Online algorithms for classification of urban objects in 3D point clouds[*]

Ioannis Stamos
Hunter College of CUNY
New York, NY 10065
http://www.cs.hunter.cuny.edu/~ioannis

Olympia Hadjiliadis
Brooklyn College of CUNY
Brooklyn, NY 11210
ohadjiliadis@brooklyn.cuny.edu

Hongzhong Zhang
Columbia University
New York, NY 10027
http://www.columbia.edu/~hz2244

Thomas Flynn
Graduate Center of CUNY
New York, NY 10016
tflynn@gc.cuny.edu

3DIMPVT Conference, ETH Zurich, Oct. 13-15 2012

Abstract

The current technology in stationary laser range-scanning enables high-resolution acquisition of 3D data in a sequential fashion. Traditionally, range scans are processed offline after acquisition, which significantly slows down the procedure. In this work we alleviate this limitation by developing low-complexity, online detection and classification algorithms. These algorithms are innovative in that they classify points into 5 distinct classes (vegetation, vertical, horizontal, car and curb regions) and robustly determine the level of the ground without requiring any prior training or parameter estimation. To construct these algorithms we extract cleverly chosen summary statistics which significantly reduce the dimensionality of the data. This reduction enables us to contrast the different classes by appropriately chosen Markov models and then use online techniques to detect a transition from one Markov model to the other. The identification of the ground level is further achieved by taking advantage of statistical properties of the distribution of the summary statistics. Our algorithms also use contextual cues to verify the existence of specific classes of objects. All our algorithms take advantage of the sequential nature of data acquisition by running in parallel and labeling points on-the-fly. Thus, these algorithms can be potentially integrated with the scanner's hardware and provide the foundation for the construction of high-resolution 3D data scanners that classify data as acquired. We have run experiments using complex urban range scans and have evaluated the classification accuracy against ground-truth.

1. Introduction

The photorealistic modeling of large-scale scenes, such as urban structures, has received significant attention in recent years (see for example [10]). The current state of the art includes the collection of high-resolution point-clouds from laser scanners. The abundance of high-resolution 3D data opens the door for new processing algorithms. Thus, the ability to segment and classify objects of interest in large-scale urban scenes efficiently and accurately is of major importance. In this work we present algorithms that allow online classification of objects as data is acquired by the sensor, enabling the efficient processing of voluminous amounts of data.

In the existing literature (see Sec. 2) point-cloud segmentation and classification algorithms use all the acquired data in order to split the scene into major surfaces. In a real-time application though decisions have to be made instantly. In these cases we need to classify as fast as possible. Thus, sequential classification techniques become relevant [2, 12]. The first attempt to achieving on-the-fly classification into vegetation, vertical and horizontal surfaces is done in [7]. Although this classification is achieved fairly accurately, it is only a preliminary step to identify the variety of objects that may arise in an urban scene. In particular, the classification of points into only three classes results in a coarse depiction of reality in which parts of a car, for example, maybe classified as horizontal or vertical surfaces or in some instances as vegetation.

In this paper we develop innovative online algorithms of low-complexity that run in parallel and achieve a far more detailed classification of objects into horizontal surfaces, vertical surfaces, curbs, cars and vegetation. We are further able to identify the level of the ground within the class of horizontal surfaces. Although achieving this detail appears as a marginal improvement from the previous classification, this task is highly non-trivial. Cars, for instance, are characterized by a level of variability and irregularity in surface that is lesser than what determines a vegetation region and is

[*]This work has been supported in part by the following NSF grants: IIS-0915971, CCF-0916452 and MRI CNS-0821384. We would like to thank Matthew Evanusa for labeling some of the datasets.

more than what determines a regular surface, whether horizontal or vertical. Moreover, car detection is made even more complex because of the existence of a large number of missing data caused by windows and metallic surfaces. Therefore, achieving a more detailed classification requires the extraction of more information from the point cloud of the urban scene, such as the level of the ground and the curbs, which serve as important contextual cues. To be more specific, a fire-hydrant or a newsstand sits behind a curb while cars do not. It is important to stress that our algorithms, unlike many others in our field, require no learning or prior parameter estimation.

2. Related work

There is a variety of range image segmentation techniques (for example [5]). These methods do not associate any classes with the extracted segments and process the data offline. There is also literature on the topic of classification of 3D point-clouds using Markov network models [1, 4, 9, 16]. [11] concentrates specifically on the detection of cars from range images. The paper uses spin images and extended Gaussian images and produces very good results. [6] also presents offline techniques for the segmentation and detection of various types of objects. These techniques assume that the data becomes available all at once, as opposed to sequentially. They also require training. Online detection techniques have been used in 3D computer vision mainly in the context of a moving sensor [14, 15], as opposed to a steady high-resolution laser-scanner. The goal is to separate between two states: drivable vs. non-drivable terrain. In [15] a hidden Markov model is used to achieve this goal. Finally, [13] describes an almost (but not exactly) real-time approach.

Our **contribution** with respect to earlier work is that we are able to achieve a far more detailed classification of data on-the-fly. This is achieved by using cleverly selected summary statistics that arise from geometrical considerations and which significantly reduce the dimensionality of the data. These statistics are sequentially computed angles, which are then appropriately analyzed and modeled to make an inference. None of the analysis carried out requires any training as it relies either on the contrast between models and the detection of change from one to the other or on the subsequent estimations of the shape of distributions of the statistics.

3. Summary Statistics and Algorithms

The scanner, placed on a steady platform, takes measurements of distance to the closest surface *sequentially* by emitting a laser beam. Note that it is also possible that no distance is going to be measured when the laser hits transparent or highly specular surfaces, or when the measured point is at a distance bigger than a threshold (300 meters in our setting). The main summary statistics used in our analysis are (a) Signed angles and (b) Line angles.

To describe the signed angles let us denote by $X_{i,k} = [x_{ik}, y_{ik}, z_{ik}]$ the vector of 3D coordinates of the k-th point in the i-th scanline. Knowledge of the vertical direction (axis **z**) is provided by many laser scanners, or can be easily acquired via hardware, or even computed from the data in urban scenes (line detection and clustering) and is thus assumed known. Most robotics application (for instance [15]) make this assumption as well. We now define $D_{i,k} = X_{i,k+1} - X_{i,k}$ (difference of two successive measurements in a given scanline i), and V_{ik}: the angle of the vector $D_{i,k}$ with the pre-determined **z** axis (0 to 180 degrees), $sV_{ik} = s_{ik} * V_{ik}$: the sign of the dot product between the vectors $D_{i,k}$ and $D_{i,k-1}$, multiplied by V_{ik} (signed angle). This sign is positive when the two vectors have the same orientation and negative otherwise. These statistics are used to achieve the coarser classification into horizontal, vertical and vegetation classes. They are also used to provide a first indication of the existence of cars in a given scan-line.

To describe the line angles let $X_A = [x_A, y_A, z_A]$ denote the position of the laser scanner and denote by θ the angle between successive laser beams. That is, θ is the angle between the vectors $X_{i,k} - X_A$ and $X_{i,k+1} - X_A$ for all $k = 1, 2, \ldots$ (θ is known). We then determine the sequence of angles $\phi_{i,k}$, for $k = 1, 2, \ldots$ by computing the angles between the vectors $D_{i,k}$ and $X_{i,k+1} - X_A$ for $k = 1, 2, \ldots$. In particular, using the law of sines we obtain

$$\frac{|X_{i,k+1} - X_A|}{|X_{i,k+2} - X_A|} = \frac{\sin(\phi_{i,k+1})}{\sin(\phi_{i,k})}. \quad (1)$$

We now notice that under the assumption that $D_{i,k}$ and $D_{i,k+1}$ are co-linear we have $\phi_{i,k+1} = \phi_{i,k} - \theta$ which leads to

$$\phi_{i,k} = \tan^{-1}\left\{ \frac{\sin\theta}{\cos\theta - \frac{|X_{i,k+1} - X_A|}{|X_{i,k+2} - X_A|}} \right\}, \quad (2)$$

for $k = 1, 2, \ldots$. We refer the reader to Fig. 1 for an illustration. We notice that if the assumption of co-linearity holds between more than two consecutive vectors $D_{i,k+n}$ and $D_{i,k+1+n}$ for all $n = 0, 1, 2, \ldots$, then $\phi_{i,k+n} = \phi_{i,k} - n\theta$. This leads us to the the line angle summary statistics $\hat{\phi}_k = \phi_{i,k} + (k-1)\theta$ for $k = 1, 2, \ldots$ within each scanline i, which are expected to be relatively close to each other if indeed co-linearity is to hold. Therefore, we use the statistical properties of the distribution of the line angles to determine the level of the ground (we assume that the ground is fairly linear), which together with the coarser classification achieved by inferential techniques developed using the signed angles and contextual truth, lead to the detailed classification into curbs, cars, vertical and horizontal surfaces. Our algorithms do not require any training. In order to achieve this we use the following contextual truth about urban scenes: **1.** Car

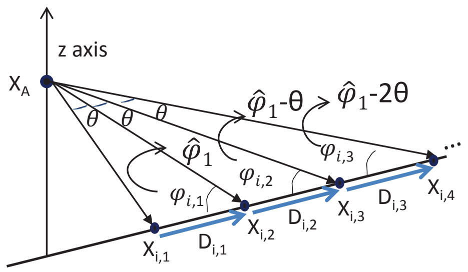

Figure 1. Four collinear points and line angles $\phi_{i,k} = \hat{\phi}_k-(k-1)\theta$ (see text).

objects are located within a finite height from the horizontal or inclined ground plane. **2.** Cars cannot be behind curbs or walls of buildings. **3.** There is a maximum height and length for cars and maximum height for curbs.

The online algorithms that we run in parallel are (1) the coarse classification algorithm, (2) the ground algorithm, (3) the curb algorithm and (4) the car algorithm. In the sections that follow we describe each of the above algorithms in detail.

4. The coarse classification algorithm

This algorithm is used to achieve a coarse classification into vegetation (T), horizontal surfaces (H) and vertical surfaces (V). It is described in full detail in [7] and is based on the fact that scene areas which include vegetation and trees produce a unique behavior in the sequence of measurements along each scanline. In vegetation areas the signed angle $sV_{i,k}$ measurements (see Sec. 3) alternate rapidly between negative (around $-90°$ degrees) and positive angles, while in horizontal and vertical surfaces they are stable on non-negative values (around $0°$ or $90°$ degrees). The detection algorithm of [7] detects vegetation by distinguishing between two hidden Markov models; the first one captures slow transitions between the $-90°$, $0°$ and $90°$ states and the second one fast transitions between the same states (each state is modeled by a Gaussian with mean $-90°$, $0°$, and $90°$ respectively). The distinction between horizontal and vertical surfaces is achieved through an online cumulative sum (CUSUM) algorithm [12] that detects a change in the mean of the signed angles $sV_{i,k}$ by $90°$.

5. The ground algorithm

The ground algorithm uses the coarse classification algorithm of Sec. 4 and consecutively computes angles $\hat{\phi}_k$ (see Sec. 3) to make an inference. It is divided into two parts (a & b) described below.

(Part a) The part that runs within each scanline i (begin with $i = 1$) which consists of the following steps. This part detects a sequence of horizontal points that could be potentially on the ground.

1. Find the first 10 consecutive points classified as horizontal H (skip missing points M). If such points cannot be found exit. Otherwise, calculate $\hat{\phi}_k$ for $k = 1, \dots, n$. Set $n = 9$.

2. Apply the mean-shift algorithm [3] to determine the number of modes in the distribution of $\hat{\phi}_k$ for $k = 1, \dots, n$.

3. If the number of modes from step 2 is one and the next point is classified as H then set $n = n + 1$, calculate $\hat{\phi}_n$ and go to step 2. If not, then stop and declare points $k = 1, \dots, n-1$ as potential ground (PG). These $n-1$ points have similar $\hat{\phi}_k$'s, meaning that they are likely to come from a linear surface. Also, we record the average of the 5 smallest (since ground is on a low-surface) z-coordinates of the points classified as (PG) and denote it by z_i. This is the estimate of the height of the ground.

Note that the above algorithm stops at the first instance it encounters a point of class V or T.

(Part b) The part that runs across scanlines. This part verifies potential ground points as actual ground and computes a robust estimate of the height of the ground. We start by running the algorithm of part (a) for the first N scanlines (N is the number of scanlines we need to visit before we have at least 50 z_i estimates). Our goal is to find a dominant estimate of the height of the ground that we call z_g. To this end we run the mean-shift algorithm on the z_i's ($i = 1, \dots, 50$) and set as z_g the main mode. This is our robust estimate for the height of the ground in the beginning of our scan. After that we do the following:

1. Start again from the first scanline (set $i = 1$).

2. If $i > N$ run the algorithm of part (a) (for $i <= N$ it has been already executed).

3. If $|z_i - z_g|/|z_g| < 10\%$ (i.e. z_i is close to the dominant estimate) we declare the potential ground points PG as ground (G). Furthermore, starting from the last G point, visit all successive horizontal (H) points that are almost collinear and give to them the G label as well (grow ground). If $i > 50$ update the dominant estimate z_g by calculating the main mode of the distribution $z_1, \dots, z_i$. Set $i = i + 1$ and go to step 2.

4. If $|z_i - z_g|/|z_g| \geq 10\%$ (i.e. z_i is not close to the dominant estimate), run the algorithm of part (a), but now start after the last PG point (remove the PG label from the previous PG points). If you are able to find a new set of PG points go back to 3. If not, set $i = i + 1$ and go to step 2.

The ground algorithm generates sequentially a robust estimate of the ground points (G) along with the height z_g of the ground at each scanline (for scanlines that do not contain ground points the following algorithms use the last updated z_g for a height estimate).

6. The curb algorithm

Curbs provide an important cue for recognition and classification in urban scenes. In our setting we are using the detected curbs to provide context for possible location of cars. Our curb detection algorithm consists of two parts: a) one part that runs within each scanline on-the-fly and provides possible starting points for curbs, and b) a second part that verifies the existence of a curb after the whole curb has been sensed from the scanner. To determine possible starting points for curbs within a scanline we just record the first vertical (V) point $\mathbf{X_{i,k}}$ after the last ground point (G) (see Sec. 5 for detection of ground points) for which *TestUnder*($\mathbf{X_{i,k}}$) is False (see Sec. 7.3 for explanation of test). We give $\mathbf{X_{i,k}}$ the label of a possible curb point (PC). Note that we terminate our search of PC points whenever the z-coordinate exceeds a reasonable level above the ground estimate (see threshold t_{height} in Sec. 7.3). In order though to verify the existence of a curb we have to look at the sequence of scanlines containing it. To achieve this we perform a sequential labeling region growing algorithm on only the vertical (V) points and sequentially produce connected components $RV_1, \ldots, RV_n$ of vertical points. If a completed such component happens to include at least one point that is a possible curb (PC) then we further investigate that particular region. We first calculate its vertical height. If it is above a threshold t_{curb} it is discarded ($t_{curb} = 0.2m$ in our settings) since curbs cannot exceed a specific height. If it consists of a significant number of scanlines (three and above in our experiments) we accept it as a curb. Otherwise (too few scanlines) we make sure that the median vertical curvature of the points in the region is small and then accept the region as a curb. Our algorithm is not missing any curb in our datasets.

7. The car algorithm

We begin by running the online detection of horizontal, vertical and vegetation regions as described in [7]. In that framework as each scan point $\mathbf{X_{i,k}}$ of scanline i is received it is classified to be either horizontal H, vertical V, or vegetation T. Note that every point gets a classification, so for instance a point that is in an inclined surface not on vegetation regions will get either an H or V classification (i.e. mostly horizontal, or mostly vertical). One problem of the vegetation detection algorithm in [7] is that it is very sensitive and regions within car objects can be labeled as T. In order to alleviate this problem we modified the online vegetation detection algorithm of [7] in order to make it less sensitive (see Sec. 7.1). We thus add one extra class T' to the classification results. Each scan point $\mathbf{X_{i,k}}$ can be classified to be in T', meaning that it is in vegetation region with higher probability. Note that now T' is missing chunks of vegetation regions. We keep both classes T and T' for each point (i.e. a point that is in both T and T' has very high probability of being in vegetation region than a point of class T only). See Fig. 2 for an example.

The car detection algorithm works as follows. At each point in a scanline the classifiers of [7] are run in parallel with the classifier of less sensitive vegetation T' (Sec. 7.1). An online CUSUM-like statistic (see [2]) that sets off a trigger at index (i, k) is used to decide whether the scanline i may contain a car or not. If the trigger is set off then the goal is to identify within that scanline i intervals of points that could potentially be parts of cars. These intervals can contain only horizontal H, vertical V, or vegetation points T that are not of class T'. Finally each interval may contain missing points. A missing point M is one for which the laser was sent but no response was recorded by the sensor. This maybe due to transparent or highly reflective surfaces for instance.

The next step in our detection consists of segmenting each scanline i into a set of intervals $[t_1(0), t_2(0)], \ldots, [t_1(m-1), t_2(m-1)]$, where $t_1(p)$ is the index of the first point and $t_2(p)$ is the last point of the p-th interval (m intervals in total). The criteria for the dividers of each interval are explained in detail in Sec. 7.3 (see Fig. 4 for one such interval). Potential car regions are then only to be decided upon within these intervals according to further tests which verify the existence or not of a car.

7.1. Less sensitive vegetation detection

As described in Sec. 4 vegetation areas can be detected coarsely. Due to the sensitivity of this algorithm to small non-vegetation areas (that can be unfortunately part of cars) we now consider the difference of signed angles $sD_{i,k} = sV_{i,k} - sV_{i,k-1}$ between successive points on a scanline. The new three states are modeled as Gaussians with means of $-150°$, $0°$, and $150°$. We want to detect a change between two hypotheses H_0 and H_1, the former corresponding to a non-vegetation region and the latter to a vegetation region, both captured by two distinct HMM models. To be more specific, let us represent the transition matrix for each of the two Markov models by

$$M = \begin{bmatrix} p_1 & p_2 & 1-p_1-p_2 \\ q_1 & q_2 & 1-q_1-q_2 \\ r_1 & r_2 & 1-r_1-r_2 \end{bmatrix} \quad (3)$$

Then under H_0: $p_1 = q_1 = r_1 = 0.1, p_2 = q_2 = r_2 = 0.8$ [i.e. tendency to reach or stay at state 2 (mean around $0°$ - no vegetation], and under H_1: $p_1 = p_2 = q_1 = q_2 = r_1 = r_2 = \frac{1}{3}$ [i.e. tendency to fluctuate between all states]. We thus use the same CUSUM-like algorithm as described in [7] with threshold $h = 10$, but we change the input and the mean and standard deviation (allowing for a larger value in

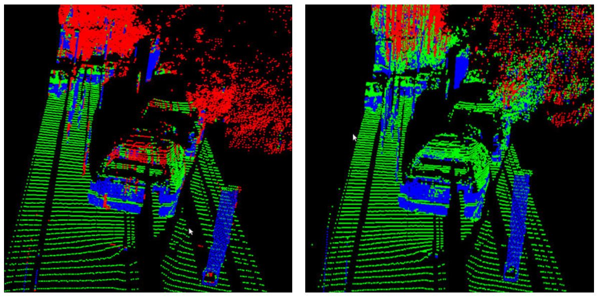

Figure 2. **(Left)** Initial vegetation detection algorithm (blue: vertical, green: horizontal, red: vegetation (class T)). **(Right)** Less sensitive vegetation detection (Sec. 7.1). Red shows vegetation regions with high probability (class T'). Some vegetation regions are missed, but there are now much fewer misclassifications on non-vegetation areas.

this case) of the random variables in each state. The selection of the exact transition probabilities, standard deviations and threshold h does not significantly change the results as long as the described behavior is maintained. Intuitively what we did was to introduce more noise in our observations since $sD_{i,k}$ is an approximation of the first derivative of $sV_{i,k}$. Therefore, the algorithm is now less sensitive to changes of regular surfaces to irregular surfaces. For a result of this algorithm see Fig. 2.

7.2. Determination of suspicious scanlines

In order to decide whether a given scanline i could potentially contain a car, we run an online CUSUM-like algorithm on the sequence of acquired data points that are currently classified as H or V, or are missing (M). Note that when we receive a point that is classified as T' we stop this online algorithm. The idea is to now distinguish a regular surface (which can also contain steps, holes or other small obstacles) from a region that could contain cars. The former are characterized by points which would customarily be identified as horizontal H and others corresponding to steps or other small obstacles which would customarily be characterized as V. Missing data is possible mainly due to inability to sense, or less frequently, small holes. Cars on the other hand are characterized by curved surfaces and continuous chunks of missing data due to mirrors, windows or metallic areas Thus, a possible way to capture and contrast a ground region to a possible car region is by using two separate Markov models; the one corresponding to a regular surface should have an enhanced probability in the horizontal H and vertical V states and low probability to missing states M. A possible car region on the other hand due to its curved surface and the persistence of missing data should be characterized by a higher likelihood of transition from a horizontal H state to a vertical V and/or a missing state M and vice versa. We therefore devise an online CUSUM-like algorithm to detect a change from one Markov model (regular surface) to another (car). The states of the models

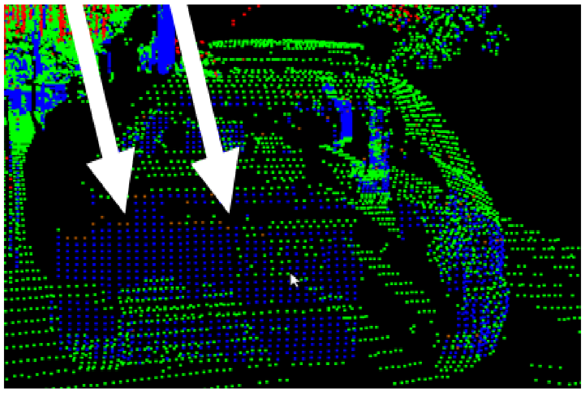

Figure 3. Determination of suspicious scanlines (Sec. 7.2). Trigger points along scanlines shown as brown [blue: vertical, green: horizontal, red: vegetation (class T')]. Two arrows help to visualize two of the triggers. These scanlines contain cars. See the pdf for color.

are H (state 1), V (state 2) and M (state 3). The algorithm is the same as in Sec. 7.1 but the transition matrix (3) is different. The transition matrix (3) specifications for each Markov model are: under H_0: $p_1 = 0.8, p_2 = 0.1, q_1 = 0.09, q_2 = 0.9, r_1 = 0.6, r_2 = 0.3$ [i.e. tendency to stay at horizontal or vertical state; state of missing data does not sustain itself and it most probably reverts to horizontal], and under H_1: $p_1 = 0.5, p_2 = 0.25, q_1 = 0.25, q_2 = 0.5, r_1 = 0.25, r_2 = 0.25$ [i.e. tendency to change between states]. This online CUSUM sets off a trigger the first time it detects a change from H_0 to H_1. The threshold used in this algorithm is $h = 1$. The selection of the exact transition probabilities and threshold h does not significantly change the results as long as the described behavior is maintained. Example of scanlines that produce triggers can be seen in Fig. 3.

It is important to note that such a trigger could be set off not only as a result of the presence of a car region but also in the presence of vegetation or even (in rare cases) as a result of a big hole on the ground. Therefore we use this trigger only as an indicator of a suspicion of the presence of a car and proceed to run further tests, including the divider algorithm that follows, before we make a final decision. The next section describes processing in scanlines that contain triggers.

7.3. Divider detection

Within a scanline that the trigger of Sec. 7.2 goes off a set of intervals $[t_1(0), t_2(0)], \ldots, [t_1(m-1), t_2(m-1)]$ that could potentially contain car regions need to be computed. These intervals can contain points currently classified as horizontal H or vertical V, as well as missing points M. This is achieved by calculating the dividers $t_1(p)$ and $t_2(p)$ ($p = 0, \ldots, m-1$). For an example see Fig. 4.

Searching for first dividers. We start by detecting the first divider $t_1(p)$. Initially $p = 0$. A potential first divider is the first point along the scanline where there is a transition from a horizontal H to a vertical V surface. This is due to the nature of the car as an object above the ground.Let us say that the change to the V surface happens at point k along the scanline. If the angle $sV_{i,k}$ corresponds to a gradual change into a vertical surface (i.e. $sV_{i,k} < l_1$ or $sV_{i,k} > l_2$)[1], then this point is considered as the first divider as it usually signifies the appearance of a tire. Otherwise (i.e. if $l_1 \leq sV_{i,k} \leq l_2$)[2] then the transition is more sudden. That can identify a bumper or the side of a car or another vertical surface (e.g. a pole or an obstacle). We thus need to further identify the case of a bumper or the side of car which is achieved by performing the following additional test: *TestUnder*($\mathbf{X_{i,k}}$). We compute the Euclidean distance $d(\mathbf{X_{i,k}} - \mathbf{X_A})$ between the k-th point and the origin $\mathbf{X_A}$ (i.e. scanner's location). We also compute the same distance for $L = 30$ previous points $d(\mathbf{X_{i,k-j}} - \mathbf{X_A})$, for all $j = 1, \ldots, L$. Out of these distances we compute the maximum M. If the difference between $d(\mathbf{X_{i,k}} - \mathbf{X_A})$ and M is small [3] that means that the k-th point is the furthest away from the scanner. *TestUnder*($\mathbf{X_{i,k}}$) is set to False and in that case we don't choose that point as the first divider (this case corresponds to a vertical obstacle, not a car) and we continue the search using the algorithm from the beginning of this paragraph. Otherwise (*TestUnder*($\mathbf{X_{i,k}}$) is set to True, i.e. the k-th is not the furthest away from the scanner) some previous points are under the surface of the divider. This is the case that identifies the location of a potential bumper or side of car and we declare $t_1(p) = \mathbf{X_{i,k}}$.

Searching for next dividers. Successive dividers are specified as sudden changes in distances between successive points $\mathbf{X_{i,n}}$ and $\mathbf{X_{i,n+N}}$ where N is either one (i.e. next point in scanline) or greater than one but all points $\mathbf{X_{i,n+1}}, \ldots, \mathbf{X_{i,n+N-1}}$ are missing. Thus if $d(\mathbf{X_{i,n}}, \mathbf{X_{i,n+N}})$ is above a threshold t_{sep}[4] and the vector $\mathbf{X_{i,n+N}} - \mathbf{X_{i,n}}$ is forward looking (i.e. does not point back towards the scanner) then divider $t_2(p) = \mathbf{X_{i,n}}$ and divider $t_1(p+1) = \mathbf{X_{i,n+N}}$. At this point the interval $[t_1(p), t_2(p)]$ and the beginning of the next interval $t_2(p+1)$ have been identified. Within the identified interval a further test is performed. This test detects whether after the beginning of the interval a long almost perfectly horizontal surface exists[5]. If such a surface is identified from point $\mathbf{K}$ to point $\mathbf{L}$ within the interval $[t_1(p), t_2(p)]$ then that means that we have a transition from an object to the ground in that interval. In that case we terminate interval p at $\mathbf{K}$ (i.e. $t_2(p) = \mathbf{K}$), we ignore the original divider $t_2(p+1)$ and we restart the search for the first divider of the next interval $p+1$ by looking for the first divider once again. This is achieved by applying the algorithm of the previous paragraph after point $\mathbf{L}$. Essentially we are looking for another object of interest within the scanline. Otherwise (i.e. in the case in which the interval $[t_1(p), t_2(p)]$ does not contain any long horizontal region), set $p = p + 1$ and look for $t_2(p)$ and $t_1(p + 1)$ by applying the algorithm of this paragraph.

The intervals $[t_1(p), t_2(p)]$ do not contain points of class T'. This is achieved by terminating the divider detection algorithm after the first point of the class T' is identified (i.e. point with high probability of being vegetation). A second **termination condition** for the divider detection algorithm is at the first measured 3D point with z-coordinate higher than a threshold t_{height}. Due to our online computation of ground surfaces we have a robust estimate of the current height of the ground at each scanline. We are thus adjusting automatically t_{height} to be at a reasonable distance above the current ground estimate (i.e. we are adding 2m to the ground height). Our algorithm is not looking for cars out of context.

Further division of intervals Before continuing with further processing of the intervals $[t_1(p), t_2(p)]$, $p = 0, \ldots, m - 1$ in order to decide whether they belong to cars or not we do a further subdivision of each of the intervals by subtracting long sequences of consecutive vertical points (these regions are obvious and normally correspond to strictly vertical obstacles or trunks of trees). This may result in cutting each interval $[t_1(p), t_2(p)]$ into disjoint intervals $[v_1(p_j), v_2(p_j)]$, $j = 0, \ldots, r - 1$.

7.4. The high verticals algorithm

This algorithm runs within each scanline i and is searching to find the first continuous sequence of vertical points $\mathbf{X_{i,k}}, \ldots, \mathbf{X_{i,k+M}}$ such that the last point's $\mathbf{X_{i,k+M}}$ z-coordinate is well beyond the ground. Since we have a reliable estimate of the height of the ground from the algorithm of Sec. 5 we can robustly set such a threshold (in our setting we add 2.5m to the current height of the ground). If point $\mathbf{X_{i,k+M}}$ indeed exists then we calculate its horizontal distance d_h from the scanner (i.e. length of projection of $X_{i,k+M} - X_A$ on known horizontal plane) . Now every point whose horizontal distance from the scanner is greater

[1]We choose $l_1 = -5°$ and $l_2 = 20°$. A value of $0°$ corresponds to an abrupt change to the vertical, since the $\mathbf{z}$ axis is the known vertical direction. We provide a slack around this sudden jump, giving more slack to positive angles because small steps can be misclassified as tires. Note that due to the rotation invariance of a tire these thresholds do not have to be adjusted in the case of inclined ground.

[2]Note that these thresholds do not have to be adjusted in cases of inclined ground surfaces. In that case the transition will appear more gradual and will be captured by the first test.

[3]We use the threshold $0.01m$. This threshold should be above the standard deviation of the noise level of the scanner (it is larger by one order of magnitude).

[4]We choose $t_{sep} = 4m$. Our goal is to provide one car within each interval. That threshold allows in most cases separation between two different car regions. In the cases where the scanline gets two cars very close to each other two or more cars can be within the same interval. But they will still be identified as car regions.

[5]We use a threshold of $3m$ since it is not possible to find long horizontal regions of that length on car regions. This threshold can be adjusted appropriately if needed.

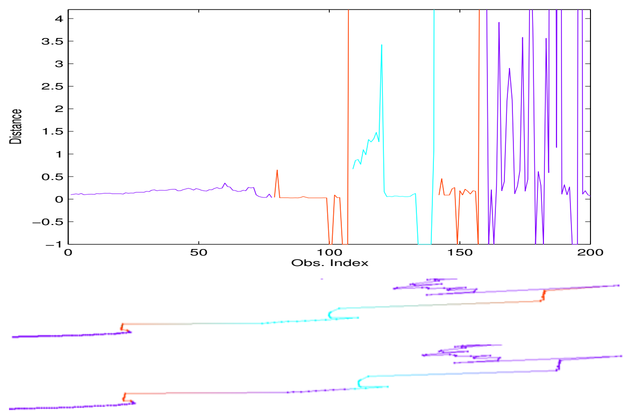

Figure 4. The divider algorithm (Sec. 7.3) in one example scanline. **(Top)** Distance between successive points (horizontal axis: index of scan point - vertical axis: distance between successive points) [distance is shown as negative when it can not be computed due to missing points]. After the first divider there are two subsequent dividers. This generates three intervals that are displayed with different colors (red-cyan-red). Parts of the scanline that are not within dividers are displayed as purple. The last interval is terminated due to reaching the limit in t_{height} (see Sec. 7.3). **(Middle)** Actual scanline that produced the distance graph shown on top. The points are displayed with the same color-code as above. Before the correction, horizontal surfaces on the ground, or tree regions can be part of the intervals. **(Bottom)** Same scanline after the corrections. Only two intervals survive and the ground points are not part of them. The last interval reverts to a tree region by the algorithm of Sec. 7.5 (note that this algorithm is applied on signed angles and not distances).

than d_h cannot take the label of a car (C) since we assume that no car can lie behind such vertical objects (i.e. vertical walls or large poles). This check is extremely useful and it allows us once again not to search for cars out of context.

7.5. Fourier transform for car vs. vegetation

As a result of the divider algorithm of Sec. 7.3 we have identified the intervals $[v_1(0), v_2(0)], \ldots, [v_1(r-1), v_2(r-1)]$ of potential cars.

Our final test consists of transforming the sequence of points in each of the intervals $[v_1(j), v_2(j)]$ $j = 0, \ldots, r-1$ into the frequency domain by applying a Fourier transform. This algorithm is used to distinguish a tree from a car region. As a result of that we are able to classify each point in the interval as part class C (car) or C' (car complement).

As a result of the less sensitive vegetation algorithm of Sec. 7.1 many regions that are part of vegetation are now given the classification H or V. We thus need to apply a test within each of the intervals $[v_1(j), v_2(j)]$ that classifies them as tree vs non-tree (likely car). This is done by transforming the signed angle data sV_i for each of the points in the interval $[v_1(j), v_2(j)]$ into the frequency domain (Fourier transform). In particular, let N be the number of points in the interval $[v_1(j), v_2(j)]$ for a fixed j. Since the numbers sV_i are real the array of frequencies repeats itself for $k > \frac{N}{2}$. We thus only need to consider the frequencies $F(k)$ for $k = 0, \ldots, \frac{N}{2}$ if N is even and $k = 0, \ldots, \frac{N-1}{2}$ if N is odd. We now select the maximum of these frequencies $MX = \max_{0 \le k \le \frac{N}{2}} |F_k|$ and compute the set of all frequencies $\mathcal{S} = \left\{\frac{N}{8} \le k \le \frac{N}{2}; |F(k)| > \frac{MX}{2}\right\}$, which are greater than $\frac{MX}{2}$. If the set $\mathcal{S}$ is empty, then the variation in the array of $\{sV_i\}$ $i = 0, \ldots, N-1$ is rather stable signifying a region that is more likely to be a car than vegetation. If, on the other hand, the array of signed angles results in at least one frequency that is above half of the maximum MX this is more likely to signify a high-variation region, in this case vegetation. Note that the $F(0)$, i.e. the 0 frequency, is most likely to be the highest one in any case. Our algorithm decides on the classification C (car) when the set $\mathcal{S}$ is empty. If the set $\mathcal{S}$ is non-empty the labeling given by the initial classification algorithm is given to each of the points in the interval $[v_1(j), v_2(j)]$.

7.6. Final processing stage

Once all the data has been processed by the Fourier transform analysis of the last subsection, everything that has not been classified as a car now recovers its original classification which was decided upon by the initial vegetation algorithm described in [7]. This concludes the car detection algorithm.

8. Combination of algorithms

The coarse classification algorithm is the base and runs online in each scanline producing labels horizontal (H), vertical (V), and vegetation (T). Part (a) of the ground algorithm is run for the first N scanlines in parallel with the coarse classification algorithm in order to estimate the initial level of the ground. We then start again from the first scanline. The following algorithms run in sequence within each scanline: (1) Part (b) of the ground algorithm that computes and verifies ground points (G) as well as the current estimate of the level of the ground, (2) the curb algorithm that generates potential curb points (PC), and (3) the car algorithm. Finally, whenever part (b) of the curb algorithm identifies a verified curb, we need to revisit (i.e. go back to) the scanlines that contain the curb. At last we correct (i.e. revert to the coarse classification labels) any car points that have been detected behind the curb.

9. Results and Conclusions

We have tested our algorithms on a number of scans in busy urban settings, which include ground, vegetation, moving objects, cars and other urban structures. Our acquisition device is the Leica ScanStation2 [8]. This is a time-of-flight scanner with a spherical field of view, that generates a sequence of 3D points at a distance of up to 300m and accuracy of 5mm per point. Visualizations of some results can be seen in Fig. 5. We have also developed an intuitive user-interface for the ground-truth labeling of 3D point

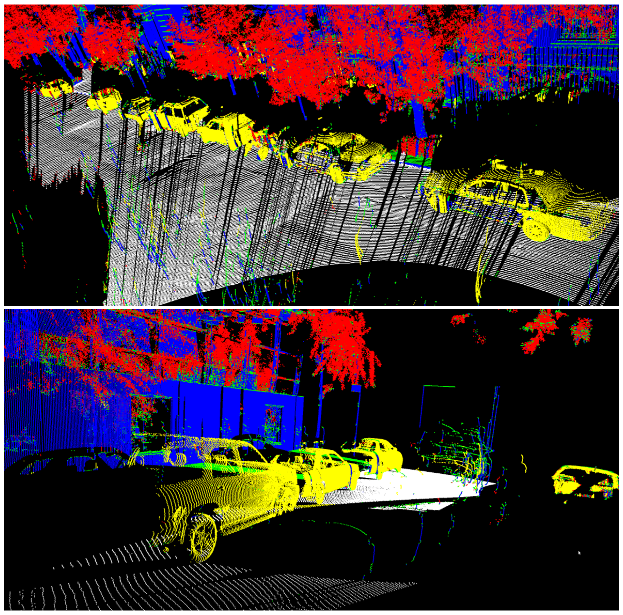
Figure 5. **(Top & Bottom)** Online car detection results. The detected car point are displayed with yellow color. Horizontal surfaces are shown in green, vertical in blue, and vegetation in red. Ground is shown in white. No offline post-processing has been performed to the dataset. See the pdf for color.

		Inferred Label				
		Vegetation	Vertical	Car	Horizontal	Recall
True Label	Vegetation	1054145	57293	2917	56784	0.9
	Vertical	266093	4083600	5399	443426	0.851
	Car	12128	13674	360891	29817	0.8665
	Horizontal	3635	12753	5927	905052	0.9759
	Precision	0.789	0.9799	0.962	0.6307	

Table 1. Confusion matrix for dataset of 7.3 million points in 11 urban range scans.

clouds. We tested our online classification results against ground-truth and produced precision-recall numbers and the confusion matrix shown in Table 1. These figures demonstrate the high accuracy of our algorithms. The precision for car objects is 0.96 and the recall 0.86. In some cases, persistent spikes (introduced by moving objects) can be identified as cars. The incorporation of context (ground, curbs, high-verticals) along with the clever online techniques provide very accurate results. The fact that no training is involved proves the possibility to generate accurate classifications using a small set of contextual rules along with innovative statistical techniques. Our algorithms consist of various stages. Each stage is significant (with the exception of the one of Sec. 7.5 that could be ommitted). Also, the detection of the ground provides an extremely significant cue. We were very careful in designing the ground-detection algorithm since possible errors in it would be detrimental for the whole pipeline. Our ground and curb detection techniques provide almost perfect results. Our future work includes the online classification of different types of urban objects, as well as combination of online methods with offline techniques. We would also like to evaluate each stage of the algorithm separately in order to discover its relevant significance. Finally, we will investigate the effect of learning the parameters that we have used in our HMM models.

References

[1] D. Anguelov, B. Taskarf, V. Chatalbashev, and etal. Discriminative learning of markov random fields for segmentation of 3D scan data. In *CVPR*, volume 2, pages 169–176, 2005. 2

[2] B. Chen and P. Willett. Detection of hidden markov model transient signals. *IEEE Transactions on Aerospace and Electronic Systems*, 36(4):1253–1268, October 2000. 1, 4

[3] D. Comaniciu and P. Meer. Mean shift: A robust approach toward feature space analysis. *IEEE PAMI*, 24:603–619, May 2002. 3

[4] B. Douillard. *Laser and Vision Based Classification in Urban Environments*. PhD thesis, The University of Sydney, 2009. 2

[5] B. Douillard, J. Underwood, N. Kuntz, and etal. On the segmentation of 3D LIDAR point clouds. In *ICRA*, 2011. 2

[6] A. Golovinskiy, V. G. Kim, and T. Funkhouser. Shape-based recognition of 3D point clouds in urban environments. In *ICCV*, Sept. 2009. 2

[7] O. Hadjiliadis and I. Stamos. Sequential classification in point clouds of urban scenes. In *3DPVT*, Paris, France, May 2010. 1, 3, 4, 7

[8] Leica Geosystems. http://hds.leica-geosystems.com/. 7

[9] D. Munoz, J. Bagnell, N. Vandapel, and M. Hebert. Contextual classification with functional max-margin markov networks. *CVPR*, pages 975–982, 2009. 2

[10] P. Musialski, P. Wonka, D. G. Aliaga, M. Wimmer, L. V. Gool, and W. Purgathofer. A survey of urban reconstruction. In *EUROGRAPHICS 2012 State of the Art Reports*, 2012. 1

[11] A. Patterson, P. Mordohai, and K. Daniilidis. Object detection from large-scale 3d datasets using bottom-up and top-down descriptors. In *ECCV*, pages 553–566. 2008. 2

[12] H. V. Poor and O. Hadjiliadis. *Quickest Detection*. Cambridge University Press, 2008. 1, 3

[13] I. Posner, D. Schroeter, and P. Newman. Online generation of scene descriptions in urban environments. *Robot. Auton. Syst.*, 56:901–914, November 2008. 2

[14] S. Thrun, M. Montemerlo, and *et al*. Stanley: the robot that won the DARPA grand challenge. *Journal of Field Robotics*, 23(9):661–692, 2006. 2

[15] D. Wolf, G. Sukhatme, D. Fox, and W. Burgard. Autonomous terrain mapping and classification using hidden markov models. In *ICRA*, pages 2038–2043, 2005. 2

[16] H. Zhao, Y. Liu, X. Zhu, Y. Zhao, and H. Zha. Scene understanding in a large dynamic environment through a laser-based sensing. In *ICRA*, 2010. 2