$;

along with its 9-cell laser blocks are fed into the lifting phase, as described in lines 9–13 in Algorithm 2.

Given a limited memory budget, the order in which road blocks are read has a significant impact on the performance of *EMF*. In order to avoid frequent re-reading of the same laser blocks, locality preserving space filling curves are considered when *EMF* loads the road blocks. In particular, we consider two space-filling curves, namely the *row-major curve* and the *z-order curve*. Figure 6 shows the orders in which road blocks are loaded in memory starting from the bottom left cell, according to the two row-major and z-order curves, respectively.

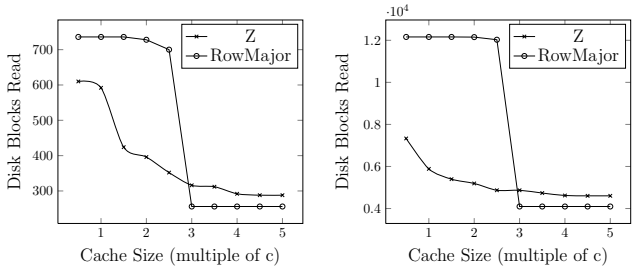


(a) Row-major curve. (b) Z-order curve.

Fig. 6. Locality Preserving Space Filling Curves

When a road block is being processed, its corresponding 9 laser blocks must be available in memory for processing. As *EMF* reads road blocks and moves along a curve, there are moments where memory is full and old laser blocks are overwritten with new ones. The dotted boxes in Figure 6 show the points where there is a high likelihood that new laser blocks must be read from disk.

Analysis: Given a grid with $n \cdot n$ cells and a fixed-size memory (a multiple of n) that employs the LRU policy [15], we investigate the effects of using Z-order and row-major curves. Two grids with sizes $16 \cdot 16$ and $64 \cdot 64$ are considered, and we vary the memory size in multiples of n with $0.5 \cdot n$ being the finest granularity. We then report the number of laser block replacements in Figure 8. The results show that



(a) $16 \cdot 16$ grid. (b) $64 \cdot 64$ grid.

Fig. 8. Performance of Z-order and Row-major Curves on Varied Memory Budget Sizes

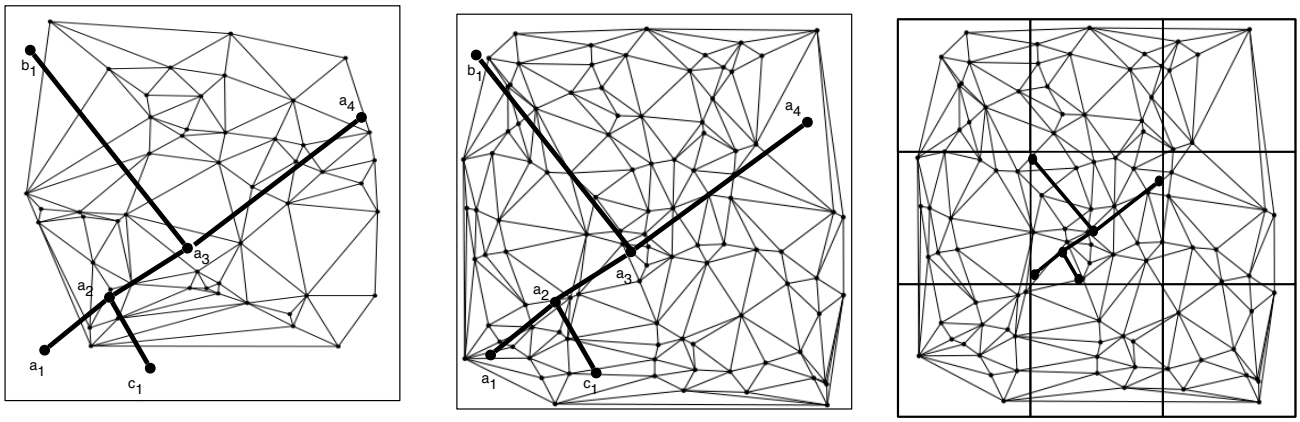
for a memory budget ranging from $0.5 \cdot n$ to $2.5 \cdot n$, the Z-order yields the fewest reads. The benefit of using the Z-order increases as the grid size n increases. Starting at $3 \cdot n$ and onwards, the row-major ordering begins to outperform Z-order and performs stably. This is because the 9 laser blocks that must be considered for a road block belong to 3 grid rows.

To illustrate, consider again Figure 6. Assuming that the memory budget is $3 \cdot n$, when the first two rows of road blocks have been processed according to the row-major curve, the memory budget is fully occupied by the first three rows of laser blocks. In order to process the next road block, i.e., $(2, 0)$ (surrounded with a dashed box), laser blocks $(3, 0)$ and $(3, 1)$ need to be read, and laser blocks $(0, 0)$ and $(0, 1)$ are over-written. The over-written blocks will never again be needed and hence with memory budget size $3 \cdot n$, no laser block needs to be read more than once. Therefore, with memory budget size no less than $3 \cdot n$, row-major behaves in a stable fashion, while with the Z-order some blocks that are over-written may be needed at a later time. Thus, if the memory budget is less than $3 \cdot n$, Z-order is preferable, while row-major wins and performs stably if the memory budget is no less than $3 \cdot n$.

B. Lifting

Upon successful filtering, the laser points in the ϵ-Neighborhoods are *triangulated* into a TIN. Then the elevation information in the TIN is assigned to the 2D road segments by projecting these onto the TINs and performing *interpolation*.

1) Triangulation: The elevation values in a given region are only available for the points where measurements were taken (e.g., the laser points in the region). To get the elevation for other points in the region, some form of approximation must



(a) $\alpha = 0.1$. (b) $\alpha = 0.4$. (c) 9-cell

Fig. 7. (a)(b) Delaunay Triangulation With Varying Fill Factor α in *OPF*. (c) Delaunay Triangulation With 9-Cell in *EMF*.

be applied. A naive approach assigns an elevation to a point that is equal to the elevation of the point's nearest neighbor in the laser point cloud or that is equal to the average elevation of its k nearest laser points. However, these approaches are unable to produce accurate results.

We adopt a different tack: given a region, all the pertinent laser points, e.g., those in ϵ-Neighborhoods, are triangulated into a TIN to approximate the surface of the region. The elevation of any point in the region can then be interpolated from the TIN.

Triangulation transforms a set of laser points, which represent discrete measurements on a surface, into a set of non-overlapping triangles where the vertices of the triangles are the laser points. We use Delaunay Triangulation [16] for triangulation. This is a specialized triangulation method where, in the resulting triangles, no triangle vertex is inside the circumscribed circles of any other triangle.

Recall that *OPF* utilizes a fill factor α to act as a parameter when determining how many laser points must be present when moving to the triangulation step. The higher the fill factor is, the more accurate the TIN approximates the real surface, thus yielding a more accurate 3D spatial network. For example, with a low fill factor of $\alpha = 0.1$, some road segments (road points a_1, b_1, and c_1) are not covered by the resulting TIN (shown in Figure 7(a)). Increasing α to 0.4 improves the coverage (Figure 7(b)).

EMF passes all the laser points in the relevant 9-cell to the triangulation step, as shown in Figure 7(c), which typically guarantees that all the road segments in the center cell are fully covered by the resulting TIN, thus achieving higher accuracy.

2) Interpolation: Figure 9 shows a 2D polyline representing a road segment and its TIN. Intuitively, projecting the 2D polyline to the TIN, the road segment's 3D representation becomes available. However, directly projecting a polyline to a TIN is computationally expensive. Instead, we sample a set of 2D road points on the 2D polyline and then project them onto the TIN in order to obtain their 3D counterparts. By connecting these 3D road points, the 3D polyline representation of the road segment is obtained, e.g., the 3D polyline $(a'_1, \ldots, a'_5, \ldots, a'_{11})$ shown in Figure 9.

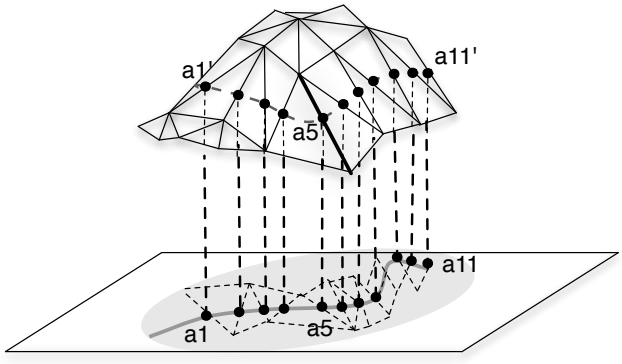


Fig. 9. *ES* With Intersection Road Points

Exact sampling: *ES* selects the points where either the direction or the grade of the road segment change. *ES* starts by projecting the TIN to the 2D plane by ignoring the z coordinates of all vertices. To illustrate, this yields the dashed triangles on the 2D plane shown in Figure 9. Next, points on the road polyline are sampled by picking the points in the original 2D polyline representation (i.e., the points where the direction of the road changes) and all the intersections of the edges of the 2D triangles and the original 2D polyline (i.e., the points where the grade of the road segment may change). The grade of a road segment may change only when the road segment crosses from one triangle to another.

Since the intersections obtained by *ES* are always on TIN triangle edges, the elevations at the vertices of the triangle edges are used in linear interpolation to compute the intersections' elevations. Figure 9 shows an example of a triangle edge in bold; its two corresponding vertices are used for computing the elevation of road point a'_5.

The remaining points in *ES* may fall in a triangle, and the elevations of these points are interpolated by taking into account the elevations of all three vertices of the triangle, i.e., using plane-based interpolation.

V. EXPERIMENTS

We detail the data sets, parameter settings, and implementation. Then we cover the empirical study of the the efficiency, accuracy, and scalability properties of the proposed filtering and lifting framework. Comparisons to existing methods, covered in Section II, were not performed because the data necessary to do so was not available to us.

A. Experimental Setup

Data Sets: Two spatial networks in Denmark are lifted in the experiments.

Aalborg (AA) covers the Aalborg region (North Jutland, Denmark) which is of size approximately $7km \cdot 5km$. The laser point cloud of *AA* occupies 2.84 GB. The spatial network of Aalborg, obtained from OpenStreetMap, has a total length of approximately $4 \cdot 10^5 m$, and its 2D polyline representation is 1 MB in storage size, consisting of $13,366$ points.

North Jutland (NJ), the northern part of Jutland, Denmark, covers a region of $185km \cdot 130km$. The laser point cloud of *NJ* occupies 342 GB. *NJ* contains a spatial network with a total length of $1.17 \cdot 10^7 m$, whose 2D polyline representation is 28 MB in storage size, containing $414,363$ points.

We report on experiments were carried out on *AA*, unless stated explicitly otherwise.

Parameter Settings: The experiments are conducted by varying several parameters to study the effect of the trade-offs between accuracy, efficiency, and memory usage. Table II shows the parameters with default values shown in bold. The equidistant parameter ed used in approximate sampling is set to $10m$. Experiments are conducted using default parameter values unless explicitly stated otherwise.

TABLE II
PARAMETER SETTINGS

Grid Cell Width δ (m)	16, **64**, 128
Filling Factor α	0.4, **0.6**, 1.0
Memory Size	c, $2 \cdot c$, **3 · c**, $4 \cdot c$

Table III shows details on the grid indices for different cell widths. In the table, $r \cdot c$ indicate the size of the grid, where r and c denotes the number of rows and columns, respectively.

TABLE III
DETAIL OF GRID INDICES

δ (m)	*AA* $(r \cdot c)$	*NJ* $(r \cdot c)$	BS (KB)
16	$316 \cdot 465$	$8082 \cdot 11526$	5
64	$79 \cdot 117$	$2021 \cdot 2882$	80
128	$40 \cdot 59$	$1011 \cdot 1441$	320

We compare variants of *EMF* that use row-major ordering and Z-curve ordering in terms of disk block reads by varying the available memory budget. The available memory is set as a multiple of c blocks (where c is the width of the grid). Note that as δ varies, the block size, which relates to how many laser points or 2D polylines fall in a cell in the grid, also varies accordingly. The size of a (laser or road) block is shown in the last column of Table III.

Implementation: The filtering and lifting framework is implemented in C and C++, and Perl is used to to perform auxiliary tasks. SCALGO Terrastream[3] is used to generate the TIN used in our ground-truth method (in Section V-B). Triangle[4] is employed for Delaunay triangulation in the lifting phase. An R-tree library[5] is applied to facilitate the interpolation step in the lifting phase. All the experiments are carried out on an Ubuntu 11.04 LINUX machine with an Intel Xeon W3565 @3.2 GHz CPU (8MB cache, hyper-threading, 4 cores), 8 GB internal memory and 16.5 TB hard disk.

B. Ground Truth Generation

The ground truth is generated by triangulating all the laser points in the *NJ* point cloud into a huge TIN and then interpolate the road points obtained by exact sampling using this TIN. The statistics of the huge TIN are listed in Table IV, where $\triangledown$ indicates triangles in a TIN.

TABLE IV
GROUND TRUTH STORAGE

Data	Laser	TIN	TIN-to-$\triangledown$	Number of $\triangledown$
AA	2.84 GB	3.1 GB	6.3 GB	$3.7 \cdot 10^7$
NJ	342 GB	372 GB	742 GB	$9 \cdot 10^9$

We use *sum of squared errors (SSE)*, which is defined in Equation 1, for quantifying the differences between the 3D spatial networks generated by the filtering and lifting framework and the ground truth.

$$SSE = \sum_{i=1}^{|Rp|} (z_i - gt_i)^2 \quad (1)$$

In the equation, $|Rp|$ is the total number of road points based on a sampling strategy (as described in Section IV-B2); z_i is the elevation reported by the proposed method, and gt_i is the elevation obtained from the ground truth. The absolute error, $A_i = |z_i - gt_i|$, is also introduced as another accuracy measure in the subsequent discussion.

C. Accuracy Studies

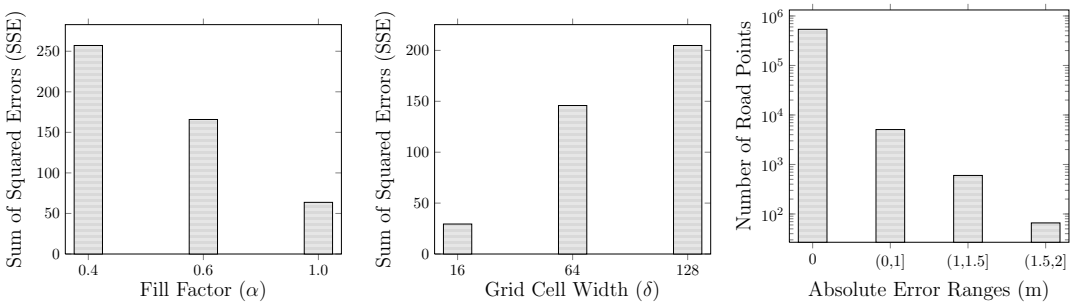
We analyze the accuracy of the proposed approaches against the ground truth.

Accuracy Analysis of *OPF*: Figure 10(a) illustrates the effect of varying the fill factor (α), while fixing the grid size to its default size ($64m$). As we increase α, more laser points are passed into Delaunay triangulation, which yields more accurate elevation values.

Setting α to its default value and increasing the grid cell width, as shown in Figure 10(b), we notice that the SSE increases.

Although α acts as a threshold value for deciding how many laser points should be processed in triangulation, it is not

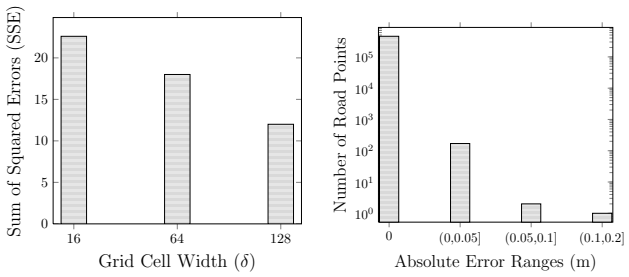
[3]http://madalgo.au.dk/Trac-TerraSTREAM
[4]http://www.cs.cmu.edu/ quake/triangle.html
[5]http://superliminal.com/sources/RTree.zip



(a) Varying α. (b) Varying δ. (c) Absolute Errors

Fig. 10. *OPF* Accuracy Study

able to control which laser points are chosen for triangulation. Since a grid cell with larger width should contain more laser points, the α fraction of laser points are more likely to be non-uniformly distributed in the cell, causing a deterioration in the triangulation and hence in the computed elevation values.



(a) Varying δ. (b) Absolute Errors

Fig. 11. EMF Accuracy Study

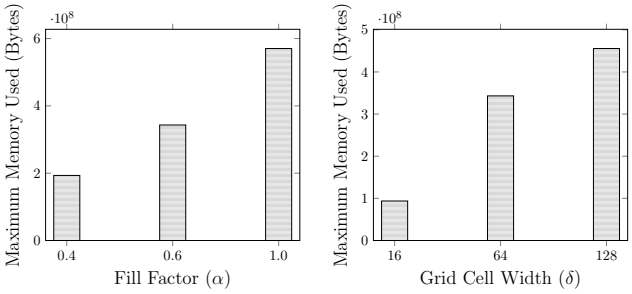
Accuracy Analysis of *EMF*: Unlike *OPF*, *EMF* employs triangulation on laser points in 9-cells, which results in a much better triangulation quality. As we increase the grid cell width, the SSE drops and the elevation values computed become much more accurate—see Figure 11(a). This occurs because the 9-cells contain increasingly more laser points, which results in higher quality triangulations.

Figures 10(c) and 11(b), for *OPF* and *EMF* respectively, display the numbers of absolute error A_i within different error ranges, thus showing the distribution of the errors. The leftmost bar in each figure indicates the road points whose elevation are computed with no error. Note that the points in the subsequent buckets drop significantly in both methods, especially in *EMF*.

D. Storage Studies

We quantify the storage requirements as the maximum amount of main memory required at any moment in the case of *OPF*; and as the number of laser and road block reads from disk in the case of *EMF*.

Memory Usage of *OPF*: Figure 12(a) shows that the maximum memory required grows as α increases. The larger α gets, the longer the wait is until there are enough laser points to proceed to the lifting phase. Hence, there is a greater demand on memory. Likewise, increasing the grid cell width yields a greater memory requirement to hold laser points as the area of the cell increases (shown in Figure 12(b)).



(a) Varying α. (b) Varying δ.

Fig. 12. OPF Memory Measurements

Memory Usage of *EMF*: Figure 13 indicates that the disk block read performance of the Z-curve and row-major orders adhere to our earlier analytical results (cf. Figure 8). The biggest memory budget in *EMF*, with size of $4 \cdot c$ and the largest grid cell width of $128m$, does not exceed 80 MB of main memory. Additionally, given the maximum α and grid cell size, *OPF* has a memory upper bound of approximately 600 MB. Hence, both *EMF* and *OPF* can function with very limited memory for the AA spatial network.

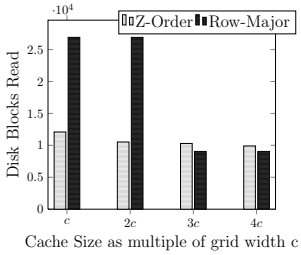


Fig. 13. EMF Block Reads

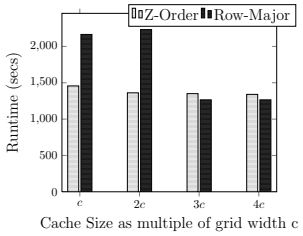
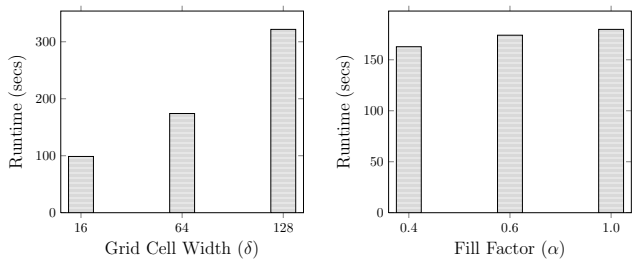


Fig. 14. EMF Runtime

E. Efficiency Analysis

We analyze the performance of our proposed methods in terms of runtime. The goal is to gain insight into the practical feasibility of *OPF* and *EMF*.

For *OPF*, Figures 15(a) and 15(b) exhibit the effects of varying grid cell width and fill factor, where both end up having to pass increasing numbers of laser points to the lifting phase where triangulation and interpolation take longer.



(a) Varying δ. (b) Varying α.

Fig. 15. One-Pass Runtime Measurements

For *EMF*, Figure 14 shows the effect on overall runtime when the available memory is varied. A very large proportion of time is spent on disk reads, which are much more time consuming than in-memory operations. We see a strong correlation between the disk reads and the overall runtime of *EMF*, which is also highlighted by the similarity to the I/O measurements shown in Figure 13.

Although results show that *OPF*, which employs no pre-processing at all, is nearly an order of magnitude faster than *EMF*, this superior run-time performance comes at the cost of an accuracy degradation (as shown in Section V-C). Note that both *OPF* and *EMF* take much less time than the ground truth computation.

F. Scalability Studies

We conduct experiments on both *AA* and *NJ* to observe the scalability of the proposed methods. Recall that the point cloud of *NJ* is almost 120 times larger than the point cloud of *AA*, as shown in Table IV. We apply *OPF* on both data sets with default parameters. The corresponding runtime and maximum memory requirement suggest good scalability, as shown in Table V. For *EMF*, we use Z-order and a memory budget

TABLE V
SCALABILITY ANALYSIS

Filtering	**Dataset**	**Runtime (mins)**	**Max Mem (MB)**
OPF (Default δ, α)	AA	2.9	327
	NJ	122.3	2,457.6
Filtering	**Dataset**	**Runtime (mins)**	**Blocks Read**
EMF (Z-Order, c)	AA	24.2	12,092
	NJ	1270.8	126,895

with c blocks (i.e., around 230 MB); all the other parameters are set to default values. The results in Table V also suggest that the runtime and number of block reads are proportional to the numbers of laser points in both data sets, which in turn suggests that *EMF* is scalability in the point cloud size. *EMF* outputs a 3D spatial network that is an order of magnitude larger for *NJ* spatial network, with a total of $4,640,865$ 3D points.

VI. CONCLUSION AND OUTLOOK

We study a spatial network lifting problem that augments a standard 2D spatial network with elevation values extracted from a laser scan point cloud. We propose a novel filtering and lifting framework that aims to produce accurate 3D spatial network models that occupy limited storage in an efficient and scalable manner using commodity hardware. The results of extensive empirical studies offer insight into the design properties of the framework and suggest that the framework is practical and is indeed capable of meeting the design goals.

Our future work aims at exploring higher accuracy estimations of eco-routes using our 3D spatial network.

ACKNOWLEDGEMENTS

The authors wish to thank the companies COWI and SCALGO for allowing us to use the aerial laser scan data and for their help with the pre-processing of this data. The work was supported by the Reduction project that is funded by the European Comission as FP7-ICT-2011-7 STREP project number 288254.

REFERENCES

[1] C. Guo, Y. Ma, B. Yang, C. S. Jensen, and M. Kaul. Ecomark: Evaluating Models of Vehicular Environmental Impact. In *ACM SIGSPATIAL*, pages 269–278, 2012.

[2] G. Tavares, Z. Zsigraiova, V. Semiao, and M.G. Carvalho. Optimisation of MSW Collection Routes for Minimum Fuel Consumption using 3D GIS Modelling. *Waste Management*, 29(3):1176–1185, 2009.

[3] L. Sugarbaker, G. Snyder, and D. Maune. Results of the National Enhanced Elevation Assessment. *Presentation to International LiDAR Mapping Forum*, 2012.

[4] NASA Jet Propulsion Laboratory. Shuttle Radar Topography Mission. http://www2.jpl.nasa.gov/srtm.

[5] USGS Global Positioning Application and Practice. U.S Geological Survey. http://water.usgs.gov/osw/gps/index.html.

[6] Tele Atlas speeding up 3D Road Map Progress. Roads and Bridges. http://www.roadsbridges.com/tele-atlas-speeding-3d-road-map-progress.

[7] M. W. Dobson. ADAS and 3D-Road Map Databases. *GeoInformatics*, pages 28–33, September 2009.

[8] M. Over, A. Schilling, S. Neubauer, and A. Zipf. Generating web-based 3D city models from OpenStreetMap: the current situation in Germany. *Computers, Environment and Urban Systems*, 34(6):496–507, 2010.

[9] Y. Liu, Z. Li, R. Hayward, R. Walker, and H. Jin. Classification of Airborne LiDAR Intensity Data using Statistical Analysis and Hough Transform with Application to Power Line Corridors. In *Digital Image Computing: Techniques and Applications*, pages 462–467, 2009.

[10] A. S. Antonarakis, K. S. Richards, and J. Brasington. Object-based Land Cover Classification using Airborne LiDAR. *Remote Sensing of Environment*, 112(6):2988–2998, 2008.

[11] Thomas Brinkhoff, Hans-Peter Kriegel, and Bernhard Seeger. Efficient Processing of Spatial Joins using R-trees. In *SIGMOD*, pages 237–246, 1993.

[12] M.-L. Lo and C. V. Ravishankar. Spatial Joins using Seeded Trees. In *SIGMOD*, pages 209–220, 1994.

[13] Y.-W. Huang, N. Jing, and E. A. Rundensteiner. Spatial joins using R-trees: Breadth-first Traversal with Global Optimizations. In *VLDB*, pages 396–405, 1997.

[14] M. Isenburg and P. Lindstrom. Streaming Meshes. In *IEEE Visualization*, page 30, 2005.

[15] J. L. Hennessy and D. A. Patterson. *Computer Architecture, Fourth Edition: A Quantitative Approach*. Morgan Kaufmann, 2006.

[16] M. de Berg. *Computational Geometry: Algorithms and Applications*. Springer, 2000.

[17] U. Ramer. *An Iterative Procedure for the Polygonal Approximation of Plane Curves*. Defense Technical Information Center, New York University, 1972.

[18] C. Zhang, E. Baltsavias, and A. Gruen. Knowledge-based Image Analysis for 3D Road Reconstruction. In *Asian Journal of Geoinformatics*, pages 3–8, 2000.

[19] C. Zhang, E. Baltsavias, and A. Gruen. Improvement and Updating of Cartographic Road Databases by Image Analysis Techniques. In *Ph.D. thesis, Institute of Geodesy and Photogrammetry*, 2002.