Digital plane detection in a point set: application to the interactive extraction of charcoal platforms from airborne LiDAR

Philippe Even, Phuc Ngo

Digital plane detection in a point set: application to the interactive extraction of charcoal platforms from airborne LiDAR

Philippe Even and Phuc Ngo

Université de Lorraine, CNRS, LORIA, Nancy, France
(philippe.even,hoai-diem-phuc.ngo)@loria.fr

Abstract. A digital blurred plane is a finite set of points with controlled thickness. This notion can be used to characterize a real world object surface, considered as planar at a large scale, but actually quite irregular at finer scale. In this paper, we introduce a new framework to detect a blurred plane in a point set, based on the analysis of two orthogonal height profiles using digital geometry tools. It is applied to the interactive extraction of relict charcoal platforms from an airborne LiDAR point cloud. This process was tested on two sets of geolocalized platforms obtained by on-site prospecting or visual survey of the LiDAR digital terrain model. Experimental validations show that a large amount of them is successfully extracted and that their estimated extent closely matches the ground truth.

Keywords: Digital plane detection · Point set processing · Airborne LiDAR

1 Introduction

Digital geometry aims at providing efficient processing algorithms to tackle the discrete nature of acquired data by most of today sensors. But many real world objects are not geometrically perfect. Discrete primitives such as digital straight lines or digital planes are not well adapted to describe their irregular surface. Therefore a notion of blurred digital object, such as blurred line segments [2] or blurred pieces of digital planes [8], has been proposed to incorporate such imperfections, the irregularity being captured by a thickness parameter. In this work, our purpose is to incorporate blurred planes in a new framework to process 3D point clouds, and to test their effectiveness in real context. Here we have chosen to address a quite difficult context involving poorly structured landscapes with hard sensing conditions.

Processing airborne LiDAR data of forest landscapes is a particularly challenging task. The LiDAR is a survey technique based on the emission of a laser beam swept over the measured scene. In forest scenes, the received signal is composed of multiple echoes corresponding to the successive hit obstacles. Lower ground points are interpolated to get a height map called *digital terrain model* (DTM). It provides bare ground visualizations of the landscape (see *Fig.* 1b). Its

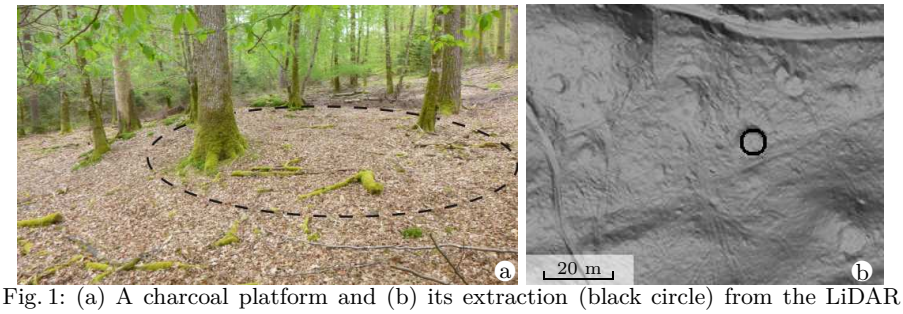

Fig. 1: (a) A charcoal platform and (b) its extraction (black circle) from the LiDAR ground points superimposed to the hill-shaded DTM.

quality depends strongly on the ground point distribution, that can be severely affected in places where the vegetation is dense. This capacity to penetrate the canopy is particularly useful for the study of geomorphological structures connected to past human activities. In forest areas, they are often well preserved from water erosion, while they have been strongly damaged by agricultural or industrial activities in open landscapes.

In a previous work [3], we proposed a framework to extract linear features such as forest roads, walls or ditches from LiDAR data. The goal addressed in this study is to design a new framework adapted to the processing of planar structures, that are also common: agricultural bankets, terrasses, pastoral enclosures, and so on. An approach based on the analysis of altimetric profiles obtained from scan lines in different directions can be successful at detecting plane patches [10]. The challenge here is to cope with possibly large terrain irregularities and important artefacts that may alter the surface, such as fallen tree remains or forest harvesting activities.

In this work, the framework is applied to the extraction of relict charcoal platforms. These structures were built in many forests areas in connection with industrial development in the past centuries. Their study may provide better understanding of the past industrial context as well as the ecological impact of charcoal production. Statistical analysis of their geometric features could help to extrapolate the acquired information on a larger scale. They are characterized by a nearly horizontal circular platform about ten meters wide (see *Fig.* 1a). In sloppy terrains, they contrast well to the local trend and are often oblong, developed perpendicularly to the slope direction. In flatter areas, they spread more uniformly in all directions, but their discrimination from natural terrain is more subtle, and sometimes, only a surrounding ditch allows their detection. This circular ditch may enclose the entire platform or only the upstream part on gentle slopes.

Some works have been led to automatize their detection in DTM visualizations using template matching [9] or convolutional neural networks [7,11,12], but proposed solutions rely on the processing of DTM visualizations, that ignore important features of the original point cloud. They also depend strongly on vegetation cover conditions and LiDAR quality. Beyond this, geometric char-

acterization of these platforms remains an open problem. In our approach, we process the point cloud directly, and the DTM is only used for interaction tasks, such as input seed selection or extracted structure validation.

Here we apply the blurred plane detection framework in order to collect richer geometric information. It takes a point set $\mathcal{S}$, a seed point P_S and a search direction $\boldsymbol{T_S}$ as input. From the recognition of support segments $\mathcal{B}$ and $\mathcal{B}'$ in two orthogonal point profiles, a thick enclosing plane $\mathcal{P}$ is constructed. The output blurred plane $\mathcal{Q}$ is composed of points of $\mathcal{S}$ that lie in $\mathcal{P}$. The input to the overall platform extraction process is a start position P_S. This seed can either be extracted from prior knowledge or manually selected in a DTM view. First, the slope direction $\boldsymbol{T_S}$ is estimated by probing the local terrain around the seed. This provides a favorable direction to find the platform plane $\mathcal{Q}$. Then the blurred plane detection framework is used to determine the possible platform extent. Finally, a circle detector is used to locate the workplace $\mathcal{C}$ where the charcoal mound was built. Charcoal platforms are good candidates to be modeled using the blurred plane concept. As a drawback, it is not possible to perform a separate evaluation of the blurred plane detection part.

The next section details the blurred plane detection framework. Its application to the extraction of charcoal platforms from LiDAR ground points is presented in *Sec.* 3, with a synthesis of the held experiments. *Sec.* 4 gives a conclusion and some perspectives to this work.

2 Blurred plane detection

In this section, discrete geometry notions used for the detection framework are introduced. First we recall briefly the notion of digital hyperplane and then explain how height profiles are extracted from the point set using directional scans. Next, we present a blurred line segment recognition procedure, that fits the assigned maximal thickness to local irregularities of the point set. Finally the blurred plane detection framework is described.

2.1 Digital hyperplanes

In the following, we note $\boldsymbol{P}$ the vector from the world reference origin to the point P. Given a non null d-dimensional vector $\boldsymbol{N}(n_1, n_2, \ldots, n_d) \in \mathbb{Z}^d$ with $d \geq 2$, a shift value $h \in \mathbb{Z}$ and an arithmetic width value $\mu \in \mathbb{Z}^+$, a **digital hyperplane** [1] is the set of integer points bounded by two bounding hyperplanes defined by equations $\boldsymbol{N} \cdot \boldsymbol{P} = h$ and $\boldsymbol{N} \cdot \boldsymbol{P} = h + \mu - 1$:

$$\mathcal{H}_d = \left\{ P(p_1, p_2, \ldots, p_d) \in \mathbb{Z}^d \;\middle|\; h \leq \boldsymbol{N} \cdot \boldsymbol{P} < h + \mu \right\} \tag{1}$$

The points of $\mathcal{H}_d$ lying in $\boldsymbol{N} \cdot \boldsymbol{P} = h$ (resp. $\boldsymbol{N} \cdot \boldsymbol{P} = h + \mu - 1$) are its **lower points**, (resp. **upper points**). Its **period** is $p(\mathcal{H}_d) = \frac{max(|n_1|,|n_2|,\ldots,|n_d|)}{gcd(|n_1|,|n_2|,\ldots,|n_d|)}$. When $\mu = p(\mathcal{H}_d)$, the hyperplane is said **naive**. The **thickness** of the digital

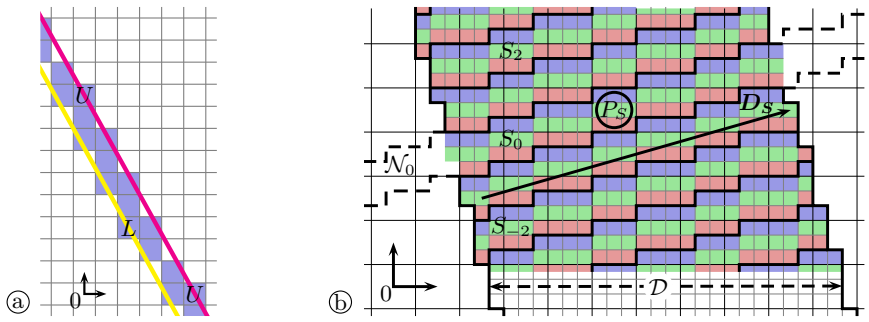

Fig. 2: a) Digital line $\mathcal{L} : 33 \leq 9x + 5y < 46$ with thickness $\tau(\mathcal{L}) = \frac{4}{3}$, bounding lines in yellow and magenta, lower point L and upper points U.
b) Directional scan with subdivision rate $N_G = 3$; the start scan S_0 is defined by point $P_S(5, 4)$, direction $\boldsymbol{D_S}(7, 2)$, arithmetic width $\mu = 7$ and radius $R_S = 4$ in the original grid $\mathcal{G}$; the scan strip $\mathcal{D}$ is defined by $46 \leq 7x + 2y < 215$ in the subdivided grid $\mathcal{G}'$.

hyperplane is the l_∞-distance between the bounding hyperplanes:

$$\tau(\mathcal{H}_d) = \frac{\mu - 1}{p(\mathcal{H}_d)} \tag{2}$$

A **digital line** is a digital hyperplane in a 2-dimensional space (see *Fig.* 2a) and a **digital plane** a digital hyperplane in a 3-dimensional space.

2.2 Height profile extraction

In this work, we assume that the points of the input set $\mathcal{S}$ are scattered on a non overlapping surface in height direction $\boldsymbol{Z}$, so that at most one point can be found at each (x, y) position. The points are embedded into a 2D grid $\mathcal{G}$ in $\boldsymbol{XY}$-plane, allowing fast access from the cell index. The grid $\mathcal{G}$ is used to define the input seed position P_S, the direction vector $\boldsymbol{D_S}$ and the search radius R_S for the plane detection framework. Height profiles can be built by sweeping the surface using directional scans.

A **directional scan** [6] is an ordered partition into scans S_i, restricted to the grid domain $\mathcal{G} \subset \mathbb{Z}^2$ of a thick digital line $\mathcal{D}$ called **scan strip**. Each scan S_i is a segment of a digital line $\mathcal{L}_i$, called **scan line**, with arithmetic width μ, and orthogonal to $\mathcal{D}$. Given a start scan S_0 defined by a center point P_S, a direction $\boldsymbol{D_S}$, a radius R_S and an arithmetic width μ, the directional scan is defined as:

$$DS(S_0) = \left\{ S_i = \mathcal{D} \cap \mathcal{L}_i \cap \mathcal{G} \;\middle|\; \begin{array}{l} \boldsymbol{N}(\mathcal{L}_i) \cdot \boldsymbol{N}(\mathcal{D}) = 0 \\ h(\mathcal{L}_i) = h(\mathcal{L}_{i-1}) + \mu \end{array} \right\} \tag{3}$$

In this definition, $\boldsymbol{N}(\mathcal{L}_i) \cdot \boldsymbol{N}(\mathcal{D}) = 0$ expresses the orthogonality between the scan lines $\mathcal{L}_i$ and the scan strip $\mathcal{D}$. The scans S_i can be iteratively parsed from

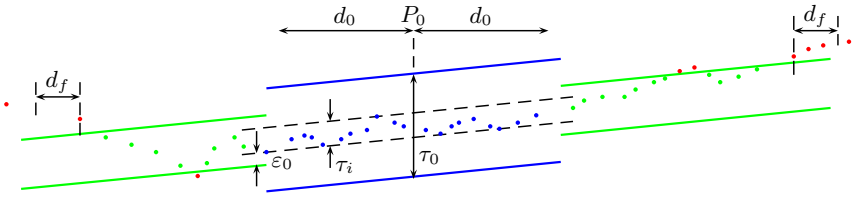

Fig. 3: Blurred segment recognition with controlled thickness: blue dots are first inserted, then green dots after maximal thickness was fit to $\tau_i + \varepsilon_0$; red dots are rejected.

S_0 to both ends. The shift μ between successive scans $\mathcal{L}_{i-1}$ and $\mathcal{L}_i$ guarantees that all points of $\mathcal{D}$ are traversed only once. The thickness of the scan strip $\mathcal{D}$ is defined by the radius value: $\tau(\mathcal{D}) = 2 \cdot R_S$.

For each scan S_i, a **height profile** is obtained by collecting the points of $\mathcal{S}$ stored in each cell, and by projecting them onto the $(P_S, \boldsymbol{D_S}, \boldsymbol{Z})$ plane. The points P_i are sorted along $\boldsymbol{D}$ direction, and their source $\kappa(P_i) \in \mathcal{S}$ is also registered for later retrieval. Directional scans are used in [6] to scan gray-level images into naive line segments. However, scan line aliasing produces an irregular point distribution in the height profile. In present work, a subdivided grid $\mathcal{G}'$ with N_G-finer resolution is used to get smoother profiles from a directional scan with arithmetic width $\mu' = N_G \cdot \mu$. *Fig.* 2b illustrates this process with $N_G = 3$. In practice, a subdivision rate $N_G = 5$ is used.

2.3 Blurred segment recognition with controlled thickness

A **blurred segment** [2] is a finite subset of a digital line $\mathcal{L}$. The covering digital line with minimal thickness is called the **optimal line** of the blurred segment $\mathcal{B}$. Hereafter, we note $\boldsymbol{V}$ its director vector.

Blurred segments can be detected in linear time by a recognition algorithm [2]. Inputs are a set of sorted points along some axis, and a maximal thickness value τ_0 for the blurred segment. Points are sequentially fed to the algorithm. At each insertion, the blurred segment and its convex hull are updated if the new thickness is less than τ_0. Otherwise the recognition process stops. The optimal line is bounded by an edge of the convex hull on one side and by a vertex on the other side. The director vector $\boldsymbol{V}$ and the thickness τ are easily derived from this edge-vertex pair.

In this algorithm, the maximal thickness is set beforehand to cope with present noise in the point set. This requires a previous analysis of input points. We rather propose an adaption of the initial algorithm to perform this noise level estimation during the blurred segment recognition process in order to better fit the segment to the set of points (see *Fig.* 3). The maximal thickness is initially set to a large value τ_0 and the blurred segment is extended to both sides of a start position P_0. Profile points are submitted to the recognition algorithm from the nearest to the furthest to P_0. Let call d_i the horizontal distance of the i^{th} point to P_0. After some observation distance is reached ($d_i > d_0$), the maximal thickness is reduced to the already detected segment thickness τ_i dilated by a

small safety margin ε_0 to incorporate some more possible terrain roughness. On each side, the detection stops when a d_f-long sequence of successive points is rejected or when the last point is processed. The output is the blurred segment, i.e. the set of accepted points, along with its final thickness τ_S, length λ and director vector $\boldsymbol{V}$.

2.4 Blurred plane detection framework

Similarly to the definition of blurred segments, a blurred plane is a finite subset of a digital plane $\mathcal{P}$. The covering digital plane with minimal thickness is said optimal. Since [5] or [8], many algorithms have been proposed to recognize a blurred plane in a point set. However the maximal thickness of the set is a fixed input for this recognition task. For our detection problem, an effective thickness value must be estimated during the recognition itself, so these algorithms are no longer suitable.

The blurred plane detection framework proposed here aims at exploring the point set in two privileged directions using the fast blurred segment recognition. The first direction can be chosen to optimize breaks-in-slope in the height profile. The found segments are enclosed in a digital plane with nearly optimal thickness. This plane is then used to select the points of the set, that compose the output blurred plane. The inputs are a start point P_S, a direction $\boldsymbol{D_S}$ and a search radius R_S in the grid $\mathcal{G}$. The search radius limits the maximal extent of the output plane patch. The parameters are geometric thresholds that are checked in the early stages. Most non-planar inputs are thus rapidly detected.

As depicted in *Fig.* 4, two blurred segments, a **main support** $\mathcal{B}$ and a **cross support** $\mathcal{B}'$, are recognized at position P_S in the height profiles built from the naive scans $S(P_S, \boldsymbol{D_S}, R_S)$ and $S'(P_S, \boldsymbol{D'_S}, R_S)$, where $\boldsymbol{D'_S}$ is orthogonal to $\boldsymbol{D_S}$. Their length is then checked: $\lambda_{min} < \lambda(\mathcal{B}) < \lambda_{max}$ and $\lambda'_{min} < \lambda(\mathcal{B}') < \lambda'_{max}$. If one is too short (resp. too long), the assigned thickness τ_0 and the observation distance d_0 are increased (resp. decreased) by a third, and a new recognition trial is attempted as a possible option. If no suitable-length segments are found, the blurred plane detection stops.

Three-dimensional vectors $\boldsymbol{B}$ and $\boldsymbol{B'}$ are then derived from the mapping of two-dimensional vectors $\boldsymbol{V}(\mathcal{B})$ and $\boldsymbol{V}(\mathcal{B}')$ in the world reference system, given the height profile directions. Let $\boldsymbol{N_P} = \boldsymbol{B} \times \boldsymbol{B'}$. The points P of each segment are examined to determine the lower and upper shift values along $\boldsymbol{N_P}$ from their source $\kappa(P)$ in the original set $\mathcal{S}$:

$$\begin{cases} l = \min_{(P \in \mathcal{B})} \boldsymbol{\kappa(P)} \cdot \boldsymbol{N_P} \\ u = \max_{(P \in \mathcal{B})} \boldsymbol{\kappa(P)} \cdot \boldsymbol{N_P} \end{cases} \quad \text{and} \quad \begin{cases} l' = \min_{(P' \in \mathcal{B}')} \boldsymbol{\kappa(P')} \cdot \boldsymbol{N_P} \\ u' = \max_{(P' \in \mathcal{B}')} \boldsymbol{\kappa(P')} \cdot \boldsymbol{N_P} \end{cases} \tag{4}$$

The digital plane $\mathcal{P}$ is defined by normal vector $\boldsymbol{N_P}$, shift value $h_P = min(l, l')$ and arithmetic width $\mu_P = max(u, u') + 1 - h_P$:

$$\mathcal{P} = \left\{ P \in \mathbb{Z}^3 \;\middle|\; h_P \leq \boldsymbol{P} \cdot \boldsymbol{N_P} < h_P + \mu_P \right\} \tag{5}$$

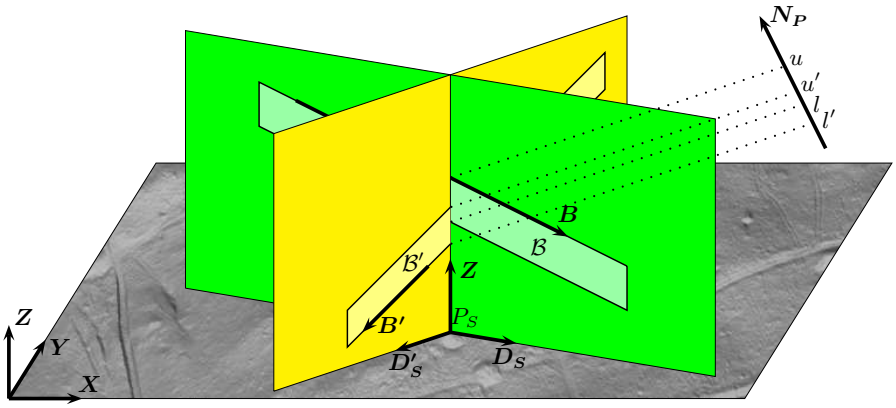

Fig. 4: Construction of the digital plane $\mathcal{P}$ from two orthogonal supports $\mathcal{B}$ and $\mathcal{B}'$.

The **crossing rate** χ_P of $\mathcal{P}$ is defined by the intersection over union factor of the blurred segments, and the **tilt** θ_P of $\mathcal{P}$ is the deviation of $\boldsymbol{N_P}$ to the vertical direction given by unit vector $\boldsymbol{Z}$:

$$\chi_P = \frac{min(u,u') - max(l,l')}{\mu_P - 1}, \quad \theta_P = \frac{||\boldsymbol{N_P} \times \boldsymbol{Z}||}{||\boldsymbol{N_P}||} \tag{6}$$

If $\chi_P < \chi_{min}$ or $\theta_P > \theta_{max}$, where χ_{min} and θ_{max} are threshold parameters, the detection stops.

Output thickness $\tau(\mathcal{P})$ is limited by the maximal assigned thickness τ_0 and the minimal crossing rate χ_{min}. The obtained digital plane $\mathcal{P}$ is not optimal. In the case of a strict intersection between the segments ($[u,l] \subset [u',l']$ or $[u',l'] \subset [u,l]$), a thinner digital plane can be found by rotating it around one of the bounding lines of the largest segment. However, this has low impact on the estimated thickness and tilt of the digital plane when the segments are long enough.

The blurred plane $\mathcal{Q}$ is the local subset of $\mathcal{S}$ belonging to $\mathcal{P}$. To get it, a directional scan is built from S_0 and the scan strip is explored in both directions. In each scan S_i, height profiles are built and inspected from the center to each end. The source of each profile point belonging to the digital plane $\mathcal{P}$ (*Eq.* 5) is inserted into a plane slice $\mathcal{K}_i$ if it belongs to the digital plane $\mathcal{P}$ (*Eq.* 5), until a d'_f-long sequence of outliers or the profile end is met. The obtained slice $\mathcal{K}_i = [\kappa(P_i^j), \ldots, \kappa(P_i^k)]$ is added to the blurred plane. To each slice, we associate a slice length $\lambda(\mathcal{K}_i) = distance(P_i^j, P_i^k)$, a lower bound width $\delta_L(\mathcal{K}_i) = distance(P_i^{j-1}, P_i^j)$ and an upper bound width $\delta_U(\mathcal{K}_i) = distance(P_i^k, P_i^{k+1})$. The plane extension stops when the search radius is reached ($|i| > \frac{R_P}{\tau(S_0)}$) or when plane slices get narrow ($\lambda(\mathcal{K}_i) < d'_f$).

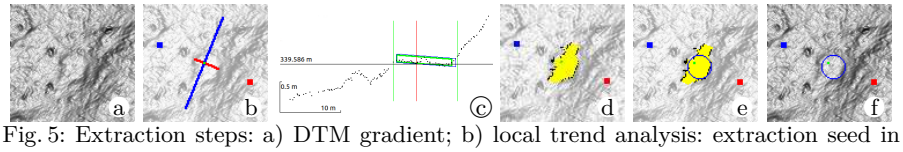

Fig. 5: Extraction steps: a) DTM gradient; b) local trend analysis: extraction seed in green, upper point in red, lower point in blue, main support in red, orthogonal support in blue; c) main height profile (Z/X ratio = 10): main support in green, digital plane in blue; d) extracted platform: extent in yellow, bounds in black; e) and f) detected circle.

3 Application to charcoal platform extraction

The blurred plane detection framework was applied to the interactive extraction of charcoal platforms from airborne LiDAR ground points. After the presentation of the extraction process, experiments held to evaluate the achieved performance are introduced.

3.1 The platform extraction process

The input of the process is a seed position P_S manually selected in a DTM view or extracted from a data base of georeferenced locations of known platforms. The extraction is performed in three steps (see *Fig.* 5). First the local terrain trend is analysed to estimate its slope gradient. Then the platform plane is detected from the input seed and the slope direction using the blurred plane detection framework. Finally, a dedicated circle detector is applied to the platform bounds to circumscribe the workplace around the charcoal hearth.

Charcoal platforms have more or less a circular shape with typical diameter ranging from $\Phi_{min} = 5\ m$ to $\Phi_{max} = 20\ m$. Along local terrain trend, they are often cut by a sharp slope or a ditch, whereas in orthogonal direction, they extend along the level line and their bounds are not as clear. Therefore it is important to get a rough estimate of the local terrain trend in the seed neighbourhood. To that end, ground point height values are examined in 16 uniformly spaced DTM cells at distance $R_T = 2\ \Phi_{max}$ to P_S. The low trend position L_T is the cell with the lowest value, and the up trend position U_T the one with the highest value (see *Fig.* 5b). Local trend direction $\boldsymbol{T}$ is defined by the vector $L_T U_T$.

The platform plane is detected using the blurred plane detection framework applied to the ground point set from seed position P_S, trend direction $\boldsymbol{T}$, and Φ_{max} as search radius. Some parameters are linked to the application context: $d_0 = \Phi_{min}$; $\tau_0 = 0.15\ m$, the largest expected platform roughness; $d'_f = 0.6\ m$, enough to cross most obstacles such as a fallen tree or a track rut, but not a possible surrounding ditch; $d_f = 0.3\ m$ as stricter conditions are required for support detection; $\varepsilon_0 = 0.05\ m$, the LiDAR vertical accuracy. Remaining ones are set to $\lambda_{min} = \lambda'_{min} = \Phi_{min}$, $\lambda_{max} = \Phi_{max}$, $\lambda'_{max} = \infty$, $\chi_{min} = 33\ \%$ and $\theta_{max} = 8\ \%$. A typical height profile is displayed in *Fig.* 5c, and the extracted platform plane in *Fig.* 5d.

In lack of perceptible bounds in perpendicular direction to the slope, the platform plane may extend quite far along the level line. Therefore a robust circle detector based on parameter space transform is used to delimit the workplace from well marked lower and upper bounds (see *Fig.* 5e-f). The chosen parametrization is based on circle center shift from the seed position (63×63 cells with 0.2 m resolution) and circle radius (77 layers with 0.1 m resolution). Only the bound pixels of the 17 central plane cuts are accumulated in the parameter space. The contribution of a bound is distributed over the crossed pixels, and set to the cut length λ so that narrow cuts have less impact on the result. The output circle is determined by the cell with the largest sum among those featuring at least 4 contributions from each of lower and upper bounds. This selection excludes the many cells where only a sequence of close pixels from a single side of the platform provide dominant scores. In that unpropitious case, wide bounds linked to possible point cloud scattering may give a decisive advantage to large radius values, resulting in diameter over-estimation.

The result of the blurred segment detection strongly depends on the input start point. Charcoal platform geolocalized positions or manual selections are not always accurate. Moreover some ground perturbation may occur right in the middle, so that even the platform center is not necessarily the best starting point. Therefore, a seed scattering option is proposed, where the extraction process is performed from 8 to 28 additional scattered spots around the seed position in a neighborhood radius of 5 pixels, until a valid solution is found.

3.2 Experiments

The charcoal platform extraction tool was tested on two LiDAR data sets mapping forested landscapes where charcoal production was developed at industrial scale in the last centuries. Both of them feature areas with a large amount of charcoal hearth platforms, typically about 150 per squarred kilometer in very dense sectors. They show different topographic contexts.

Bitche forest is located in North-East Lorraine. It is a sandstone area in the northern part of Vosges massif, featuring deep valleys enclosed by steep slopes with flat plateaux on the top. Most platforms lie in the slopes, where they highly contrast with the local trend. They are easily detected on-site. Many assume an oblong shape, extending perpendicularly to the slope. Poor planarity is often observed, maybe caused by upper bank erosion. On the plateaux, they have a large circular shape. Their detection is much more difficult, sometimes helped by the presence of a $30 - 40$ cm deep ditch all around.

Spincourt forest lies in the Woëvre plain (North-West Lorraine), in clayey terrain with gentle slopes. Platforms are difficult to find at field work. Many of them feature a subtle embankment upwards, while their downstream edge is often lost in low vegetation. Their detection is much easier in DTM gradient views.

LiDAR acquisition data is available for each site, along with a large data base of platform locations obtained by DTM surveys and field verification (see *Tab.* 1). In complement, a large set of platform dimensions was collected by

Table 1: Experimental site features and platform diameter comparisons (mean and standard deviation); A: site area, d: ground point mean density, N_M: number of on-site measured platforms, N_A: number of annotated platforms, Φ_M: on-site measured diameter, Φ_A: annotated diameter, Φ_E: extracted diameter, t: extraction time

	$A(km^2)$	$d(m^{-2})$	N_M	N_A	$\Phi_A - \Phi_M\ (m)$	$\Phi_E - \Phi_M\ (m)$	$t\ (ms)$
Bitche	144	6.7	164	11754	0.64 ± 1.20	-0.12 ± 2.09	10.69 ± 2.17
Spincourt	49	16.3	43	1378	0.13 ± 0.65	1.26 ± 3.06	9.53 ± 0.49

manual annotation and a smaller one by on-site surveys, both mostly based on available locations. On-site measures may feature large inaccuracy in flat areas where platform limits are often hard to discriminate. On the other hand, the annotation consists in a delineation of the platform contour in DTM gradient views. Visualization parameters strongly impact the result. A sharp slope gradient was used for the Spincourt set to better discriminate the platforms from the background. On the contrary for the Bitche set, a relaxed slope gradient with larger cut threshold was preferable to outline the whole platform. Despite of the great care put on this task, errors are still important. Two annotation sets of the same 6 km^2-wide area by two different persons showed a mean diameter difference of 2.5 meters.

In order to assess the reliability of the obtained diameters, we compared the annotated and the extracted values to on-site surveys. The extraction seed used for this test is the estimated position of the platform based on available geolocalizations. The results of *Tab.* 1 for Bitche set show reliable mean values but a larger standard deviation for extracted diameters compared to annotated ones. A 1 meter over-estimation appears in the small set surveyed in Spincourt forest. In this smooth relief, the platform bound may not be found and the blurred segment recognition may stop against a further break-in-slope. This error source is not compensated by fail cases dealing with too short supports, that are much fewer as they are more easily diagnosed in the extraction process.

Execution time for a complete extraction was also recorded and found quite compatible to interactive applications. A mean extraction time of 10 ms was obtained on a standard portable computer with an Intel Core i7 processor.

We then performed extraction tests on the complete sets (see *Tab.* 2). The used extraction seed is the center of the annotated platform. Different values were tested for each parameter of the blurred plane detection process in order to analyse their influence on the result. A large part of the platforms are successfully extracted. For most parameters (θ_{max}, χ_{min}, d_f, ε_0), small variations do not significantly change the output values. Seemingly, no careful setting is required. On the contrary, the seed scattering option has a strong effect. The method depends a lot on the selected seed. Only two thirds of the platforms are extracted when only the input seed is tested. Much more platforms are found if the seed scattering option is used with N_S additional near positions. This points out the large sensitivity of the approach on the local context around the input seed. The other main factor is the second trial performed when a blurred segment

Table 2: Comparison of annotated and extracted diameters on the whole set of annotated platforms; N_R: maximal number of blurred segment recognition trials, N_S: maximal number of tested seeds, S: successful extraction rate, $\Phi_E - \Phi_A$: difference between extracted and annotated diameters, R: recall, P: precision

		Bitche				Spincourt			
N_R	N_S	$S(\%)$	$\Phi_E - \Phi_A(m)$	$R(\%)$	$P(\%)$	$S(\%)$	$\Phi_E - \Phi_A(m)$	$R(\%)$	$P(\%)$
1	1	62.2	0.79 ± 1.81	85.5	71.8	62.7	1.70 ± 2.65	88.1	69.1
1	9	84.2	0.83 ± 1.93	84.4	70.3	91.6	2.20 ± 3.10	87.4	64.9
1	17	86.4	0.84 ± 1.95	84.2	70.0	95.6	2.28 ± 3.17	87.3	64.3
1	29	87.7	0.85 ± 1.99	84.1	69.8	97.8	2.35 ± 3.23	87.2	63.8
2	1	66.5	0.81 ± 1.83	85.5	71.4	65.1	1.75 ± 2.74	88.1	68.8
2	9	86.0	0.88 ± 1.96	84.7	69.9	94.2	2.19 ± 3.11	87.2	64.8
2	17	87.9	0.89 ± 2.00	84.6	69.6	97.8	2.26 ± 3.21	87.0	64.2
2	29	88.9	0.90 ± 2.02	84.5	69.4	99.1	2.28 ± 3.22	86.9	64.1

recognition produces a too long or too short support. The results show that the setting of underlying parameters, i.e. assigned maximal thickness τ_0 and observation distance d_0, depends a lot on the local terrain context.

We also compared the disk diameters and areas of extracted and annotated platforms. Diameter values show similar results to those obtained on the smaller set of on-site surveyed platforms. Circle areas are compared on the basis of recall and precision values. For each platform, the area A_I of the intersection between annotated and extracted circles is computed and compared to the area A_D of the annotated circle, and to the area A_E of the extracted circle. The recall value is defined as $R = \frac{A_I}{A_D}$ and the precision value as $P = \frac{A_I}{A_E}$. Mean values are displayed in *Tab.* 2. The performance decreases when more platforms are found using additional neighbour seeds or a second recognition trial for the supports. These options lead to the integration of lower quality platforms, that contributes to decrease the coincidence of extracted and annotated platforms. In Spincourt set, the recall is largely higher than the precision. This observation could be linked to the already observed over-estimation of the platform extent. In many cases, the annotated platform is probably covered by a larger extracted one.

On the whole, the extraction results fit rather well to those of a manual annotation, and could advantageously be used to speed up this task in semi-supervised mode. Moreover it delivers additional information on the platforms, such as the deviation from the horizontal plane or the surface roughness.

4 Conclusion

In this paper, we propose a new framework to detect a digital blurred plane in a set of points. The blurred plane is constructed as the union of two blurred line segments in orthogonal directions, and then completed with points collected into directional scans. This framework was successfully applied to the extraction of charcoal platforms from LiDAR ground points. The local trend is analysed

beforehands to get a favorable direction to facilitate the recognition of the main blurred segment. After platform plane detection in the point set, a dedicated circle detector is used to determine the workplace. The experimental work showed that platforms are quickly and correctly localized, so that the produced tool can be used for their fast and reliable annotation in DTM views. Moreover in the scope of a large scale study, the produced geometrical information can valuably be collated to other kinds of factors to disclose some associated geometrical patterns.

The blurred plane detection framework may be used for the extraction of other structures featuring plane surfaces. However it does not optimally enclose the blurred segments, so that some restriction should be put on cases where the plane normal vector largely deviates from the depth direction.

Future work deals with more advanced geometrical analyses to provide a finer estimation of the platform extent based on an elliptical approximation, and also to better exploit the local topographical context to tune the extraction parameters. Then, similarly to earlier work on forest roads [4], large-scale automatic detection of charcoal platforms will be studied. To that end, a detection of circular spots in DTM gradient views may provide input seeds to the extraction framework. At longer term, a combination of the approach with CNN-based solutions will be studied for training set preparation as well as output validation.

Acknowledgments. The authors would like to thank all the organisms and persons who contributed to this work by making available all the necessary data:

- SRA-Lorraine, ONF and *Association des Amis de Senon et de Spincourt*, for the LiDAR acquisition of Spincourt forest,
- LIEC, *Université de Lorraine*, for the LiDAR acquisition of Bitche area,
- Simon Ritz, INRAP / ArTeHiS, *Université de Bourgogne*, for the data base of geolocalized charcoal platforms in Spincourt forest,
- Vincent Robin, LIEC, *Université de Lorraine*, for the data base of geolocalized charcoal platforms in Bitche area,
- Anne Gebhardt, INRAP / LIEC, *Université de Lorraine*, for her lead participation to the platform survey in both sites and to the annotation work.

Disclosure of Interests. The authors declare no conflicts of interest.

References

1. Andres, E., Acharya, R., Sibata, C.: Discrete analytical hyperplanes. Graphical Models and Image Processing **59**(5), 302–309 (1997)
2. Debled-Rennesson, I., Feschet, F., Rouyer-Degli, J.: Optimal blurred segments decomposition of noisy shapes in linear time. Computers and Graphics **30**(1), 30–36 (2006)
3. Even, P., Ngo, P.: Live extraction of curvilinear structures from LiDAR raw data. ISPRS Annals of the Photogrammetry, Remote Sensing and Spatial Information Sciences pp. 211–219 (2020)
4. Even, P., Ngo, P.: Automatic forest road extraction from lidar data of mountainous areas. In: 1st Joint Conf. on Discrete Geometry and Mathematical Morphology. pp. 93–106 (2021)

5. Gerard, Y., Debled-Rennesson, I., Zimmermann, P.: An elementary digital plane recognition algorithm. Discrete Applied Mathematics **151**, 169–183 (2005)
6. Kerautret, B., Even, P.: Blurred segments in gray level images for interactive line extraction. In: 13th Int. Workshop on Combinatorial Image Analysis. LNCS, vol. 5852, pp. 176–186. Springer (2009)
7. Oliveira, C., Aravecchia, S., Pradalier, C., Robin, V., Devin, S.: The use of remote sensing tools for accurate charcoal kilns' inventory and distribution analysis: Comparative assessment and prospective. Int. Journal of Applied Earth Observations and Geoinformation **105**, 102641 (2021)
8. Provot, L., Buzer, L., Debled-Rennesson, I.: Recognition of blurred pieces of discrete planes. In: Int. Conf. on Discrete Geometry for Computer Imagery (Springer LNCS 4245). pp. 65–76 (2006)
9. Schneider, A., Takla, M., Nicolay, A., Raab, A., Raab, T.: A template-matching approach combining morphometric variables for automated mapping of charcoal kiln sites. Archaeological Prospection **22**(1), 45–62 (2015)
10. Sithole, G., Vosselman, G.: Automatic structure detection in a point-cloud of an urban landscape. In: 2nd GRSS/ISPRS Joint Workshop on Data Fusion and Remote Sensing over Urban Areas. pp. 67–71 (2003)
11. Trier, Ø.D., Reksten, J.H., Løseth, K.: Automated mapping of cultural heritage in Norway from airborne lidar data using faster R-CNN. Int. Journal of Applied Earth Observations and Geoinformation **95**, 102241 (2021)
12. Verschoof-van der Vaart, W., Bonhage, A., Schneider, A., Ouimet, W., Raab, T.: Automated large-scale mapping and analysis of relict charcoal hearths in Connecticut (USA) using a deep learning YOLOv4 framework. Archaeological Prospection **30**, 251–266 (2023)

Addendum to the paper: Digital plane detection in a point set: application to the interactive extraction of charcoal platforms from airborne LiDAR

Philippe Even and Phuc Ngo

Université de Lorraine, CNRS, LORIA, Nancy, France
(philippe.even,hoai-diem-phuc.ngo)@loria.fr

This document includes the algorithms used for the blurred plane detection in a 3D point set, and for its application to the extraction of charcoal platforms.

1 Blurred plane detection algorithm

Algorithm 1: `detectBlurredPlane`
`Detects a blurred piece of digital plane.`

Input data : a set $\mathcal{S}$ of 3D points,
a 2D point P_S as sead point,
a vector $\boldsymbol{D}_S$ along the main direction of the detection,
a neighborhood radius R_S to limit the detection area
Parameters: grid subdivision factor N_G for scan smoothing,
minimal cross ratio between supports χ_{min},
maximal tilt value θ_{max}
Output : a blurred plane $\mathcal{Q}$

$\mathcal{Q} \leftarrow \emptyset$;
$\mathcal{DS}_1 \leftarrow$ `directionalScan` $(P_S, \boldsymbol{D}_S, R_S, N_G)$;
$\mathcal{DS}_2 \leftarrow$ `directionalScan` $(P_S, \boldsymbol{D}_S^{\perp}, R_S, N_G)$;
$\mathcal{B}_1 \leftarrow$ `detectPlaneSupport` $(\mathcal{S},$ `firstScan` $(\mathcal{DS}_1))$;
if $\mathcal{B}_1 \neq \emptyset$ **then**
  $\mathcal{B}_2 \leftarrow$ `detectPlaneSupport` $(\mathcal{S},$ `firstScan` $(\mathcal{DS}_2))$;
  if $\mathcal{B}_2 \neq \emptyset$ **then**
    $\boldsymbol{V}(x_v, y_v) \leftarrow$ `directorVector` $(\mathcal{B}_1)$;
    $\boldsymbol{B}_1 \leftarrow$ `vector` $(x_v \cdot \cos(angle(\boldsymbol{D}_S)), x_v \cdot \sin(angle(\boldsymbol{D}_S)), y_v)$;
    $\boldsymbol{V}(x_v, y_v) \leftarrow$ `directorVector` $(\mathcal{B}_2)$;
    $\boldsymbol{B}_2 \leftarrow$ `vector` $(x_v \cdot \cos(angle(\boldsymbol{D}_S^{\perp})), x_v \cdot \sin(angle(\boldsymbol{D}_S^{\perp})), y_v)$;
    $\boldsymbol{N} \leftarrow \boldsymbol{B}_1 \times \boldsymbol{B}_2$;
    $l \leftarrow \{\infty, \infty\}$;
    $u \leftarrow \{-\infty, -\infty\}$;
    for $i \leftarrow 1$ **to** 2 **do**
      foreach $P \in \mathcal{B}_i$ **do**
        $shift \leftarrow \kappa(P) \cdot \boldsymbol{N}$; `/* κ(P): origin of P in S */`
        if $shift < l[i]$ **then**
          $l[i] \leftarrow shift$;
        if $shift > u[i]$ **then**
          $u[i] \leftarrow shift$;
    $h_P \leftarrow \min(l[1], l[2])$;
    $\mu_P \leftarrow \max(u[1], u[2]) + 1 - h_P$;
    $\chi_P \leftarrow (\min(u[1], u[2]) - \max(l[1], l[2])) \,/\, (\mu_P - 1)$;
    $\theta_P \leftarrow ||\boldsymbol{N} \times \boldsymbol{Z}|| \,/\, ||\boldsymbol{N}||$;
    if $\theta_P < \theta_{max}$ *and* $\chi_P > \chi_{min}$ **then**
      $\mathcal{P} \leftarrow$ `digitalPlane` $(\boldsymbol{N}, h_P, \mu_P)$;
      $\mathcal{Q} \leftarrow$ `extractBlurredPlane` $(\mathcal{P}, \mathcal{S}, P_S, \boldsymbol{D}_S, R_S)$;

Algorithm 2: `detectPlaneSupport`
`Detects a plane support (blurred segment) in the point set.`

Input data : a set $\mathcal{S}$ of 3D points,
a 2D scan segment S
Parameters: maximal number of support recognition trials N_R,
maximal assigned thickness to support recognition τ_0,
support recognition observation distance d_0,
support minimal length λ_{min},
support maximal length λ_{max},
maximal tilt value θ_{max}
Output : a "support" blurred segment $\mathcal{B}$

```
1  P_C ← center (S);
2  V ← direction (S);
3  C ← collect (S, S);
4  H ← ∅;
5  foreach P(x_P, y_P, z_P) ∈ C do
6  |  H ← H + (P_C P · V, z_P) ;
7  H ← sort_V (H);        /* Sorts the height profile along V        */
8  B ← ∅;
9  n ← 0 ;
10 k ← 1 ;
11 do
12 |  B ← pinchedBlurredSegmentRecognition (H, k · τ_0, k · d_0) ;
13 |  n ← n +1;
14 |  if length (B) < λ_min or sin (angle (B)) > θ_max then
15 |  |  B ← ∅;
16 |  |  k ← k + k / 3 ;
17 |  else
18 |  |  if length (B) > λ_max then
19 |  |  |  B ← ∅;
20 |  |  |  k ← k − k / 3 ;
21 |  |  else
22 |  |  |  n ← N_R;
23 while n < N_R;
```

Algorithm 3: `pinchedBlurredSegmentRecognition`
`Detects a blurred segment with controlled thickness.`

Input data : a height profile $\mathcal{H}$: ordered sequence of 2D points $P(x, y)$, an initial assigned thickness τ_{ass}, an observation distance d_{obs}
Parameters: additional thickness margin ε_0, fail distance d_f
Output : a blurred segment $\mathcal{B}$

```
1  B ← ∅;
2  s ← 0 ;                                  /* Index of start point in H          */
3  while s < #H and x[s] < 0 do
4  |  s ← s + 1 ;
5  if s ≠ 0 and |x[s − 1]| < |x[s]| then
6  |  s ← s − 1 ;
7  initBlurredSegment (P[s], τ_ass) ;
8  i ← 0 ;                                  /* Side of closest next point         */
9  j ← 1 ;                                  /* Side of furthest next point        */
10 side ← [−1, +1];                         /* -1 to the right, +1 to the left    */
11 last ← [s, s] ;
12 n ← [s − 1, s + 1] ;
13 d_max ← x[#H − 1]- x[0];
14 d ← [d_max, d_max] ;
15 if n[0] ≥ 0 then
16 |  d[0] ← x[s]-x[n[0]];
17 if n[1] < #H then
18 |  d[1] ← x[n[1]]-x[s];
19 observing ← true ;
20 while d[0] < d_max or d[1] < d_max do
21 |  if observing and x[last[1]] − x[last[0]] > d_obs then
22 |  |  setMaxThicknessOfBlurredSegment (B, thickness(B) + ε_0) ;
23 |  |  observing ← false ;                          /* pinched    */
24 |  if d[j] < d[i] then
25 |  |  i ← j;                              /* updates nearest next point */
26 |  |  j ← i + 1 % 2 ;
27 |  if insertPointToBlurredSegment (B, side[i], P[n[i]]) then
28 |  |  last[i] ← n[i];
29 |  else
30 |  |  if |x[n[i]] − x[last[i]]| > d_f then
31 |  |  |  d[i] ← d_max;                  /* stops searching on this side */
32 |  if d[i] < d_max then
33 |  |  n[i] ← n[i] + side[i];
34 |  |  if n[i] < 0 or n[i] ≥ #H then
35 |  |  |  d[i] ← d_max;                          /* side end reached */
36 |  |  else
37 |  |  |  d[i] ← |x[n[i]] − x[s]|;      /* updates distance to next point */
```

Algorithm 4: `extractBlurredPlane`
Extracts a blurred piece of a digital plane from a point set.

Input data : a digital plane $\mathcal{P}$,
a set $\mathcal{S}$ of 3D points,
a 2D seed point P_S,
a vector $\boldsymbol{D}_S$ along the main direction of the detection,
a neighborhood radius R_S to limit the detection area
Parameters: slice extraction fail distance d'_f
Output : a blurred plane $\mathcal{Q}$ structured as a deque of parallel slices $\mathcal{K}_i$

```
1  Q ← ∅;
2  DS1 ← directionalScan (P_S, D_S, R_S, N_G) ;
3  S ← firstScan (DS1) ;
4  K ← extractBlurredPlaneSlice (P, S, S, P_S) ;
5  Q ← pushBack (Q, K) ;
6  P_C ← center (K);
7  toright ← true ;
8  r ← 0 ;
9  d ← 0 ;
10 do
11     if toright then
12         S ← nextScanToRight (DS1) ;
13         K ← extractBlurredPlaneSlice (P, S, S, P_C) ;
14         Q ← pushFront (Q, K) ;
15     else
16         S ← nextScanToLeft (DS1) ;
17         K ← extractBlurredPlaneSlice (P, S, S, P_C) ;
18         Q ← pushBack (Q, K) ;
19     r ← r + τ(S);
20     if length(K) < d'_f then
21         d ← d + τ(S);
22     else
23         d ← 0 ;
24     if toright and (r > R_S or d > d'_f) then
25         toright ← false ;
26         r ← 0 ;
27         d ← 0 ;
28 while r ≤ R_S and d ≤ d'_f;
```

Algorithm 5: `extractBlurredPlaneSlice`
Extracts a blurred plane slice.

Input data : a digital plane $\mathcal{P}$,
a set $\mathcal{S}$ of 3D points,
a scan S,
a 2D point P_C
Parameters: plane slice extraction fail distance d'_f
Output : a sequence $\mathcal{K}$ of 3D points

```
1  K ← ∅;
2  V ← direction(S);
3  E ← collect (S, S) ;
4  E ← sort_V (E) ;                    /* Sorts collected point along V  */
5  P_0 ← argmin_(P∈E) |P_C P · V| ;
6  P ← P_0;
7  d ← 0 ;
8  fwd ← true ;
9  do
       /* Updates current point                                          */
10     if fwd then
11         if P = front (E) or d > d'_f then
12             if P = front (E) and P ∈ P then
13                 K ← P + K;
14             P ← P_0;
15             d ← 0 ;
16             fwd ← false ;
17         else
18             P ← previous (E, P);
19     else
20         P ← next (E, P);
       /* Processes current point                                        */
21     if P ∈ P then
22         if fwd then
23             K ← P + K;
24         else
25             K ← K + P;
26         d ← 0 ;
27         P_f ← P;
28     else
29         d ← distance (P_f, P) ;
30 while P ≠ back (E) and d ≤ d'_f;
```

2 Charcoal platform extraction from LiDAR data

Algorithm 6: `detectCharcoalPlatform`
`Extracts the circular workplace of a localized platform.`

Input data : a set $\mathcal{S}$ of 3D points,
a 2D point P_S assumed to belong to the platform,
a maximal number N_S of test points in the neighborhood of P_S
Parameters: charcoal platform search radius R_P,
minimal number K_{min} of voting slices,
manimal number K_{max} of voting slices
Output : a circle C, enclosing the workplace of the platform

$C \leftarrow$ `circle` $(0,0,0)$;
$\delta_S \leftarrow [(0,0),$
$(2,1),(1,2),(-1,2),(-2,1),(-2,-1),(-1,-2),(1,-2),(2,-1),$
$(3,0),(2,2),(0,3),(-2,2),(-3,0),(-2,-2),(0,-3),(2,-2),$
$(5,0),(4,2),(2,4),(0,5),(-2,4),(-4,2),$
$(-5,0),(-4,-2),(-2,-4),(0,-5),(2,-4),(4,-2)]$;
$n \leftarrow 0$;
while `radius` $(C) = 0$ *and* $n < N_S$ **do**
  $P \leftarrow P_S + \delta_S[n]$;
  $\boldsymbol{T} \leftarrow$ `findLocalTrendDirection` $(\mathcal{S}, P)$;
  $\mathcal{Q} \leftarrow$ `detectBlurredPlane` $(\mathcal{S}, P, \boldsymbol{T}, R_P)$;
  if $\mathcal{Q} \neq \emptyset$ **then**
    $\mathcal{K} \leftarrow$ `selectLeadSlices` $(\mathcal{Q}, K_{min}, K_{max})$;
    if $\mathcal{K} \neq \emptyset$ **then**
      $C_{loc} \leftarrow$ `fitCircleOnSliceBounds` $(\mathcal{K}, K_{max})$;
      if $C_{loc} \neq C$ **then**
        $\boldsymbol{T}_N \leftarrow \frac{\boldsymbol{T}}{||\boldsymbol{T}||}$;
        $x_C \leftarrow \mathtt{x}(P) + \cos(\boldsymbol{T}_N) \cdot \mathtt{xCoord}(C_{loc}) - \sin(\boldsymbol{T}_N) \cdot \mathtt{yCoord}(C_{loc})$;
        $y_C \leftarrow \mathtt{y}(P) + \sin(\boldsymbol{T}_N) \cdot \mathtt{xCoord}(C_{loc}) + \cos(\boldsymbol{T}_N) \cdot \mathtt{yCoord}(C_{loc})$;
        $C \leftarrow$ `circle` $(x_C,$ `y`$_C,$ `radius` $(C_{loc}))$;
  $n \leftarrow n + 1$;

Algorithm 7: `findLocalTrendDirection`
`Estimates the local trend direction around given position.`

Input data : a set $\mathcal{S}$ of 3D points,
a 2D point $P_{in}(x_{in}, y_{in})$, center of the analysis
Parameters: local trend search radius R_T,
local trend search resolution N_T
Output : a 2D vector $\boldsymbol{T}$ from lowest to highest points of local trend

$z_{min} \leftarrow +\infty$;
$z_{max} \leftarrow -\infty$;
$i \leftarrow 0$;
while $i < N_T$ **do**
  $P_C \leftarrow (x_{in} + R_T \cos(\frac{2\pi i}{N_T}), y_{in} + R_T \sin(\frac{2\pi i}{N_T}))$;
  $\mathcal{C} \leftarrow$ `collect (`$\mathcal{S}$`,` P_C`)`;
  foreach $P(x, y, z) \in \mathcal{C}$ **do**
    if $z < z_{min}$ **then**
      $z_{min} \leftarrow z$;
      $P_{min} \leftarrow P(x, y, z)$;
    if $z > z_{max}$ **then**
      $z_{max} \leftarrow z$;
      $P_{max} \leftarrow P(x, y, z)$;
  $i \leftarrow i + 1$;
$\boldsymbol{T} \leftarrow \boldsymbol{P_{min}P_{max}}$;

Algorithm 8: `selectLeadSlices`
`Selects the largest connected slices in given blurred plane.`

Input data : a blurred plane $\mathcal{Q}$ structured as a deque of parallel slices, a minimal size K_{min} and a maximal size K_{max} of the slice subset
Output : a connected subset $\mathcal{K}$ of the blurred plane $\mathcal{Q}$

```
1  K ← ∅;
2  K' ← ∅;
3  λ_med ← medianSliceWidth (Q) ;
   /* Selects the widest connected sequence of large slices          */
4  wide ← false ;
5  K ← front (Q) ;
6  if length (K) > λ_med then
7  |  K' ← pushBack (K', K) ;
8  |  wide ← true ;
9  do
10 |  K ← next (Q, K) ;
11 |  if wide then
12 |  |  if length (K) ≤ λ_med then
13 |  |  |  if #K' > #K and #K' ≥ K_min then
14 |  |  |  |  K ← K';
15 |  |  |  K' ← ∅;
16 |  |  |  wide ← false ;
17 |  |  else
18 |  |  |  K' ← pushBack (K', K) ;
19 |  else
20 |  |  if length (K) > λ_med then
21 |  |  |  wide ← true ;
22 while K ≠ back (Q);
23 if wide and #K' > #K and #K' ≥ K_min then
24 |  K ← K';
   /* Keeps only K_max centre slices                                 */
25 if #K > K_max then
26 |  K' ← ∅;
27 |  K ← front (K) ;
28 |  n ← (K_max − #K)/2 ;
29 |  if n ≥ 0 then
30 |  |  K' ← pushBack (K', K) ;
31 |  n ← n + 1;
32 |  while n < K_max and K ≠ back (K) do
33 |  |  K ← next (K, K) ;
34 |  |  if n ≥ 0 then
35 |  |  |  K' ← pushBack (K', K) ;
36 |  |  n ← n + 1;
37 |  K ← K';
```

Algorithm 9: `fitCircleOnSliceBounds`
`Finds the best circle crossing the bounds of given slices.`

Input data : a set $\mathcal{K}$ of blurred plane slices in DTM grid,
a maximal size K_{max} of the slice subset
Parameters: number N_P (resp. N_R) of cells in x-y (resp. radius) dimension,
size Δ_P (resp. Δ_R) of cells in x-y (resp. radius) dimension,
value R_C of the centre cell in radius dimension
Output : a circle C in blurred plane reference system

```
   /* Initialisation of the accumulator                                  */
 1 A ← 3dAccumulator (N_P, N_P, N_R, Δ_P, Δ_P, Δ_R, 0, 0, R_C) ;
 2         /* A_P: domain of A in x-y dimensions                         */
 3         /* A_R: domain of A in radius dimension                       */
 4 foreach a ∈ A do
 5     a ← (0,0) ;              /* weight, contribution from each side  */
 6 ctrb ← [1, aPowerOfTwoGreaterThan(K_max)] ;
   /* Vote                                                               */
 7 foreach K ∈ 𝒦 do
 8     W_k ← length (K);
 9     foreach side ∈ [LOW, UP] do
10         λ ← sliceBoundSegment (K, side);
11         Δ ← discretization (λ, A_P) ;
12         foreach p ∈ Δ do
13             w_p ← W_k / #Δ ;
14             foreach r ∈ A_R do
15                 C_r ← drawCircle (p, r) ;
16                 foreach c(x,y) ∈ C_r ∩ A_P do
17                     w ← weight (A[x,y,r]) + w_p;
18                     t ← contribution (A[x,y,r]) + ctrb[side];
19                     A[x,y,r] ← (w,t) ;
   /* Counting                                                           */
20 w_max ← 0 ;
21 a_max ← (0,0) ;
22 foreach a ∈ A do
23     t_low ← contribution (a) % ctrb[1];
24     t_up ← contribution (a) / ctrb[1];
25     if t_low ≥ 4 and t_up ≥ 4 then
26         w ← weight (a) ;
27         if w > w_max then
28             w_max ← w;
29             a_max ← a;
   /* Output circle construction                                         */
30 C ← circle (0,0,0) ;
31 if w_max ≠ 0 then
32     C ← circle (xValue (a_max), yValue (a_max), radiusValue (a_max)) ;
```