Visual Informatics 5 (2021) 1–14

Contents lists available at ScienceDirect

Visual Informatics

journal homepage: www.elsevier.com/locate/visinf

Robust reconstruction of curved line structures in noisy point clouds

Marcel Ritter [a,b,*], Daniel Schiffner [c], Matthias Harders [a]

[a] Interactive Graphics and Simulation Group, Department of Computer Science, University of Innsbruck, Austria
[b] Airborne Hydromapping GmbH, Innsbruck, Austria
[c] DIPF | Leibniz Institute for Research and Information Education, Frankfurt, Germany

ARTICLE INFO

Article history:
Received 28 January 2021
Received in revised form 13 April 2021
Accepted 7 May 2021
Available online 14 May 2021

Keywords:
Computational geometry
Noisy point clouds
Line reconstruction
Automatic
Adaptive control

ABSTRACT

Point-based geometry representations have become widely used in numerous contexts, ranging from particle-based simulations, over stereo image matching, to depth sensing via light detection and ranging. Our application focus is on the reconstruction of curved line structures in noisy 3D point cloud data. Respective algorithms operating on such point clouds often rely on the notion of a local neighborhood. Regarding the latter, our approach employs multi-scale neighborhoods, for which weighted covariance measures of local points are determined. Curved line structures are reconstructed via vector field tracing, using a bidirectional piecewise streamline integration. We also introduce an automatic selection of optimal starting points via multi-scale geometric measures. The pipeline development and choice of parameters was driven by an extensive, automated initial analysis process on over a million prototype test cases. The behavior of our approach is controlled by several parameters — the majority being set automatically, leaving only three to be controlled by a user. In an extensive, automated final evaluation, we cover over one hundred thousand parameter sets, including 3D test geometries with varying curvature, sharp corners, intersections, data holes, and systematically applied varying types of noise. Further, we analyzed different choices for the point of reference in the co-variance computation; using a weighted mean performed best in most cases. In addition, we compared our method to current, publicly available line reconstruction frameworks. Up to thirty times faster execution times were achieved in some cases, at comparable error measures. Finally, we also demonstrate an exemplary application on four real-world 3D light detection and ranging datasets, extracting power line cables.

1. Introduction

Datasets based on points as geometric primitives have become very popular in recent years. Noisy point cloud data are, for example, obtained from different capturing devices, such as depth cameras, during stereo matching, or in light detection and ranging (LiDAR) scanners (e.g. Toth and Jóźków (2016), Foix et al. (2011)). Moreover, such data are also common in particle-based simulations (e.g. Ihmsen et al. (2014)). In this context, we currently focus on the automatic reconstruction of curved line structures from noisy 2D/3D point cloud data. To this end, we investigated the development of geometric measures for analyzing such data, as well as the automatic choice of optimal local point neighborhoods for further processing.

Our application area is the reconstruction of airborne scans. In these, data come with high noise, uncertainty, geometric diversity, as well as also at high volumes. As stated in Devore et al. (2013), the reconstruction of airborne scans has received lesser attention than that of small-sized objects. Related work often focuses on small and well-defined objects, for which normal vectors may even be known (see e.g. Lu et al. (2017)). This notion is also supported in the review in Berger et al. (2013), where the authors point out the lack of a comprehensive evaluation specific for different sub-classes of reconstruction algorithms.

In our framework curved line structures are extracted based on a mesh-free streamline reconstruction strategy (e.g. Tao et al. (2013)). Starting from automatically determined initial seed positions, important and dominant directions are identified in point cloud neighborhoods. The latter are based on weighted second order tensors of local point covariance. Size variations in the geometric features are captured through analysis over multiple scales. Based on the determined geometric measures, we automatically set algorithm parameters: the selection of starting points and integration directions, the radii of the directional

* Corresponding author at: Interactive Graphics and Simulation Group, Department of Computer Science, University of Innsbruck, Austria.
E-mail addresses: marcel.ritter@uibk.ac.at (M. Ritter), Schiffner@dipf.de (D. Schiffner), matthias.harders@uibk.ac.at (M. Harders).

https://doi.org/10.1016/j.visinf.2021.05.001

search neighborhoods, as well as the direction evaluation. A large set of varying 2D and 3D geometries serves as a test-bed for automatic analysis, including different types of noise and data holes. Reconstruction errors are quantified with metrics that compare to the original undistorted point cloud geometry. Our heuristic framework is capable of robustly reconstructing 3D curved lines from distorted point sets; with the following main contributions:

- Analysis of geometric measures in point clouds to automatically set neighborhood radii.
- Scoring functions to automatically select integration start points and optimized integration radii.
- Use of a hybrid vector field, for reconstruction via streamline integration.
- Noise rate estimation via geometric measures.
- New error measures for line reconstruction evaluation.
- Extensive automated analysis of parameter choices, such as covariance centroids and weighting factors.

After presenting the related work, we first provide an overview of the complete reconstruction pipeline in Section 3. Thereafter, we introduce the key elements of the framework in Section 4. First, the distance-weighted geometric measures are addressed; next, we present the hybrid vector field employed for streamline integration, based on Eigenvectors and angular-weighted directions; and finally, the use of multiple line-lets for streamline integration is introduced. In Section 5 we focus on the automation of the method: detecting good start point candidates and identifying optimal radii for Eigenvector pre-computation. Finally, performance evaluation and curved line reconstruction results are presented in Section 6. We introduce adequate error metrics, compare to recently proposed alternative techniques, and indicate the real-world application. In the next section we will cover prior research results, separated into several associated domains.

2. Related work

Line Reconstruction in Points Clouds: In Dey and Wenger (2001) an algorithm based on a Voronoi diagram of the point cloud was introduced; nearest neighbors were connected using an angle-to-Voronoi edge ratio and a topological condition. They were able to connect irregular point samples with sharp corners; however, existing points were connected directly, which is not appropriate for the noisy and densely sampled geometry targeted in our work. A Voronoi-based approach was also followed in Zeng et al. (2008), augmented with a human vision inspired criterion, directly connecting points. They also provide an overview of curve reconstruction algorithms, such as CRUST, or NN; and outperformed these with their DISCUR algorithm. They successfully performed line reconstructions of small point sets, including sharp corners, boundaries, and multiple components. However, they could not handle large and very noisy data, interpolation between samples, or 3D reconstructions. A method based on a uniform grid was developed in Lin et al. (2005). A sequence of fitting rectangles was computed containing points of the cloud. The center points of the rectangles were then connected and used along with border intersections to control a B-spline as curve approximation. While the method could handle jittered data, it was not robust against speckle noise; also, branches or sharp corners were not supported. Other works focused on fitting polynomials to noisy point clouds. In Ruiz. et al. (2013) a noise-adaptive smoothing term was added to the curve fitting; they employed a principal component analysis (PCA) in a pre-processing step for segmentation, and constructed piecewise curves, supporting branches and crossings. While our approach follows a different direction, similarities exist in the use of a weighted PCA, as well as the piecewise reconstruction strategy. Nevertheless, they only provided examples for 2D; and the method was hampered by outliers and data holes. Fitted B-splines were employed in Flöry (2009), where the authors extended line and surface reconstructions to avoid user-defined regions. Nevertheless, for 3D they focused on surfaces, not on lines; and branches or crossings were not targeted in their work. A line reconstruction algorithm based on half disks and an angle-weighted probability function was suggested in Philsu and Hyoungseok (2010). From a start point, additional ones were selected and concatenated into a line. The method could handle line crossings; further, when noise was introduced, lines could still be reconstructed. However, the result would always be located at the outermost border of a curved point cloud. In our approach we also employ an angle-weighted directional component. In Hasirci and Ozturk (2011) lines were reconstructed via a PCA, using an adaptive radius selection. They employed a normalized maximal Eigenvalue to decide on an optimal radius, and then generated 3rd order polynomials to reconstruct a line. Albeit, their method was limited to smooth lines and could not handle crossings, branches, or data holes. They extended their work in Ozturk and Hasirci (2013) by applying a minimal Euclidean spanning tree for point thinning. This enabled robust support for branches and crossings, but also introduced gaps at the intersections. In a different context, transmission lines were reconstructed in Jaw and Sohn (2017), via segmentation and piecewise regression. They focused on an automatic, robust, and precise method dealing with noise induced by wind; they compared to and outperformed the Hough Transform.

Most recent publications on 2D line reconstructions of point clouds can be found in Ohrhallinger et al. (2016), Ohrhallinger and Wimmer (2018), and Ohrhallinger and Wimmer (2019). First the CRUST and NN approaches were extended, focusing on very sparsely sampled data. They select a local nearest neighbor and the opposite half-space neighbor, and prove that this permits connecting up to 60° sharp corners with a larger ϵ-sampling than previous methods. The authors further extended their introduced HNN-CRUST algorithm to *FitConnect*, by first estimating a local feature size and generating new points via blending in noisy regions. An extensive analysis on many examples is provided and compared to other work on line reconstruction. They were able to successfully deal with sharp corners and varying noise. A further extension improves line smoothness. However, their focus was on 2D-manifolds; T-junctions, crossings, and 3D reconstructions were not covered.

Surface Reconstruction in Point Clouds: In the early 1990s, tangent planes computed via covariance analysis to define implicit surface functions were introduced in Hoppe et al. (1992). Using these, they reconstructed surfaces with the Marching Cubes isosurface algorithm. However, the covariance was not weighted and, thus, more prone to noise. In McIvor and Valkenburg (1997), several methods for local surface and normal estimations were compared. They analyzed the estimation performance on artificial geometries and added growing noise. They focused on quadratic fitting and mentioned a covariance-based technique, but did not include it in their comparison. A benchmark for surface reconstruction algorithms was provided in Berger et al. (2013), comparing ten state-of-the-art approaches for point clouds with defined normals; among them: compactly supported radial basis functions (CSRBF) (Ohtake et al., 2005; Wendland, 2005), simple point set surfaces (SPSS), implicit moving least squares (IMLS), multilevel partition of unity (MPU). The results showed that there was no superior, general technique for surface reconstruction. Polynomial basis functions for distance weighting, as introduced e.g. for CSRBF in Wendland (2005), were also included as weighting candidates in our work. In Giraudot et al. (2013) they also aimed at the reconstruction of point sets, automatically adjusting to locally varying sampling and noise properties. A noise adaptive

distance function was employed and an implicit function was found as a zero iso-line (or surface). The method yielded robust results to outliers and jittered point sets, but differed in that it required closed smooth shapes as prerequisite.

Tensor and Covariance Techniques: An algorithm to compute Gaussian and mean curvature on polygon meshes was presented in Taubin (1995), where a tensor product of direction vectors to neighboring vertices was introduced to estimate a surface curvature. They constructed a curvature represented by a 2×2 matrix in the local surface tangent plane. The method is limited to meshes and not applicable to 3D point cloud data. In Berkmann and Caelli (1994) a 2nd fundamental form and a Gauss map estimation of surfaces was introduced, based on covariance in a local neighborhood. They detected regions in depth images and distinguished between planar, parabolic, and curved segments. They indicated that their method was robust to noise, due to the use of covariances, instead of the closed fundamental form. However, this only holds for low noise ratios. A similar weighted product to compute a covariance matrix in point cloud neighborhoods was proposed in Alexa and Adamson (2004) to estimate normals. The method operates on a point cloud, but the points are known to originate from a surface geometry, again, with low noise in the data. In Liu et al. (2012) a voting technique was applied to make the normal vector estimation robust against noise. They employed shape factors for coloring and as a confidence measure. In our work, the shape factors are defined differently; our method achieves robustness by weighted covariance.

Line Following in Tensor Fields: Lines extracted from diffusion tensor fields have been used in the medical domain to visualize features in magnetic resonance images. Here, the connection between different regions of the brain are of interest, illustrated by fiber tracking. Early approaches employed Eigenvector streamlines following the major Eigenvector of the 2nd order tensor (Basser et al., 2000), or a vector tensor multiplication to compute the next step direction (Chou et al., 2006). Crossing fibers could attract streamline integration to switch to a different fiber. Thus, extensions have been proposed to favor the original direction, when at a crossing (Weinstein et al., 1999). We also investigate direction selection strategies, but operate on mesh-less data instead of the uniform grids in diffusion tensor works.

Point Cloud Classification: In Natale et al. (2010) shape factors of a PCA of indoor 3D flash LiDAR images were analyzed by a decision network for point classification. They parameterized shape factors by the number of neighbors in rather small neighborhoods. We use different shape factor definitions and parameterize by radius. Similarly, also in Weinmann et al. (2015) shape factors were employed as input to a random forest classifier. They improved their true positives by 3% via relying on the geometric median in the data.

3. Reconstruction pipeline overview

Our framework for the extraction of curved line structures in noisy point cloud data is composed of several individual key steps (see also Fig. 1):

1. **Geometric measures**: Geometric measures are locally computed via weighted covariances, in multi-scale neighborhoods of the point cloud data. An optimal neighborhood radius is determined in a point-wise fashion, based on these local measures.
2. **Start points**: Seed positions for the piecewise streamline integration are automatically selected based on the previously computed measures.
3. **Grow lines**: From the start points line-lets are grown in parallel, following streamlines in a hybrid vector field. They follow point samples, which are likely to have been sampled from continuous line geometry.

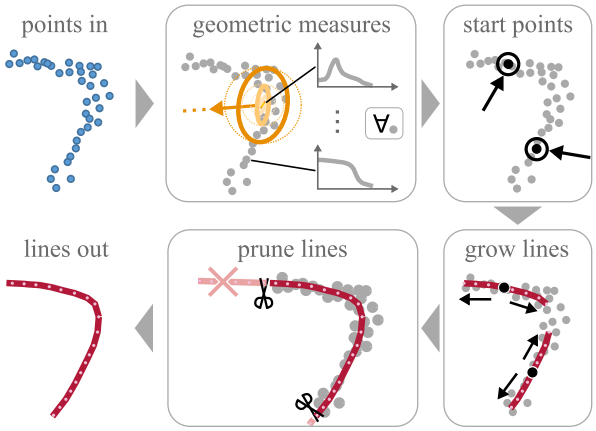


Fig. 1. Main processing steps of the reconstruction pipeline.

4. **Prune lines**: The growing line-lets are tested for crossings and corners, and extended to overcome data holes. Finally, they are pruned, and then concatenated into curved line structures as output.

Each pipeline step involves a number of parameters for control of the outcome. For most of these, we have carried out extensive automated initial tests to determine and fix best performing parameters. A few parameters remain that can be manually adapted, if desired, providing still some user-control over the pipeline outcome (see Section 5.3). In the following, we will first describe the individual steps in more detail, and then address the automation.

4. Reconstruction of linear structures

4.1. Geometric measures

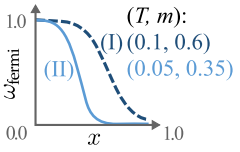
A central element of our reconstruction framework is numerical *geometric measures*, characterizing local *linearity*, *planarity*, and *sphericity*; following Westin et al. (Westin et al., 1997). To obtain these measures for any location in a point cloud we first calculate weighted second order tensors $\underset{\sim}{t}$, which are constructed from all direction vectors within a neighborhood:

$$\underset{\sim}{t} = \frac{1}{\sum_{i=1}^{N} \omega(d_i)} \sum_{i=1}^{N} \omega(d_i)(\boldsymbol{v}_i \otimes \boldsymbol{v}_i), \quad \text{with} \tag{1}$$
$$\boldsymbol{v}_i = \boldsymbol{p}_i - \boldsymbol{c}, \quad d_i = |\boldsymbol{v}_i|/r,$$

where $\boldsymbol{c}$ is a specifically determined 3D center location (denoted below as *centroid*), $\boldsymbol{p}_i$ a point in the local neighborhood of $\boldsymbol{c}$ within neighborhood radius r, N the number of neighbor points for that radius, d_i normalized distances, and $\omega(x)$ a radial distance weighting function. Note that without using the weighting function and employing the mean position for $\boldsymbol{c}$, we would obtain standard covariance matrices of the local point cloud as these tensors, on which then a principal component analysis could be carried out. Also note that distance queries and computations in the point cloud are accelerated with an octree data structure.

We tested thirty different normalized scalar functions for the weighting in Eq. (1). Possible candidates were, for instance, quadratic, cubic, compactly-supported RBF, or SPH smoothing kernels. In an initial controlled, automated study we computed reconstruction errors for these candidate functions; varying parameters such as neighborhood radius, number of points, jitter noise strength, etc. In total, 1.35 million parameter combinations were automatically tested, on a circle and a rectangle test shape, in about 7 hours of run time. In the end, the best performance

was found for a weighting function similar to the Fermi–Dirac distribution (originating from quantum statistics, but here without physical constants) (McDougall et al., 1938):



$$\omega_{\text{fermi}}(x) = \frac{1}{e^{(x-m)/T}+1}. \tag{2}$$

The shape of the graph is controlled by two parameters $T \in [0.0, 0.4]$ and $m \in [0.0, 1.0]$. The former blends from a step to a linear function, while the latter shifts the function along the x-axis. In our automated experiments, we found weighting curve (I), with parameters $(0.1, 0.6)$ to be best for the tensor computation (also see plot above). Note that further below we will also make use of a similar weighting function for mesh-less interpolation; in that context, curve (II), with parameters $(0.05, 0.35)$ performed best. These values resulted from analyzing scatter-plots of the best reconstructions in the automatic testing mentioned above.[1]

Further stabilization against noise can be achieved by choosing a suitable method for determining the centroid $\boldsymbol{c}$. Several strategies in the tensor computation above have been proposed in the past. Especially the presence of noise has a strong influence on the computation, which makes using the mean (MN) not necessarily the optimal choice. Eq. (1) is a weighted PCA when employing the MN. Another option is to instead employ the geometric median (MD), which has been found to be robust against noisy data (Lipman et al., 2007; Lin et al., 2014). Further, we also investigated the option to introduce a weighting into the mean and median computation (WMN, WMD); the centroid locations then vary dependent on the chosen weight function. For computing the geometric median we employ the Weiszfeld algorithm (Plastria, 2011), which makes use of an exponential function in its iterations. Adding a weight function in the algorithm controls the strength of the iterative shift. Such weighted centroid variants yield locations in-between the geometric median and the mean. Finally, a further option could be to use the actual points of the point cloud itself as the "centroids" in the tensor estimation, thus skipping any mean or geometric median computation; this yields the so-called point distribution tensor (PDT) (Ritter and Benger, 2012). We compared these different options in performance tests with varying 3D test geometries (see Section 6.5). As a generally well-performing candidate, WMN with quadratic inverse stood out.

Employing the WMN as centroid and the Fermi–Dirac weighting function in the tensor calculation, we finally obtain our geometric measures. An Eigenvector decomposition of $\underset{\sim}{t}$ yields measures for local linearity, planarity, and sphericity, according to Westin et al. (1997):

$$C_L = (\lambda_3 - \lambda_2)/L, \quad C_P = 2(\lambda_2 - \lambda_1)/L, \quad C_S = 3\lambda_1/L, \tag{3}$$

with Eigenvalues of $\underset{\sim}{t}$: $\lambda_3 > \lambda_2 > \lambda_1$, $L = \lambda_1 + \lambda_2 + \lambda_3$, and associated Eigenvectors $\boldsymbol{e}_3, \boldsymbol{e}_2, \boldsymbol{e}_1$. Both the Eigenvalues and Eigenvectors will be central for the reconstruction of the curved line structures via vector field integration. It should be noted that both depend on the neighborhood radius, which controls the set of points considered for their computation. Both can vary significantly depending on how the neighborhood radius is selected. Therefore, we explored the measures further, as functions of radii.

[1] Further details available in the supplementary material.

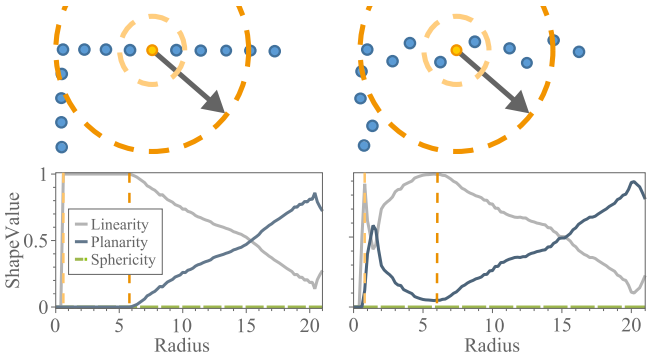


Fig. 2. Geometric measures, computed at a centroid in small example 2D point clouds, with and without noise (top). The geometric measures (bottom) change with the radius; linearity is initially high and decreases when a corner is reached.

4.2. Dependence on neighborhood radius

In the tensor computation above one could employ a fixed radius for the neighborhood of a certain point. However, depending on local feature scale this may or may not include representative geometric information. Moreover, if 3D jitter noise is present, with an amplitude larger than the radius, the sphericity measure will be dominant, independent of other structures. Thus, it will be challenging, or even impossible, to find only a single radius that is optimal in all cases; especially, when sampling resolution changes; or linear features occur at different scales.

The change of the geometric measures dependent on a selected radius is visualized in Fig. 2; the three measures are computed for two exemplary 2D point clouds (top), yielding graphs as functions of growing radii (bottom). Note that for this 2D case the sphericity measure yields zero; while the sum of the measures is one. The centroid (orange), for which the measures are computed, is located on a part of a point-sampled rectangle corner (blue), once without (left) and once with jitter noise (right). In the former case, as soon as the radius includes at least two neighboring points the linearity graph reaches a maximum of 1.0, and remains there until the growing radius covers the corner (stippled, larger dark-orange circle). At this point the linearity starts to decrease, while planarity increases. In the latter, noisy case, an initial peak at small scale marks the radius, at which the two closest points are covered, yielding high linearity. This measure decreases down to the jitter noise amplitude; and then increases, also until the corner is reached. Using different weighting functions in Eq. (1) results in changes in the linearity graph. Therefore, we also evaluated the fitness of the weighting functions with regard to properties of the linearity graph: the maxima/minima at small radii, the plateau at large radii, and the overall smoothness of the resulting graph. Regarding the smoothness, the Fermi–Dirac (I) weighting showed a good behavior, especially, when employing the MD, WMD, and WMN centroids. Quadratic weighting also performed well, but occasionally produced discontinuity artifacts in the geometric measure graphs. Further, a direct visualization of the multi-scale space was developed illustrating different parameter choices Ritter et al. (2021).

4.3. Vector field integration

In order to reconstruct curved line structures in point clouds, we employ vector field integration (see e.g. Tao et al. (2013)). This denotes streamlines evolving along a certain direction in the mesh-less data, following the original geometry. Integration steps are numerically calculated using an explicit Runge–Kutta method. We tested various variants up to order five, but the influence of

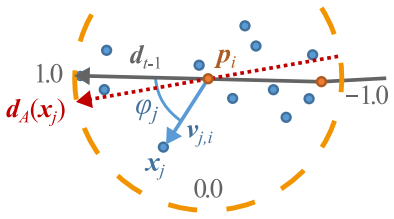


Fig. 3. Angle-weighted direction vectors; the previous integration direction (gray) is used as reference. With it, cosines of angles are employed, obtained via dot products with all neighbor directions (one example shown in blue). The final direction (red) is the normalized mean of the weighted vectors in the neighborhood.

this was found to be small; thus, for our experiments we rely on a simple and fast third order RK32 scheme.

For the integration process, local direction vectors are required, which will be determined based on the previously computed Eigenvectors (note that the latter are bidirectional and may have to be flipped). We have analyzed different methods for computing directions; in the end two were combined into a hybrid approach.

Since we work with mesh-less data we have to either: interpolate between directions computed at existing data cloud points $\boldsymbol{p}_i$ or to directly compute a direction at an arbitrary position $\boldsymbol{x}_j$. The first approach for finding an integration direction at a location $\boldsymbol{x}_j$ computes an average of neighboring major Eigenvectors; weighted based on normalized distances:

$$\boldsymbol{d}_E(\boldsymbol{x}_j) = \frac{1}{\sum_{i=1}^{N} \omega(d_i)} \sum_{i=1}^{N} \omega(d_i)\, \boldsymbol{e}_{3,i}, \tag{4}$$

with major Eigenvectors $\boldsymbol{e}_{3,i}$ of N neighboring points, given by the selected radius; and d_i the normalized distances. The Eigenvectors are aligned according to the direction used in the previous integration step $\boldsymbol{d}_{t-1}$; i.e. they are flipped if they oppose the current streamline major direction. For the weighting function $\omega(.)$ we employ the Fermi–Dirac (II) function mentioned above, which led to slightly improved results in the interpolation. This first approach is useful for noisy data and larger scale features, providing a very smooth and robust direction selection. Since the Eigenanalysis has already been performed, no additional computation is required.

A second option for finding an integration direction is to employ the (normalized) sum of angle-weighted local difference vectors:

$$\boldsymbol{d}_A(\boldsymbol{x}_j) = \mathrm{nrm}\left(\sum_{j=1}^{N} \omega\left(1 - \frac{\cos(\varphi_j)+1}{2}\right) \boldsymbol{v}_{j,i} \right), \tag{5}$$

with $\boldsymbol{v}_{j,i} = \boldsymbol{x}_j - \boldsymbol{p}_i$ being difference vectors, between location $\boldsymbol{x}_j$ and N neighboring points $\boldsymbol{p}_i$, again for a given radius. The cosine term is computed using the angles φ_i between the normalized vectors $\boldsymbol{d}_{t-1}$ and $\boldsymbol{v}_{j,i}$ (see also Fig. 3); finally, $\omega(.)$ again is the Fermi–Dirac (II) weighting function, this time evaluated with the angle-based term. Directions computed with the second method stay closer to the geometry, and overall reconstruction performance is better for smaller-scale features. As this method includes angle computations at the current streamline integration position it must be executed at "run-time" for each integration step. Later, we will obtain the final integration direction as a weighted average of $\boldsymbol{d}_E$ and $\boldsymbol{d}_A$ (see Section 5.2). For this, optimal weights and radii will be automatically determined based on the pre-computed geometric measures.

4.4. Piecewise reconstruction

As an improvement of the previously outlined vector field integration, we further propose to grow multiple, smaller streamlines simultaneously, in breadth first fashion. Starting from different points, integrations are evolved in parallel, both in forward and backward direction, yielding so-called *line-lets*. In each integration step the evolving line-lets are checked for various conditions (and possibly terminated): i.e. exceeding a maximum step count, exiting the bounding box of the point cloud, passing beyond a maximum distance threshold, colliding with another line-let end, or intersecting another line-let segment. In case of a collision, line endings are merged; in case of an intersection, line endings are placed at the intersection point. In 3D, intersections of skew lines (i.e. segments) are determined by checking their closest points. Line segment endings and closest skew line segment points are merged, when their distance falls below a distance threshold: $\Delta_{\mathrm{mrg}} h$, where h is the integration step size and Δ_{mrg} a weighting factor (set to 1.4). Furthermore, if during the integration a line direction from one step to the next changes by more than 70°, a new line-let is seeded and a branch created. Endings at intersections, collisions, as well as branch seeds are labeled as *closed*; whereas line segment endings generated due to exceeding the bounding box or the distance threshold are labeled as *open*.

In a final (pruning) step, points are deleted from all line-lets, starting from the open endings. The deletion process propagates backwards until the distance to the closest point in the point cloud is smaller than a user defined threshold. A second user controlled parameter is the maximum integration distance. Increasing the latter enables to overcome data holes and decreasing the former allows to delete long open branches. Finally, note that for efficient intersection computations, the points of the line-lets are also organized in a separate octree. In the next section, the automation of the process will addressed.

5. Process automation

5.1. Selection of start points

A key element in our piecewise reconstruction is the setting of start points for multiple line-let creation. We propose to automate this according to the linearity graphs introduced above. Points for starting the vector field integration should be located on "good" linear regions in a point cloud. Thus, we analyze each linearity graph to determine such locations. As outlined above, linearity measures are determined for different radii. First, in order to accelerate the process, and to focus on features at different scales, we propose to increase the search radii r_k for this computation according to an exponential function (instead of a linear increase):

$$r_k = \Delta_{mdn}\ 1.5^k, \tag{6}$$

where k can be considered as a discrete index to the set of examined radii, proportional to $\log(r)$; the parameter Δ_{mdn} denotes the (approximate) median minimum distance between any two points in the point cloud (computed either for the whole cloud or for a random subset). Distances are computed for the Nth-closest neighbors (we employed $N = 6$). Based on these, the median for Δ_{mdn} is determined.

We found that the integral area underneath a specific part of the linearity graph C_L already represents a robust feature for start point selection. To illustrate this Fig. 4 shows graphs for six locations on an exemplary noisy point cloud of a rectangle (note that the abscissas in the subplots employ a log scale for r_k). Locations on a linear portion of the cloud exhibit a larger integral

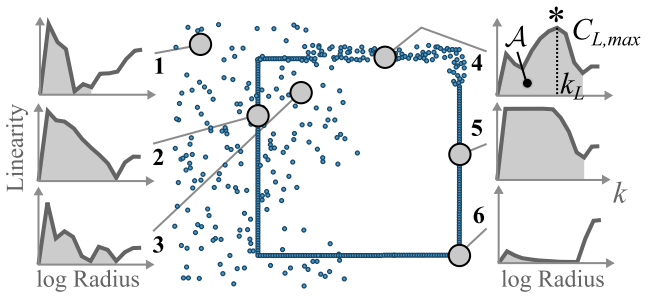


Fig. 4. Six linearity graphs on a rectangle at distinct locations. The abscissas employ a log-scale for the radii. The approximated integrals $\mathcal{A}$, up to the last local minimum, as well as the maximum $C_{L,max}$ are used for start point detection.

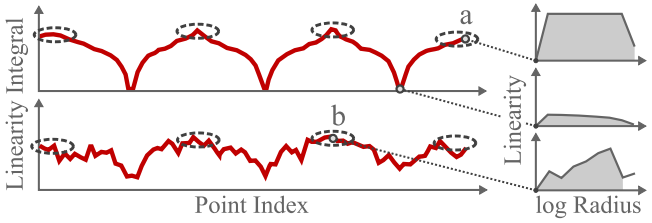


Fig. 5. (Left:) Linearity integral sum values over (ordered) point indices, determined for the two point clouds shown in Fig. 6. (Right:) Examples of linearity graphs for three selected points. Locations with a high sum (indicated by ellipses) are in linear regions and considered as good start points for the streamline integration.

(see 2, 4, 5). In contrast, corner points (6) and points in noise (1, 3) typically show a smaller integral and graph maxima. Thus, we propose to estimate an approximate integral $\mathcal{A}$ of the linearity graph C_L for points in the cloud. It is computed by summing up linearity values for (exponentially growing) radii r_k, from zero to the last occurring local minimum in the graph; numerically approximating the analytic integral (shaded gray areas in plots). Note that from a certain maximal radius, linearity values will remain constant.

As a further illustration of the proposed feature, Fig. 5 shows the linearity integral values computed at the centroids of the open rectangles depicted in Fig. 6. This example was computed without (top) and with noise (bottom). As can be seen, the areas with large integral values (marked by ellipses) remain relatively robust. Therefore, large values of $\mathcal{A}$ may yield points in linear regions.

Based on the above, to obtain good start points we employ a score function, which can be computed either for all points or for a reduced random subset (initially with distance parameter d set to 1.0):

$$O_s = d \cdot N_1^{0.01} \left(C_{L,max} \cdot \mathcal{A}\right)^4, \tag{7}$$

where $C_{L,max}$ is the maximum linearity value in the examined interval of radii and N_1 the number of neighbors in the first non-empty neighborhood encountered (when growing the radius). Thus, orphaned points are slightly down-voted. Further, in this specific case a small computational improvement could be achieved by using the PDT for computing $C_{L,max}$, instead of using one of the other centroid computation approaches.

Employing the scoring function, we determine the highest-scoring initial point; setting $d = 1$. Next, using the latter as a seed, we then progressively identify additional start points, by evaluating the same (cached) equation, now with d being set to the shortest distance to all so far selected start points. Note that candidates can be skipped if a minimum number in N_1 is not

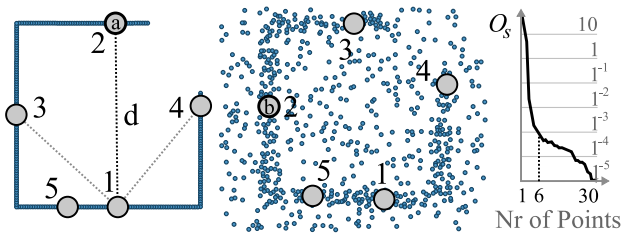


Fig. 6. Start points selected for two example geometries — without and with noise. Points are chosen dependent on their score O_s; numbers indicate the order of selection. Linearity integrals at locations (a) and (b) are shown in Fig. 5. On the right, the decreasing scoring function for consecutive candidates is depicted.

reached (by default 2 points). In very noisy data, employing a minimum number of 6 points improved the results. The candidate in the current points with the highest score O_s is then added to the set of selected start points, and the process is repeated with the remaining ones. The process stops if a maximum number of starting points has been found.

Fig. 6 illustrates identified start points on two geometries, one without and one with jitter noise. The numbers in the plot indicate the order in which candidates were selected according to the score O_s. As can be seen, the first start point is found in both cases at the center of a long rectangle edge, while corner locations and noise points are avoided. For the initial point (1), distances to start point candidates are shown by dotted lines. Larger distance is favored. Therefore, start points remain at a reasonable distance from each other. The graph on the right shows the decreasing score O_s for a progressively increasing number of selected start points. The score function is designed multiplicative, since the scale of d cannot be normalized; this still permits a relative sorting.

5.2. Radii and weights for integration direction

As indicated above, the final streamline integration direction for the current step $\boldsymbol{d}_t$ will be determined as a weighted average of vectors $\boldsymbol{d}_E$ and $\boldsymbol{d}_A$:

$$\boldsymbol{d}_t = \mu\, \boldsymbol{d}_E + \big(1-\mu\big)\boldsymbol{d}_A, \tag{8}$$

with μ being a blending weight. The computation of both direction vectors depends on the neighborhood radius, which we also propose to determine automatically. Similar to the automatic start point selection, the graph of the linearity measure can be utilized for this.

After a single numerical integration step, the streamline end position will usually not be located exactly at a point of the point cloud. Therefore, we first find the point $\boldsymbol{p}_i$ closest to the current integration position $\boldsymbol{x}_t$, within a search radius of $r_t = 1.25 \cdot \Delta_{mdn}$. At this point the pre-computed linearity graph will be further examined. This permits finding an optimal neighborhood radius, for the current step in the streamline integration process.

Large radii would be a good selection in point cloud regions with noisy and coarse structures, while for regions with fine structures small radii should be chosen. As an example for the former, consider the maximum in the linearity graph for point 4 in Fig. 4 (marked with a *); for the latter, consider the (first) maximum in the graph at location 6 in Fig. 8. Thus, the analysis of the extrema in the linearity graph will be a means for automatically selecting the neighborhood radius.

Linearity graphs will generally exhibit several local minima and maxima at varying radii; typically, up to four extrema were present in our examined geometries. Accordingly, we designed a second scoring function. It is evaluated for each local maximum

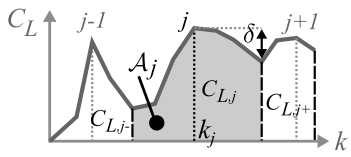


Fig. 7. Elements of scoring function, based on local maximum at k_j and neighboring minima at k_{j-} and k_{j+} of the linearity graph.

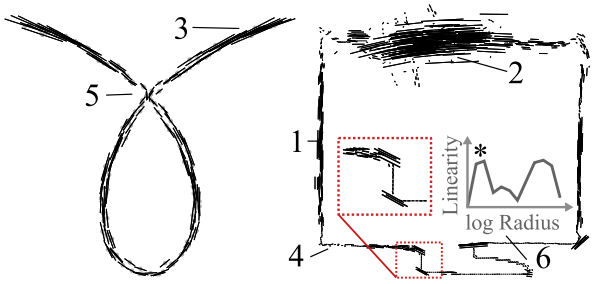


Fig. 8. Optimal radii and directions $\boldsymbol{d}_E$ shown via line segments on two exemplary point cloud geometries.

in the linearity graph, at the corresponding radius (with index k_j) for that specific maximum:

$$O_m = \left(1 - \frac{k_j}{K}\right) C_{L,j} + \frac{\mathcal{A}_j}{\mathcal{A}} - \frac{\delta}{2}. \tag{9}$$

Here, j denotes the radius index of the currently examined local maximum, for the radius given by k_j. $C_{L,j}$ denotes the respective linearity value, and K the total number of discrete radii indices considered for the current linearity graph. Further, $\mathcal{A}$ and $\mathcal{A}_j$ are again (numerically approximated) integrals of the graph. The former spans the same interval of radii, as explained above; however, the latter is only computed for an interval between the closest local minima, to the left and to right of j (indicated by $j-$ and $j+$, respectively). Finally, δ denotes the difference between $C_{L,j}$ and the larger of the two linearity values at the neighboring local minima, i.e. $\delta = C_{L,j} - \max(C_{L,j-}, C_{L,j+})$. For better illustration, the involved quantities are visualized in Fig. 7.

The scoring function is comprised of three terms, which can take values ranging from 0.0 to 1.0; higher being better. Note that in the first term small radii are preferred (i.e. small k_j), since it is beneficial to adjust to small-scale features if they are present. Additionally, larger linearity values $C_{L,j}$ generally also indicate good radii candidates. In the second term, large local integrals $\mathcal{A}_j$ are favored, which also hints at linear structures. Finally, peaky maxima will be voted down via δ in the third term. The score O_m will be computed for all local maxima, and the radius associated with the highest score will be retained for the computation of $\boldsymbol{d}_E$ and $\boldsymbol{d}_A$. In order to illustrate this step, Fig. 8 depicts obtained radii and directions $\boldsymbol{d}_E$ for two example point clouds. Line segments are shown, each with length given by the determined optimal radius and direction by vector $\boldsymbol{d}_E$. The radius, and thus length of line segments, increases in regions of large linear structures (1), in noise (2), and at low curvature (3); in contrast, it is smaller at corners (4), crossings (5), and fine details (6).

We blend the two direction vectors via μ. When the linearity and the integral at the chosen optimal local maximum j is large, then $\boldsymbol{d}_E$ would be a stable choice; otherwise $\boldsymbol{d}_A$ should be prioritized to follow close-by geometry. Thus, we propose to automatically set the blending parameter as:

$$\mu = \frac{1}{2}\,\frac{\mathcal{A}_j}{\mathcal{A}} + \left(C_{L,j} - \frac{1}{2}\right). \tag{10}$$

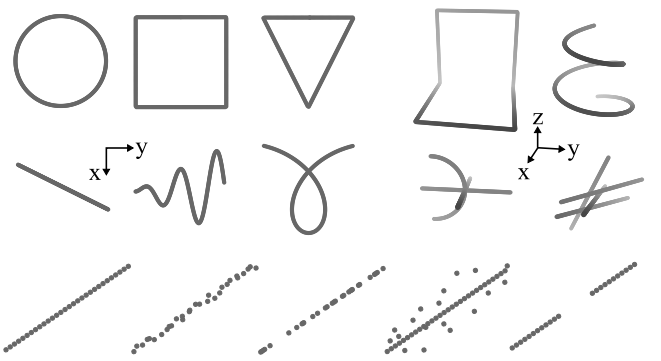


Fig. 9. (Top/Left): Six test "2D" geometries: Circle, Rectangle, Triangle, Line, Wave, and Crossing. (Top/Right): Four 3D geometries: Elbow, Mikado, Helix, and Crossing3D (depth indicated by gray gradient). (Bottom): Added noise types; from left to right: undistorted reference, jitter noise, distribution noise, outlier noise, and hole(s).

Furthermore, to ensure that both methods for direction selection still influence the final outcome, we additionally clamp this blending factor to the interval [0.05, 0.95].

Finally, the step size of the numerical streamline integration is also automatically set to $h = 0.5\ \Delta_{mdn}$; note that this ensures on average a neighborhood size of about six points. In order to improve computational efficiency, we carry out in parallel the computation of geometric measures and Eigenvectors, as well as the radius selection. The direction $\boldsymbol{d}_E$ is determined at the current integration neighborhood and interpolated on the pre-computed Eigenvectors using the Fermi–Dirac (II) weighting. To compute $\boldsymbol{d}_A$ the previous integration direction and current streamline position have to be known; thus, it is obtained progressively.

5.3. User control

While our framework exhibits several parameters, we propose to fix most of them. This is achieved either through the outlined automatic selection processes, or through extensive prior automated parameter analysis runs. In our current framework, a user has to manually control mainly three parameters:

- *MaxIterations*: The maximum number of streamline integration steps.
- *StartPointsNr*: The number of start points for line-let integration.
- *DistanceCutoff*: The maximally allowed distance from an integration point to the closest data point; used to overcome holes.

All remaining parameters are pre-defined; albeit, a user could adjust them still, if desired, for additional control. Examples are the pruning distance *DistancePrune*: the maximally allowed distance from a line-end to a closest point in the cloud, as well as *MaxRadius*: the maximum geometric analysis radius (set to $60\ \Delta_{mdn}$). We provide a user with an interactive visualization interface, for easy control and assessment of the reconstruction.

6. Analysis and results

6.1. Point cloud test geometries

For our analysis we employ six "2D" and four 3D test geometries: *Circle*, *Rectangle*, *Triangle*, *Line*, *Wave*, and *Crossing*; as well as *Elbow*, *Helix*, *Mikado*, and *Crossing3D* (see Fig. 9); these exhibit features such as varying curvatures, sharp corners, and line crossings. All are sampled as 3D point clouds (the "2D"

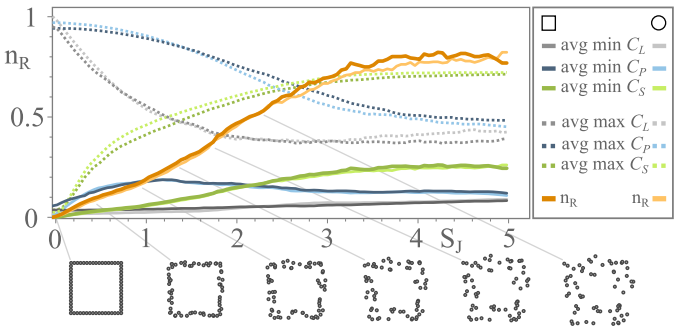


Fig. 10. Noise rate n_R (orange), as well as averaged minimum and maximum geometric measures per jitter noise – for linearity (gray), planarity (blue), and sphericity (green) – for a rectangle (saturated colors) and a circle (desaturated colors) point cloud. The averaged minimum sphericity grows linearly at lower jitter values. Thus, it was chosen as a noise rate measure and scaled for normalization, see Eq. (11).

geometries are displaced out of the plane with a cosine function). Different types of noise were added to these geometries; we employ four categories, inspired by noise that may occur during real-world data capture: unsteady trajectories and vibrations may produce *jitter* noise; shadowing or multiple scans may introduce density *distribution* noise; levitating particles or other small objects may produce random *outliers*; intensity cut-offs in sensors or occlusions may lead to *holes*. Fig. 9 also illustrates the effects of these on a line segment (bottom). All effects are controllable in our framework by appropriate parameters: jitter amplitude S_j, distribution blend S_d, data hole start t_a and end t_e, and the number of additional, random outliers M.

6.2. Noise estimate

Below, we will test the reconstruction process on various noisy example geometries. Note that only for our own, artificially generated data these noise parameters will be known exactly. In contrast, for arbitrary point clouds the latter have to be estimated. We found that the mean of all sphericity graph minima directly related to the noise in the data. Therefore, we propose a metric quantifying noise strength n_R in arbitrary data:

$$n_R = c \cdot \frac{1}{N} \sum_{i=1}^{N} \min_r C_{S,i}(r), \tag{11}$$

with $C_{S,i}(r)$ denoting sphericity of point with index i, at multi-scale radius r. Further, a constant scaling factor $c = 3.15$ was included, normalizing n_R for our test cases to interval [0.0, 1.0]. Fig. 10 illustrates the change of this noise estimate, with respect to increasing 3D jitter noise. Plots are shown for the "2D" test cases of Rectangle and Circle. Moreover, also the change in minimum and maximum of the three shape measures is indicated. At the bottom, a view of the rectangular point cloud with increasing noise is provided. For both geometries, n_R initially increases mostly linearly, later plateauing for higher, more extreme noise.

The noise rate n_R is independent of distribution noise and data holes. Below we will use this measure for evaluating and comparing results, dependent on noise magnitude. As indicated, also real-world datasets, for which the noise parameters are not known, could be compared in this way.

6.3. Error metrics

In order to evaluate the automatic reconstructions in noisy point clouds, we require appropriate error metrics. An option would be to adapt existing error measures, such as in Berger et al. (2017), to our case of curved line geometries. Nevertheless, we decided to develop metrics tailored to point-sampled linear structures, comparing differences to a ground truth in geometry, reconstruction length, as well as amount of coverage.

First, geometric differences are quantified using an adapted Hausdorff metric and an average minimal distance metric. For comparison, we employ as ground truth equidistant point samples Λ on the (known) test geometries. These are compared to our reconstructions given by the streamline integration points Ω, obtained with constant step size. For these we compute two errors — firstly, the point-based Hausdorff metric:

$$\mathcal{E}_H = \max\left\{ \max_{x\in\Lambda} \min_{y\in\Omega} d(x,y), \max_{y\in\Omega} \min_{x\in\Lambda} d(x,y) \right\}, \tag{12}$$

with $d(.,.)$ being the Euclidean distance between two points; and secondly, an average minimal distance metric:

$$\mathcal{E}_V = \frac{1}{N_\Omega + N_\Lambda} \left(\sum_{x\in\Lambda} \min_{y\in\Omega} d(x,y) + \sum_{y\in\Omega} \min_{x\in\Lambda} d(x,y) \right), \tag{13}$$

with N_Ω and N_Λ being the number of points in both compared sets. These two metrics indicate the local geometric quality of a reconstruction.

As these two metrics remain stable even if only a part of a point cloud is reconstructed, we developed two additional measures, accounting for the reconstruction completeness.

The first is given as the ratio between the (estimated) length of the reconstruction and the (known) length of the original line:

$$\mathcal{E}_{Len} = \text{len}(\Omega) \,/\, \text{len}(\Lambda), \quad \text{with} \tag{14}$$
$$\text{len}(\Omega) = \sum_{i=1}^{N_\Omega - 1} |\boldsymbol{x}_{i+1} - \boldsymbol{x}_i|,$$

with $\boldsymbol{x}_i$ being consecutive points of the reconstruction. If the reconstructed line is shorter than the original one, then $\mathcal{E}_{Len} < 1.0$; which we consider as *incomplete* coverage. Moreover, if $\mathcal{E}_{Len} > 1.0$, then the reconstruction may, in some form, cover the original geometry several times (note that it may still not be fully reconstructed).

The second completeness metric determines the amount of coverage. For each line-let a corresponding arc-length interval $[s, e]_l$ is computed, with s and e being the start and end parameters (in arc-length) of Λ, and l the line-let index. The union of all intervals divided by the length of the original point cloud yields a completeness measure:

$$\mathcal{E}_{Com} = \frac{1}{\text{len}(\Lambda)} \sum_{|\mathbf{U}|} (e_i - s_i), \quad e_i, s_i \in U_i \tag{15}$$
$$\mathbf{U} = \bigcup_{l\in L} [s,e]_l.$$

$\mathcal{E}_{Com}$ approaches 1.0 if the union of all arc-lengths covers the full original geometry; if $\mathcal{E}_{Com} < 1.0$, then the reconstruction is again *incomplete*. Using these additional two metrics, a reconstruction is considered as unique, if $\mathcal{E}_{Len} \simeq \mathcal{E}_{Com}$.

6.4. Initial exemplary reconstructions

In order to illustrate the performance of our complete reconstruction pipeline we first show a smaller set of qualitative results. Examples of the previously described test geometries were reconstructed (see Figs. 11 and 12), both without as well as with varying degrees of noise. If not explicitly indicated for a specific case, then jitter and distribution noise were both set to 1.0. Automatically determined start points are shown as black circles. As can be seen, the combination of start point selection and line-let integration allowed for handling of sharp corners and line crossings in these initial test examples. The adaptive

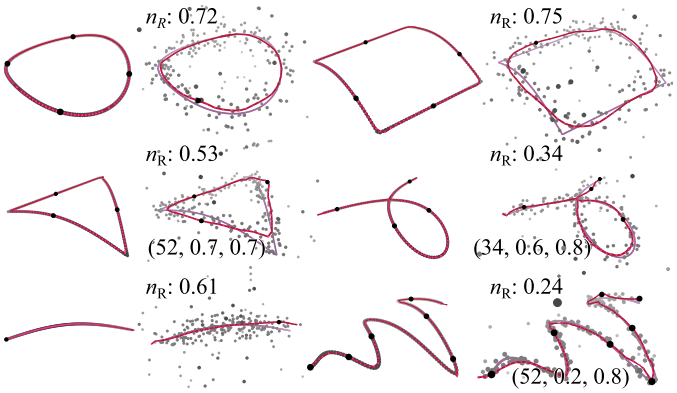


Fig. 11. Automatic reconstructions (red lines) of six "2D" test geometries, without and with noise (with n_R specified). The number of points is $N = 174$. Black dots are automatically set start points. Noise cases are either generated with parameters $M = 87, S_j = 1.0, S_d = 1.0$, or according to values included inline via triplets.

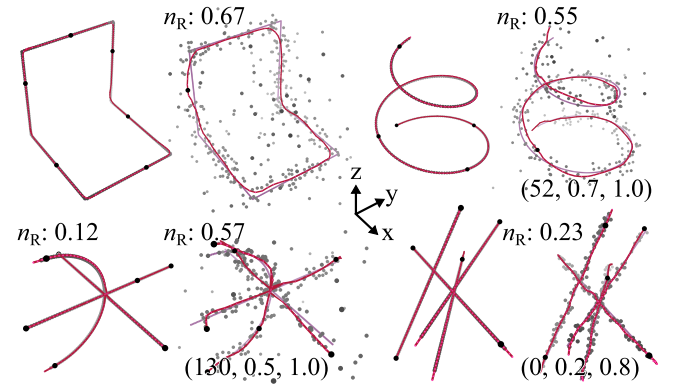


Fig. 12. Automatic reconstructions (red lines) of four 3D test geometries, with and without noise (with n_R specified). Number of points is $N = 260$. Black dots mark detected start points. Noise parameters are shown as triplets; for the elbow: $M = 130$, S_j=S_d=1.0.

Table 1
Average error measures for 3D reconstructions of ten example cases, as in Figs. Fig. 11 and Fig. 12. For each geometry, the numbers for the noise-free (left) as well as the noisy (right) case are indicated (separated by '/'). Only reconstructions deemed as successful were used for the error computation.

Geometry	ε_{Len} [%]	ε_H [10^{-2}]	ε_V [10^{-2}]	Success [%]	Time [ms]
Circle	100/99	4/68	2/27	100/91	8/17
Rectangle	98/96	14/108	2/36	100/82	8/17
Triangle	99/91	17/90	3/35	100/73	20/24
Crossing	103/105	10/95	3/26	100/73	6/14
Line	103/109	7/117	3/32	100/91	6/19
Wave	102/108	33/52	3/16	100/91	8/19
Elbow	99/100	18/109	4/35	100/91	13/27
Helix	101/106	12/110	4/26	100/82	11/26
Crossing3D	103/123	7/88	3/33	100/45	12/26
Mikado	104/108	4/39	2/10	100/45	14/13

selection of radius and integration direction made reconstruction of noise-free as well as noisy point clouds possible.

For these example geometries we also examined the previously mentioned error metrics. To this end, first 11 noise seeds were randomly generated for each of them. The noise parameters were not limited, possibly generating extreme cases, which could not be reconstructed. Thus, the maximum noise amplitudes were then lowered, until at least five visually successful (according to an observer) reconstructions were obtained. Then, the error

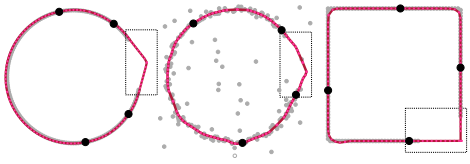


Fig. 13. Reconstructions (red) of incomplete data. The stippled gray rectangles indicate the location of data holes in the original point clouds.

metrics were computed per case, and averaged. The final results are compiled in Table 1; both for noise-free (left number) and noisy (right number) cases. Here, success indicates the percentage of successful reconstructions, per test case, as assessed by an observer. As can be seen, when (stronger) noise is present, the success rate reduces; for these examples on average to about 60%. Geometries with corners exhibited higher errors ε_V. This is also due to smoothing effects, introduced by the weighting functions and smoothing radii. Further, Crossing, Line, and Wave were quite susceptible to outlier noise. The latter represented the most difficult case, especially if jitter noise became too high.

In a further test, we also qualitatively examined the robustness of the reconstruction to data holes. The latter have to be overcome by numerical integration across empty regions. In this case, either another line-let may be encountered or a user-specified length be exceeded. Fig. 13 shows three qualitative examples on the circle and the rectangle geometry, again with and without noise. The size of the holes (indicated by stippled rectangles in the figure) is about 12% of the total line length.

6.5. Comprehensive analysis of reconstruction

6.5.1. Test dataset

In order to perform a more quantitative and broad performance evaluation, we obtained reconstructions with our described approach in an extensive batch processing. Overall, 4.45 million parameter runs were executed, each representing a reconstruction attempt. Runs were grouped by five centroid types (see Section 4), and 6 configuration sets. Each set with one centroid consisted of 127k runs. All ten test geometries were sampled with 128, 256 or 512 points, and reconstructed with varying noise. The distribution noise parameter S_d as well as the jitter parameter S_j were increased from 0.0 to 1.0; both with a step size of 0.1. Similarly M, the number of outliers, was increased from 0 to 0.4N, with a step of $\lfloor 0.1N \rfloor$. Finally, due to the randomness in the noise generation, reconstructions were repeated seven times with different noise seeds.

6.5.2. Computational performance

The experiment was run on an Intel i7-9700 @ 3.60 GHz. Computation times as well as error metrics were recorded for each reconstruction. Over all parameter sets, the mean computation time for obtaining the geometric measures was 32.8 ms, while the integration took on average 7.8 ms. Fig. 14 (left) provides box plots of the overall computation time for the reconstructions, per centroid type. Also, the median times are given as numbers. The point distribution tensor is obviously the fastest, the geometric median the slowest. However, note that the PDT is anyhow used in all methods, as discussed in Section 5.1.

6.5.3. Success rate

Instead of relying on the judgment of an observer, we now chose to automatically determine reconstruction success based on the previously introduced error measures. We consider a reconstruction as *successful*, when $\varepsilon_{Len} \in [0.9, 1.2]$, $\varepsilon_{Com} \in$

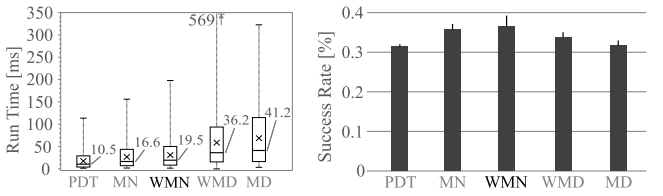


Fig. 14. (Left:) Computation time of the reconstruction (multi-scale analysis and integration). (Right:) Averaged success-rates of 4.45 million parameter runs. Six configuration sets were tested with five different centroid settings.

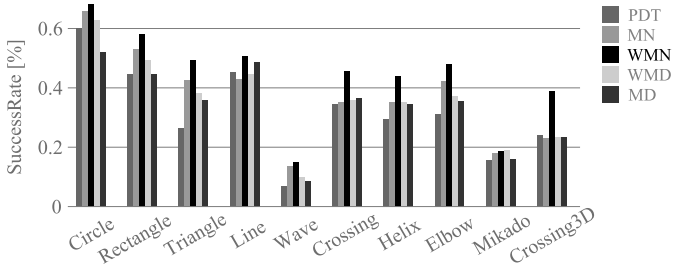


Fig. 15. Success-rates per geometry, for the best performing configuration set of each centroid candidate. Overall, the weighted mean (WMN) performs best, followed by the mean (MN).

[0.95, 1.05], and $\varepsilon_V \leq 0.25$. Fig. 14 (right) illustrates the collected success rates for the different centroids. On average, the weighted mean (WMN) yields a 19% higher rate than the PDT (0.374 vs. 0.315). Note that in this study the success in general is lower, since considerable noise is added to the runs.

Next, Fig. 15 indicates the success rates, per geometry. However, here only the best performing configuration set (of six) per centroid was kept. WMN performs better in almost all cases, with the exception of the Mikado geometry. It especially handles cases well that include corners; the employed quadratic inverse weighting function helps in stabilizing the reconstructions. Due to this, we recommend using WMN for the centroid computation, for generic noisy 3D point clouds. Unless otherwise specified, it has been employed in the majority of our presented cases.

Generally, Wave, Mikado, and Crossing3D are the most challenging cases for our reconstruction approach. The curved nature of the wave makes it very sensitive to jitter noise, usually failing when $S_j \geq 0.3$. At similar jitter amplitudes also the closely passing Mikado lines cannot successfully be reconstructed anymore.

6.5.4. Influence of noise

Next, we examine the effect of noise type and parameter setting on the reconstruction results. In Fig. 16 success rates are plotted, again separated by geometry. This time, rates are shown for different noise values/intervals; from top to bottom: jitter noise amplitude S_j, outlier noise M, and distribution noise S_d. Note that here, WMN was employed for the centroid computation. Further note, that a bar plotted for e.g. $S_j = 0$ includes the full ranges of the other noise parameters, i.e. in this case M and S_d. As can be seen, the success rate mainly depends on jitter noise. In contrast, changing distribution and outlier noise has a smaller effect. Also, as already discussed above, the Wave and Mikado represent the most difficult cases.

6.5.5. Comparisons using new noise rate

Above we had introduced a new noise rate measure in Eq. (11). It can be computed for arbitrary point clouds to characterize overall noise, thus also making comparisons easier. Fig. 17 depicts success rates, per geometry, according to intervals of noise rate n_R. As expected, success rates drop when the noise rate increases.

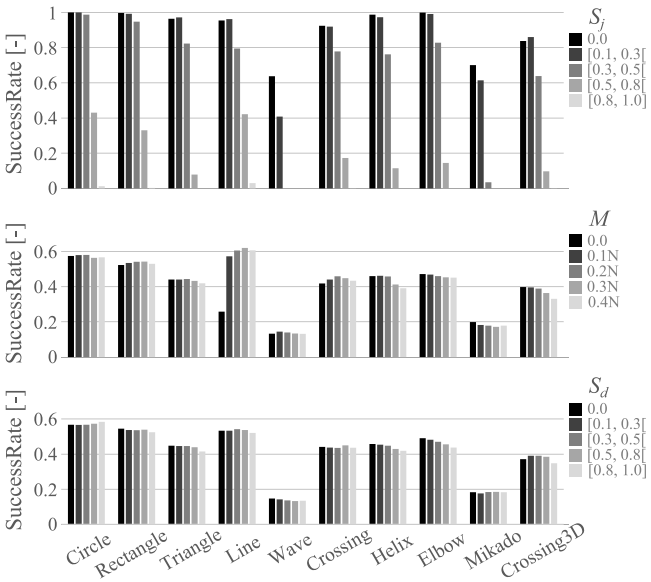


Fig. 16. Effect of increasing noise on success rates, per geometry; using WMN and the best configuration set. Shown are jitter noise amplitude S_j (top), outlier noise M (center), and distribution noise S_d (bottom). Reconstruction performance is mostly independent of outlier and distribution noise. For $S_j \geq 0.8$ most reconstructions fail. Wave and Mikado are highly sensitive to jitter noise.

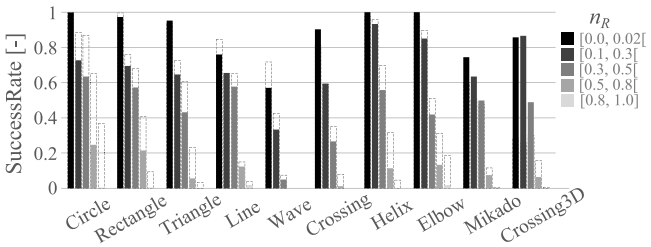


Fig. 17. Success rates, plotted for varying noise rate intervals; again separated per geometry. Using a single noise rate facilitates overall comparisons of reconstruction success vs. noise, even for unknown datasets. Moreover, also effects of algorithm variations can be compactly visualized. Here, for instance, results are also shown for the variation of enforcing a minimum of 6 candidate points in the computation of O_s; here visualized with stippled bars. This yields improvements in the success rate, in presence of high noise.

Again, the poor performance of the Wave example becomes apparent. In addition, also possible variations of our approach for special cases can be studied this way. As an example, for high noise cases using 6 candidate points for computing O_s could be enforced, as discussed above. As can be seen via the stippled bars in the plot, this sometimes would yield improvements.

6.6. Comparison to FitConnect/StretchDenoise

Next, we compared our approach to the recently published reconstruction methods *FitConnect* (Ohrhallinger and Wimmer, 2019) and *StrechDenoise* (Ohrhallinger and Wimmer, 2018). Four challenging geometries introduced in their work were also tested in our framework; point clouds of a "monitor" outline, a spiral, a circle, and a "bunny". The examples include different geometric features, such as corners, curved and straight sections, sparse geometry, as well as variable jitter noise. All geometries were tested using the original geometries specified by the respective authors. In addition, we also tested our geometry of a crossing with *FitConnect*.

For the experiment we used the authors' codes, which are publicly available online: (FitConnect, 2018) and StretchDenoise

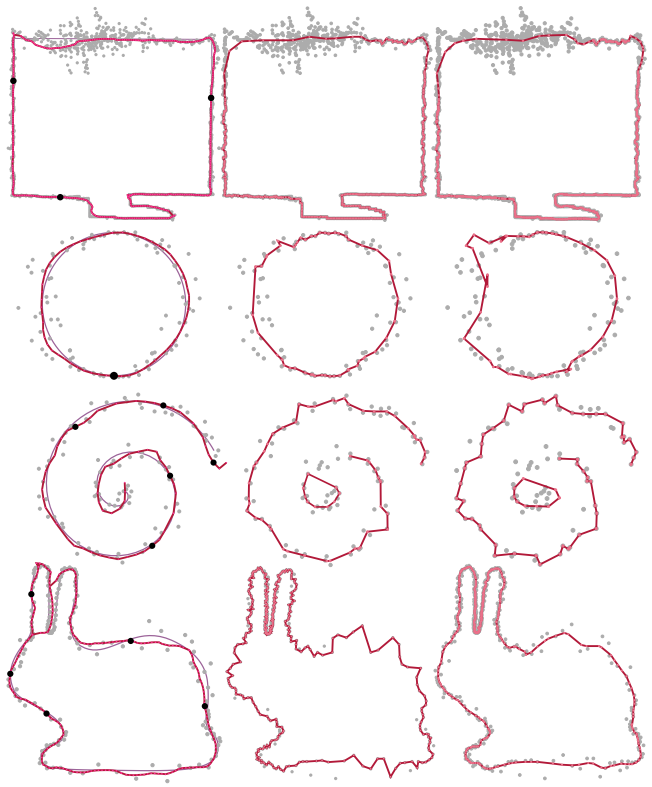


Fig. 18. Comparison of the reconstruction of three geometries of Ohrhallinger and Wimmer (2019) – using our approach (left), *FitConnect* (middle), and *StrechDenoise* (right).

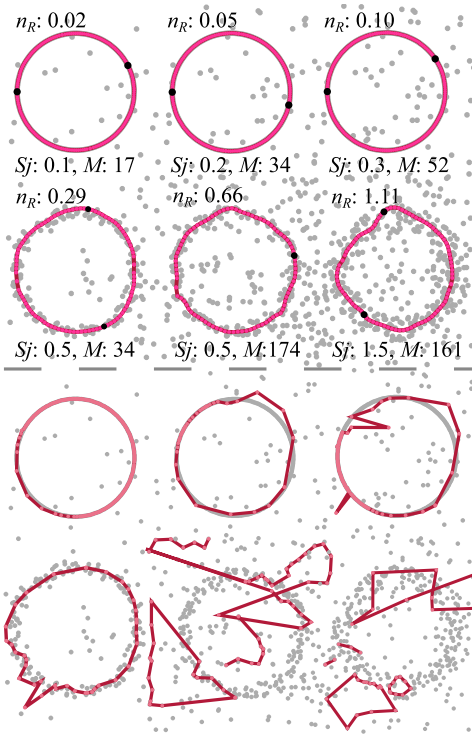


Fig. 19. Comparison of a noisy circle with different strengths of noise: our approach (top half) and *StrechDenoise* (bottom half); the same noise settings are employed for both methods ($N = 174$). *StretchDenoise* exhibited difficulties with outlier noise, deteriorating when e.g. 30% outliers are present (bottom half, row 1, right). In contrast, our approach could handle jitter noise $S_j = 0.5$ and 100% additional outliers (top half, row 2, middle). Further, even strong noise with $S_j = 1.0$ and 150% can be handled, by enforcing 6 candidate points for O_s (top half, row 2, right).

(2018). Only minor modifications were made, to ensure comparable computation time measurements, vector graphics plots of results after computation, and loading of our crossing example. Fig. 18 compares the reconstructions of four test cases, for our approach, *FitConnect*, and *StrechDenoise*. Our method yields smoother reconstructions; but corners may be smoothed out, unless line-lets meet. The reconstructions with *FitConnect* follow occasionally a more zigzaggy path, especially for lower noise rates. However, it can produce sharper corners. Regarding the bunny, our approach exhibited some artifacts: the small gap between the ears was not reconstructed correctly; a merged line resulted. Nevertheless, in contrast, the crossing lines example could not be reconstructed with *FitConnect* or *StrechDenoise*.

We compiled error measures and computation times for these experiments, comparing all three approaches in Table 2. Note that for this, the results from the *FitConnect* and *StrechDenoise* framework and the ground truth data were loaded into our framework to compute the error measures. We found that our method was capable of producing results of similar accuracy, while in most cases being faster.

A further experiment was carried out, comparing *StretchDenoise* and our method, when increasing the noise levels for a 2*D* circle. Fig. 19 depicts six configurations of increasing jitter and outlier noise. The parameters, including the aggregated noise rates n_R, are indicated. *StrechDenoise* (bottom half) could handle jitter noise up to $S_j \approx 0.4$, with up to 30% additional outliers. Our approach (top half) was capable of handling $S_j \approx 0.5$, with up to 100% additional outliers. Moreover, by enforcing a 6-point neighborhood for computing O_s, this can be further improved to $S_j \approx 1.0$, with 150% outliers.

Table 2
Error measures and timings of five reconstructions, comparing with *FitConnect (FitC.)* (Ohrhallinger and Wimmer, 2019) and *StrechDenoise (StrD.)* (Ohrhallinger and Wimmer, 2018). Our method can be several times faster; with similar results. The best performing method is indicated in bold font.

Geometry	Method	ε_H [10^{-2}]	ε_V [10^{-2}]	ε_{Com} [10^{-2}]	ε_{Len} [10^{-2}]	Time [ms]
Monitor	Our	02	0.3	100	**101**	**67**
	FitC.	*02*	***0.2***	*100*	*106*	*2420*
	StrD.	*02*	***0.2***	*100*	*103*	*2470*
Circle	Our	**09**	04	100	103	**4**
	FitC.	*11*	***03***	*100*	***102***	*10*
	StrD.	*36*	*08*	*100*	*125*	*12*
Spiral	Our	36	08	**100**	108	**5**
	FitC.	*31*	***06***	*75*	*106*	*23*
	StrD.	***20***	*07*	*80*	***99***	*34*
Bunny	Our	39	13	90	**97**	7
	FitC.	*71*	*12*	***100***	*124*	***4***
	StrD.	***31***	***6***	***100***	***103***	*6*
Crossing	Our	46	02	100	103	**9**
	FitC./StrD.	–	–	–	–	*>94*

6.7. Application on real-world LiDAR data

Four smaller subsets ranging from 39*k* to 111*k* points of a massive LiDAR dataset were selected to test our method on real-world data; including houses, bushes, trees, and power lines. The data contain jitter, distribution, and outlier noise. Figs. 20 to 23

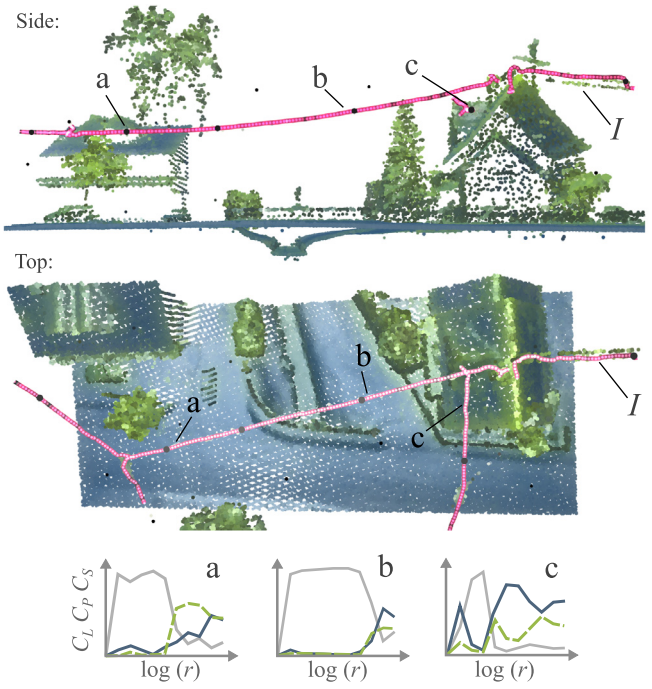


Fig. 20. Power line reconstruction of an urban LiDAR scan. Two different views of the reconstruction; 57*k* points in 2.1 mins. (Bottom): Computed geometric measures at selected locations: (a) close to a tree with sphericity becoming the second dominant shape value; (b) automatic start point, with dominant linearity integral on an isolated linear structure; (c) roof location where planarity becomes more dominant. The line (*I*) first follows a bundle of cables and then continues along a pole onto the roof's gable; i.e. a linearly directed structure of the roof-surface. Here, the WMN was employed.

illustrate the results, using the WMN and exponential radius growth.[2] The algorithm extracts the main cable geometries of the high-resolution scans. Coloring according to the median of the geometric features for each LiDAR point provides a good visual classification. The maximum analysis radius was chosen manually and set to 15 m, to capture all linearity scales. The integration step size was increased to about 0.3 m. The datasets shown in Figs. 22 and 23 also performed well with the default parameters. The main linear structures from the larger, noisy real-world datasets were reconstructed. Computation time ranged from 0.25 to 5.8 mins dependent on data size, point distributions, and maximum radius.

Note that in Fig. 20 an artifact appeared on the roof of the right house. Also, the integration continued onto the roof and followed some of its edges. Stopping criteria were not adjusted to this specific dataset. The WMN reconstructed the cable system successfully. Further, employing the WMD enhanced the reconstruction of the cable bundle as well as it enabled to integrate better along the roof surface edges.

A part of a power line consisting of three parallel cables was tested, see Fig. 21. Here, the start point selection favors the outer cables and ignores the inner one. The upper cable is covered completely and the lower by about 80%. Also one of the poles becomes selected and reconstructed. For the dataset shown in Fig. 22 the single Y-shaped cables were reconstructed as well as parts of a bundle of cables. Due to the sparse sampling of the single cable on the right, starting points are not detected and, thus, it is not reconstructed. Fig. 23 shows another scene where the main cable is successfully reconstructed, whereas again a

[2] Further examples are also shown in the supplementary video.

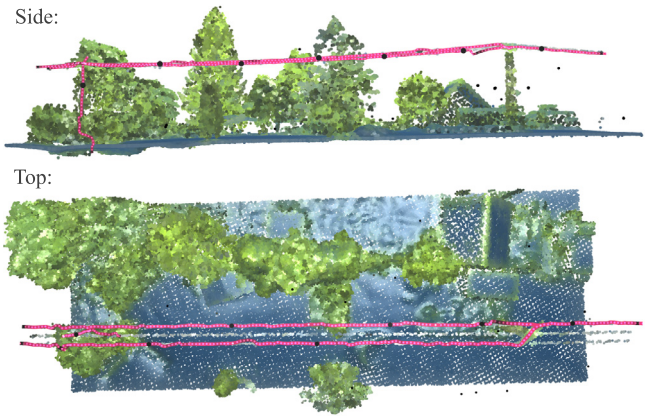


Fig. 21. Automatic reconstruction of a triple power line of about 66 m length sampled with holes and jitter noise; 111*k* points in 5.8 mins. (Top-View:) Start points are selected at the outer two cables. The upper cable is fully reconstructed and the lower to about 80%. At the right pole connection the streamline switches sides. Besides the cables, the left pole gets selected also reconstructed.

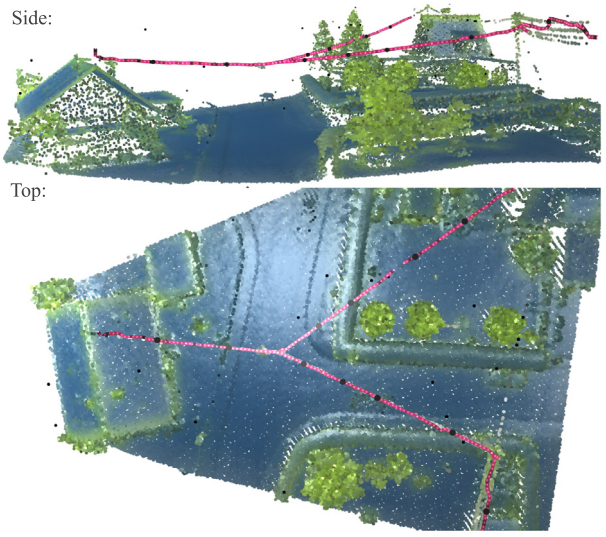


Fig. 22. Three of four single cables are successfully reconstructed of a larger urban scene; 97*k* points in 2.7 mins (0.8 mins with default). (Side-View:) At the right side, due to the very sparse sampling cable was left out in the start point selection and could, thus, not be reconstructed. The cable bundle (right) is covered, but the streamline switches between cables. The cable crossing is well connected.

sparsely sampled cable and a section of a bundle is not covered by the curve reconstruction.

6.8. Discussion of centroid variants

As outlined above, different methods for computing the centroids can be selected. This influences both the tensor computation and the geometric features. The latter will differ in scale and smoothness. Thus, the selection will influence the results of the score function and the vector field. In general, WMN exhibited the best performance. Still, in Table 3 we indicate the advantages and disadvantages of the different options.

PDT clearly outperforms the others concerning computation time. WMN performs best in the overall success rate, and it is beneficial for reconstructing crossings. Both MN and WMN show the most success for the difficult Wave case. Moreover, all approaches could handle the test cases in FitConnect (2018),

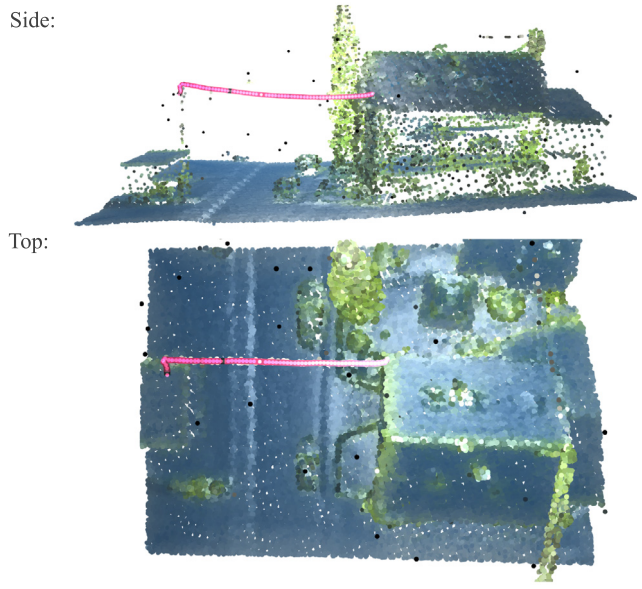


Fig. 23. The main linear feature, a cable on the left, is successfully traced; 39*k* points in 0.9 mins (0.25 mins with default). Whereas, a sparsely sampled cable and a bundle of cables are not covered by the reconstruction on the right.

Table 3
Centroid performance by different aspects.

	PDT	MN	WMN	WMD	MD
Computation time	++	+	+	–	– –
Success rate	–	+	++	+	–
Crossings	–	–	+	–	–
Wave	– –	+	+	–	–
Examples	+	+	+	+	+
Extreme noise	– –	++	+	–	–
LiDAR	–	–	+	+	–

StretchDenoise (2018) (see above). For computing the maximum linearity $C_{L,max}$ in the starting point score O_s, using the PDT has the advantage that it down-votes noisy starting points. In extreme noisy cases, as the circle above, MN and WMN performed best. For our real-world LiDAR datasets, WMN and WMD performed best in reconstructing the cables.

7. Conclusion

We have presented a framework for the reconstruction of curved line structures from noisy 3D point clouds. The method implements a piecewise streamline integration, in a neighborhood tensor field. The approach employs weighing functions at several steps, e.g. tensor computation, point cloud interpolation, and angular weighting. The Fermi–Dirac weighting function gave good results and stabilized against noise. Linearity graphs for different radii were examined to automatically set parameters and steer the algorithm. This includes setting start points, detecting optimal neighborhood radii, and finding directions for integration by using two new scoring functions based on the geometric measures. Further, multiple line-lets are grown forward and backward in parallel; they intersect to form the final global reconstruction. This permits to handle non-manifold lines and sharp corners.

A thorough analysis was carried out on different test geometries, with different types and amounts of noise. This highlighted the performance and the limits of the method for each test geometry. With the automated parameter runs, we explored different centroids for the tensor computation: e.g. mean or geometric median and weighted variants; opting for the weighted mean. Further, we compared our method to the recently introduced *FitConnect* and *StretchDenoise* algorithms, with respect to error metrics and computation time. Our reconstructions were smoother, supported higher noise rates, and could handle crossings, while usually being faster. However, in some test cases the geometry reconstruction was not fully successful. Finally, we tested our method on four noisy real-world LiDAR datasets and were able to reconstruct single cables including holes and crossings. However, some artifacts such as partial coverage were also encountered.

In future work, we will investigate further the application to LiDAR cases; especially, including linearity and sphericity into the stopping criteria and/or the pruning step, increasing robustness against sparse sampling and improve on parallel curve reconstruction.

The executable of the tool and the source code for geometric measure computation and line reconstruction are provided online as well as the data and evaluation tools for the parameter run benchmark (MssfReconstruct, 2019). A video demonstrating the method, selected test cases, and the LiDAR reconstruction can be found here: https://marcel-ritter.com/mssfreconstruct.

CRediT authorship contribution statement

Marcel Ritter: Conceptualization, Methodology, Software, Validation, Formal analysis, Investigation, Visualization, Writing - original draft. **Daniel Schiffner:** Methodology, Software, Writing - review & editing. **Matthias Harders:** Methodology, Supervision, Resources, Writing - review & editing.

Declaration of competing interest

The authors declare that they have no known competing financial interests or personal relationships that could have appeared to influence the work reported in this paper.

Acknowledgments

This research was funded through the Vice Rectorate of Research of the University of Innsbruck within the scope of the doctoral program *Computational Interdisciplinary Modelling (DK CIM)*.

Appendix A. Supplementary data

Supplementary material related to this article can be found online at https://doi.org/10.1016/j.visinf.2021.05.001.

References

Alexa, M., Adamson, A., 2004. On normals and projection operators for surfaces defined by point sets. In: Gross, M., Pfister, H., Alexa, M., Rusinkiewicz, S. (Eds.), SPBG'04 Symposium on Point - Based Graphics 2004. The Eurographics Association.
Basser, P.J., Pajevic, S., Pierpaoli, C., Duda, J., Aldroubi, A., 2000. In vivo fiber tractography using DT-MRI data. Magn. Reson. Med. 44 (4), 625–632.
Berger, M., Levine, J.A., Nonato, L.G., Taubin, G., Silva, C.T., 2013. A benchmark for surface reconstruction. ACM Trans. Graph. 32 (2), 20:1–20:17.
Berger, M., Tagliasacchi, A., Seversky, L.M., Alliez, P., Guennebaud, G., Levine, J.A., Sharf, A., Silva, C.T., 2017. A survey of surface reconstruction from point clouds. Comput. Graph. Forum 36 (1), 301–329.
Berkmann, J., Caelli, T., 1994. Computation of surface geometry and segmentation using covariance techniques. IEEE Trans. Pattern Anal. Mach. Intell. 16 (11), 1114–1116.
Chou, M.-C., Wu, M.-L., Chen, C.-Y., Wang, C.-Y., Huang, T.-Y., Liu, Y.-J., Juan, C.-J., Chung, H.-W., 2006. Tensor deflection (TEND) tractography with adaptive subvoxel stepping. J. Magn. Reson. Imaging 24 (2), 451–458.
Devore, R., Petrova, G., Hielsberg, M., Owens, L., Clack, B., 2013. Processing terrain point cloud data. SIAM J. Imag. Sci. 6, 1–31.
Dey, T.K., Wenger, R., 2001. Reconstructing curves with sharp corners. Comput. Geom. 19 (2), 89–99.

FitConnect, 2018. https://github.com/stefango74/fitconnect.
Flöry, S., 2009. Fitting curves and surfaces to point clouds in the presence of obstacles. Comput. Aided Geom. Design 26 (2).
Foix, S., Alenya, G., Torras, C., 2011. Lock-in time-of-flight cameras: A survey. IEEE Sen. J. 11 (9), 1917ff.
Giraudot, S., Cohen-Steiner, D., Alliez, P., 2013. Noise-adaptive shape reconstruction from raw point sets. In: Proc. of the 11th Eurographics/ACMSIGGRAPH Symposium on Geometry Processing. In: SGP '13, Eurographics Ass., Goslar, DEU, pp. 229–238.
Hasirci, Z., Ozturk, M., 2011. An eigenvalue analysis based bandwidth selection method for curve reconstruction from noisy point clouds. In: 34th International Conference on Telecommunications and Signal Processing (TSP). pp. 478–482.
Hoppe, H., DeRose, T., Duchamp, T., McDonald, J., Stuetzle, W., 1992. Surface reconstruction from unorganized points. SIGGRAPH Comput. Graph. 26 (2), 71–78.
Ihmsen, M., Orthmann, J., Solenthaler, B., Kolb, A., Teschner, M., 2014. SPH fluids in computer graphics. In: Eurographics 2014 - State of the Art Reports. The Eurographics Association.
Jaw, Y., Sohn, G., 2017. Wind adaptive modeling of transmission lines using minimum description length. ISPRS J. Photogr. Remote Sens. 125, 193–206.
Lin, C.-H., Chen, J.-Y., Su, P.-L., Chen, C.-H., 2014. Eigen-feature analysis of weighted covariance matrices for LiDAR point cloud classification. ISPRS J. Photogr. R. Sens. 94.
Lin, H., Chen, W., Wang, G., 2005. Curve reconstruction based on an interval B-spline curve. Vis. Comp. 21, 418ff.
Lipman, Y., Cohen-Or, D., Levin, D., Tal-Ezer, H., 2007. Parameterization-free projection for geometry reconstruction. ACM Trans. Graph. 26 (3).
Liu, M., Pomerleau, F., Colas, F., Siegwart, R., 2012. Normal estimation for pointcloud using GPU based sparse tensor voting. In: IEEE Intern. Conf. on Robotics and Biomimetics. pp. 91–96.
Lu, X., Chen, W., Schaefer, S., 2017. Robust mesh denoising via vertex pre-filtering and L1-median normal filtering. Comput. Aided Geom. Design Preprint.
McDougall, J., Stoner, E.C., Whiddington, R., 1938. The computation of Fermi-Dirac functions. Philos. Trans. R. Soc. Lond. Ser. A.
McIvor, A.M., Valkenburg, R.J., 1997. A comparison of local surface geometry estimation methods. Mach. Vis. Appl. 10 (1), 17–26.
MssfReconstruct, 2019. https://github.com/gileoo/MssfReconstruct.
Natale, D., Baran, M.S., Tutwiler, R.L., 2010. Point cloud processing strategies for noise filtering, structural segmentation, and meshing of ground-based 3D Flash LIDAR images. In: 2010 IEEE 39th Applied Imagery Pattern Recogn. Worksh. (AIPR). pp. 1–8.
Ohrhallinger, S., Mitchell, S.A., Wimmer, M., 2016. Curve reconstruction with many fewer samples. In: Proceedings of the Symposium on Geometry Processing. In: SGP '16, Eurographics Association, Goslar Germany, Germany, pp. 167–176.
Ohrhallinger, S., Wimmer, M., 2018. Stretchdenoise: Parametric curve reconstruction with guarantees by separating connectivity from residual uncertainty of samples. In: Proc of the 26th Pacific Conf on Comp Graph and App: Short Papers. In: PG '18, Eurographics Association, Goslar, DEU, pp. 1–4.
Ohrhallinger, S., Wimmer, M., 2019. Fitconnect: Connecting noisy 2D samples by fitted neighborhoods. Comput. Graph. Forum 38 (1), 126ff.
Ohtake, Y., Belyaev, A., Seidel, H.-P., 2005. An integrating approach to meshing scattered point data. In: Proceedings of the 2005 ACM Symposium on Solid and Physical Modeling. In: SPM '05, ACM, New York, NY, USA, pp. 61–69.
Ozturk, M., Hasirci, Z., 2013. A novel method for thinning branching noisy point clouds. Turk. J. Elec. Eng. Comp. Sci. 21, 2239–2258.
Philsu, K., Hyoungseok, K., 2010. Point ordering with natural distance based on brownian motion. Math. Probl. Eng. 450460, 17.
Plastria, F., 2011. The weiszfeld algorithm: Proof, amendments, and extensions. In: Eiselt, H., Marianov, V. (Eds.), Foundations of Location Analysis. Vol. 155, Springer, Boston, MA.
Ritter, M., Benger, W., 2012. Reconstructing power cables from LIDAR data using eigenvector streamlines of the point distribution tensor field. In: Journal of WSCG Vol20, 20-th Intern. Conf. in Central Europe on Computer Graphics, Visualization and Computer Vision.
Ritter, M., Schiffner, D., Harders, M., 2021. Visual analysis of point cloud neighborhoods via multi-scale geometric measures. In: Eurographics 2021 - Short Papers. The Eurographics Association.
Ruiz., O.E., Cortés., C., Aristizábal., M., Acosta., D.A., Vanegas., C.A., 2013. Parametric curve reconstruction from point clouds using minimization techniques. In: Proc of the Intern Conf on Comp Graph Theory and App and Intern Conf on Inf Vis Theory and App - Vol. 1: GRAPP, (VISIGRAPP 2013). SciTePress, pp. 35–48.
StretchDenoise, 2018. https://github.com/stefango74/stretchdenoise.
Tao, J., Ma, J., Wang, C., Shene, C., 2013. A unified approach to streamline selection and viewpoint selection for 3D flow visualization. IEEE TVCG 19 (3), 393–406.
Taubin, G., 1995. Estimating the tensor of curvature of a surface from a polyhedral approximation. In: Proceedings of the Fifth International Conference on Computer Vision. In: ICCV '95, IEEE Computer Society, Washington, DC, USA, p. 902.
Toth, C., Jóźków, G., 2016. Remote sensing platforms and sensors: A survey. ISPRS J. Photogr. R. Sens. 115, 22ff.
Weinmann, M., Jutzi, B., Hinz, S., Mallet, C., 2015. Semantic point cloud interpretation based on optimal neighborhoods, relevant features and efficient classifiers. ISPRS J. Photogramm. Remote Sens. 105, 286–304.
Weinstein, D., Kindlmann, G., Lundberg, E., 1999. Tensorlines: Advection-diffusion based propagation through diffusion tensor fields. In: Proceedings of the Conference on Visualization '99: Celebrating Ten Years. In: VIS '99, IEEE Computer Society Press, Los Alamitos, CA, USA, pp. 249–253.
Wendland, H., 2005. Scattered Data Approximation. Cambridge University Press.
Westin, C., Peled, S., Gudbjartsson, H., Kikinis, R., Jolesz, F., 1997. Geometrical diffusion measures for MRI from tensor basis analysis. In: Proceedings of ISMRM, Canada. p. 1742.
Zeng, Y., Nguyen, T.A., Yan, B., Li, S., 2008. A distance-based parameter free algorithm for curve reconstruction. Comput. Aided Des. 40 (2), 210–222.