Grundsätzlich kann die Interpretation von Punktwolken anhand der Zielstellung der Modellierung (Zielsystem) sowie zunehmender Generalisierung (von der elementbasierten Betrachtung bis hin zur funktionellen Semantik) eingeteilt werden.

(Quelle: Geist et al., 2018)

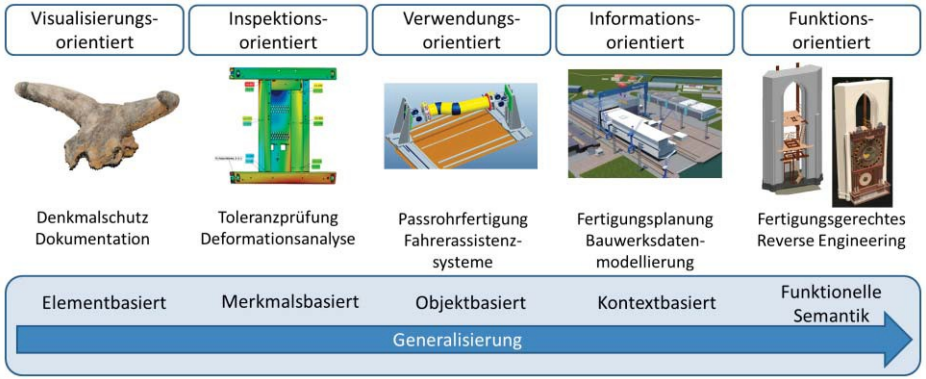


Abb. 1: Interpretation von 3D-Punktwolken.

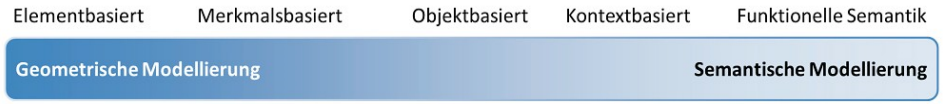


Abb. 2: Zusammenhang geometrischer und semantischer Modellierung.

Abb. 3: Generalisierungsgrad.

Eine Modellierung ist somit immer ein Kompromiss zwischen Datentreue und der Abbildung komplexer Zusammenhänge.

Anforderungen an die Datenbasis (Quelle: Geist et al., 2018)

- Oberflächengüte einer einzelnen Aufnahme (Messrauschen und oberflächenbedingte Unsicherheiten)
- Anzahl der Punktwolken
- Qualität der Registrierung der Punktwolken (wenn mehr als eine)
- Segmentierungslevel

Kriterien für die Wahl der Modellierungsmethode

- Ausdrucksfähigkeit: Können alle geometrischen Objekte mit der Methode beschrieben werden?
- Korrektheit: Sind die dargestellten Geometrien eindeutig interpretierbar bzw. kann überprüft werden, ob sie überhaupt realisierbar sind?
- Effizienz: Können Operationen auf dem Modell eine Transformation in ein anderes Modell so erfolgen, dass sie Nutzanforderungen gerecht wird?

Die wichtigsten Parameter zur Wahl einer Modellierungsstrategie

- Für welche Anwendung ist das Modell bestimmt?
- Welche Ressourcen stehen zur Verfügung (Zeit, Personal, Software)?
- Grad der gewünschten Komplexität und Generalisierung.
- A-priori geschätzte Genauigkeit der Repräsentation.

Die Art der Modellverwendung im Zielsystem definiert die Datenstrukturen und Operationen, die für die Modellierung zur Verfügung stehen. Mit steigender Semantik bzw. dem Grad des formulierten Vorwissens kann die Modellierung geeignet unterstützt und beschleunigt werden.

Prozessablauf von den Daten zum Modell

(Quelle: Geist et al., 2018)

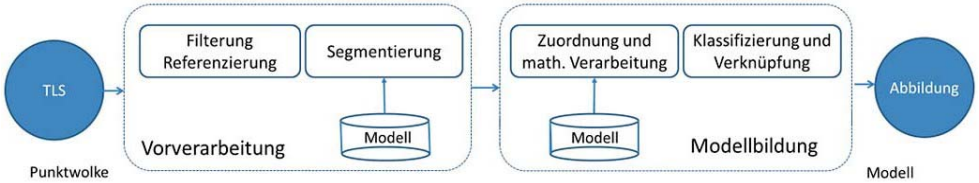


Abb. 4: Prozessablauf von der Punktwolke zum Modell.

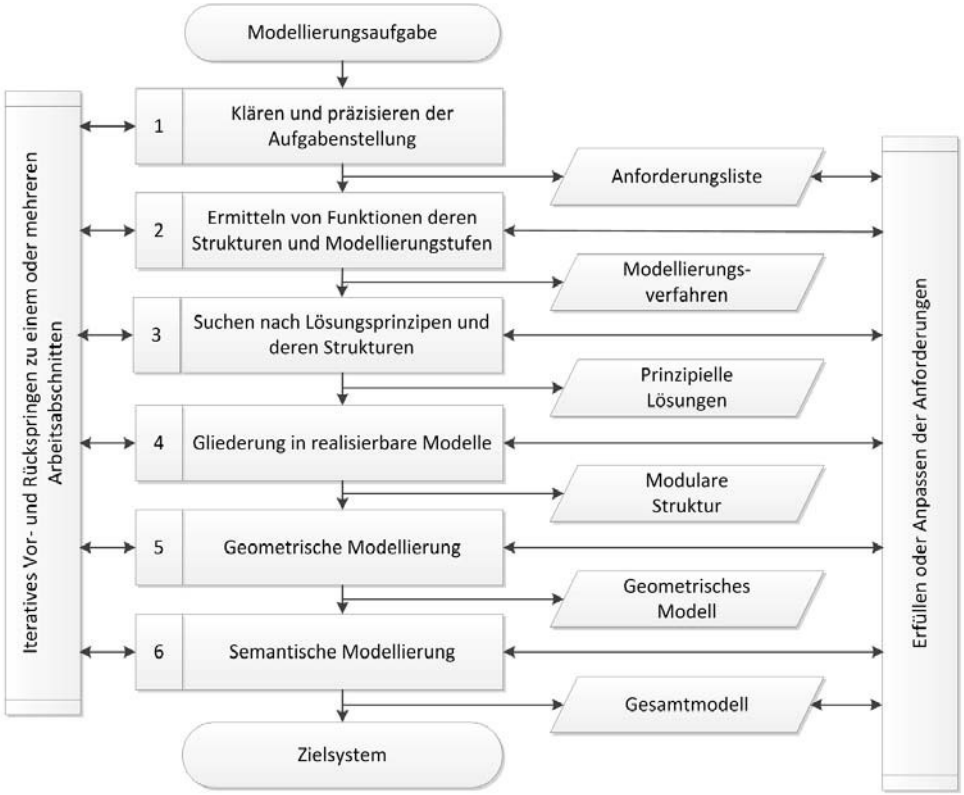


Abb. 5: Methodik bei der Modellierung.

Data structures
Octrees (1/2)

Octrees are efficient tree data structures for handling large point sets. In addition to spatial indexing, common uses of octrees in the context of point cloud processing are compression, streaming of levels of detail and viewing frustrum culling. Octrees are the 3D analogue of quadtrees. Each node of the tree represents a cuboidal volume, also called a cell. Starting with a 3D bounding box cell enclosing the entire data space as the root, each internal cell containing data is recursively subdivided into up to eight non-empty octants. The region octree uses a regular binary subdivision of all dimensions. Note that in this way the location of the internal cells is implicitly defined by their level and position in the octree (see Fig. 1). A point octree shares the same features of all octrees but the centre of subdivision is always at an arbitrary point inside a cell. While the point octree adapts better to the data than the region octree, it needs to store additional data for split positions.
A node of an octree is similar to a node of a binary tree, with the major difference being that it has eight pointers, one for each octant, instead of two for “left” and “right” as in the ordinary binary tree. A point tree additionally contains a key which is usually decomposed into three parts, referring to *x, y* and *z* coordinates. Therefore a node contains the following information:

- eight pointers: oct[1],…, oct[8];
- (optionally) a pointer to parent: oct;
- pointer to list of data points (commonly stored in leaves only, oct[i] can be used); and
- split coordinates x, y, z (optionally for the region octree).

(Source: Vosselman & Maas, 2010)

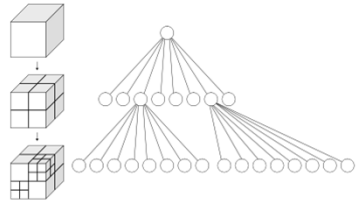


Fig. 2: Octree.
(Source: Wikipedia, 2019)

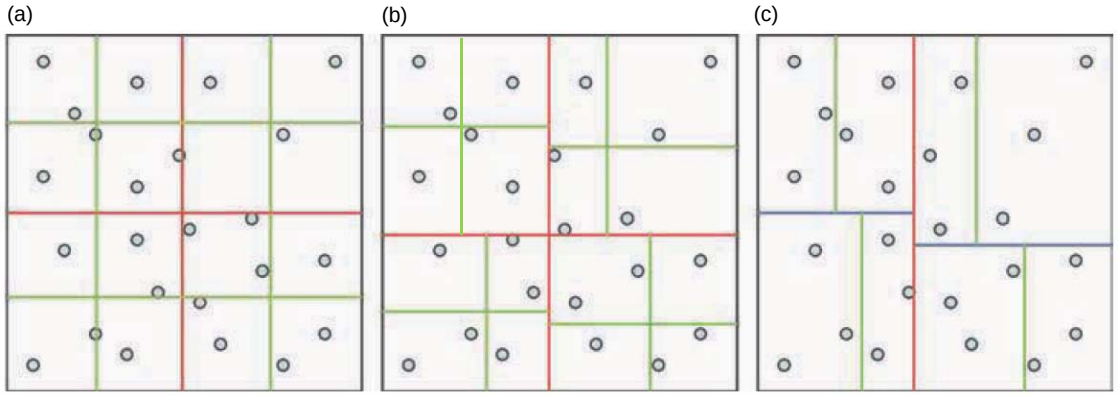


Fig. 1: Comparison of (a) a regular region quadtree, (b) an adaptive point quadtree, and © a 2-D tree.
(Source: Vosselmann & Maas, 2010)

An octree is said to be complete if every internal node has exactly eight child nodes. The number of internal nodes (n_{nodes}) in an complete octree with resolution s is given by

storage overhead

$$n_{nodes} = \sum_{i=0}^{(\log_2 s)-1} 8^i = \frac{s^3 - 1}{7} \approx 0.14 n_{\text{data points}}$$

$$s = 2^x$$

A complete octree has the advantage that it allows for a simple array-like data structure and organisation. No pointers are needed. Such a representation is referred to as a linear octree.

Adaptive Visualisierung mit Frustrum Culling (Kegelstumpf-Auslese)

Frustrum Culling ist eine Optimierungsmethode, bei der alle 3D-Punkte vom Zeichen ausgeschlossen werden, die außerhalb des Sichtbereichs liegen. Es sorgt dafür, dass akzeptable Bildwiederholraten auch bei großen Szenen möglich sind, denn durch Frustrum Culling werden viele der Punkte der Szene nicht gezeichnet, die man ohnehin nicht sehen kann. Ein Octree als Hierarchie von Geometriebegrenzungen von Volumina kann dazu verwendet werden, effizient 3D Punkte anzuzeigen. (Source: Nüchter, 2016)

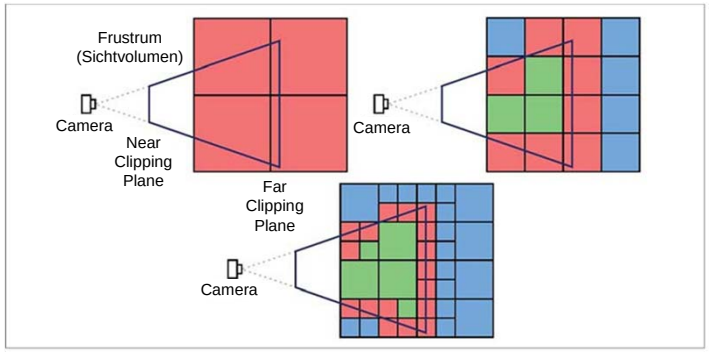


Fig. 3: Das Prinzip des Frustrum Cullings mit einem Octree.
(Source: Nüchter, 2016)

Data structures
k-D tree (1/2)

In contrast to octrees, k-D trees can guarantee a fully balanced hierarchical data structure for point sets sampled from a k-dimensional manifold (Bentley, 1975). K-D trees are a compact spatial data structure well suited for important tasks like the location of nearest neighbours. The k-D tree is a k-dimensional binary search tree in which each node is used to partition one of the dimensions. Therefore, to organize a 3D point set a 3D tree is used. Each node in the tree contains pointers to the left and right subtrees. All nodes except for the leaf nodes have one axis-orthogonal plane that cuts one of the dimensions (*x, y* or *z*) into two pieces. All points in the left subtree are below this plane and all points in the right subtree are above the plane. Each leaf node contains a pointer to the list of points located in the corresponding cell defined by the intersection of all half-spaces given by nodes in the path of the root node to the leaf node itself. This data structure makes it possible to locate a point in the k-D tree with N points in time O(log$N$) on average but at worst in time O(N) if the tree is very skewed. If the tree is balanced, the worst case time becomes O(log$N$). The average time to locate the k nearest neighbours is in the order of O(k + log$N$).

The construction of a balanced k-D tree starts with the bounding box enclosing all points and recursively splits the current cell along one dimension into two subregions enclosing an equal number of elements by calculating the median of the corresponding coordinate of the point set. A comparison of the 2D analogue of an octree with a k-D tree is shown in Figure 1.

(Source: Vosselman & Maas, 2010)

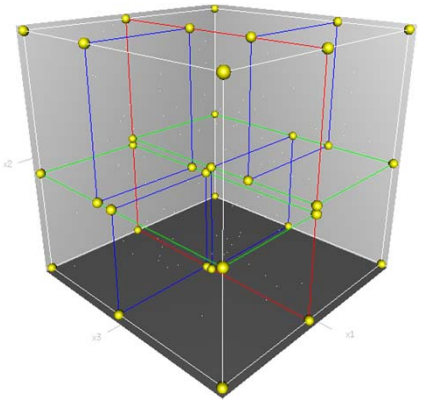


Fig. 4: A 3-dimensional k-D tree.
(Source: Wikipedia, 2019)

A 3-dimensional *k*-d tree. The first split (the red vertical plane) cuts the root cell (white) into two subcells, each of which is then split (by the green horizontal planes) into two subcells. Finally, four cells are split (by the four blue vertical planes) into two subcells. Since there is no more splitting, the final eight are called leaf cells.

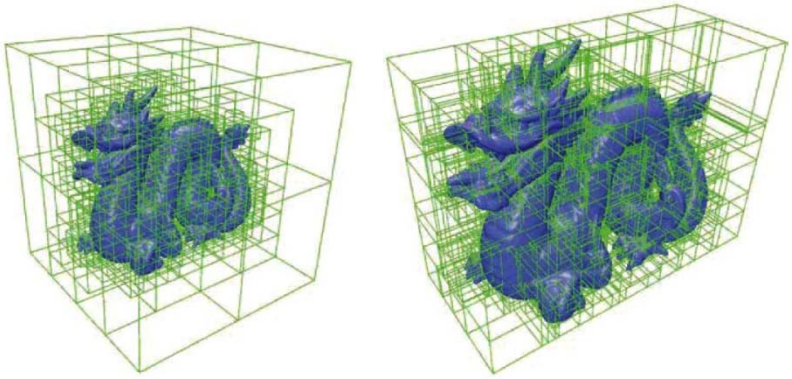


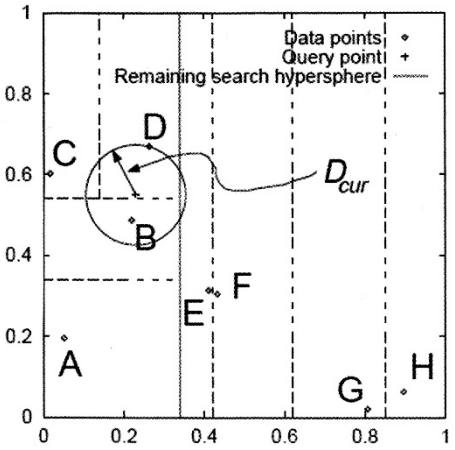
Fig. 5: (a) An octree, and (b) a 3D tree.
The 3D adapts better to the given data than the octree.
(Source: Vosselmann & Maas, 2010)

Nearest neighbour search (NN) algorithm

Der Suchalgorithmus für k-D-Bäume ist eine rekursive Prozedur. Das Argument der Prozedur ist der aktuelle zu analysierende Knoten, also wird die Wurzel als Startwert übergeben. Die Suche nach dem nächsten Punkt für einen gegebenen Punkt p_q besteht aus dem Vergleich von p_q mit der Teilerdimension und dem Teiler, d.h. im Fall $k = 3$ mit der entsprechenden Separierungsebene. Dadurch bestimmt der Suchalgorithmus, in welchem Nachfolgerknoten weitergesucht werden muss und setzt die Suche dort fort. Dies wird solange wiederholt, bis ein Blatt, das den nächsten Datenpunkt p_q enthält, erreicht ist. In den Blättern muss der Algorithmus alle Punkte betrachten. Es ist jedoch möglich, dass der nächste Punkt in einem anderen Blatt liegt. Dieser Fall ist zu untersuchen, falls der Abstand zwischen p_q und der Grenze der Region des Blattes kleiner als der Abstand $\|p_q - p_b\|$. Es tritt Backtracking auf, bis alle Blätter analysiert wurden, die im Radius $\|p_q - p_b\|$ liegen. Diese Bedingung ist als Ball-WithinBounds bekannt (Bentley, 1975).

(Source: Nüchter, 2016)

(a)



(b)

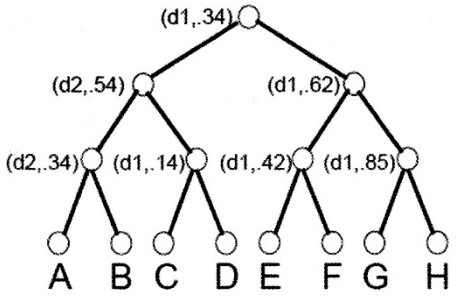


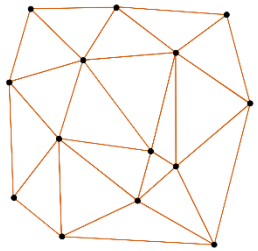
Fig. 6: K-D tree and best bin first (BBF) search. (Source: Szeliski, 2011)

Figure 6 shows an example of a two-dimensional k-D tree. Here, eight different data points A-H are shown as small diamonds arranged on a two-dimensional plane. The k-D tree recursively splits this plane along axis-aligned (horizontal or vertical) cutting planes. Each split can be denoted using the dimension number and split value (Fig. 6b). The splits are arranged so as to try to balance the tree, i.e., to keep its maximum depth as small as possible. At query time, a classic k-D tree search first locates the query point (+) in its approximate bin (D), and then searches nearby leaves in the tree (C, B, …) until it can guarantee that the nearest neighbour has been found. The best bin first (BBF) search (Beis & Lowe, 1997) searches bins in order of their spatial proximity to the query point and is therefore usually more efficient. (Source: Szeliski, 2011)

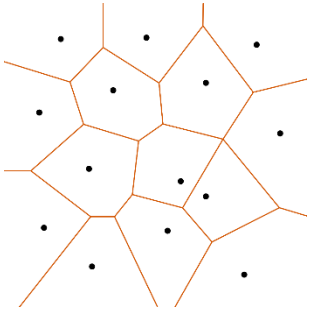
Meshing
Delaunay triangulation (2.5D)

A Delaunay triangulation for a given set **P** of discrete points in a plane is a triangulation DT (**P**) such that no point in **P** is inside the circumcircle of any triangle in DT(**P**). Delaunay triangulations maximize the minimum angle of all the angles of the triangles in the triangulation; they tend to avoid sliver triangles.

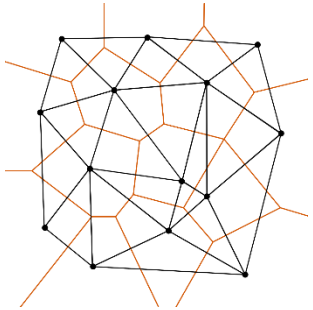
The Delaunay triangulation of a discrete point set **P** in general position corresponds to the dual graph of the Voronoi diagram for **P**. The circumcenters of Delaunay triangles are the vertices of the Voronoi diagram. In the 2D case, the Voronoi vertices are connected via edges, that can be derived from adjacency-relationships of the Delaunay triangles: If two triangles share an edge in the Delaunay triangulation, their circumcentres are to be connected with an edge in the Voronoi tessellation. (Source: Wikipedia, 2019)



Delaunay triangulation of a given set **P** of discrete points.

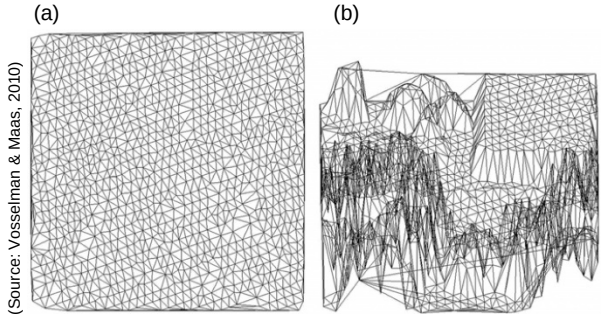


Voronoi diagram for **P**.



Voronoi diagram (orange) and Delaunay triangulation (black).

(Source: Wikipedia, 2019)



(a) Top view; and (b) slanted view of a triangulated point cloud with a building, terrain and trees.

It should, however, be noted that a triangulation is defined in a plane, typically the *XY*-plane. The height distribution of points has no influence on a triangulation in the *XY*-plane. As a consequence, points that are very close in the XY-plane, and therefore share a triangle edge, may not be close in 3D. Because the Delaunay triangulation is still a 2D data structure, it cannot solve all raster data structure problems. In particular, in the presence of multiple surfaces above each other, the triangulation will generate many edges between points of these surfaces that may be far apart. (Source: Vosselman & Maas, 2010)

Projection of 3D points:

- XY-plane
- best fit plane(s)
  → patch-wise 2.5D triangulation

➔ **3-D Delaunay triangulation using tetrahedral meshes** (Maur, 2002)
(2D: 3 vertices → triangle, circle; 3D: 4 vertices → tetrahedron, sphere)

Quelle: https://de.wikipedia.org/wiki/Delaunay-Triangulierung
Source: https://en.wikipedia.org/wiki/Delaunay_triangulation

Meshing
Poisson triangulation (3D)

Poisson surface reconstruction (Kazhdan et al., 2006, 2007; Kazhdan & Hoppe, 2013) uses a smoothed 0/1 inside-outside (characteristic) function, which can be thought of a clipped signed distance function. The gradients for this function are set to lie along oriented surface normals near known surface points and 0 elsewhere. The function itself is represented using a quadratic tensor-product B-spline over an octree, which provides a compact representation with larger cells away from the surface or in regions of lower point density, and also admits the efficient solution of the related Poisson equations.
(Source: Szeliski, 2011)

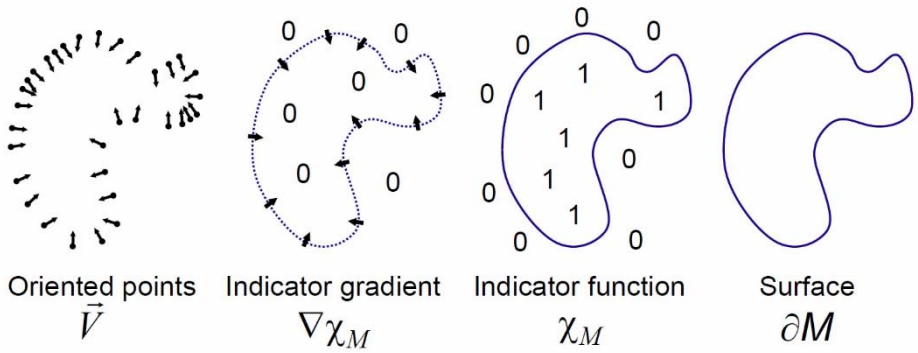


Fig. 1: Intuitive illustration of Poisson reconstruction in 2D. (Source: Kazhdan et al., 2006)

Many implicit surface fitting methods segment the data into regions for local fitting, and further combine these local approximations using blending functions. In contrast, Poisson reconstruction is a global solution that considers all the data at once, without resorting to heuristic portioning or blending. Poisson reconstruction creates very smooth surfaces that robustly approximate noisy data (Kazhdan et al., 2006).

Poisson‘s equation: https://en.wikipedia.org/wiki/Poisson's_equation
GitHub: https://github.com/mkazhdan/PoissonRecon
http://www.cs.jhu.edu/~misha/Code/PoissonRecon/Version8.0/
MeshLab: http://www.meshlab.net/
CloudCompare: http://www.danielgm.net/cc/release/

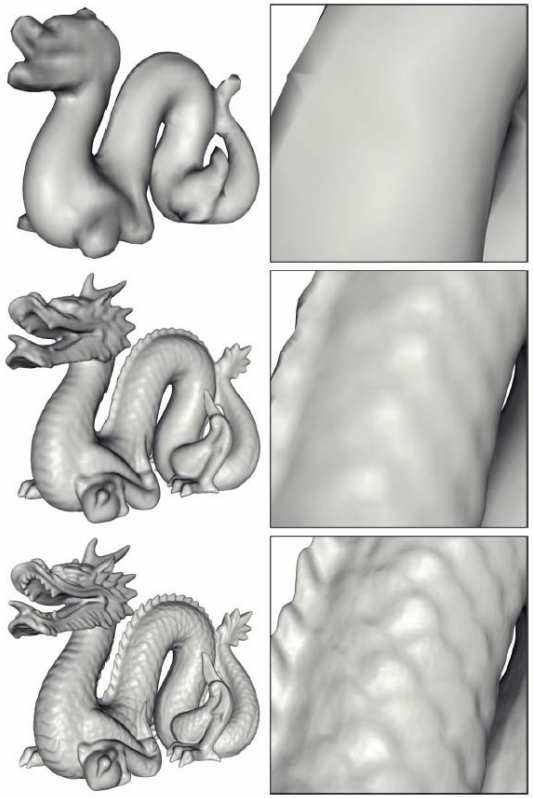


Fig. 2: Reconstruction of the dragon model at octree depths 6 (top), 8 (middle), and 10 (bottom). (Source: Kazhdan et al., 2006)

Modellierung
Ausdünnen

Mögliche Verfahren:
(Quelle: Wieser & Wunderlich, 2017)

- zufällige Auswahl, jeder n-te Punkt
- Abstandsfilterung
- Octree basiert
  - nächster Punkt zum Zentrum (Originalpunkt bleibt erhalten.)
  - Zentrum
  - Schwerpunkt

  → Neupunkt! (resampling process)

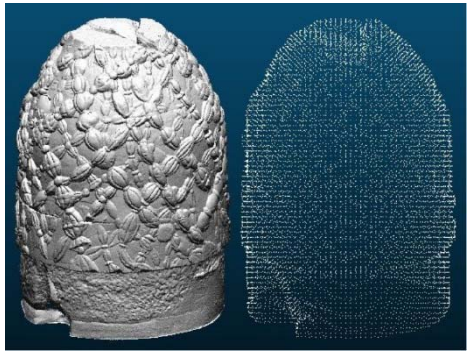


Fig. 1: Octree-based resampling process on a point cloud.
(Source: CloudCompare, V2)

- Krümmungsbasierte Filterung

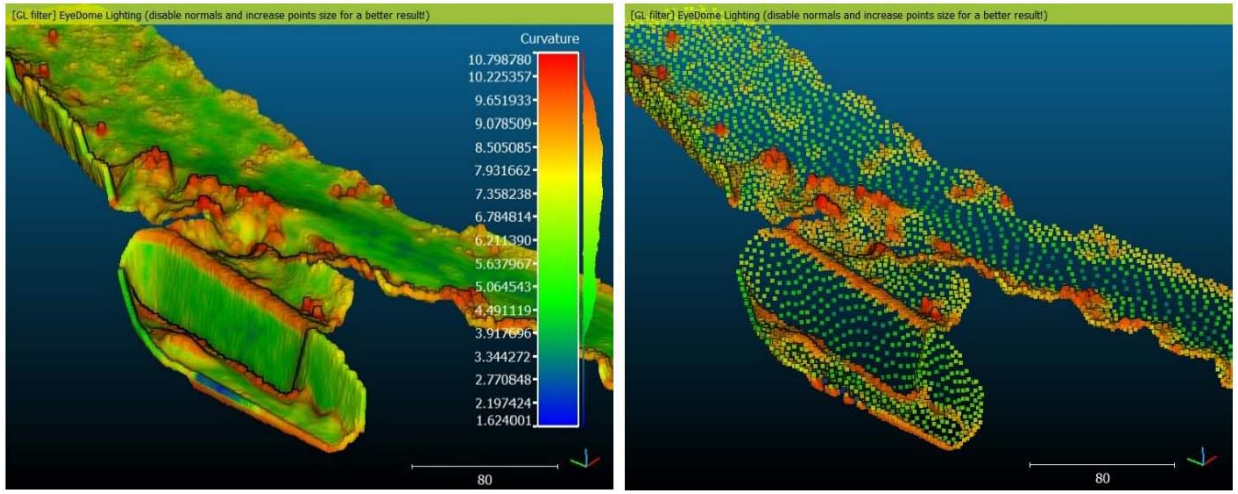


Fig. 2: (a) original cloud with a scalar field proportional to the local curvature, (b) subsampling output.
(Source: CloudCompare, V2)

User Manual: https://www.cloudcompare.org/doc/qCC/CloudCompare%20v2.6.1%20-%20User%20manual.pdf

Modellierung
Filterung (1/5)

Daten auf objektrelevante Information reduzieren (Quelle: Wieser & Wunderlich, 2017)

Mögliche Verfahren:

- Ausreißerfilter
- Glättungsfilter
- Distanz/Winkel-Filter
- Intensität
- Hidden Points (z-Buffer)

(a) Depth coded point clouds (b) Corresponding image

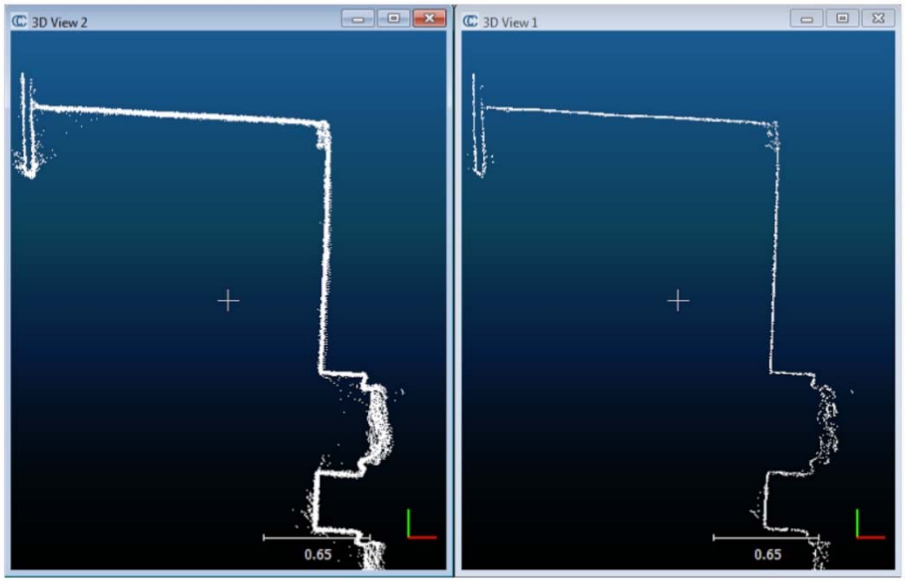
Fig. 1: Z-buffer problem to project point clouds onto images: points in the back face are projected onto the front through interspaces of the front faces. (Source: Qin & Grün, 2014)

- Klassifizierung
- Hough-Transformation

Ausreißerfilter (cp. CloudCompare, V2)

SOR (Statistical Outlier Removal) filter: It computes first the average distance of each point to its neighbours (considering k nearest neighbours for each). Then it rejects the points that are farther than the average distance plus a number of times the standard deviation.

The 'Noise filter' tool resamples a bit the SOR filter but it considers the distance to the underlying surface instead of the distance to the neighbours. This algorithm locally fits a plane (around each point of the cloud) then removes the point if it's too far away from the fitted plane. To estimate the underlying (planar) surface, the user can define a radius or a (constant) number of neighbours. The user has the choice between a relative error (as SOR) and an absolute error. Eventually isolated points can be removed in the same run.



(Source: https://www.cloudcompare.org/doc/)

Fig. 2: Typical result of the 'Noise filter'. (Source: CloudCompare, 2019)

☺ It is very powerful on flat surfaces (walls, etc.).
☹ If a too high kernel radius (or two low error threshold) is used it will 'eat' the corners. To save the corners or sharp edges one can try to run the algorithm repeatedly with a small radius and relatively high error threshold.

Interactive deletion of erroneous LiDAR data, i.e., points of mixed distances, ghost points, rain drops, dust etc.

Glättungsfilter

Octree-based surface smoothing/visualization

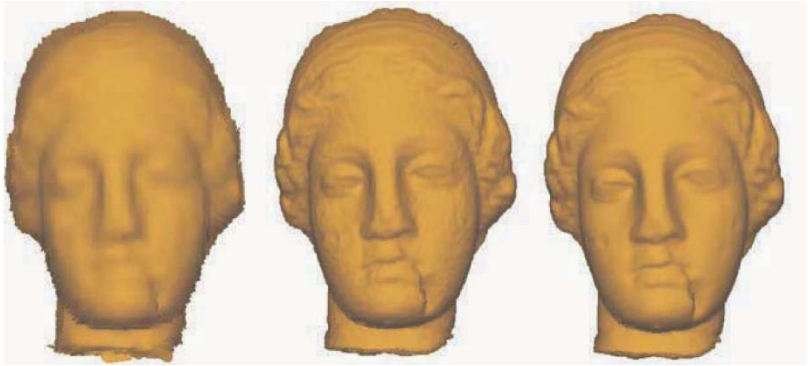
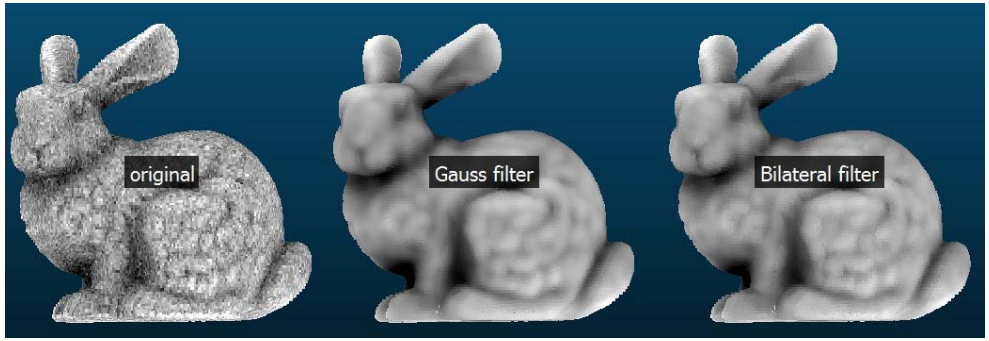


Fig. 3: Progressive evolution of the point cloud during decoding of the octree. The decoder can be stopped at any time and the centres of the leaves of the tree that has been decoded so far give a coarse approximation of the original point cloud. The images were rendered using splatting. (Source: Vosselman & Maas, 2010)



(Source: https://www.cloudcompare.org/doc/)

Fig. 4: Gaussian and Bilateral filters.
The two parameters of the filter are: spatial sigma and scalar sigma. A bilateral filter is a non-linear, edge-preserving, and noise-reducing smoothing filter for images.
.(Source: CloudCompare, V2)

Klassifizierung

- Merkmalsbasiert (feature-based)
  Mayr et al., 2017: The work presents an approach based on point clouds for automatically classifying multitemporal scenes of a hillslope affected by shallow landslides. The first objective is to extract discrete and geomorphologically meaningful objects from TLS point clouds and maximise the objects' consistency throughout the time series. The objects are associated into one to seven target classes (scarp, eroded area, deposit, rock outcrop and different classes of vegetation). The second objective is to maintain the spatially accurate, detailed and 3D representation of the scene as a point cloud throughout the analysis. These objectives are persued by integrating machine-learning methods for supervised classification and a topological rule-set in an object-based analysis workflow.

Fig. 5:
Test area displaying two landslide scars.

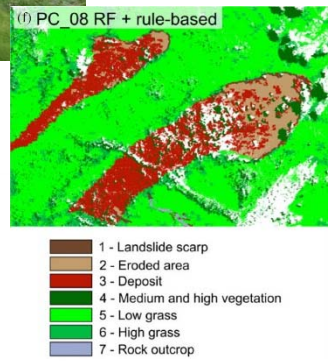


Fig. 8: Rule-based reclassification.

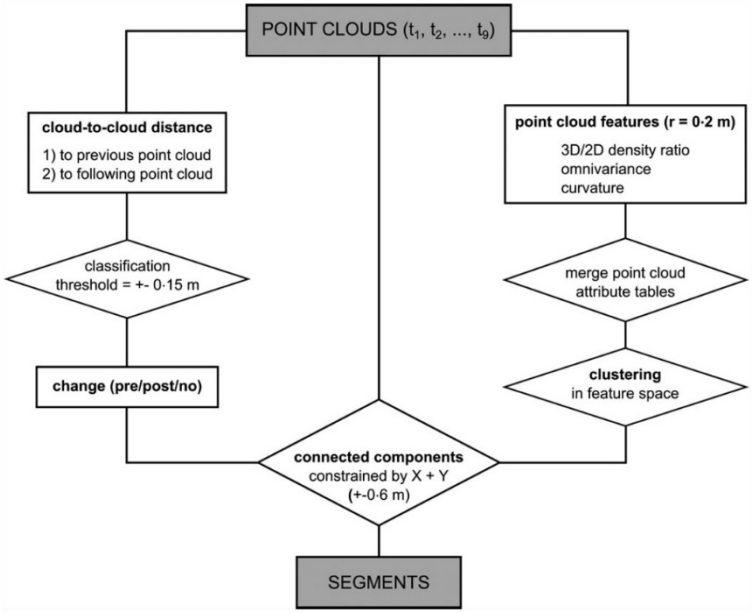


Fig. 6: Point cloud segmentation method. (Source: Mayr et al. 2017)

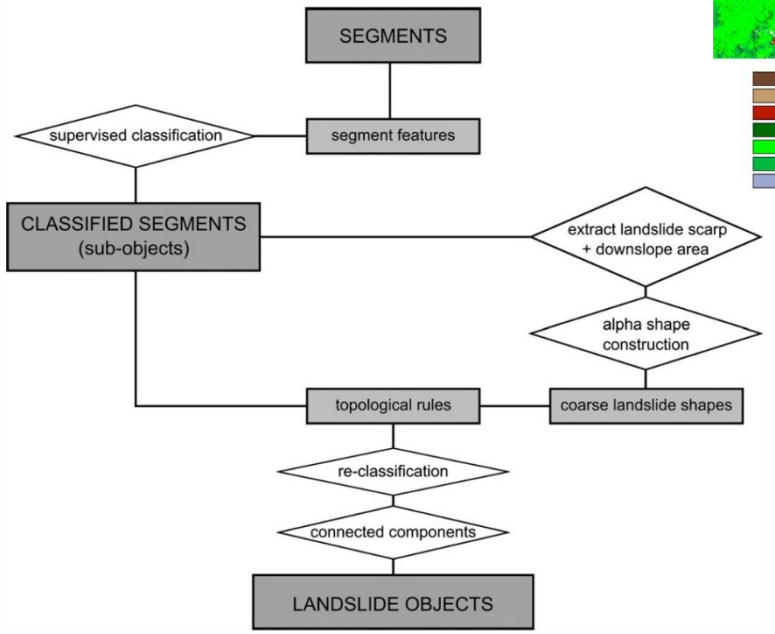


Fig. 7: Two-step classification strategy with supervised classification and rule-based reclassification. (Source: Mayr et al. 2017)

Klassifizierung

- First pulse, intermediate pulses, last pulse – Full-waveform analysis (→ waveform features) (Doneus et al., 2009)

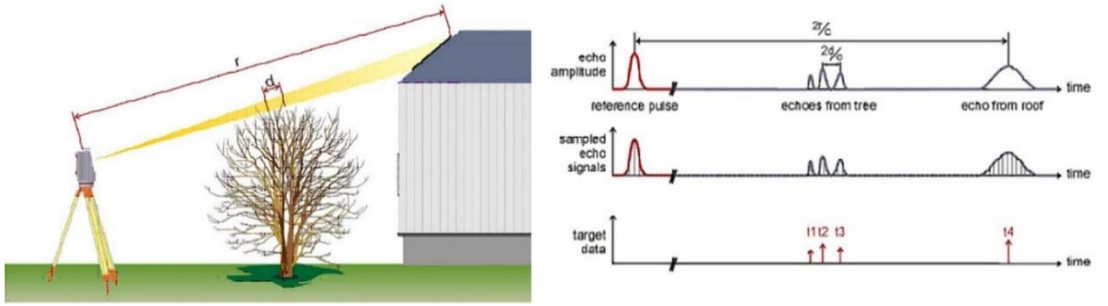


Fig. 9: Full waveform analysis. (Source: www.riegl.com)

- Calibrated reflectance (amplitude information)
- Höhenunterschied in benachbarten Punkten

- Machine learning/ deep learning

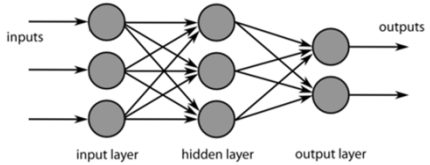


Fig. 10: Structure of a DNN (deep neural network). https://en.wikipedia.org/wiki/Deep_learning

The transfer of convolutional neural networks (CNNs) to 3D point data requires rasterization and manual selection of attributes that are represented in the raster.

Winiwarter & Mandlburger (2019) presented a neural network based on PointNet by Qi et al. (2017) that instead directly uses 3D point clouds, along with attributes attached to the points, as input, considering neighbourhoods of points. The proposed method has the option of further improvement by use of additional data such as echo or RBG information.

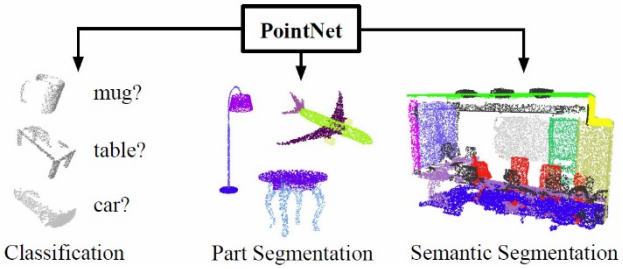
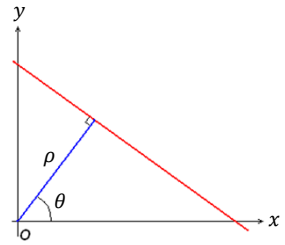


Fig. 11: Application of PointNet. (Source: Qi et al., 2017)

Detection of 2D curves (lines)



Hough (1962) exploited the point-line duality to identify the supporting lines of sets of collinear pixels in images. In his approach, pixels are mapped to lines in a discretized 2D parameter space using a slope-intercept parameterization ($y = mx + b$). Each cell of the parameter space accumulates the number of lines rasterized over it. At the end, the cells with the largest accumulated numbers (votes) represent the lines that best fit the set on input pixels. Unfortunately, the use of slope-intercept requires a very large parameter space and cannot represent vertical lines.
Duda and Hart (1972) replaced the slope-intercept with an angle-radius parametrization based on the normal equation, naturally handling vertical lines and significantly reducing the memory requirements of the algorithm: $\rho = x\cos\theta + y\sin\theta$ (Hesse normal form)

Given a single point in the plane, then the set of all straight lines going through that point corresponds to a sinusoidal curve in the (ρ, θ) plane, which is unique to that point. A set of two and more points that form a straight line will produce sinusoids which cross at the (ρ, θ) for that line.

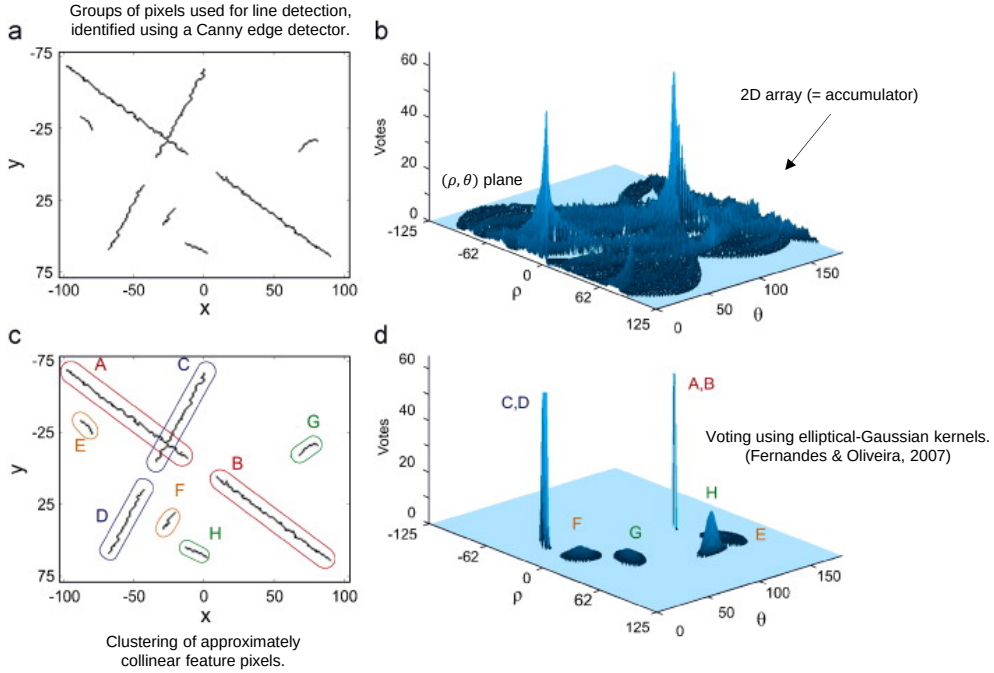


Fig. 1: The conventional versus the proposed voting procedureof Fernandes & Oliveira (2007). (a) A simple image containing approxiamtley straight line segments. (b) A 3D visualization of the voting map produced by the conventional voting scheme. (c) Clustering of pixels into segments. (d) A 3D visualization of the voting map produced by the proposed technique of Fernandes & Oliveira (2007).

Detection of 3D objects (planes and cylinders)

Hough transform can be used for the detection of 3D objects in range data or 3D point clouds. For generalized plane detection using Hough transform, the plane can be parametrized by its vector **n** **(**using spherical coordinates) and its distance from the origin ϱ resulting in a three dimensional Hough space. This results in each point in the input data voting for a sinusoidal surface in the Hough space. The intersection of these sinusoidal surfaces indicates presence of a plane (Rabbani, 2006).

$$\mathbf{n} = (\cos\theta\sin\phi \quad \sin\theta\sin\phi \quad \cos\phi) \qquad 0 \le \theta < 2\pi \qquad 0 \le \phi < 2\pi$$

$$\varrho_k = x_P\cos\theta_i\sin\phi_j + y_P\sin\theta_i\sin\phi_j + z_P\cos\phi_j \quad \text{(Hesse form of the plane)}$$

Hough transform has also been used to find cylindrical objects in point clouds using a two step approach. The first step finds the orientation of the cylinder and the second step finds the position and radius (Rabbani & van den Heuvel, 2005).

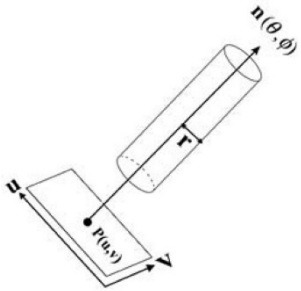
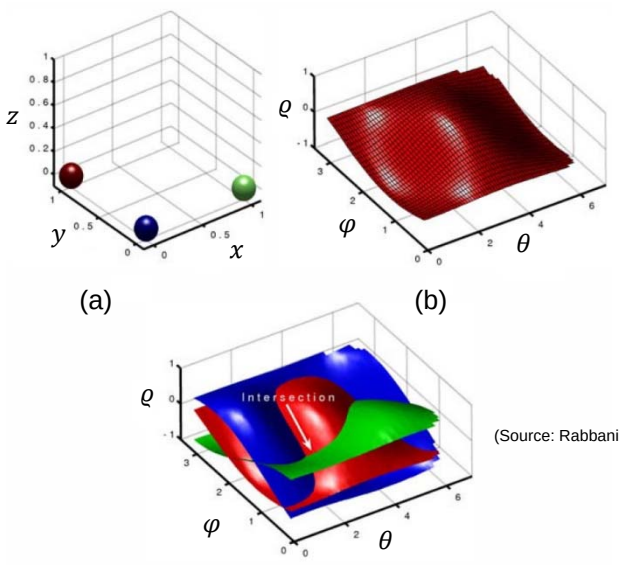


Fig. 3: The 5 parameters of a cylinder.



(Source: Rabbani, 2006)

Fig. 2: Voting in the Hough space for plane detection: (a) input point cloud, (b) the Hough space for one point, (c) the Hough space for three points, with the intersection of the three voting manifolds giving the parameter of the plane.

Source: https://en.wikipedia.org/wiki/Hough_transform

3-D Kernel-based Hough transform for plane detection (3DKHT)

Limburger & Oliveira (2015) suggested a deterministic technique for plane detection in unorganized point clouds whose cost is $n\log(n)$ in the number of samples, achieving real-time performance for relatively large data sets. It is based on a fast Hough-transform voting strategy for planar regions, inspired by the Kernel-based Hough transform (KHT), Fernandez & Oliveira, 2008.

Modellierung
Literaturquellen (1/2)

Auswahl:

- CloudCompare: https://www.cloudcompare.org/
  Dokumentation https://www.cloudcompare.org/doc/
- Beis, J., Lowe, D.G., 1997: Shape indexing using approximate nearest-neighbour search in high-dimensional spaces. Conference on Computer Vision and Pattern Recognition, Puerto Rico, 1000-1006.
- Bentley, J.L., 1975: Multidimensional Binary Search Trees Used for Associative Searching. Communication of the ACM, Volume 18, Issue 9, Sept. 1975, 509-517.
- Doneus, M., Pfenningbauer, M., Studnicka, N., Ullrich, A., 2009: Terrestrial waveform laser scanning for documentation of cultural heritage in woodland. In: Takkase, Y., Proceedings of the 22nd CIPA-Symposium, Kyoto, Japan, p. 1-6.
- Duda, R.O., Hart, P.E., 1972: Use of Hough transformation to detect lines and vurves in pictures. Comm. ACM, 15(1), 11-15.
- Fernandes, L.A.F.,Oliveira, M.M., 2007: Real-time line detection through an improved Hough transform voting scheme. Pattern recognition, Vol. 41, Issue 1, 299-314.
- Franz, M., Carrea, D., Abellán, A., Derron, M.H., Jaboyedoff, M., 2016: Use of targets to track 3D displacements in highly vegetated areas affected by landslides. Landslides, 13, 821-831.
- Geist, M., Niemeyer, F., Gierschner, F., 2018: Modellierung – Strategien zur Interpretation von 3D-Punktwolken. DVW-Schriftenreihe, Terrestrisches Laserscanning 2018 (TLS 2018), Band 93, Wißner-Verlag, 59-74.
- Holst., C., Schmitz, B., Kuhlmann, H., 2016: TLS-basierte Deformationsanalyse unter Nutzung von Standardsoftware. DVW-Schriftenreihe, Terrestrisches Laserscanning 2016 (TLS 2016), Band 85, Wißner-Verlag, 39-58.
- Hough, P.V.C., 1962: Method and means for recognizing complex patterns. United States Patent Office, No. 3,069,654.
- Kazhdan, M., Bolitho, M., Hoppe, H., 2006: Poisson Surface Reconstruction. In: Polthier, K., Sheffer, A. (Eds.), Eurographics Symposium on Geometry Processing, 61-70.
- Kazhdan, M., Klein, A., Dalal, K., Hoppe, H., 2007: Unconstrained Isosurface Extraction on Arbitrary Octrees. In: Belyaev, A., Garland, M. (Eds.), Eurographics Symposium on Geometry Processing, 9 pages.
- Kazhdan, M., Hoppe, H., 2013: Screened Poisson Reconstruction. ACM Transactions on Graphics (TOC), Volume 32, Issue 3, June 2013, 13 pages.
- Limberger, F.A., Oliveira, M.M., 2015: Real-time detection of planar regions in unorganized point clouds. Pattern Recognition, 48(6), 2043-2053.
- Maur, P., 2002: Delaunay Triangulation in 3D. Technical Report No. DCSE/TR-2002-02, University of West Bohemia in Pisen, Czech Republic.
- Mayr, A., Rutzinger, M., Bremer, M., Elberlink, S.O., Stumpf, F., Geitner, C., 2017: Object-based classification of terrestrial laser scanning point clouds for landslide monitoring. The Photogrammetric Record, 32, 160, 377-397.
- Nüchter, A., 2016: Effiziente Speicherung großer Punktwolken – Datenstrukturen für Algorithmen für mobile und terrestrische Laserscansysteme. DVW-Schriftenreihe, Terrestrisches Laserscanning 2016 (TLS 2016), Band 85, Wißner-Verlag, 105-120.
- Qi, C.R., Su, H., Mo, K., Guibas, L.J., 2017: PointNet: Deep Learning on Point Sets for 3D Classification and Segmentation. IEEE Computer Vision and Pattern Recognition, 652-660.

Modellierung
Literaturquellen (2/2)

Auswahl:

- Rabbani, T., 2006: Automatic reconstruction of industrial installations using point clouds and images. Publications on Geodesy, 62, Netherlands Geodetic Commission, Delft, Dissertation, 153 p.
- Rabbani, T., van den Heuvel, F., 2005: Efficient Hough transform for automatic detection of cylinders in point clouds. ISPRS Archives, Volume XXXVI, Part 3-W19, 60-65.
- Szeliski, R., 2011: Computer Vision – Algorithms and Applications. Texts in Computer Science, Springer, 812 p.
- Wang, M., Tseng, Y.-H., 2004: Lidar data segmentation and classification based on octree structure. ISPRS Archives, Volume XXXV, Part B3, 308-313.
- Vosselman, G., Maas, H.-G. (Eds.), 2010: Airborne and Terrestrial Laser Scanning. CRC Press, Taylor & Francis Group, 318 p.
- Winiwarter, L., Mandlburger, G., 2019: Classification of 3D point clouds using deep neural networks. Dreiländertagung der DGPF, der OVG und der SGPF in Wien, Österreich – Publikationen der DGPF, Band 28, 663-674.