ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences, Volume I-3, 2012
XXII ISPRS Congress, 25 August – 01 September 2012, Melbourne, Australia

A MARKED POINT PROCESS MODEL FOR VEHICLE DETECTION IN AERIAL LIDAR POINT CLOUDS

Attila Börcs, Csaba Benedek

Distributed Events Analysis Research Laboratory
Computer and Automation Research Institute
H-1111, Budapest, Kende utca 13-17
{borcs,bcsaba}@sztaki.hu

Commission III/2

KEY WORDS: LIDAR, aerial laser scanning, vehicle, urban, Marked Point Process

ABSTRACT:

In this paper we present an automated method for vehicle detection in LiDAR point clouds of crowded urban areas collected from an aerial platform. We assume that the input cloud is unordered, but it contains additional intensity and return number information which are jointly exploited by the proposed solution. Firstly, the 3-D point set is segmented into ground, vehicle, building roof, vegetation and clutter classes. Then the points with the corresponding class labels and intensity values are projected to the ground plane, where the optimal vehicle configuration is described by a Marked Point Process (MPP) model of 2-D rectangles. Finally, the Multiple Birth and Death algorithm is utilized to find the configuration with the highest confidence.

1 INTRODUCTION

Vehicle detection on urban roads is a crucial task in automatic traffic monitoring and control, environmental protection and surveillance applications (Yao et al., 2011). Beside terrestrial sensors such as video cameras and induction loops, airborne and spaceborne data sources are frequently exploited to support the scene analysis. Some of the existing approaches rely on aerial photos or video sequences, however in these cases, it is notably challenging to develop a widely applicable solution for the recognition problem due to the large variety of camera sensors, image quality, seasonal and weather circumstances, and the richness of the different vehicle prototypes and appearance models (Tuermer et al., 2010). The Light Detection and Ranging (LiDAR) technology offers significant advantages to handle many of the above problems, since it can jointly provide an accurate 3-D geometrical description of the scene, and additional features about the reflection properties and compactness of the surfaces. Moreover the LiDAR measurements are much less sensitive on the weather conditions and independent on the daily illumination. On the other hand, efficient storage, management and interpretation of the irregular LiDAR point clouds require different algorithmic methodologies from standard computer vision techniques.

LiDAR based vehicle detection methods in the literature follow generally either a grid-cell- or a 3-D point-cloud-analysis-based approach (Yao and Stilla, 2011). In the first group of techniques (Rakusz et al., 2004, Yang et al., 2011), the obtained LiDAR data is first transformed into a dense 2.5-D Digital Elevation Model (DEM), thereafter established image processing operations can be adopted to extract the vehicles. On the other hand, in point cloud based methods (Yao et al., 2011), the feature extraction and recognition steps work directly on the 3-D point clouds: in this way we avoid loosing information due to projection and interpolation, however time and memory requirement of the processing algorithms may be higher.

Another important factor is related to the types of measurements utilized in the detection. A couple of earlier works combined multiple data sources, e.g. (Toth and Grejner-Brzezinska, 2006) fused LiDAR and digital camera inputs. Other methods rely purely on geometric information (Yao et al., 2010, Yang et al., 2011), emphasizing that these approaches are independent on the availability of RGB sensors and limitations of image-to-point-cloud registration techniques. Several LiDAR sensors, however, provide an intensity value for each data point, which is related to the intensity of the given laser return. Since in general the shiny surfaces of car bodies result in higher intensities, this feature can be utilized as an additional evidence for extracting the vehicles.

The vehicle detection techniques should also be examined from the point of view of object recognition methodologies. Machine learning methods offer noticeable solutions, e.g. (Yang et al., 2011) adopts a cascade AdaBoost framework to train a classifier based on edgelet features. However, the authors also mention that it is often difficult to collect enough representative training samples, therefore, they generate more training examples by shifting and rotating the few training annotations. Model based methods attempt to fit 2-D or 3-D car models to the observed data (Yao et al., 2011), however, these approaches may face limitation for scenarios where complex and highly various vehicle shapes are expected.

We can also group the existing object modeling techniques whether they follow a *bottom-up* or an *inverse* approach. The *bottom-up* techniques usually consist in extracting *primitives* (blobs, edges, corners etc.) and thereafter, the objects are constructed from the obtained features by a sequential process. To extract the vehicles, (Rakusz et al., 2004) introduces three different methods with similar performance results, which combine surface warping, Delaunay triangulation, thresholding and Connected Component Analysis (CCA). As main bottlenecks here, the Digital Terrain Model (DTM) estimation and appropriate height threshold selection steps critically influence the output quality. (Yao et al., 2010) applies three consecutive steps: geo-tiling, vehicle-top detection by local maximum filtering and segmentation through marker-controlled watershed transformation. The output is a set of vehicles contours, however, some car silhouettes are only partially extracted and a couple of neighboring objects are merged into the same blob. In general, bottom-up techniques can be relatively fast, however construction of appropriate primitive filters may be difficult/inaccurate, and in the sequential work flows, the

Parameter	Domain	Description
x_p, y_p, z_p	$\mathbb{R}^3$	coordinates of the 3-D geometric location of the point p
g_p	[0,255]	intensity (or gray level) value associated to the point p
n_p	$\{1,2,3,4\}$	number of echoes (or returns) from the direction of p
r_p	$\{1,2,3,4\}$	index (ordinary number) of the echo associated to point p from its direction (i.e. $r_p \leq n_p$)

Table 1: Parameters associated to a point p of the input cloud $\mathcal{L}$

failure each step may corrupt the whole process. In addition, we have limited options here to incorporate a priori information (e.g. shape, size) and object interaction.

Inverse methods, such as Marked Point Processes, MPPs, (Benedek et al., 2012, Descombes et al., 2009), assign a fitness value to each possible object configuration, thereafter an optimization process attempts to find the configuration with the highest confidence. In this way complex object appearance models can be used, it is easy to incorporate prior shape information (e.g. only searching among rectangles) and object interactions (e.g. penalize intersection, favor similar orientation). However, high computational need is present due searching in the high dimension population space. Therefore, applying efficient optimization techniques is a crucial need.

In this paper, we propose an MPP based vehicle detection method with the following key features. (i) Instead of utilizing complex image descriptors and machine learning techniques to characterize the individual vehicle samples, only basic radiometric evidences, segmentation labels and prior knowledge about the approximate size and height of the vehicle bounding boxes are exploited. (ii) We model interaction between the neighboring vehicles by prescribing prior non-overlapping, width similarity and favored alignment constraints. (iii) Features exploited in the recognition process are directly derived from the segmentation of the LiDAR point cloud in 3-D. However, to keep the computational time tractable, the optimization of the inverse problem is performed in 2-D, following a ground projection of the previously obtained class labels. (iv) During the projection of the LiDAR point cloud to the ground (i.e. a regular image), we do not interpolate pixel values with missing data, but include in the MPP model the concept of *pixel with unknown class*. In this way we avoid possible artifacts of data interpolation.

2 POINT CLOUD SEGMENTATION

The input of the proposed framework is a LiDAR point cloud $\mathcal{L}$. Let us assume that the cloud consists of l points: $\mathcal{L} = \{p_1, \ldots, p_l\}$, where each point, $p \in \mathcal{L}$, is associated to geometric position, intensity and echo number parameters, as detailed in Table 1.

The point cloud segmentation part consists of three steps. First, a density based clustering technique is adopted to remove clutter points (i.e. points not belonging to connected regions, like most reflections from walls), and vegetation is filtered out by using return number information. Let us denote by $\mathcal{V}_\epsilon(p)$ the ϵ neighborhood of p:

$$\mathcal{V}_\epsilon(p) = \{q \in \mathcal{L} : ||q - p|| < \epsilon\},$$

where $||r - p||$ marks the Euclidean distance of points r and p. Then with using $|\mathcal{V}_\epsilon(p)|$ for the cardinality of a neighborhood:

$$\mu(p) = \text{clutter iff } |\mathcal{V}_\epsilon(p)| < \tau_\mathcal{V},$$

where ϵ and $\tau_\mathcal{V}$ threshold parameters depend on the point cloud resolution and density. For efficient neighborhood calculation, we need to divide the point cloud into smaller parts by making a nonuniform subdivision of the 3-D space using a *k*-d tree data structure.

For estimating the vegetation, we utilize that trees and bushes cause multiple laser returns:

$$\mu(p) = \text{vegetation iff } r_p < n_p$$

Note that this step removes some points of car and buiding edges where the echo number is bigger than one, but we experienced that these regions do not significantly corrupt the vehicle separation process. We denote by $\mathcal{L}_{\rm cv} \subset \mathcal{L}$ the points labeled as clutter or vegetation:

$$\mathcal{L}_{\rm cv} = \{p \in \mathcal{L} : \mu(p) \in \{\text{clutter}, \text{vegetation}\}\}.$$

Second, we identify the ground points, by estimating the the best plane $\mathcal{P}$ in the cloud $\mathcal{L} \setminus \mathcal{L}_{\rm cv}$ using the RANSAC-based algorithm of (Yang and Foerstner, 2010). This technique selects in each iteration three points randomly from the input cloud, and it calculates the parameters of the corresponding plane. Then it counts the points in $\mathcal{L} \setminus \mathcal{L}_{\rm cv}$ which fit the new plane and compares the obtained result with the last saved one. If the new result is better, the estimated plane is replaced with the new candidate. The process is iterated till convergence is obtained. Note that since the ground is usually not planar in a greater area, large point clouds should be divided into smaller segment, and the ground plane is estimated within each segment separately. Next, we label the point $p \in \mathcal{L} \setminus \mathcal{L}_{\rm cv}$ as ground as:

$$\mu(p) = \text{ground iff } d(p, \mathcal{P}) < \tau_\mathcal{P},$$

where $d(p, \mathcal{P})$ denotes the distance of point p from plane $\mathcal{P}$, and the $\tau_\mathcal{P}$ threshold depends on the geometric accuracy of the LiDAR data. We denote by $\mathcal{L}_{\rm gr} = \{p \in \mathcal{L} : \mu(p) = \text{ground}\}$

Third, for the remaining points in $\mathcal{L} \setminus (\mathcal{L}_{\rm cv} \cup \mathcal{L}_{\rm gr})$, a floodfill-based segmentation algorithm is propagated, which aims to detect the large connected building roofs. We mark the points selected by the algorithm with label 'roof', and compose the set: $\mathcal{L}_{\rm rf} = \{p \in \mathcal{L} : \mu(p) = \text{roof}\}$. Meanwhile, the points of the remaining blobs of the cloud are labeled as vehicle candidates, if their height coordinate is less than the maximal vehicle height:

$$\mu(p) = \text{vehicle iff } p \in \mathcal{L} \setminus (\mathcal{L}_{\rm cv} \cup \mathcal{L}_{\rm gr} \cup \mathcal{L}_{\rm rf}) \text{ AND } z_p < h_{\max}. \quad (1)$$

To make the tuning of $h_{\max}$ less critical for the process, we used an overestimation of the possible vehicle heights. In this way we exclude obvious outliers, such as traffic lights, while further false possible points in the vehicle candidate set (denoted by $\mathcal{L}_{\rm vl}$) should be eliminated in a later step. Finally, points in $\mathcal{L}$ not clustered yet are merged into the clutter class.

After the 3-D segmentation process, we stretch a 2-D pixel lattice S (i.e. an image) onto the ground plane, where $s \in S$ denotes a single pixel. Then, we project each LiDAR point to this lattice, which has a label of ground, vehicle or building roof: $L^\star = \mathcal{L}_{\rm gr} \cup \mathcal{L}_{\rm vl} \cup \mathcal{L}_{\rm rf}$. This projection results in a 2-D class label map and an intensity map, where multiple point projections to the same pixel are handled by a point selection algorithm, which gives higher precedence to vehicle point candidates. On the other hand, the projection of the sparse point cloud to a regular image lattice results in many pixels with undefined class labels and intensities. In contrast to several previous solutions, we do not interpolate these missing points, but include in the upcoming model

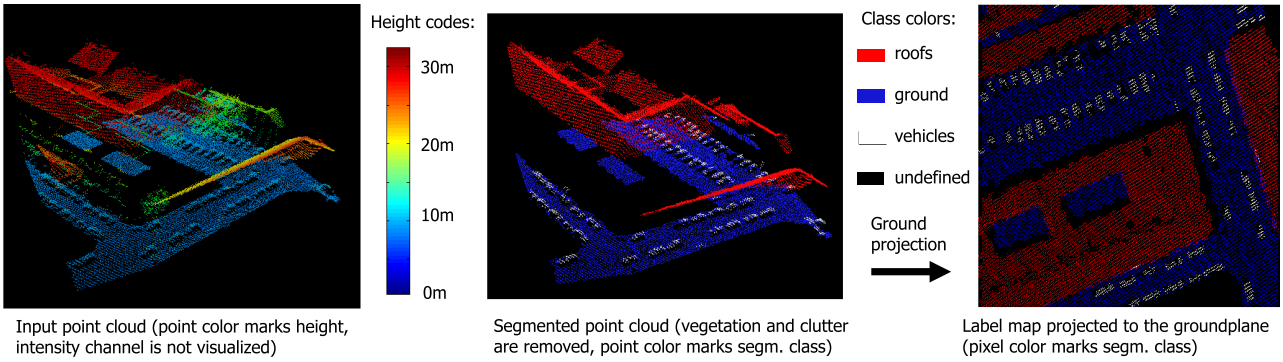


Figure 1: Workflow of the point cloud filtering, segmentation and projection steps. Test data provider: Astrium GEO-Inf. Services - Hungary©

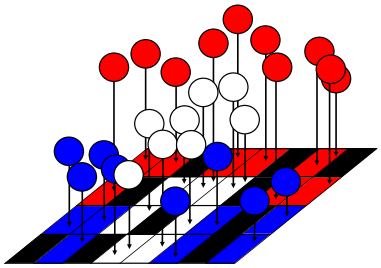


Figure 2: Demonstration of the projection step (best viewed in color). LiDAR points are denoted by spheres, and pixels on the image lattice by cells, with the following color codes: red - roof, blue - ground, white - vehicle. Roof and ground pixels represent the *background* class in the lattice, while black cells correspond to pixels with class label *undefined*.

the concept of unknown label at certain pixels. In this way, our approach is not affected by the artifacts of data interpolation.

Let us denote by $\chi(s) \subset \mathcal{L}^\star$ the set of points of $\mathcal{L}^\star$ projected to pixel s. After the projection (Fig. 2), we distinguish vehicle, background and undefined classes on the lattice as follows:

$$\nu(s) = \begin{cases} \text{vehicle} & \text{if } \exists p \in \chi(s) : \mu(p) = \text{vehicle} \\ \text{background} & \text{if } \forall p \in \chi(s) : \begin{cases} \mu(p) = \text{roof} \\ \quad \text{OR} \\ \mu(p) = \text{ground} \end{cases} \\ \text{undefined} & \text{if } \chi(s) = \emptyset. \end{cases}$$

Note that for easier visualization, in Fig. 1 and 2 we have distinguished pixels of roof (red) and ground (blue) projections, but during the next steps, we consider them as part of the background class. We also assign to each pixel s and intensity value $g(s)$, which is 0, if $\nu(s) = \text{undefined}$, otherwise we take the average intensity of points projected to s.In the following part of the algorithm, we purely work on the previously extracted label and intensity images. The detection is mainly based on the label map, but additional evidences are extracted from the intensity image, where several cars appear as salient bright blobs due to their shiny surfaces.

3 MARKED POINT PROCESS MODEL

The inputs of this step are the label and intensity maps over the pixel lattice S, which were extracted in the previous section. We will also refer to the input data jointly by $\mathcal{D}$. We assume that each vehicle from top view can be approximated by a rectangle, which we aim to extract by the following model. A vehicle candidate u is described by five parameters: c_x and c_y center coordinates, e_L, e_l side lengths and $\theta \in [-90°, +90°]$ orientation (Fig. 3(c)). The vehicle population of the scene is described by a configuration of an unknown number of rectangles, which is realized by a Marked Point Process (MPP) (Descombes et al., 2009). Note that with replacing the rectangle shapes for parallelograms, the "shearing effect" of moving vehicles may also be modeled (Yao and Stilla, 2011), but in the considered test data this phenomenon could not be reliably observed.

Let $\mathcal{H}$ be the space of u objects. The Ω configuration space is defined as:

$$\Omega = \bigcup_{n=0}^{\infty} \Omega_n, \quad \Omega_n = \left\{ \{u_1, \ldots, u_n\} \in \mathcal{H}^n \right\}.$$

Denote by ω an arbitrary object configuration $\{u_1, \ldots, u_n\}$ in Ω. We define a neighborhood relation $\sim$ in $\mathcal{H}$: $u \sim v$ iff the distance of the object centers is smaller than a threshold. The neighborhood of u in configuration ω is defined as $\mathcal{N}_u(\omega) = \{v \in \omega | u \sim v\}$ (hereafter, we ignore ω in the notation).

Taking an inverse modeling approach, an energy function $\Phi_{\mathcal{D}}(\omega)$ is defined on the object configuration space, which evaluates the negative fitness of each possible vehicle population. Thereafter, we search for the configuration estimate which exhibits the Minimal Energy (ME): $\omega_{\text{ME}} = \text{argmin}_{\omega \in \Omega} \left[\Phi_{\mathcal{D}}(\omega) \right]$. $\Phi_{\mathcal{D}}(\omega)$ can be decomposed into subterms, which are defined on the the $\mathcal{N}-$neighborhoods of each object in ω:

$$\Phi_{\mathcal{D}}(\omega) = \sum_{u \in \omega} \Psi_{\mathcal{D}}(u, \mathcal{N}_u).$$

The above neighborhood-energies are constructed by fusing various data terms and prior terms, as introduced in the following subsections in details.

3.1 Data-dependent energy terms

Data terms evaluate the proposed vehicle candidates (i.e. the $u = \{c_x, c_y, e_L, e_l, \theta\}$ rectangles) based on the input label- or intensity maps, but independently of other objects of the population. The data modeling process consists of two steps. *First*, we define different $f(u) : \mathcal{H} \to \mathbb{R}$ features which evaluate a vehicle hypothesis for u in the image, so that 'high' $f(u)$ values correspond to efficient vehicle candidates. In the *second step*,

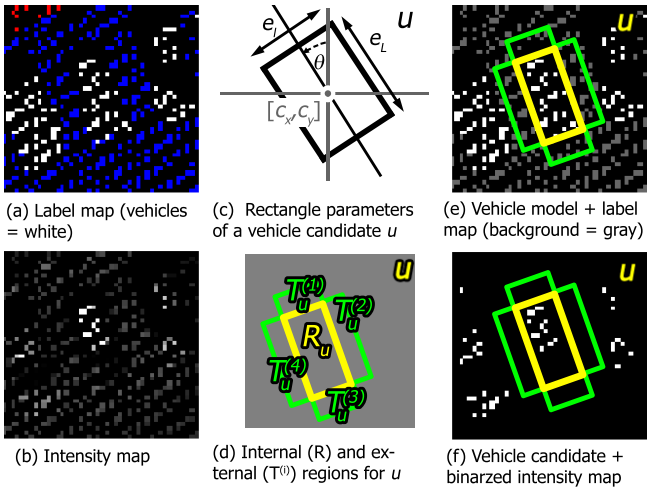


Figure 3: Demonstration of the (a)-(b) input maps (c) object rectangle parameters and (d)-(f) dataterm calculation process

we construct $\varphi_d^f(u)$ *data driven* energy subterms for each feature f, by attempting to satisfy $\varphi_d^f(u) < 0$ for real objects and $\varphi_d^f(u) > 0$ for false candidates. For this purpose, we project the feature domain to $[-1, 1]$ with a monotonously decreasing function: $\varphi_d^f(u) = \mathcal{Q}\big(f(u), d_0^f\big)$, where

$$\mathcal{Q}(x, d_0) = \begin{cases} \left(1 - \frac{x}{d_0}\right), & \text{if } x < d_0 \\ \exp\left(-\frac{x-d_0}{0.1}\right) - 1, & \text{if } x \geq d_0. \end{cases} \quad (2)$$

Observe that the $\mathcal{Q}$ function has a key parameter, d_0^f, which is the object acceptance threshold for feature f: u is acceptable according to the $\varphi_d^f(u)$ term iff $f(u) > d_0^f$.

We used four different data-based features. To introduce them, let us denote by $R_u \subset S$ the pixels of the image lattice lying inside the u vehicle candidate's rectangle, and by T_u^{up}, T_u^{bt}, T_u^{lt}, and T_u^{rg} the upper, bottom, left and right object neighborhood regions, respectively (see Fig. 3). The feature definitions are listed in the following paragraphs.

The *vehicle evidence* feature $f^{\text{ve}}(u)$ expresses that we expect several pixels classified as vehicle within R_u:

$$f^{\text{ve}}(u) = \frac{1}{|R_u|} \sum_{s \in R_u} \mathbf{1}\left\{\nu(s) = \text{vehicle}\right\},$$

where $|R_u|$ denotes the cardinality of R_u, and $\mathbf{1}\{.\}$ marks an indicator function: $\mathbf{1}\{\text{true}\} = 1$, $\mathbf{1}\{\text{false}\} = 0$.

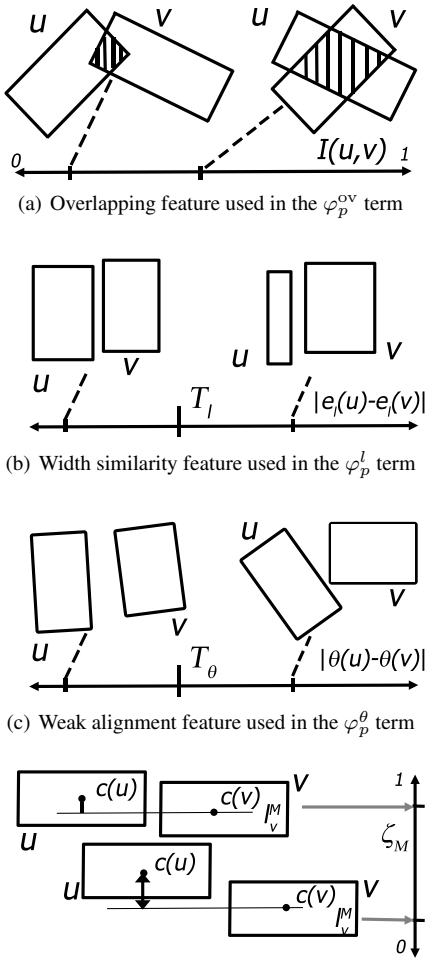
The *external background* feature $f^{\text{eb}}(u)$ measures if the vehicle candidate is surrounded by background regions:

$$f^{\text{eb}}(u) = \underset{i \in \{\text{up,bt,lt,rg}\}}{\text{min2nd}} \left(\frac{1}{|T_u^i|} \sum_{s \in T_u^i} \mathbf{1}\left\{\nu(s) = \text{background}\right\} \right),$$

where the min2nd operator returns the second smallest element from the background filling ratios of the four neighboring regions: with this choice we also accept vehicles which connect with at most one side to other vehicles or undefined regions.

The *internal background* feature $f^{\text{ib}}(u)$ prescribes that within R_u only very few background pixels may occur:

$$f^{\text{ib}}(u) = \frac{1}{|R_u|} \sum_{s \in R_u} 1 - \mathbf{1}\left\{\nu(s) = \text{background}\right\}.$$



(a) Overlapping feature used in the φ_p^{ov} term

(b) Width similarity feature used in the φ_p^{l} term

(c) Weak alignment feature used in the φ_p^{θ} term

(d) Strong alignment feature used in the φ_p^{at} term

Figure 4: Demonstration of the prior constraints used in the proposed model

Demonstration of the f^{ve}, f^{eb} and f^{ib} feature calculation can be followed in Fig. 3(e).

Finally, the *intensity* feature provides additional evidence for image parts containing high intensity regions (see Fig. 3(b) and (f)).

$$f^{\text{it}}(u) = \frac{1}{|R_u|} \sum_{s \in R_u} \mathbf{1}\left\{g(s) > T_g\right\},$$

where T_g is an intensity threshold.

After the feature definitions, the data terms $\varphi_d^{\text{it}}(u)$, $\varphi_d^{\text{ve}}(u)$, $\varphi_d^{\text{ib}}(u)$, $\varphi_d^{\text{eb}}(u)$ can be calculated with the $\mathcal{Q}$ function by appropriately fixing the corresponding d_0^f parameters for each feature.

3.2 Prior terms

In contrast to the data-energy functions, the prior terms evaluate a given configuration on the basis of prior geometric constraints, but independently of the input label and intensity maps. We used three types of prior terms in the model, realizing *non-overlapping*, *width similarity* and *alignment* (*weak & strong*) constraints between different objects.

First, we have to avoid configurations which contain multiple objects in the same or strongly overlapping positions. Therefore, measure an overlapping coefficient $I(u, v)$, which penalizes intersection between different object rectangles (see Fig. 4(a)):

$I(u,v) = 2 \cdot |R_u \cap R_v|/(|R_u| + |R_v|)$, and derive the overlapping term as:

$$\varphi_p^{\mathrm{ov}}(u, \mathcal{N}_u) = \sum_{v \in \mathcal{N}_u} I(u,v).$$

Second, to prevent us from merging contacting vehicles into the same object candidate, we penalize rectangles with significantly different width (e_l) parameters in local neighborhoods (Fig. 4(b)):

$$\varphi_p^{l}(u, \mathcal{N}_u) = \frac{1}{|\mathcal{N}_u|} \sum_{v \in \mathcal{N}_u} \mathbf{1}\left\{|e_l(u) - e_l(v)| > T_l\right\}.$$

We set T_l as the half of the average vehicle width.

Third, we favor, if objects in a local neighborhood are aligned i.e. they form regular lines or rows. This later effect can be often observed either by parking cars, or by vehicles waiting at crossroads or in traffic jams. Note that the alignment assumptions cannot be used as hard constraints, since we should always expect some irregularly oriented vehicles in the scene. However, we propose to reward the object groups meeting the alignment criterion, in two ways. *On one hand*, we moderately favor, if the orientation of u is similar to most of its neighbors; and moderately penalize, if not (Fig. 4(c)):

$$\varphi_p^{\theta}(u, \mathcal{N}_u) = \gamma_\theta \cdot \left(\frac{1}{|\mathcal{N}_u|} \sum_{v \in N_u} \mathbf{1}\left\{|\theta(u) - \theta(v)| > T_\theta\right\} - 0.5 \right),$$

with a small $0 < \gamma_\theta$ weight. We used $T_\theta = 40^\circ$ and $\gamma_\theta = 0.1$.

On the other hand, we strongly favor, if the central point of u (denoted by $c(u)$) is close either to the major (l_v^M), or to the minor (l_v^m) axis lines of its neighbors $v \in \mathcal{N}_u$. We shall consider here cases when vehicles park or run parallel or perpendicular to the road side. The corresponding energy term is obtained as follows. $\varphi_p^{\mathrm{at}}(u, \mathcal{N}_u) = 0$ if $|\mathcal{N}_u| < N_{\min}$, otherwise:

$$\varphi_p^{\mathrm{at}}(u, \mathcal{N}_u) = \frac{1}{|\mathcal{N}_u|} \cdot \max\Big(\sum_{v \in N_u} \mathbf{1}\left\{\zeta_M(u,v) < T_M\right\}, \sum_{v \in N_u} \mathbf{1}\left\{\zeta_m(u,v) < T_m\right\} \Big)$$

where $\zeta_M(u,v)$ (resp. $\zeta_m(u,v)$) is the normalized distance of $c(u)$ and l_v^M (resp. l_v^m) as shown in Fig. 4(d). T_M and T_m depend on the resolution of the lattice, and we used $N_{\min} = 4$. Note that the fulfillment of the axis alignment criterion is not necessary, however, if it is satisfied, we give further rewards as explained in the next section.

3.3 Integration of the energy components

As introduced before, the data energy terms provide different feature based conditions for the acceptance of the vehicle candidates, while the prior terms penalize/favor given configurations based on preliminary expectations about geometry and object interactions. In general, we prescribe that the vehicles satisfy each of the four feature constraints from Sec. 3.1 (i.e. all energy subterms are negative), therefore, we derive the joint data term (first row of (3)) by the maximum operator, which is equivalent to the logical AND in the negative fitness domain. However, if the axis distance criterion is satisfied ($\varphi_p^{\mathrm{at}}(u, \mathcal{N}_u) > 0.5$), we are less strict regarding the data terms, and only investigate the internal and external background energy parts (see eq. (4)). Finally, we use the remaining prior energy functions, as additive terms in $\Psi_{\mathcal{D}}$ (second row of (3)). Based on these arguments, the local object energies are calculated by the following formula:

$$\begin{aligned} \Psi_{\mathcal{D}}(u, \mathcal{N}_u) = &\max\Big(\varphi_d^{\mathrm{ib}}(u), \varphi_d^{\mathrm{eb}}(u), \Upsilon_{\mathcal{D}}(u, \mathcal{N}_u)\Big) + \\ &+ \varphi_p^{\mathrm{ov}}(u, \mathcal{N}_u) + \varphi_p^{\theta}(u, \mathcal{N}_u) + \varphi_p^{l}(u, \mathcal{N}_u), \end{aligned} \tag{3}$$

where the $\Upsilon_{\mathcal{D}}$ term is responsible for considering or avoiding the f^{it} and f^{ve} features, depending on φ_p^{at}:

$$\begin{aligned} &\Upsilon_{\mathcal{D}}(u, \mathcal{N}_u) = \\ &= \min\Big(1 - 2 \cdot \mathbf{1}\big\{\varphi_p^{\mathrm{at}}(u, \mathcal{N}_u) > 0.5\big\}, \max\big(\varphi_d^{\mathrm{it}}(u), \varphi_d^{\mathrm{ve}}(u)\big)\Big). \end{aligned} \tag{4}$$

4 OPTIMIZATION

We estimate the optimal object configuration by the Multiple Birth and Death Algorithm (Descombes et al., 2009) as follows:

Initialization: start with an empty population $\omega = \emptyset$.

Main program: set the birth rate b_0, initialize the inverse temperature parameter $\beta = \beta_0$ and the discretization step $\delta = \delta_0$, and alternate birth and death steps.

1. *Birth step*: Visit all pixels on the image lattice S one after another. At each pixel s, if there is no object with center s in the current configuration ω, choose birth with probability δb_0.

   If birth is chosen at s: generate a new object u with center $[c_x(u), c_y(u)] := s$, and set the e_L, e_l and θ parameters randomly. Finally, add u to the current configuration ω.

2. *Death step*: Consider the actual configuration of objects $\omega = \{u_1, \ldots, u_n\}$ and sort it by decreasing values depending on the data term. For each object u taken in this order, compute $\Delta\Phi_\omega(u) = \Phi_{\mathcal{D}}(\omega/\{u\}) - \Phi_{\mathcal{D}}(\omega)$, derive the *death rate* as follows:

   $$d_\omega(u) = \frac{\delta a_\omega(u)}{1 + \delta a_\omega(u)}, \quad \text{with} \quad a_\omega(u) = e^{-\beta \cdot \Delta\Phi_\omega(u)},$$

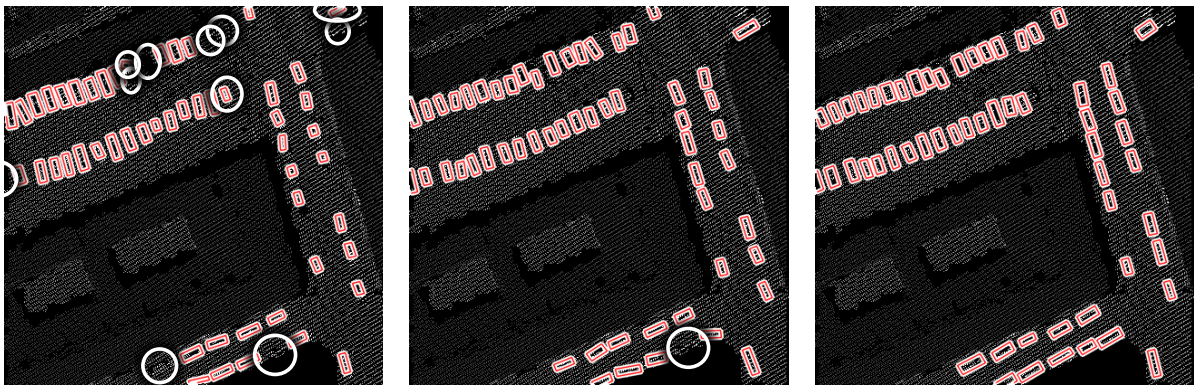
   and remove u from ω with probability $d_\omega(u)$.

Convergence test: if the process has not converged yet, increase the inverse temperature β and decrease the discretization step δ with a geometric scheme, and go back to the birth step. The convergence is obtained when all the objects added during the birth step, and only these ones, have been killed during the death step.

5 EVALUATION AND PARAMETER SETTINGS

We evaluated our method in four aerial LiDAR data sets (provided by Astrium GEO-Inf. Services - Hungary), which are captured above dense urban areas. For accurate Ground Truth (GT) generation, we have developed an accessory program with graphical user interface, which enables us to manually create and edit a GT configuration of rectangles, which can be compared to the output of the algorithm.

As for parameter settings, the data term thresholds were set based on a limited number of training samples (around 10% of the vehicles in each test set), using similar Maximum-Likelihood strategies to (Benedek et al., 2012). The prior term parameters, which prescribe the significance of the object interaction constraints, must be determined by the user: our applied values have been



(a) DEM-PCA method (b) Proposed MPP method (c) Ground Truth

Figure 5: Comparison of the detection results with the DEM-PCA model and the Proposed MPP method, for the point cloud segment marked as Set#1 in Table 2. Circles denote missing or false objects.

Data Set	NV	DEM-PCA		Prop. MPP	
		MO	FO	MO	FO
Set#1	57	8	3	1	0
Set#2	31	3	4	6	0
Set#3	18	1	5	3	0
Set#4	14	1	2	1	1
Overall	120	13	14	11	1
Overall F-rate		88%		95%	

Table 2: Numerical comparison of the detection results obtained by the DEM-PCA and the proposed MPP models. Number of Vehicles (NV), Missing Objects (MO) and False Objects (FO) are listed for each data set, also and in aggregate.

given in Sec. 3.2. Regarding the MBD optimization settings, we followed the guidelines from (Descombes et al., 2009).

To perform quantitative evaluation, we have measured how many vehicles are correctly or incorrectly detected in the different test sets, by counting the Missing Objects (MO), and the Falsely detected Objects (FO). These values are compared to the Number of real Vehicles (NV), and the F-rate of the detection (harmonic mean of precision and recall) is also calculated.

For comparison, we have selected a *bottom-up* grid-cell-based algorithm from (Rakusz et al., 2004), called later as *DEM-PCA*, which consists of three consecutive steps: (1) Height map (or *D*igital *E*levation *M*odel) generation by ground projection of the elevation values in the LiDAR point cloud, and missing data interpolation. (2) Vehicle region detection by thresholding the height map followed by morphological connected component extraction. (3) Rectangle fitting to the detected vehicle blobs by *P*rincipal *C*omponent *A*nalysis.

Some qualitative results are shown in Fig. 5, and the quantitative evaluation is provided in Table 2. The proposed MPP model surpasses the DEM-PCA method by 7% regarding the F-rate, due to the fact that our method results in significantly less false objects. We can also observe in Fig. 5 that the vehicle outlines obtained with the MPP model are notably more accurate.

6 CONCLUSION

This paper has proposed a novel MPP based vehicle extraction method for aerial LiDAR point clouds. The efficiency of the approach has been tested with real-world LiDAR measurements, and its advantages versus a reference method has been demonstrated. The authors would like to thank Astrium GEO-Information Services - Hungary for test data provision. This work was supported by the Hungarian Research Fund (OTKA #101598), the APIS Project of EDA and the i4D Project of MTA SZTAKI. The second author was partially funded by the János Bolyai Research Scholarship of the Hungarian Academy of Sciences.

REFERENCES

Benedek, C., Descombes, X. and Zerubia, J., 2012. Building development monitoring in multitemporal remotely sensed image pairs with stochastic birth-death dynamics. IEEE Trans. Pattern Anal. Mach. Intell. 34(1), pp. 33–50.

Descombes, X., Minlos, R. and Zhizhina, E., 2009. Object extraction using a stochastic birth-and-death dynamics in continuum. J. Mathematical Imaging and Vision 33, pp. 347–359.

Rakusz, Á., Lovas, T. and Barsi, Á., 2004. Lidar-based vehicle segmentation. International Archives of Photogrammetry and Remote Sensing XXXV(2), pp. 156–159.

Toth, C. and Grejner-Brzezinska, D., 2006. Extracting dynamic spatial data from airborne imaging sensors to support traffic flow estimation. ISPRS Journal of Photogrammetry and Remote Sensing 61(3-4), pp. 137 – 148.

Tuermer, S., Leitloff, J., Reinartz, P. and Stilla, U., 2010. Automatic vehicle detection in aerial image sequences of urban areas using 3D HoG features. In: ISPRS Photogrammetric Computer Vision and Image Analysis, Paris, France, p. B:50.

Yang, B., Sharma, P. and Nevatia, R., 2011. Vehicle detection from low quality aerial LIDAR data. In: IEEE Workshop on Applications of Computer Vision (WACV), pp. 541 –548.

Yang, M. Y. and Foerstner, W., 2010. Plane Detection in Point Cloud Data. Technical Report TR-IGG-P-2010-01, Department of Photogrammetry, University of Bonn.

Yao, W. and Stilla, U., 2011. Comparison of two methods for vehicle extraction from airborne lidar data toward motion analysis. IEEE Geoscience and Remote Sensing Letters 8(4), pp. 607–611.

Yao, W., Hinz, S. and Stilla, U., 2010. Automatic vehicle extraction from airborne lidar data of urban areas aided by geodesic morphology. Pattern Recogn. Letters 31(10), pp. 1100 – 1108.

Yao, W., Hinz, S. and Stilla, U., 2011. Extraction and motion estimation of vehicles in single-pass airborne lidar data towards urban traffic analysis. ISPRS Journal of Photogrammetry and Remote Sensing 66, pp. 260–271.