Contour detection in unstructured 3D point clouds

Timo Hackel, Jan D. Wegner, Konrad Schindler
Photogrammetry and Remote Sensing, ETH Zürich

Abstract

We describe a method to automatically detect contours, i.e. *lines along which the surface orientation sharply changes, in large-scale outdoor point clouds. Contours are important intermediate features for structuring point clouds and converting them into high-quality surface or solid models, and are extensively used in graphics and mapping applications. Yet, detecting them in unstructured, inhomogeneous point clouds turns out to be surprisingly difficult, and existing line detection algorithms largely fail. We approach contour extraction as a two-stage discriminative learning problem. In the first stage, a contour score for each individual point is predicted with a binary classifier, using a set of features extracted from the point's neighborhood. The contour scores serve as a basis to construct an overcomplete graph of candidate contours. The second stage selects an optimal set of contours from the candidates. This amounts to a further binary classification in a higher-order MRF, whose cliques encode a preference for connected contours and penalize loose ends. The method can handle point clouds* $> 10^7$ *points in a couple of minutes, and vastly outperforms a baseline that performs Canny-style edge detection on a range image representation of the point cloud.*

1. Introduction

By and large, modern 3D reconstruction techniques like dense multi-view matching, laser scanning or structured light projection deliver 3D point clouds as primary output. But raw point clouds are of limited use for most applications, and mostly serve as a basis for further processing. Either they are triangulated into an unstructured surface model, *e.g.* a triangle mesh, for low-level tasks such as visualization; or they are processed into more compact and structured CAD-like higher-level structures, *e.g.* [24, 16, 32]. Perhaps the most widely used approach in this context, especially for indoor applications, is to locally fit parametric surfaces or solids to the data. Line features then follow as a last step, by intersecting surface primitives. Such an approach can work well for schematic surface geometries, but quickly reaches its limits when portions of the scene are not covered by the primitive library.

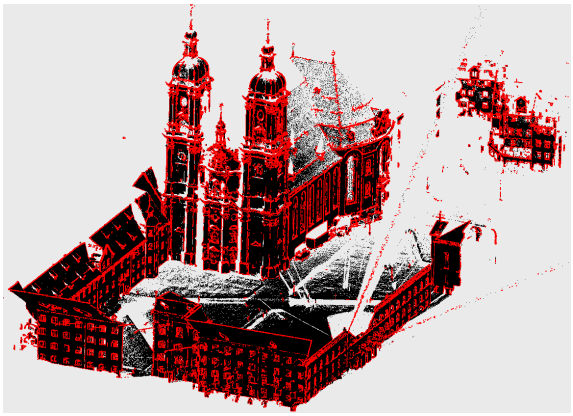

Figure 1: Given an unstructured 3D point cloud (black), our method detects contours (red). Challenges include strongly varying point density (*e.g.*, front facade *vs.* roof of the church); occlusions and missing data (*e.g.*, dormer on the left); and sheer data volume (here, $\approx 3 \cdot 10^7$ points).

Here, we propose to proceed the other way round. Our goal is to extract *contours* in outdoor point clouds such as those acquired by terrestrial laser scanners, see Fig. 1. By contours we mean linear features along which the orientation (normal) of the underlying surface exhibits an unusual discontinuity. Arguably, such wireframe-like line features are a lower-level representation than CAD surfaces or solids, in the sense that humans can "see" them in the raw data – the point cloud alone is sufficient to draw contours, even for relatively unconstrained surfaces that cannot be represented with a small library of primitives.

In that sense, it seems natural to detect contours *before* surface reconstruction, and use them to drive the segmentation and/or to fit surface patches – which is in fact how interactive modelling systems operate, both in graphics and vision [25] and in commercial mapping [26, 6].

At first sight, it may seem quite simple to detect crease edges in point clouds. In practice, it turns out to be surprisingly challenging, for several reasons. On the conceptual side, the definition of what constitutes a contour is not as clear-cut as it seems, and hard to formalise. While it should certainly have one high and one relatively low principal curvature at some (not necessarily fixed) scale, there are fur-

ther relevant properties like sufficient length, few sudden changes of the line direction, *etc*. Informally speaking, it is "where a CAD operator would draw a line or curve."

On a more technical note, scanners and cameras are line-of-sight sensors, hence real point clouds inherently suffer from occlusions and incomplete data, and exhibit extreme variations in point density (depending on the distance to the sensor and the surface orientation). Both effects are a lot stronger outdoors, where they are not mitigated by the limited size and constrained shape of rooms. See Figs. 1, 3.

The contribution of the present paper is a contour detector, which casts the problem as a two-level discriminative learning task. First, a binary classifier is trained which, for each 3D point, predicts the likelihood of lying on a contour, given the geometry in a local neighborhood and the point's latent semantic class probabilities. Based on these contour scores, we design a hypothesize-and-verify framework that takes into account the line structure: points with high score serve as seed points to construct an overcomplete contour graph; an optimal subset of the graph edges is then selected in another binary labeling task on a higher-order random field, to obtain the final contours. The first step can be interpreted as a discriminative version of classical multiscale point set analysis such as [22]. By using supervised learning, the detector implicitly learns which properties define (in the eyes of the users who labeled the training data) a point on a contour. In practice this gives much improved scores compared to simple curvature values, especially in terms of precision. The second step can be seen as a global model for linking individual edge points to an contour network, while taking into account evidence collected along longer (candidate) lines; rather than greedily linking individual points, as for example in the classical Canny detector [4]. We are not aware of any comparable method for 3D contour extraction.

In experiments on several laser scans each containing on the order of 10^7–10^8 points the method delivers high-quality contours. In a quantitative comparison on a real outdoor scan, the proposed method achieves an average $F1$-score of 80%, whereas a 3D range image variant of the Canny edge detector badly fails.

2. Related Work

The task of 3D contour extraction is related to the elementary image processing task of line detection. In this context it is important to note that even in 2D images (respectively, regularly sampled 3D image stacks) line extraction is surprisingly difficult, if one adopts a more high-level definition of what a "line" is. Although low-level algorithms like the Canny edge detector [4] are widely used to detect and link adjacent high-contrast pixels, the extraction of structured line information is still an active research topic, for example in medical imaging [29, 28], stereo reconstruction [7] and topographic mapping [27, 19]. These more recent works have in common that they rely on a hypothesize-and-verify strategy: *(i)* find points that are likely to lie on lines with low-level image processing; *(ii)* link those points to candidate lines, with shortest-path search or minimum spanning trees; *(iii)* use context features extracted along their entire length and/or CRF-type connectivity priors to reassess the candidate lines, and retain only an optimal subset.

When working on the basis of images, one possible solution is to extract line features in 2D and then triangulate them to 3D lines. Still, even that approach has in practice largely been limited to "clouds" of disconnected, straight line segments [23, 20, 13]. The problem becomes even more difficult when moving to 3D point clouds, with strongly varying point density and no regular neighborhood structure. A possible approach is to first turn the point cloud into a triangle mesh with all-purpose methods like Poisson reconstruction [15], and then analyze the mesh to find contours. Unfortunately, such approaches tend to wash out sharp edges and struggle to distinguish them from less pronounced regions of relatively high curvature. This is indeed not surprising, because surface reconstruction from noisy data typically involves a prior that smoothes the surface, which contradicts the goal to find sharp contours. Solutions have been developed which attempt to identify sharp creases before or during mesh generation. This is in accordance with our claim that line features should be identified early and used to support surface modelling. However, existing approaches [2, 8, 21] work locally, without taking into account long-range line structure, and use only basic surface features. Thus they struggle to distinguish between contours and areas of high surface roughness (*e.g*. vegetation), which renders them unsuitable for outdoor scenarios.

An alternative to triangle meshes are parametric 3D primitives like boxes, spheres, or cylinders. Several authors fit such primitives to point cloud data and combine them with simple set operations like union and intersection, leading to CAD or CSG (constructive solid geometry) models [24, 17, 32]. Again, it appears that fitting solids works best if the line structures have been made explicit beforehand [16]. That work again uses a rather simplistic, purely local definition of lines. Moreover, a small library of CSG primitives is too limited to faithfully represent realistic scenes, which is why high-quality reconstructions fall back to filling the gaps with triangle meshes [16].

The starting point of the present work is the recurring need for complete and accurate contours to support subsequent processing steps. We do not commit to any particular surface or solid modelling technique, rather our aim is to generate wireframe (WF) models. Their greater flexibility makes them applicable to both parametric and free-form surface regions, and they can even serve to select the most

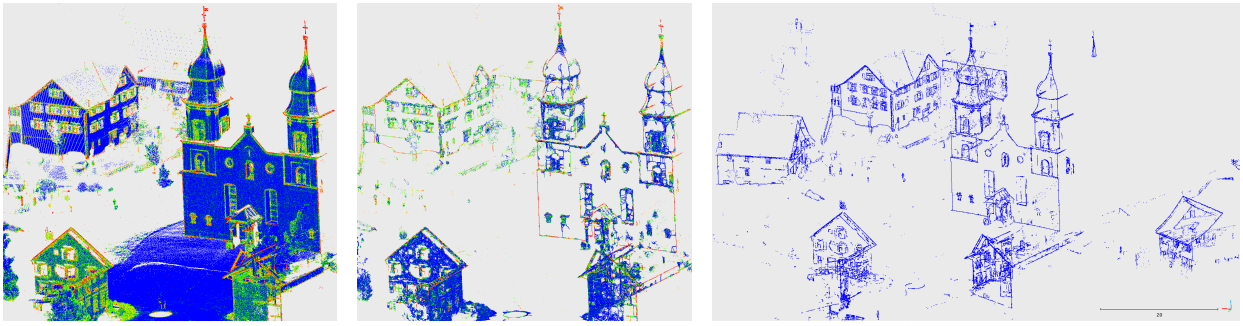

Figure 2: Illustration of our contour detection pipeline. (left) A binary classifier predicts pointwise contour scores (red: high contour probability, blue: low contour probability); (middle) Seed points with high contour scores are linked into an overcomplete graph of contour candidates; candidates are rescored with another round of classification; (right) candidates are pruned to an optimal set of contours by MRF inference.

suitable representation for a given point cloud or region, and to optimally fit it to the data. It has been suggested to extract line segments with robust model fitting algorithms like RANSAC or the Hough transform, and then link the soup of segments into wireframes [12]. Such a strategy will however restrict the geometry of the lines to a small set of predefined parametric models, and thereby lose much of the advantage of wireframes compared to solid-based representations. Early work on contour extraction, often from the more isotropic point clouds captured by aerial laser scanners, used rule-based expert systems to exhaustively cover all expected types of contours, *e.g.* [2]. The higher complexity of close-range point clouds renders it infeasible to design and tune such expert systems by hand, which calls for a machine learning approach.

3. Method

Our method to extract wireframe edges consists of three main steps as shown in Fig. 2. Starting from the raw point cloud without any preprocessing, we proceed as follows:

- for each individual point, discriminatively predict the probability of lying on a contour;
- find regularly spaced points with high contour scores, and link them into a graph of candidate contours.
- select an optimal subset of those candidates as final wireframe edges, by approximate inference in a higher-order random field defined over the graph edges and their adjacency relations.

This pipeline bears some similarity to recent work that aims to extract curvilinear networks from images [29, 19, 11]. Other than those works, we operate on irregular 3D point clouds rather than regular 2D or 3D image rasters. But like them, we greatly improve over classical Canny-type detectors by following three recurrent lessons of modern computer vision research: *(i)* use discriminative learning with rich feature sets, rather than raw differential geometry, to obtain low-level evidence at the scale of points/pixels; *(ii)* aggregate the points/pixels into higher-level primitives and use orientation statistics over the entire neighborhood to describe those primitives; *(iii)* include a prior about the expected neighborhood structure to capture long-range relations between primitives. In the following, we describe in detail how we implement those principles for contour extraction in unstructured point clouds.

3.1. Pointwise contour scores

The first step of the proposed pipeline is to compute low-level evidence for the presence of a contour. At each 3D point, we predict the likelihood that it lies on a contour with a binary, discriminative classifier, based on multi-scale surface properties in the point's local neighborhood. In that sense the method can be seen as a discriminative extension of early methods for feature extraction from point clouds [9, 22]. Those methods relied on (possibly multi-scale) curvature values computed from the point neighborhood. We found that, unsurprisingly, curvature alone is not a very good feature to identify contours. The definition of what constitutes a contour is unsharp and ill-posed, and includes properties like that the line should separate two regions of different, but well-defined surface orientation; or, that it should be part of of a sparse wireframe network. This clearly goes beyond individual curvature values, *e.g.* rough surfaces and volumes such as bushes and tree crowns in outdoor scans exhibit high per-point curvature, but are not perceived as contours. We thus rely on an extended feature set to describe the point neighborhood, including those proposed in [31] for semantic labeling of 3D point clouds, as well as newly developed features that emphasize occlusion boundaries and transitions between smooth surfaces.

In our framework we avoid using color and/or intensity values, which, depending on the recording technology, are

Sum	$\lambda_1 + \lambda_2 + \lambda_3$
Omnivariance	$(\lambda_1 \cdot \lambda_2 \cdot \lambda_3)^{\frac{1}{3}}$
Eigenentropy	$-\sum_{i=1}^{3} \lambda_i \cdot ln(\lambda_i)$
Anisotropy	$(\lambda_1 - \lambda_3)/\lambda_1$
Planarity	$(\lambda_2 - \lambda_3)/\lambda_1$
Linearity	$(\lambda_1 - \lambda_2)/\lambda_1$
Surface Variation	$\lambda_3/(\lambda_1 + \lambda_2 + \lambda_3)$
Sphericity	λ_3/λ_1
First Order Moment	O, see Eq. (2)
Line Feature	Q, see Eq. (3)
Orientation Feature	R, see Eq. (4)
Verticality	$1 - \lvert\langle [0\,0\,1], \mathbf{e}_3 \rangle\rvert$

Table 1: Geometric features based on the eigenvalues and eigenvector of the structure tensor.

not always available; and often also unreliable because of strong lighting effects. Thus, we face a purely geometric classification problem: learn a binary classifier for the two classes *contour* ($c_i = 0$) and *non-contour* ($c_i = 1$). With that classifier, predict a contour score $p(c_i|\mathbf{x}_i)$ for each individual point $\mathbf{p}_i$, given the point's feature vector $\mathbf{x}_i$.

Several authors [10, 31] have proposed geometric features for point cloud classification (usually into semantic object classes) that are based on the covariance tensor Σ_i,

$$\Sigma_i = \frac{1}{N} \sum_{n \in \mathcal{P}_i^N} (\mathbf{p}_n - \overline{\mathbf{p}})(\mathbf{p}_n - \overline{\mathbf{p}})^\top \ , \tag{1}$$

where $\mathcal{P}^N$ denotes the set comprising the N nearest neighbours of $\mathbf{p}_i$, and $\overline{\mathbf{p}} = \text{med}_{n \in \mathcal{P}^N}(\mathbf{p}_n)$ is the medoid. The features are, by and large, based on different arithmetic combinations of the eigenvalues $\lambda_1 \geq \lambda_2 \geq \lambda_3 \geq 0$ and eigenvectors $\mathbf{e}_1, \mathbf{e}_2, \mathbf{e}_3$ of the covariance tensor. We use the features of [31], see Table 1.

The original feature vector lacks information about occlusion boundaries. We thus add an additional feature dimension based on the first order moment $\mathsf{m}_\uparrow$ of $\mathbf{p}_i$ around the first eigenvector $\mathbf{e}_1$ of the structure tensor,

$$O = \frac{\mathsf{m}_\uparrow^2}{\mathsf{m}_\Uparrow} \qquad \text{with}$$
$$\mathsf{m}_\uparrow = \sum_{n \in \mathcal{P}_i^N} \langle \mathbf{p}_n - \mathbf{p}_i, \mathbf{e}_2 \rangle \ , \ \mathsf{m}_\Uparrow = \sum_{n \in \mathcal{P}_i^N} \langle \mathbf{p}_n - \mathbf{p}_i, \mathbf{e}_2 \rangle^2 \ , \tag{2}$$

where $\langle ., . \rangle$ denotes the scalar product. The normalization with the second order moment $\mathsf{m}_\Uparrow$ ensures $O \in [0, 1]$, to suppress the effects of varying point density. Moreover, the standard eigenvalue features do not explicitly capture the fact that contour points should *(i)* be arranged along locally smooth 1D paths of consistently high curvature and *(ii)* separate surface areas with different orientations. We thus design two more feature dimensions for those properties.

The first one is similar to a 3D line detector: we approximate the tangent of the putative contour with the first moment and split the points in the neighborhood into a subset $\mathcal{C}_{\mathsf{near}}$ of size $\lceil \alpha \cdot N \rceil$ which are closest to the tangent, and the remaining points $\mathcal{C}_{\mathsf{far}}$ (reusing the projections $d_{i,n} = \lvert\langle \mathbf{p}_n - \mathbf{p}_i, \mathbf{e}_2 \rangle\rvert$ from the previous feature). For each point we examine the local surface variation $\gamma_n = \lambda_3/(\lambda_1 + \lambda_2 + \lambda_3)$. The ratio between the average surface variations of the two subsets serves as feature

$$Q = \frac{\text{mean}_{n \in \mathcal{C}_{\mathsf{near}}}(\gamma_n)}{\text{mean}_{n \in \mathcal{C}_{\mathsf{far}}}(\gamma_n)} \ . \tag{3}$$

Empirically we found the best split to be $\alpha = 0.2$. The second feature targets the change in surface normal along the tangent. Again the N neighbours in $\mathcal{P}_i^N$ are split into two subsets, one set $\mathcal{P}^l$ for the points on the "left" side of the tangent and the other $\mathcal{P}^r$ for the "right" side. The feature is then the angle between the left and right medoid normal,

$$\mathcal{P}^l = \{\mathbf{p}_n \in \mathcal{P}^N | \langle \mathbf{p}_n, \mathbf{e}_3 \rangle < 0\} \ , \ \mathcal{P}^r = \mathcal{P}^N \setminus \mathcal{P}^l$$
$$R = \left| \left\langle \underset{n \in \mathcal{P}^l}{\text{med}}(\mathbf{n}_n), \underset{n \in \mathcal{P}^r}{\text{med}}(\mathbf{n}_n) \right\rangle \right| . \tag{4}$$

It is well-known that methods which rely on the local orientation and curvature in general benefit from multi-scale representations, and this is even more true for unevenly sampled 3D point clouds. Like [22, 3] we therefore vary the size of the neighborhood $\mathcal{P}^N$ and extract all geometric features at 9 different scales, by voxel-grid down-sampling of the raw point cloud with voxel sizes $s \in \{0.01, 0.05, 0.1, 0.2, 0.4, 0.8, 1.6, 3.2, 6.4\}$ $[m]$.

It is reasonable to assume that the definition of a contour also depends on the semantic object class(es) one is looking at, *e.g.* the transition from man-made ground surface to building wall is rather likely to form a contour, and on buildings an occluding contour is more likely to be a contour than on a tree. Moreover, it has been shown by several authors that geometric features of the type described above perform reasonably well also for the task of semantic point cloud labeling. We thus run a random forest classifier on the geometric feature set described above, with seven possible class labels *natural ground, man-made ground, low vegetation, high vegetation, buildings, scanning artifacts*, and a *rejection class* for other objects. The estimated class probabilities per point are also included in the feature vector for contour detection. Note that, although the geometric features are the same, this step does add the information that is contained in the semantic labels of the training data.

Given the complete feature set described so far, we train another, binary random forest to predict the contour score $p(c_i|\mathbf{x}_i)$ of a point $\mathbf{p}_i$, see Fig. 2.

3.2. Contour candidate generation

Given point-wise scores, we identify contour points and link them to contours following a hypothesize-and-verify strategy, *i.e.* we generate an over-complete set of *contour candidates* that is pruned to an optimal subset. The idea

Figure 3: If the horizontal point density decreases more rapidly than the vertical density, "scan lines" become visible (on the left).

is that explicitly processing contour segments (instead of points) delivers more expressive, long-range evidence along the entire line length. Also, they allow one to impose connectivity constraints to favor longer, connected contours.

Candidate generation consists of *(i)* local non-maxima suppression (NMS) to find seed points with high contour probability $p(c_i = 0|\mathbf{x}_i)$; and *(ii)* linking seed points into an over-complete graph of putative contour segments.

Voxel-grid non-maxima suppression. Traditional line detectors like Canny select points with high scores as seeds to bootstrap line tracing. However, contour likelihoods of 3D point clouds are very unevenly distributed. Consequently, simple thresholding tends to either miss contours (if set too high), or to generate too many seeds (if too low), which in turn would lead to a huge candidate set that defies further processing. Note that both cases will also result in candidate segments of greatly varying length. We therefore opt for an adaptive NMS that produces a more uniform distribution of seed points: we discard only points below a conservative threshold $p(c_i = 0|\mathbf{x}_i) < 0.5$, then perform voxel-grid filtering (with $s = 0.1$ $[m]$ spacing). Only the point with the highest score per voxel is retained. Of those remaining, points that have a neighbor in $\mathcal{P}^N$ with a higher score are removed, except for neighbors along the local tangent (approximated by the eigenvector $\mathbf{e}_1$ of the covariance tensor). Furthermore, the less confident of two points along the tangent is removed if their distance is $< 0.5 \cdot s$. With the described NMS rules, seed points will have a spacing $< 2 \cdot s$ if they lie on potential contours, and $< 3 \cdot s$ otherwise.

Graph construction. Next, seed points must be locally connected to a neighborhood graph. Recall that outdoor laserscan point clouds have a very anisotropic point distribution (Fig. 3) due to quadratic decrease in point density with increasing distance from the sensor. Voxel-grid filtering can partially reduce this effect in high-density areas, but is less efficient in the far-field. Another characteristic property that complicates neighborhood graph construction is varying point spacing in horizontal and vertical direction on slanted surfaces. Previous works usually connect points across large neighborhoods, which however quickly becomes intractable in terms of memory and computation time when dealing with millions of points. An alternative strategy is to sample random points on (estimated) surfaces [18]. This strategy seems inappropriate for contour delineation because it risks to alter lines by hallucinating points in low-density regions.

Our method first connects seed points $\mathbf{p}_i \in \mathcal{P}$ into a k-nearest neighbor graph with a low number of neighbors $k_1 = 5$, to obtain an initial edge set $\mathcal{E}_1$. Second, a larger neighborhood $k_2 = 50$ is constructed per point and reduced to a minimum spanning tree (MST) with Prim's method. MST edges of all points are added to $\mathcal{E}_1$ to yield the complete edge set $\mathcal{E}$. Finally, edge set $\mathcal{E}$ is reduced to a set of candidate contours, defined as chains of adjacent edges, by searching for up to $\beta = 15$ minimum-cost paths that connect a seed point to its neighbors within a radius $r_{\text{link}} = 0.85$ m. We define graph edge costs as

$$e_{ij} = 2 - \frac{p(c_i|\mathbf{x}_i) + p(c_j|\mathbf{x}_j)}{2} - \frac{d^*}{\|\mathbf{p}_i - \mathbf{p}_j\|} , \quad (5)$$

with $\|\mathbf{p}_i - \mathbf{p}_j\|$ the Euclidean length of the edge between $\mathbf{p}_i$, $\mathbf{p}_j$, and d^* the smallest distance from either of the two endpoints $\mathbf{p}_i$, $\mathbf{p}_j$ to all other points. In practice it is not necessary to run Dijkstra's algorithm globally over long distances, but only within the local neighborhood $2 \cdot r_{\text{link}}$ thus keeping computation efficient. Edges that are not part of any shortest path are discarded. All (overlapping) shortest-path edge chains form the set of candidate contours. We point out that more sophisticated methods have been proposed to remove unwanted edges from the raw neighbourhood graph, *e.g.* ant colony optimisation in [29]. For large graphs with millions of points such methods are computationally demanding and this step quickly becomes a bottleneck of the overall system. Local shortest-path search is a pragmatic compromise that, in our experience, gives very similar results in much less time.

3.3. Contour edge labeling

The final step of the proposed contour detector is to select a subset of the candidate edges, such that it best covers the actual contours. This is another binary labelling problem, this time on a Markov Random Field defined by the candidate graph, in which the candidate contours (rather than the original 3D points) are the variables. To that end we design a novel unary term that takes into account both the contour scores of the associated points and their spatial layout. The expectation that contours should mostly form a connected wireframe is encoded via higher-order terms that discourage free endpoints.

Unary Term. The unary term encodes statistics about both the per-point scores and the point distribution along a putative contour. For the point distribution we introduce a new descriptor, *cumulative shape context*, based on the pointwise shape context of [1]. The original shape context captures the relative locations of points in an image,

by picking one point $\mathbf{p}_i$ as the origin of a local coordinate system and building a polar histogram over the relative positions $\mathbf{v}_{ij} = \mathbf{p}_j - \mathbf{p}_i$ of other points in its neighbourhood. The procedure to gather statistics about the distribution of discrete points makes the method appealing for our data. Compared to the original application, we require statistics about an entire contour candidate, which consists of a sequence of points. Moreover, the descriptor in our case should be invariant to rotation and scale. To achieve scale invariance and at the same time simplify the descriptor to 1D, we discard relative distance and encode only directions, by normalising the vectors. Rotation invariance is achieved by projecting them onto the canonical direction $\mathbf{v}_{se} = \mathbf{p}_e - \mathbf{p}_s$ defined by the start point $\mathbf{p}_s$ and end point $\mathbf{p}_e$ of the contour candidate. To collect the directional statistics from all points along the candidate, we simply add their histograms together into a single one. Taken together, we visit each point $\mathbf{p}_i$ along the putative contour, and let each of the other points $\mathbf{p}_j$ generate a scalar value

$$v_{ij} = \left\| \left\langle \frac{\mathbf{v}_{se}}{\|\mathbf{v}_{se}\|}, \frac{\mathbf{v}_{ij}}{\|\mathbf{v}_{ij}\|} \right\rangle \right\| \quad \forall\, i,j \in \{s,\ldots,e\} : i \neq j \ . \quad (6)$$

The v_{ij} are then quantised into 5 equally spaced bins to form a 1D histogram $\mathbf{z}_1$. To also include information about the overall straightness/curvature of the contour candidate, we repeat the computation of the cumulative shape context descriptor, but this time normalise the rotation w.r.t. the local tangent (first eigenvector) $\mathbf{e}_{1,i}$:

$$w_{ij} = \left\| \left\langle \mathbf{e}_{1,i}, \frac{\mathbf{v}_{ij}}{\|\mathbf{v}_{ij}\|} \right\rangle \right\| \quad \forall\, i,j \in \{s,\ldots,e\} : i \neq j \ , \quad (7)$$

to obtain a second histogram $\mathbf{z}_2$.

To retain the pointwise contour likelihoods $p(c_i|\mathbf{x}_i)$ from section 3.1, they are again histogrammed over the points on the candidate contour, in two different ways. The first histogram simply quantises the scores directly into 5 bins to obtain $\mathbf{z}_3$. For the second one the candidate is chopped into b segments of equal length, and the mean probabilities of the segments form a further set of features $\mathbf{z}_4$, which captures the variation of the score along a candidate. Overall, a much stronger descriptor can be obtained by aggregating information along contour candidates (*i.e.* edge chains), rather than looking only locally at the level of points. In that sense the approach is similar in spirit to recent work on line feature extraction in medical imaging, *e.g.* [28] compute HOG-style features along similar candidate edges.

To obtain a discriminative unary, the descriptors $\mathbf{z} = [\mathbf{z}_1, \mathbf{z}_2, \mathbf{z}_3, \mathbf{z}_4]$ are again fed into a two-class random forest to obtain a class-conditionals $p(g|\mathbf{z})$ for being $(g_i = 1)$ or not being $(g_i = 0)$ a contour. For the subsequent MRF inference the class-conditional probabilities are converted to log-likelihoods ("energies") in the usual manner, $h_i = -\log p(g_i|\mathbf{z}_i)$.

Markov Random Field. To obtain a wireframe-type model in which the contours are long, and if possible form closed contours, we encourage contours that meet at their endpoints. Perhaps the simplest way to do so is to embed the candidate contours as variables (nodes) in an MRF. Each contour candidate $\mathbf{l}_i$ defines a clique ℓ_i together with all other candidates with which it shares either the start node or the end node.[1] Denoting the set of contours that connect to $\mathbf{l}_i$ at the start node as $\mathcal{L}_i^s$, and similarly those which meet $\mathbf{l}_i$ at the end node $\mathcal{L}_i^e$, we have

$$\ell_i = \begin{cases} \gamma & g_i = 1 \text{ AND } \forall\, \mathbf{l}_j \in \mathcal{L}_i^s, \mathcal{L}_i^e : g_j = 0 \\ & \text{(isolated contour)} \\ \delta & g_i = 1 \text{ AND } (\exists\, \mathbf{l}_j \in \mathcal{L}_i^s : g_j = 1 \text{ XOR } \exists\, \mathbf{l}_j \in \mathcal{L}_i^e : g_j = 1) \\ & \text{(continuation only on one side)} \\ 0 & g_i = 1 \text{ AND } \exists\, \mathbf{l}_j \in \mathcal{L}_i^s : g_j = 1 \text{ AND } \exists\, \mathbf{l}_j \in \mathcal{L}_i^e : g_j = 1 \\ & \text{(continuation on both sides)} \\ 0 & g_i = 0 \\ & \text{(candidate is not a contour)} \end{cases} \quad (8)$$

The cliques are of varying order, depending on how many candidates meet at the start/end points of $\mathbf{l}_i$; but there are only as many cliques as variables, and each clique depends only on the direct neighbours and can be computed efficiently. There are MRF formulations that enforce connectivity over long distances, but they need known foreground seeds and are computationally a lot more demanding [30].

The overall energy in our MRF simply reads $E = \sum_i h_i + \sum_i \ell_i$. We minimize it with Iterated Conditional Modes (ICM), because of its low computational cost [14]. Stronger inference methods like loopy believe propagation yield comparable results, with much higher runtime.

Note that there is no penalty for selecting overlapping candidates. In our setting it does not hurt if multiple contours overlap, since they are based on the same points and can be unambiguously repaired in post-processing. On the contrary, a pairwise potential that suppresses overlapping contours would be harmful, because wireframe junctions are not known in advance, and it can happen that candidates extend past a junction. In such cases, overlapping contours are necessary to recover the junction correctly.

4. Experiments

We evaluate the proposed contour detector on a large database of laserscans with a total of more than one billion 3D points. The scans were captured in different cities and villages across Europe and Asia, using professional, surveying-grade laser scanners. Visually, the results on 16 different point clouds demonstrate that our method generalises well across different scenes and locations, see Fig. 4

[1] "Start" and "end" are used for convenience, the graph is undirected.

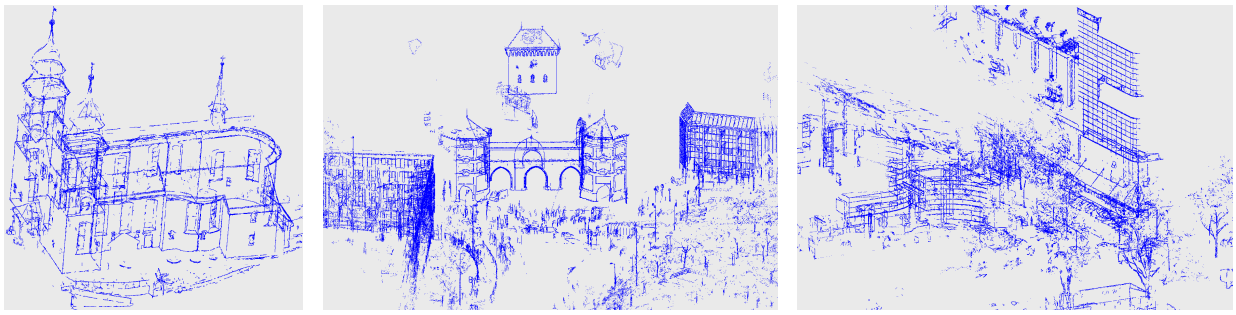

Figure 4: Contour detection results at different locations. (left) Bildstein; (middle) Munich; (right) Singapore.

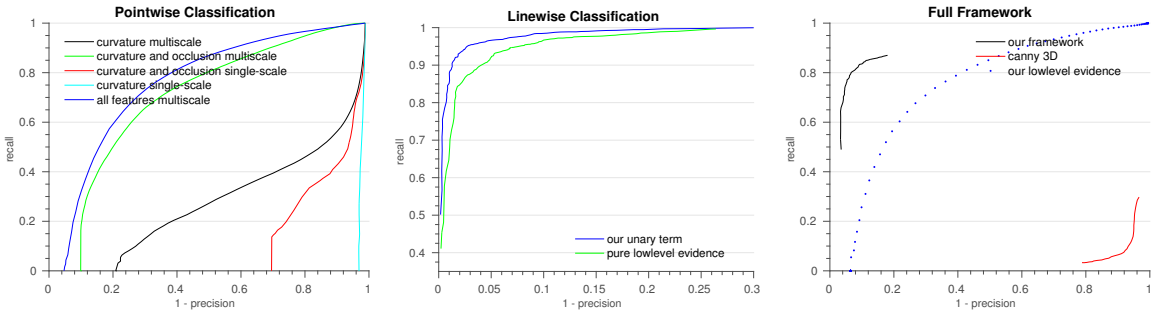

Figure 5: Precision-Recall Curves. Please note, each stage requires different ground truth, so values are not comparable across diagrams. See text for details.

for examples. One of the scans was hand-labeled to serve as ground truth, comprising $101'614$ points on contours and $7'681'061$ background points. Trees and clutter caused by moving occluder were left unlabeled to allow for a fair comparison against methods that do not have access to semantically annotated training data. This is a bias *against* our proposed method. Our contour scores (Sec. 4.1) can handle these object classes, largely thanks to the semantic class-conditionals included in the feature vector. On the contrary, methods based only on curvature, like the Canny baseline, would generate massive amounts of false positives on them, hence we exclude them from the comparison.

4.1. Pointwise Classification

As quality metrics for quantitative evaluation, we use precision-recall curves, evaluated over individual 3D points. The pointwise contour detector (Sec. 4.1) is trained on $48'964$ positive and $85'853$ negative examples taken from other cities than the test set. Our Random Forests always consist of 50 trees, their tree depth is determined automatically by grid search, with 5-fold cross-validation over the training set. Figure 5 shows the performance of our pointwise contour score, as well as standard baselines. Most related methods, which also perform some sort of edge or line detection in point clouds, threshold either curvature values or a combination of curvature and occlusion features. In particular, single-scale curvature and occlusion form the basis of RGB-D edge detection in [5], while multi-scale curvature is the descriptor used by [22]. Our goal is to compare the power of different feature sets, thus we do not set thresholds, but instead learn Random Forest classifiers also for the baseline feature sets. Together with the cross-validation described above this is more or less guaranteed to perform at least as well as a single threshold per feature dimension. In the comparison there are several interesting observations. Single-scale detectors are practically useless in large-scale outdoor point clouds, because of the strongly varying point density. Even in the multi-scale version, curvature alone is not suitable, because outdoor point clouds are never complete, thus many contours appear as occlusion edges with data only on one side. And even multi-scale curvature and contour features do not capture all important information: our full feature set still performs significantly better than the best baseline, over the entire precision-recall curve.

4.2. Linewise Classification

We go on to separately evaluate the proposed features at the level of entire contour candidates, *i.e.*, the unary term of our MRF (Sec. 3.2). Since a contour candidate spans many points, the training set for lines has only 1862 contours and 832 negatives (automatically generated contour candidates that do not coincide with ground truth contours) The Random Forest was trained with the same settings as above. Figure 5 shows the results on the test set (2227 posi-

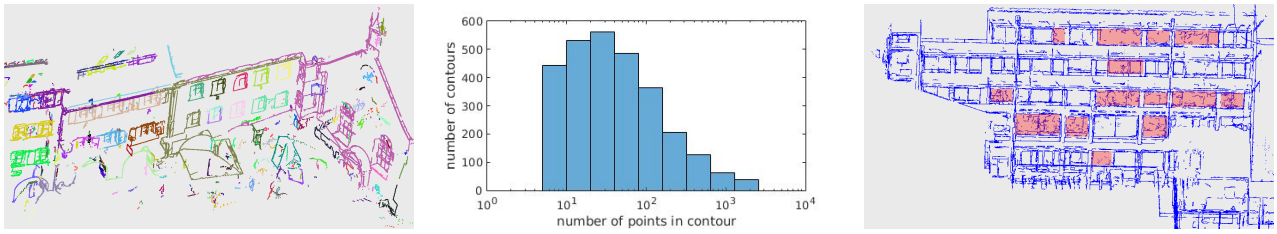

Figure 6: (left) Connected contour segments on our test set; (middle) Histogram of contour lengths; (right) A failure case: if objects with strong geometric structure did not appear in the training set, then they tend to get high contour scores.

tives, 1013 negatives). As expected, simply aggregating the pointwise scores as evidence for a contour already works reasonably well. Adding information about the shape of the contour, in the form of cumulative shape context features, improves the performance further, most notably is the high-precision regime (up to ≈20 percent points).

4.3. Full Framework

We do not have access to any direct competitor that also extracts contours in 3D point clouds. The closest independent baseline that we could find is the 3D Canny variant of [5]. That detector requires a so-called "organised point cloud", *i.e.* a depthmap with regular neighbourhood structure, both to compute low-level evidence such as depth discontinuities and to link the edge points into lines. We thus convert our test set, which stems from a single scan position and therefore comes in a regular (angular) grid, into a depth image on a cubemap.

At this point the evaluation faces a subtle, but important problem: the Canny baseline does not return line segments, but only a list of all points that form part of a line. Hence, we can only evaluate against the pointwise ground truth. Our method, on the other hand, aims to find lines. To that end it suppresses many of the individual points along a contour through non-maxima suppression (NMS), which would all be flagged as false negatives.

As a compromise, we emulate the NMS during evaluation, based on the detection result. On the one hand, we do not count false negatives that are less than 3 cm away from the nearest true positive. On the other hand, we identify false negatives in the detection results, and ignore additional false negatives within 3 cm in the ground truth. This filtering approximately corrects for point dropped by NMS during candidate generation. Still, it counts gaps of > 6 cm in the contour network as false negatives, and it also counts false positives not present in the ground truth, with their expected point count after NMS. We believe that for a relative comparison the approximation is meaningful, however note that the absolute precision/recall values cannot be compared to the one for pointwise classification. As can be seen in Fig. 5, the proposed detector vastly improves over a simple Canny edge detector on range images, which essentially fails. We have tried to tune the two thresholds of Canny for seed generation and linking separately, so as to maximize the performance on our test set, but did not manage to improve over the depicted curve.

Recall that our contours are ordered sequences of 3D data points. Colour-coded example contours are shown in Fig. 6. We currently do *not* break up contours at points of high curvature ("corners") or where more than two segments meet ("junctions"), since there is no need to do so in our application. In this setting the detected contours' median and average lengths are 28, respectively 187 points. For our specific data – urban outdoor scans in metric world units – this corresponds to 0.5 m, respectively 3.7 m. The full histogram of contour lengths is also shown in Fig. 6.

4.4. Failure cases

We have observed two main sources of failure for our contour detector. The main type of error happens at disconnected contours that are very close to each other. If the distance between two contours is less than the voxel size s of our NMS, then the candidate generation as well as the MRF prior will tend to hallucinate spurious connections.

Another problem comes from the machine learning backbone of our method. Regions that have a large amount of geometric structure, but have not been seen in the (negative) training data, tend to get high contour scores and induce false positives. Figure 6 shows a case with window blinds that were never seen during training.

5. Conclusion

We have proposed a novel approach to detect contours in unorganized 3D point clouds, which is able to handle data from complex outdoor environments. In our experiments the proposed detector work robustly across a range of laser-scanning datasets, whereas a standard Canny-like baseline badly fails. Our method is also computationally efficient enough for practical application: processing ten million points only takes a couple of minutes an a single desktop PC.

References

[1] S. Belongie, J. Malik, and J. Puzicha. Shape matching and object recognition using shape contexts. *IEEE TPAMI*, 24(4), 2002.

[2] C. Briese. Three-dimensional modelling of breaklines from airborne laser scanner data. *ISPRS Archives*, 35(B3), 2004.

[3] N. Brodu and D. Lague. 3d terrestrial lidar data classification of complex natural scenes using a multi-scale dimensionality criterion: Applications in geomorphology. *IJPRS*, 68:121–134, 2012.

[4] J. Canny. A computational approach to edge detection. *IEEE TPAMI*, 8(6), 1986.

[5] C. Choi, A. J. Trevor, and H. I. Christensen. RGB-D edge detection and edge-based registration. *IROS 2013*.

[6] Eos Systems Inc., Photomodeler. http://www.photomodeler.com/index.html.

[7] R. Fabbri and B. Kimia. 3d curve sketch: Flexible curve-based stereo reconstruction and calibration. *CVPR 2010*.

[8] S. Fleishman, D. Cohen-Or, and C. T. Silva. Robust moving least-squares fitting with sharp features. *ACM ToG*, 24(3), 2005.

[9] S. Gumhold, X. Wang, and R. McLeod. Feature extraction from point clouds. *Int'l Meshing Roundtable*, 2001.

[10] L. Guo, N. Chehata, C. Mallet, and S. Boukir. Relevance of airborne lidar and multispectral image data for urban scene classification using random forests. *IJPRS*, 66(1), 2011.

[11] Y. Guo, N. Kumar, M. Narayanan, and B. Kimia. A multi-stage approach to curve extraction. *ECCV 2014*.

[12] K. Hammoudi, F. Dornaika, B. Soheilian, and N. Paparoditis. Extracting wire-frame models of street facades from 3d point clouds and the corresponding cadastral map. *ISPRS Archives*, 38(3A), 2010.

[13] M. Hofer, M. Maurer, and H. Bischof. Line3d: Efficient 3d scene abstraction for the built environment. *GCPR 2015*.

[14] J. H. Kappes, B. Andres, F. A. Hamprecht, C. Schnörr, S. Nowozin, D. Batra, S. Kim, B. X. Kausler, J. Lellmann, N. Komodakis, and C. Rother. A comparative study of modern inference techniques for discrete energy minimization problems. *CVPR 2013*.

[15] M. Kazhdan and H. Hoppe. Screened poisson surface reconstruction. *ACM ToG*, 32(3), 2013.

[16] F. Lafarge and C. Mallet. Creating large-scale city models from 3d-point clouds: a robust approach with hybrid representation. *IJCV*, 99(1), 2012.

[17] Y. Li, X. Wu, Y. Chrysathou, A. Sharf, D. Cohen-Or, and N. J. Mitra. Globfit: Consistently fitting primitives by discovering global relations. *ACM ToG*, 30(4), 2011.

[18] Z. C. Marton, R. B. Rusu, and M. Beetz. On fast surface reconstruction methods for large and noisy point clouds. *ICRA 2009*.

[19] J. A. Montoya-Zegarra, J. D. Wegner, L. Ladickỳ, and K. Schindler. Mind the gap: modeling local and global context in (road) networks. *GCPR 2014*.

[20] A. O. Ok, J. D. Wegner, C. Heipke, F. Rottensteiner, U. Soergel, and V. Toprak. Matching of straight line segments from aerial stereo images of urban areas. *IJPRS*, 74, 2012.

[21] A. C. Öztireli, G. Guennebaud, and M. Gross. Feature preserving point set surfaces based on non-linear kernel regression. *Computer Graphics Forum*, 28(2), 2009.

[22] M. Pauly, R. Keiser, and M. Gross. Multi-scale feature extraction on point-sampled surfaces. *Computer Graphics Forum*, 22(3), 2003.

[23] C. Schmid and A. Zisserman. Automatic line matching across views. *CVPR 1997*.

[24] R. Schnabel, R. Wahl, and R. Klein. Efficient RANSAC for point-cloud shape detection. *Computer Graphics Forum*, 26(2), 2007.

[25] C. Taylor, P. E. Debevec, and J. Malik. Reconstructing polyhedral models of architectural scenes from photographs. *ECCV 1996*.

[26] Trimble, Inpho Geo-Modeling Module. http://www.trimble.com/imaging/inpho.aspx?tab=Geo-Modeling_Module.

[27] F. Tupin, H. Maitre, J.-F. Mangin, J.-M. Nicolas, and E. Pechersky. Detection of linear features in SAR images: application to road network extraction. *IEEE TGRS*, 36(2), 1998.

[28] E. Türetken, F. Benmansour, and P. Fua. Automated reconstruction of tree structures using path classifiers and mixed integer programming. *CVPR 2012*.

[29] E. Türetken, G. González, C. Blum, and P. Fua. Automated reconstruction of dendritic and axonal trees by global optimization with geometric priors. *Neuroinformatics*, 9(2-3), 2011.

[30] S. Vicente, V. Kolmogorov, and C. Rother. Graph cut based image segmentation with connectivity priors. *CVPR 2008*.

[31] M. Weinmann, B. Jutzi, and C. Mallet. Feature relevance assessment for the semantic interpretation of 3d point cloud data. *ISPRS Annals*, 5(W2), 2013.

[32] J. Xiao and Y. Furukawa. Reconstructing the world's museums. *IJCV*, 110(3), 2014.