A Survey of Simple Geometric Primitives Detection Methods for Captured 3D Data

Adrien Kaiser, Jose Alonso Ybanez Zepeda, Tamy Boubekeur

A Survey of Simple Geometric Primitives Detection Methods for Captured 3D Data

Adrien Kaiser[+,*] Jose Alonso Ybanez Zepeda[*] Tamy Boubekeur[+]

[+]: LTCI, Telecom ParisTech, Paris-Saclay University [*]: Ayotle

Abstract

The amount of captured 3D data is continuously increasing, with the democratization of consumer depth cameras, the development of modern multi-view stereo capture setups and the rise of single-view 3D capture based on machine learning. The analysis and representation of this ever growing volume of 3D data, often corrupted with acquisition noise and reconstruction artifacts, is a serious challenge at the frontier between computer graphics and computer vision. To that end, segmentation and optimization are crucial analysis components of the shape abstraction process, which can themselves be greatly simplified when performed on lightened geometric formats. In this survey, we review the algorithms which extract simple geometric primitives from raw dense 3D data. After giving an introduction to these techniques, from the acquisition modality to the underlying theoretical concepts, we propose an application-oriented characterization, designed to help select an appropriate method based on one's application needs, and compare recent approaches. We conclude by giving hints for how to evaluate these methods and a set of research challenges to be explored.

Keywords: 3D data, geometric primitives, shape analysis, shape abstraction, computational geometry, data fitting

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computing Methodologies / Computer Graphics]: Computational Geometry and Object Modeling—Curve, surface, solid, and object representations

1 Introduction

1.1 Objectives

This survey aims at providing an overview, a classification and a comparison of the methods developed over the years to detect simple geometric primitives in 3D data, captured from different possible sources. As described by Woodford et al. [WPM*12], these methods are "model fitting algorithms that, given a set of features (here, points), find the most likely model instance (here, primitives) that generated those features". The concept of simple geometric primitives is further developed in Section 2.2.

The process of approximating and abstracting 3D shapes by a simple parameterization allows extreme simplification of the geometry while keeping an accurate representation of the input data. Therefore, explaining 3D data using simple geometric primitives is a way of representing it in a compact manner and makes easier any subsequent analysis that would be performed, with consequences on both performances and the ability to perform high level tasks. Figure 1 shows the goals of the detection process with the different types of input, detailed in Section 2.1, and the expected output primitives with different layers of abstraction, detailed in Section 4.2.1.

In order to exploit raw captured 3D data in practical applications, one often has to reconstruct a high-level representation, of a single object or an entire scene, providing a visual summary which is similar to the understanding acquired by a human brain. This often amounts to the description of complex objects using only a couple of simple geometric primitives, such as spheres, cylinders, planes or boxes. Such visual abstractions not only simplify the geometry and topology of the input data, but also help clarify the spatial relationships between shape components, can act as economic substitutes for visibility queries, rendering effects and physics simulation, or be used as super-structures to quickly distribute filters, edits or enriched semantics over the data, on a per-component basis. These compact primitive lists that summarize dense sampling sets are later used for processing, reasoning or interaction, with applications ranging from path finding in robotics to object placement in augmented reality, through domain meshing in CAD, control structures for freeform design and level-of-detail selection in game engines.

1.2 Historical Background

The first apparition of 3D capture happened during the 1960s with tedious techniques using lights, cameras and projectors [Ebr15].

This is the authors' draft. The final paper will be published in COMPUTER GRAPHICS *Forum*.

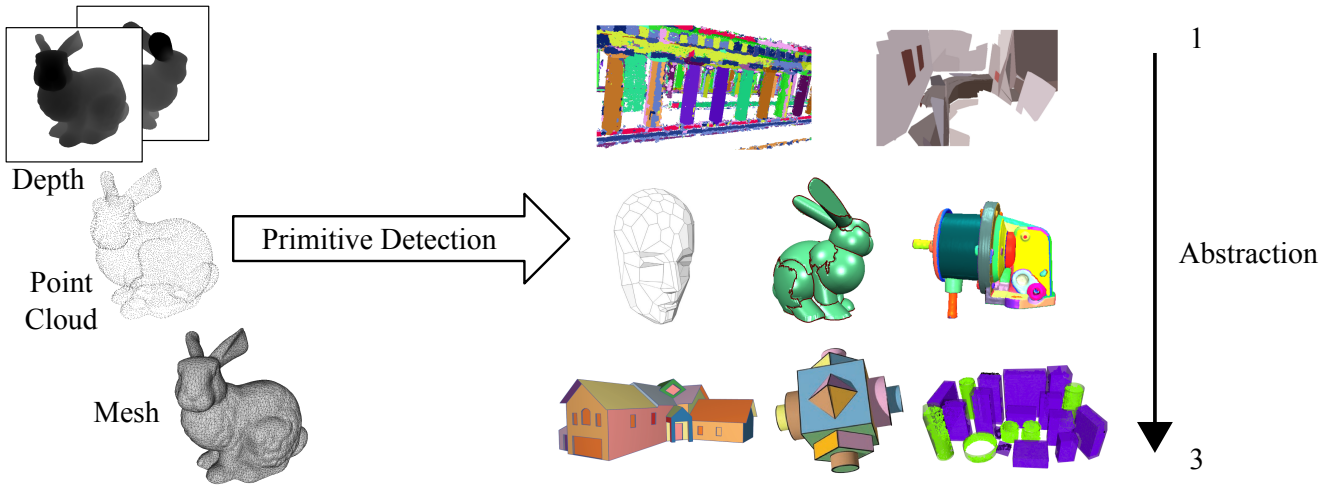


Figure 1: ***Objectives of the detection procedure***. *Using one of the input data types discussed in Section 2.1 (left), the detection process outputs primitives with different levels of abstraction, detailed in Section 4.2.1. Images courtesy of [LPRM02] (original bunny mesh), [WPM*12] (columns), [TJRF13] (reconstructed office), [CSAD04] (patched face), [WK05] (patched bunny), [SWK07] (oil pump), [LWC*11] (mechanical object), [ASF*13] (house) and [GMLB12] (cuboids and cylinders)*

However at this time, most 3D scanning devices made use of physical contact probes. In 1972, Shirai et al. [Shi72] use a basic range finder to fit polyhedrons to acquired 3D data and mark the beginning of the fitting of simple geometric primitives to 3D data. While a first attempt to model curved objects is made the next year [AB73], it is only three years after the original work of Shirai et al, that Popplestone et al. [PBAC75] successfully extend it to the modeling of cylindrical objects. In the mid-1970s, increasing interest for 3D geometric representations [Req80] fostered research in automatic modeling of captured data. In 1982, Hebert et al. [HP82] use parameter spaces to segment 3D scenes represented by depth maps into planes, cylinders and cones. The year 1983 sees the first application of simple 3D geometric primitive shapes to the registration of different views of a laser range finder [FH83]. Efficient optical methods for 3D scanning were developed in the mid-1980s and allowed faster and more accurate 3D modeling of real objects. This led to more algorithms detecting these simple shapes and multiple applications as the sensors became more available to researchers.

Although research has been intensively conducted in the area since 1972, the first studied method in this survey is from the year 1998 [LMM98], as we consider modern methods and applications of simple geometric primitive detection.

1.3 Applications

In the following, we link the application spectrum of primitive detection to selected scenarios. Indeed, substituting a dense (e.g., point) sampling by a small set of geometric primitives has numerous applications in computer graphics, computer vision, augmented and virtual reality.

Robotics. Visual SLAM (Simultaneous Localization And Mapping) is an ongoing research area in robotics whose goal is to localize an agent within a 3D map of the environment, constructed at the same time. This area grew from feature-based to dense, to using depth data, which now allows a detailed geometric analysis of the environment around the sensor. The tracking of the currently viewed scene is the main component of SLAM systems and is improved with the use of primitive shapes detected in the environment, in terms of both accuracy and speed. Recent SLAM systems mainly use large planes present in the scene to improve localization. The development of autonomous mobile robots, with embedded localization capabilities, can also benefit from a lightweight representation of their environment for faster decision making during both indoor and outdoor navigation. Explaining 3D objects with simple geometric primitives also makes it easier for household robots to grasp anything that could be of use to them.

Modeling. Building accurate and lightweight models of existing scenes is possible through scene reconstruction with geometric primitive detection as a basis. In the context of a house for instance, such a model can be used to measure the dimensions of rooms or estimate the available free space. It can also be exploited to easily move furniture or other objects around the house to try different setups.

Shape Processing. Bounding 3D shapes by simple geometry or more accurate primitives eases resampling, segmentation and deformation of the underlying data. Abstracted shapes are intuitively lighter and easier to manipulate. To this end, the Sphere Mesh representation [TGB13] has been used to apply deformations to meshes at different scales, for instance.

Rendering. Rendering techniques benefit greatly from proxies made of simple geometric primitives. For instance, 3D levels-of-details [DDSD03] can be generated by fitting a collection of primitives to complex shapes, and project the high resolution

geometric and materials attributes onto maps, parameterized over the primitives and used for distant shading. Approximate occlusion culling can also be performed by substituting high resolution geometry with a compact set of primitives, used for occlusion queries. Similarly, such primitives can be used for fast soft shadowing in real time [RWS*06].

Interaction. When computing the navigation spaces in a 3D scene, simple primitives help quickly cull away entire regions of the 3D space, to restrict the authorized navigation to areas that are free from object occlusion. Laying out simple primitives over dense meshes also helps locate interactors meant for the user to interact with the scene.

Animation. The design of skinning weights and the construction of control rigs can exploit structured sets of simple primitives, to reconstruct skinned animated sequences from performance capture data [TGBE16], better track hands in real-time [TPT16] or quickly determine collision detections in physical simulation.

Architecture. The modeling of a full building requires reconstruction from 3D data, and the use of geometric primitives simplifies the consistency of the model to the constraints brought by architectural rules of regularity. For instance, Furukawa et al. [FCSS09b] present a fully automatic system to generate floor plans, that can be used as a basis for extending the building.

1.4 Related Surveys

This survey aims at discussing algorithms that make use of shape analysis, segmentation and reconstruction methods. It provides tools which are complementary to the following previous surveys, since none of them presents a clear classification of primitive detection methods:

- 1998: "A Survey of Shape Analysis Techniques" [Lon98]
- 2009: "A Benchmark for 3D Mesh Segmentation" [CGF09]
- 2014: "State of the Art in Surface Reconstruction from Point Clouds" [BTS*14]

1.5 Contributions and Organization

In this paper, we provide:

- a summary of existing methods for simple primitive detection;
- a discussion on the links between different algorithms and
- a classification of the methods.

In Section 2, we recall basic concepts involved in the primitive detection procedures. Section 3 defines the different theoretical paradigms used in the presented methods, giving a first classification. Section 4 lists specific context criteria and properties that allow comparing methods. Section 5 analyzes and compares the actual methods. Section 6 provides insight into the way to evaluate primitive detection, lists available implementations and potentially useful test datasets. Section 7 concludes the survey and describes the future challenges in this area.

In the text, mathematical notations are as follows:

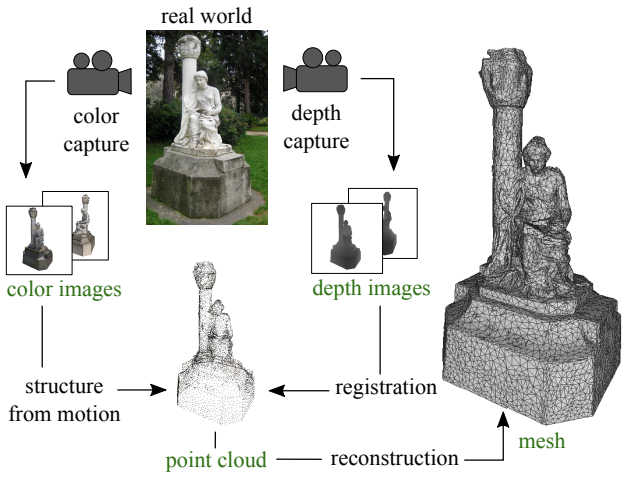


Figure 2: ***3D acquisition pipeline*** *for the statue in the Arenes de Lutece, Paris, France. The green captions indicate potential bootstrap stages for primitive detection.*

- **bold** letters denote vectors, i.e. triplets of real values representing coordinates, either a position or direction.
- non-bold letters denote real and integer values or matrices when specified.

2 Background

2.1 Input Data

Three dimensional data can be modeled with different representations, each of which favoring different families of processing methods. The specific forms of input data used for simple geometric primitive detection are further presented below, and the full acquisition pipeline is shown in Figure 2. As presented in this figure, the data suitable as input for primitive detection can be formatted as depth images, point clouds, or polygonal meshes. Although most methods are developed for one particular type of data, some implementations handle any form of input by performing a conversion to their specific input format.

Images. The raw format of acquired 3D data is the depth map directly extracted from depth sensors and represented by a grey level image. Depth sensors are often coupled with a regular RGB camera to provide additional color information. The natural 2D grid structure of these RGB-D images allows fast processing of 3D data.

Image Sequence. When scanning an object or a room, users can move a handheld sensor around them. The goal is to acquire as many views as possible and build the most accurate reconstruction. This generates a high number of 2D images that can be used as they are or post-processed. Typically, a sequence of RGB-D images gives a sequence of 2.5D colored point clouds representing the same items, which are then consolidated into a single 3D point cloud. On the other hand, a sequence of RGB images must first be processed via stereo vision algorithms to produce a 3D point cloud ready for primitive detection.

Point Cloud. A point cloud is the most common representation of acquired 3D data, as it is simply a point sampling of the real world. It takes the form of a list of 3D positions, possibly with additional attributes, such as per-sample normals and color values. If computed from an image or an image sequence, the point cloud may be organized in multiple 2D grid structures, which allow fast browsing and pointwise neighborhood query using the sensor topology.

Meshes. In computer graphics, numerous advanced processes exploit polygonal representations of surfaces, such as triangle or quad meshes. Their explicit topology makes easier operators such as filtering, resampling and rendering, either based on projection or intersection search. They are typically generated from acquired point clouds using surface reconstruction algorithms. These may use an inside/outside volumetric indicator function, e.g., *moving least squares* [ABCO*03], *implicit multiple radial basis functions* [OBA*03] or *gradient-based Poisson solution* [KBH06]. Other algorithms are based on a Delaunay-based triangulation, e.g., use tangent planes [HDD*92], Voronoi filtering [AB99] or Power crust [ACK01]. See the work of Berger et al. [BTS*14] for a complete recent survey.

2.2 Simple Geometric Primitives

A simple 3D geometric primitive is defined as a 3D geometric shape with the following characteristics:

- fixed and limited number of global intrinsic parameters i.e., that only define the global size, orientation and position of the shape;
- convex (except for the torus);
- symmetric;
- basic shape which can be assembled with others to construct more complex shapes.

This definition matches the use of primitives in Constructive Solid Geometry [Fol96], where complex shapes are built using Boolean compositions of these simple objects. We classify the geometric primitives used for detection in 3D data into four categories described in detail below, from the most simple with few parameters to the most complicated ones:

- planes;
- cuboids and boxes;
- spheres, cylinders and cones;
- other shapes: ellipsoids, tori, non-rectangular parallelepipeds.

Figure 3 shows different simple primitives of varying complexity, sorted by category.

Our interest lies in the fitting of primitive shapes to surfaces in order to approximate the boundary of objects and not their volume. We consider all simple primitive shapes as surface patches rather than as volumes of solid shapes. To be more specific, most of the methods that detect primitives in 3D data output trimmed shape surfaces, whose extent corresponds to that of the modeled object. Details on primitive trimming are given in the paragraph below.

Moreover, this survey does not consider the following primitives:

- conservative bounding primitives, i.e. bounding envelopes that strictly contain the input data, as our focus is on fitting primitives to surfaces rather than building envelopes to model volumes. This includes axis-aligned bounding boxes (AABB), oriented bounding boxes (OBB) or their generalization, the Discrete Oriented Polytopes (k-DOP). In contrast to methods that seek a single, extremely simplified shape as envelope of a complex object, we rather look into sets of simple primitives which model the boundaries of the objects as closely as possible;
- specific Computer Aided Design primitives such as rolling ball blends and other parametric patches [FS96];
- non-natural quadric surfaces; natural quadrics are spheres and right circular cylinders and cones [HHNM80].

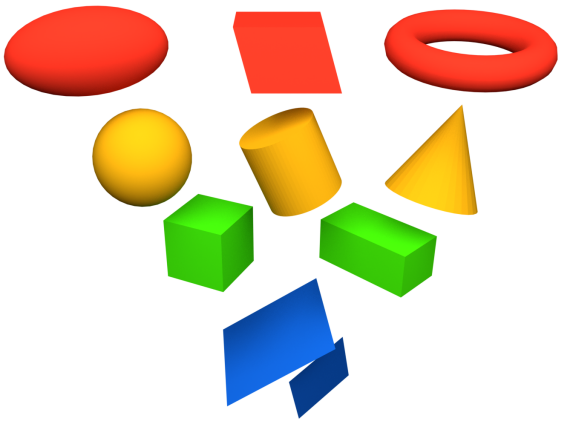


Figure 3: ***Common geometric primitives.***

Planes and Planar Surfaces. A plane is the most basic isotropic three-dimensional shape and the most commonly seen primitive in human-made environments. It can simply be defined by a normal vector $\mathbf{n}$ and its distance $d \in I\!R$ to the origin point of the reference frame. Methods using planar patches instead of infinite planes usually compute the convex hull or bounding rectangle of the set of plane inliers, or determine the limits of patches by clipping planes to each other.

Boxes and Cuboids. Boxes and cuboids are sets of assembled orthogonal planes and are defined by their center $\mathbf{C}$, orientation vector $\mathbf{n}$ and the three lengths of their sides. They can also be represented by their eight vertices or by the parameters of the planes forming them.

Spheres, Cylinders, Cones. A sphere is an isotropic shape and can be defined by its center $\mathbf{C}$ and a scalar $r \in I\!R_+$ representing its radius.

A cylinder can be parameterized with a point $\mathbf{C}$ belonging to its axis, a radius $r \in I\!R_+$ and an additional vector $\mathbf{n}$ representing its orientation.

A cone is parameterized by its center $\mathbf{C}$, called apex, an orientation vector $\mathbf{n}$ and an angle α between its axis and surface.

Ellipsoids, Tori, Parallelepipeds. Ellipsoids, tori and non-rectangular parallelepipeds are geometric shapes of higher complexity.

An ellipsoid is defined by its center **C**, axis **n** and three radii $a, b, c \in IR_+$, also called semi axes.

A torus is defined by its center **C**, axis **n**, minor and major radii m and M.

A general parallelepiped is defined by its center **C**, orientation vector **n**, the three lengths of its sides and three interaxial angles.

Primitive Shape Trimming. In the context of this survey, simple primitive shapes are considered as surface patches whose goal is to approximate the bounded surface of objects. When parts of the data has been identified as belonging to a primitive shape, some methods stop the processing while the shape has an infinite (planes, cylinders, cones) or full extent (spheres, cuboids, boxes, parallelepipeds, ellipsoids, tori). However, multiple algorithms go further in the analysis and estimate the extent of the fitted shape that models the actual object. In particular, Schnabel et al. [SWK07] define a regular grid on the surface of detected shapes in order to identify connected components. Such a grid can also be used to compute the *convex hull* of inliers in the space of the shape [And79]. Another solution is to fit smaller shape patches to the surface and merge them in the final model [BV11]. This extent appears naturally in region growing algorithms such as the method proposed by Feng et al. [FTK14]. However, in the spirit of shape approximation, it should be carefully taken into consideration that the boundary curve should also be designed in a compact way using simple 2D shapes such as rectangles, circles or regular convex polygons. The definition of more complex trimming curves such as convex hulls or the use of a 2D grid on the surface might indeed compromise the simplicity and light weight of the primitive decomposition.

3 Theoretical Foundations

This section presents the different theoretical concepts used as a basis for simple geometric primitive detection in existing methods. Families of such methods include:

- stochastic: RANSAC, local statistics;
- parameter spaces: Hough-like voting methods, parameter space clustering;
- other clustering techniques: primitive-driven region growing, Lloyd-like automatic clustering, primitive-oblivious segmentation followed by fitting.

Note that a number of methods are based on more than a single theoretical paradigm.

Table 1 and Figure 4 give strengths and weaknesses for these different theoretical frameworks. In Table 1, the reference algorithm is the most representative of each framework and usually has the most features and the best quality output.

3.1 Stochastic

Framework	Strengths	Weaknesses	Reference algorithm
RANSAC	- Simple - General - Accurate - Robust to outliers	- Many parameters to tune - Dependent on a minimum set - No spatial consistency	Fast RANSAC [SWK07]
Local statistics	- Application-specific	- Model-dependent	Monocular Occupancy Maps [CSM12]
Hough transform	- Handles missing data - Supports many model instances - Relatively robust to noise	- Unbounded space size - Dependent on parameter space quantization	Primitive-based registration [RDvdHV07]
Clustering parameter space	- Robust to outliers	- Restricted to low dimensions	Cluster Normal Space [HHRB11]
Primitive growing	- Meaningful segmentation - Spatial consistency	- Slow - Local - Sensitive to initial conditions (seeds) - Sensitive to noise - Sensitive to outliers	Hierarchical Modeling [AP10]
Automatic clustering	- No prior on location - Few parameters	- Dependent on seeds - Sensitive to outliers - Can require numerous clusters (K-means)	Quadric Surface Fitting [YWLY12]
Segmentation then fitting	- Vast literature for segmentation	- Can merge different primitives - Sensitive to noise - Sensitive to outliers	Hybrid City Representation [LM12]

Table 1: ***Strengths and weaknesses of the main theoretical frameworks***

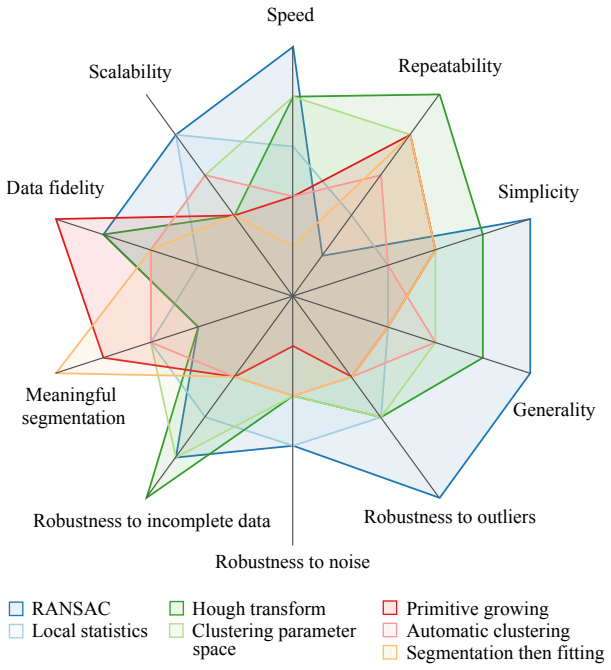


Figure 4: ***Qualitative comparison of theoretical methods.***

RANSAC-based. RANdom SAmple Consensus (RANSAC) is a popular stochastic method used to estimate model parameters iteratively given a data set. It is known to be particularly robust to outliers. First introduced by Fischler et al. [FB81], RANSAC has been used in a variety of applications, especially in computer vision and image processing.

The basic principle of the algorithm is to try many possible randomized models that could fit the data and evaluate how good this model is in order to find a consensus, i.e. an agreement of most of the data samples. Here, the term "most" is to be defined depending on the application. Specifically, the algorithm is composed of two main steps. The first one is a randomized sampling over the data to find a minimum set of samples allowing to compute parameters for the model. The second step consists in counting how much of the dataset is represented by this model and keep the model that fits most of the data.

Variants of *RANSAC* usually keep the random sampling part but use a more elaborate method to compute the score and choose the best model, instead of the simple inlier count. The score to maximize or minimize can be the median of squared errors to the model [Rou84], a squared error function where outliers are given a fixed penalty in *MSAC* [TZ00] or the maximum likelihood in *MLESAC* [TZ00]. *Randomized RANSAC* [MC04] introduces a speed-up to the original *RANSAC* algorithm by running a pre-test on a few data points before score computation, called $T_{d,d}$ test, which allows detecting and discarding wrong models very early. *PROSAC* [CM05] makes a change to the random sampling strategy by selecting samples within a small subset of data, ordered by confidence values, and increasing its size until a suitable model has been found.

Algorithm 1 provides the algorithm in detail, in the context of geometric primitive detection in an oriented point cloud. In that specific case, the randomized sampling occurs on line 4 where three oriented vertices are randomly selected. The consensus search then happens on line 8, where we keep only the model with the most votes, i.e., the geometric primitive that has the most inliers.

Algorithm 1 RANSAC

1: **Input:** List of vertices and associated normals
2: **while** there are too many unassigned vertices **do**
3: **for** N times **do**
4: Randomly select three vertices
5: Compute the parameters of a primitive that sweeps them
6: Compute the number of inliers for this primitive that are close enough to it
7: **end for**
8: Keep the parameters with the most inliers assigned to it and remove them from the pointcloud
9: **end while**
10: **Output:** List of primitive parameters and the corresponding inliers.

Examples of RANSAC Shape Detectors. The efficient iterative primitive detection method presented by Schnabel et al. [SWK07] makes use of the RANSAC paradigm to detect planes, spheres, cylinders, cones and tori given an unorganized oriented point cloud as input. At each iteration, primitives of differents types are fitted to a minimal set chosen with several heuristics. They claim that three 3D points and associated normals are sufficient to estimate the parameters for all these shapes (Section 4.1 of their paper). Furthermore, methods have been developed in order to efficiently compute shape parameters from minimal sets [DMPT01, BGZ16]. Then, the best fit is kept and the corresponding inliers are removed from the point cloud for the algorithm to carry on with the next iteration. This method and its implementation have since been used in many follow ups, as it is designed to be efficient by giving a stochastic answer to the problem of RANSAC iteration count. Its direct application [SWWK08] uses the primitives to build a topology graph and match shapes in 3D point clouds. Another application by Li et al. [LWC*11] adds a regularization step for mechanical objects. Assuming regular relations of coplanarity, coaxiality and orthogonality between object parts, the high-level modeling made of geometric primitives is optimized to form a cleaner and more regular model. Following the same paradigm, the *multiBaySAC* algorithm [KL15] uses Bayes rule to randomly generate multiple primitive hypothesis that fill up a parameter space in which the best candidates are identified.

Some formulations of RANSAC shape detectors add constraints to the input data, such as connectedness [SWK07], specific orientation of the shapes [SHFH11] or adjacency [ASF*13]. However, the stochastic nature of RANSAC can be controlled to alter the speed, quality and completeness of the shape detection. This control is possible using the probabilistic distribution sampled to find minimal sets, or by applying filters to the output of the random selection. It is often given to the user through parameters such as the maximum distance from inlier to its shape or the minimum number of inliers for a shape [SWK07, LA13]. More details on the control and tuning of algorithms are given in Section 4.2.2.0.4 and metrics are discussed in Section 6.

Local Statistics. The definition of occupancy probabilities at space locations allows inferring local primitive parameters from these distributions. Mostly, such methods aim at bounding detected objects with boxes or cylinders. The probabilities can be defined at all or some locations in space using e.g., a simple Gaussian, Gaussian Mixture Models or Bayesian inference [BFF15]. The analysis of this probabilistic field enables the detection of the objects positions.

Examples of Shape Detectors using Local Statistics. For example, Carr et al. [CSM12] create occupancy maps from registered RGB views, where each ground location is associated with an occupancy probability for vehicles or pedestrians. By deconvolving these maps with primitive specific kernels corresponding to the projection of boxes and cylinders, the algorithm is able to highlight object locations. A mean shift procedure then allows recovering the position and orientation of cuboids and cylinders that bound vehicles and pedestrians in the scene.

Bagautdinov et al. [BFF15] identify objects, especially persons, standing on the floor of indoor rooms acquired through depth sensors. The algorithm builds a Bayesian generative model of probabilistic occupancies at each location. Its optimization using

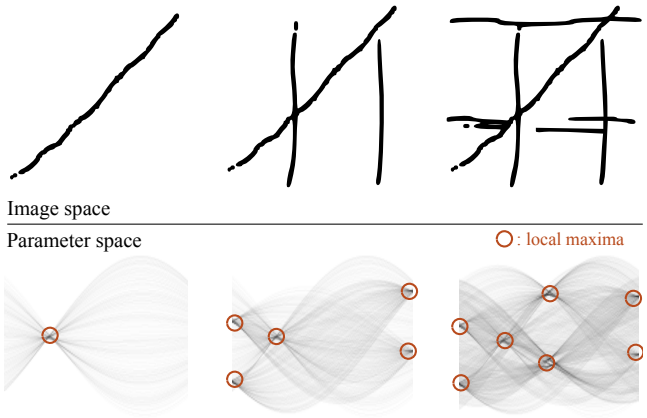


Figure 5: ***2D Hough transform for line detection****. Each pixel in image space (top) corresponds to a curve in the Hough space defined by polar coordinates (bottom). Aggregating these per-pixel curves reveals local maxima in Hough space, that model lines in image space.*

current depth observations makes it converge towards a discrete number of positions on the ground. Boxes are then fitted to the detected objects to form a clear boundary in image space.

3.2 Parameter Space

Hough Transform The Hough transform defines an accumulation space built upon a parameter space in which similar geometric elements coincide. After quantizing this space into regular bins, all samples of the data cast votes for all geometric elements of which they are inliers. The most voted parameter set identifies the object that best explains the input data. Figure 5 shows an example of 2D Hough transform for line detection. Named after Paul Hough's 1962 patent [Hou62], and originally used to detect lines in images [DH72], it has since been generalized [Bal81] to detect 2D and 3D common geometric objects, such as circles. In particular, it can be used to detect 3D shapes [WPM*14]. In spite of its generic nature, one of the most important drawbacks of this method is the lack of boundary of the parameter space. Given its potential dimension, it can become an issue in terms of memory consumption and processing time.

Research has been conducted in order to improve the performance and usability of the Hough transform. The *Fast Hough Transform* [LLLM86] uses the slope-intercept form for line equations which transforms any Euclidean point into a line. It recursively divides the Hough space into hierarchical squares which are only filled when a given line produces an intersection, computed efficiently. This structure allows fast counting of votes and identification of the model. With the same parameterization, the *Randomized Hough Transform* [XOK90] detects curves of arbitrary dimensions. Random sets of image points of the same size as the dimension of the sought curve are picked, each set leading to exactly one point in the Hough space. This point activates a cell in parameter space and iterations of random pick and cell update are run, while the list of activated cells is kept with their corresponding scores. A simple threshold on the score allows recovering curves. The *Probabilistic Hough Transform* [KEB91] also gives a stochastic answer to the complexity of the Hough Transform but keeps its one-to-many mapping from image to parameter space, thus only solves the speed issue. They show that only a random subset of points in the image is enough to correctly recover the models, although the size of the subset is not automatically defined. In contrast to the usual algorithmical definition of the Hough Transform, Stephens [Ste91] derives an analytic formulation based on the maximum likelihood method which leads to continuous values in Hough space and allows the use of known mathematical tools for locating maxima. Introduced in 2008, the *Kernel-based Hough Transform* [FO08] forms approximately collinear image edge pixels clusters in which the best fitting line and its error kernel are computed. All kernels are then merged into the Hough space and a peak detection allows identifying the correct candidates within a simplified parameter space.

Algorithm 2 Hough Transform

```
Input: List of vertices and associated normals
Quantize parameter space of dimension d into regular bins
for all vertices do
    for all combinations of parameters from 1 to d − 1 do
        Compute value of parameter d
        Increment corresponding bin
        Store vertex as inlier of this bin
    end for
end for
Detect local maxima of the quantized parameter space
Store all detected maxima as primitives
Output: List of primitive parameters and the corresponding inliers.
```

Algorithm 2 details the Hough Transform, in the context of geometric primitive detection in an oriented point cloud.

Examples of Hough-based Shape Detectors. Planes can be detected using their spherical parameterization. Each 3D point and its associated normal extracted from depth maps contributes to only one voxel in the discrete Hough space. By smoothing this space, local maxima and candidate planes are identified. Through time, correct candidates form peaks in a time-global Hough space and get activated [HSSM14]. The detected planes can then be refined using Singular Value Decomposition over the inlier sets [WO02]. Figure 6 shows a three-dimensional Hough space where each 3D Euclidean location builds a surface. The intersection of these surfaces yields the plane passing through all three points.

These planes are used for robot automation [HSSM14] or scene reconstruction [WO02, LO15]. Their main drawback – the size and memory footprint of the parameter space – can be overcome using hierarchical voxel grids depending on the number of point contributions in each cell [HSSM14]. In order to solve processing time issues, Limberger et al. [LO15] apply the *Kernel-based Hough Transform* [FO08] to 3D plane detection using an efficient spherical accumulator.

In order to simplify a given mesh model, *Billboard Clouds* [DDSD03] can also be created using a voxelized 3D

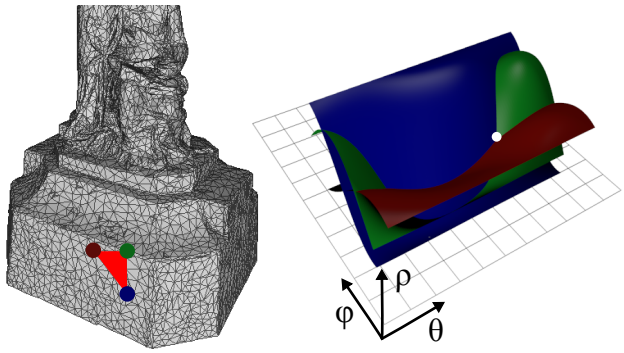


Figure 6: ***3D Hough space for plane detection****. Each 3D Euclidean location (left) builds a surface defined by spherical coordinates (right). The 3D Hough points at the intersections of these surfaces give the planes passing through many points.*

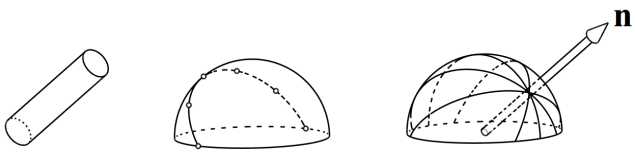


Figure 7: ***Hough transform to find the orientation of a cylinder****. Normals of all points along the cylinder (left) are represented as a unique arc on the Gauss sphere (middle). Each point along this arc translates on the Hough Gauss sphere to an arc corresponding to all possible cylinders associated with this point (right). Hence, the intersection of all arcs on the Hough Gauss sphere gives the orientation of the cylinder. Image courtesy of [RVDH05].*

Hough space. The local maxima of this space give the planes that approximate the shape, called *billboards* (textured planar polygons) that can be used to render the shape using very few primitives. The resolution of the voxel grid gives control over the approximation of the mesh. A coarse voxel grid tends to merge many triangles into a single billboard, and a finer grid will produce more billboards for a better approximation.

Rabbani et al. [RVDH05] use a Hough-based method to detect cylinders in point clouds. First, the orientation of the cylinder is detected using a 3D Hough parameter space lying upon the Gauss sphere (see Figure 7). Once the orientation of the cylinders have been found, the corresponding inliers are projected on the orthogonal plane and the Hough transform for circle fitting in 2D allows recovering the radius and position of the cylinder.

Clustering Parameter Space. Some methods directly exploit and analyze parameter spaces to detect simple geometric primitives. These methods mainly detect plane clusters in point clouds, as the parameter space for planes can be divided into two simple disjoint spaces. These are the parameter space for normal vectors i.e., Gauss map modeling the plane orientation as a point on the unit sphere – and the Euclidean distance to the origin. Algorithm 3 provides details on the clustering of the plane parameter space applied to

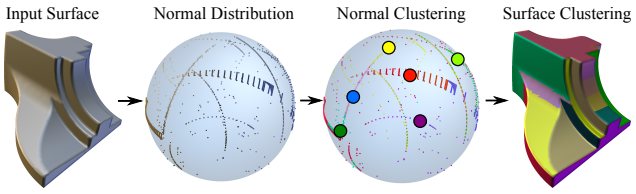


Figure 8: ***Normal space clustering****. Clusters are generated from the point density map of the surface normal vectors on the Gauss map.*

an oriented point cloud. Step 1 (line 2) represents the clustering of all data points based on their normal orientation, which leads to a list of local maxima of the normal space (line 8). At this stage, the output does not represent primitive shapes, but rather groups of points of similar orientation. Step 2 (line 9) takes as input the output of Step 1, hence the list of normal space maxima (line 11), and clusters the points within each maximum (line 13) based on their distance to the origin (line 17). Eventually, the list of points clustered with their normal and distance represents the detected primitive shapes (line 18).

Examples of Shape Detectors using Parameter Spaces. By clustering the space of normal orientations over a hemisphere using voxels [HHRB11] or a mean shift procedure [CC08], as shown in Figure 8, dominant orientations can be found. The individual plane clusters can be extracted using a threshold on the distance in Euclidean space [HHRB11] or point density peaks along the detected directions [FCSS09a]. Parameters can then be estimated using Principal Component Analysis within the clusters [CC08].

Algorithm 3 Clustering Plane Parameter Space

1: **Input:** List of vertices and associated normals
2: **Step 1:** Cluster normal space
3: Quantize normal space into regular bins
4: **for** all vertices **do**
5: Increment bin corresponding to the normal of the vertex
6: Store vertex as inlier of this normal bin
7: **end for**
8: Detect local maxima of the quantized normal space
9: **Step 2:** Cluster distance to origin space
10: Quantize distance space into regular bins
11: **for** all local maxima of the normal space **do**
12: Empty all bins of the distance space
13: **for** all inlier vertices of this maximum **do**
14: Increment bin corresponding to distance of the vertex
15: Store vertex as inlier of this distance bin
16: **end for**
17: Detect local maxima of the quantized distance space
18: Store all detected maxima as primitives
19: **end for**
20: **Output:** List of primitive parameters and the corresponding inliers.

3.3 Clustering

Several methods use known segmentation and clustering techniques to discover primitives in 3D data. As described by Lukacs et al. [LMM98], "segmentation is most commonly treated as a local-to-global aggregation problem with similarity constraints employed to control the process". In the context of 3D geometric primitive detection, these similarity constraints drive the detection towards different paradigms. Specifically, three main types of clustering are used in the literature:

- primitive-driven region growing;
- automatic clustering and Lloyd-based algorithms;
- primitive-oblivious segmentation followed by primitive fitting to the regions.

Primitive-driven Region Growing. The region growing algorithm is used to extract connected components in a depth map or a point cloud with neighborhood information (e.g., k-nearest neighbors). A label is assigned to a seed sample and its neighbors are iteratively analyzed and assigned the seed's label if their characteristics are similar enough to the seed's. These characteristics, such as color, depth or normal orientation are considered similar given a threshold which is usually application-dependent. Algorithm 4 provides the region growing procedure in the context of primitive detection, called "Primitive Growing".

Algorithm 4 Primitive Growing

```
Input: List of vertices and associated normals
while there are unprocessed vertices do
    Initialize a new region with a vertex
    Add the vertex to the list of vertices to process
    for all vertices to process do
        mark the vertex as processed
        for all neighbors of the current vertex do
            if the neighbor is similar to the vertex then
                add the neighbor to the region
                add the neighbor to the list of vertices to process
            end if
        end for
    end for
    if the current region is large enough then
        keep the region as new primitive
    end if
    empty the current region
end while
Output: List of primitive parameters and the corresponding inliers.
```

Examples of Primitive Growing Methods. The processing is started by assigning seeds to random [OLA16, LMM98] or regular [ZYH*15] positions in the data. The growing of points into regions can be performed through a neighbor search using efficient data structures such as a neighbor graph [FTK14, AEH15]. A shape-based flood filling can also be used [LLL*12, ZXTZ15]. Some methods over-segment the data into patches, *super-points* or *super-regions* [LGZ*13] that can be joined together to form the final segmentation. Algorithms used to merge these patches include rectangle fitting [MPM*14, OLA16], candidate generation and selection [MMBM15], linear interpolation [TGB13] or automatic merge of neighboring shapes [LMM98, AFS06, ZYH*15]. Making use of the efficient structure of images, a few methods offer a speed-up over point cloud based methods [TGRC13, KHB*15, AEH15]. A refinement step is usually applied to estimate plane parameters. It can be based on least squares fitting [TGRC13, BSG*11], principal component analysis [SMGKD14], RANSAC [LLL*12, SXZ*12, AEH15] or shape-driven pixel-wise region growing [FTK14].

Points are aggregated into regions using similarities with their neighbors in terms of different heuristics related to the primitive shapes to detect. For plane detection, planar heuristics include Euclidean distance [SMGKD14, SXZ*12, LGZ*13], normal orientation [MMBM15, OLA16, TGRC13] and surface curvature [ZXTZ15, MPM*14, BSG*11]. More general methods often use the primitive fitting error, computed as the mean square error resulting of a potential aggregation [AP10, XZZ*11, LMM98]. Other metrics include the *spherical quadric error* [TGB13], *cylindricity measure* [ZYH*15] and tensors capturing the local dimensionality of the data [Sch04]. Gelfand et al. [GG04] compare point clusters in terms of *slippable motions*, defined as rigid transformations that, applied to a simple geometric shape, will not form any gaps between the original and transformed shapes.

A special kind of primitive growing algorithms use a tree representation of the data [AP10, TGB13]. Points are initialized as separate clusters and ordered according to the cost of aggregating them with nearby clusters into a single primitive, based on previously discussed metrics. Iterative aggregations are then performed, starting from the least costly, in order to build a hierarchical partition of the data. The hierarchical model allows navigating through different levels of approximation, that can be used as an intermediate control structure to deform the input model.

Automatic Clustering. Automatic clustering methods in machine learning are often based on Lloyd's clustering algorithm [Llo82], developed privately in 1957 to create a partition of point sets in Euclidean spaces. This algorithm proceeds iteratively in order to cluster all points into regularly distributed regions. The two main Lloyd-based algorithms are the *K-Means* and *Mean Shift* clustering methods, described below.

The *K-Means* procedure, first defined by MacQueen [Mac67], takes as input a spatial data set and a fixed number k of clusters to detect within this set. After selecting k random data points called *means*, the remaining points are assigned to the closest mean according to a chosen distance metric. Using the so-initialized clusters, new means are created at their respective centroids. The cluster assignment and means update steps are repeated until convergence. These two steps can also be seen as iterations of expectation and maximization (EM) steps, making the k-means algorithm a variant of the EM method. Figure 9 shows the process in a simple 2D example. Algorithm 5 explains the procedure in detail, in a general N-dimensional vectors context.

In contrast to the k-means algorithm, the *Mean Shift* algorithm

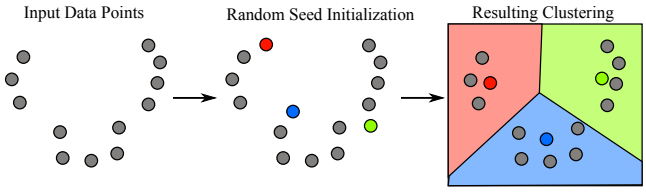


Figure 9: ***K-means clustering***. *Starting from raw data points (left), 3 initial seeds are randomly generated (middle). The location of each seed is then iteratively refined to the barycenter of the data points that are closer to it than to the other seeds, until obtaining a final stable clustering (right).*

Algorithm 5 K-Means Clustering

1: **Input:** list of nD points
2: initialize means as K random positions
3: **while** means are updated **do**
4: assign each point to its closest mean (expectation step)
5: update all means as centroid of their inliers (maximization step)
6: **end while**
7: **output:** list of K mean positions and their corresponding point assignments that form clusters.

does not require a prior number of clusters to be found. Originally presented by Fukunaga and Hostetler [FH75], and first applied to computer vision by Comaniciu and Meer [CM02], it creates partitions of a feature space. The data is considered as samples of a probability distribution made of kernel instances and points are assigned to the corresponding kernels. In other words, this algorithm is seeking the modes of a distribution modeled by kernels, among which the two commonly used ones are the flat and Gaussian kernels. These kernels are recovered by iteratively shifting each point towards a kernel center using weighted contributions of the original points in the dataset. The kernels then appear naturally to form the data clusters. Algorithm 6 explains the Mean Shift procedure in detail, in a general N-dimensional vectors context.

Algorithm 6 Mean-Shift Clustering

1: **Input:** list of N nD points, fixed bandwidth σ
2: **for** each data point $\mathbf{x}$ **do**
3: initialize the current mean m at $\mathbf{x}$
4: **while** m is shifted **do**
5: compute the center of gravity $\mathbf{C}$ of all points using the chosen kernel density function $p()$ with a fixed bandwidth σ centered at current mean m (Expectation step): $\mathbf{C} = \sum_{i=1}^{N} p(\mathbf{x}_i, m, \sigma)\mathbf{x}_i$
6: shift m to $\mathbf{C}$ (Maximization step)
7: **end while**
8: record the mean m for $\mathbf{x}$
9: **end for**
10: **Output:** list of means positions and their corresponding point assignments that form clusters.

Examples of Shape Detectors using Automatic Clustering. Several methods were inspired by these automatic clustering algorithms to fit geometric primitives to 3D data. They were pioneered by *variational shape approximation (VSA)* [CSAD04], which aims to find a fixed number of planar proxies to simplify a meshed object with the best possible approximation. Randomly picked data samples are used as initial shapes to bootstrap the process. Then, the optimization starts with an iteration of geometry partitioning and shape fitting that is repeated until convergence of the partitions. The partitioning is done using region growing started from the centers of the current shapes and using their parameters. Regions are grown based on distortion error between adjacent samples and the surface of the shapes. The fitting step computes the shapes that minimize the error to their associated samples and the partitioning starts again with the newly computed shapes. Yan et al. [YWLY12] developed a similar method to estimate quadric surfaces from a given mesh model. In the specific case of the method proposed by Woodford et al. [WPM*12], points are assigned by expanding or contracting neighboring primitives, leading to an unspecified number of primitive shapes.

Primitive-oblivious Segmentation. Several methods apply a segmentation step to the data prior to fitting, in order to reduce the number of outliers, thus getting a more accurate model. The segmentation of three-dimensional data, in the form of a point cloud or depth map, is carried out through algorithms such as region growing, watershed by flooding, classification or other clustering methods presented below. In this context, the segmentation methods do not take into account primitive search: they are completely oblivious to the primitive detection. Instead, they use heuristics such as location, color, distance or other application-specific features.

Examples of Segmentation-based Shape Detectors. Region growing and the search for connected components are the most commonly used techniques to segment images and 3D data. This method, presented earlier in this manuscript with the use of primitive-specific neighbor comparison, can also be applied to the raw data with a comparison based on heuristics such as depth discontinuities or color. For example, Martinovic et al. [MKRVG15] first label a point cloud for different architectural elements, independently of primitive shape considerations. Connected components are then extracted for each element of the facade using architecturally-based features.

Supervised classification methods use previously labeled data to characterize points into a number of classes. Depending on the application, these semantic classes are defined using features that discriminate them well from one another. In scene analysis, they are typically used to segment buildings from vegetation [LM12], walls from desks, or object parts [KLM*13]. For example, Lalonde et al. [LVHH06] use local geometry features in natural outdoor scenes to label flat areas as trails, linear areas as trees and scattered areas as leaves. Other geometric attributes include elevation or horizontality [VLA15]. See more details about learning methods in Section 4.2.2.0.3.

The watershed algorithm is a segmentation method that uses the gradient magnitude image computed from the input image and

considers it as a heightmap inside which water would be dropped. Starting from a number of markers in the image, the watershed by flooding technique [BL79] makes the water level rise up from these locations to find local maxima of the height map. These local maxima represent the watershed of the image gradient and thus are the limits of the segments in the image. Such flood filling algorithms are most suitable for depth maps as input images, in the context of 3D data segmentation [WGC99].

Using 3D data as input, applying geometry rules to detect different parts in the data is easy. These rules can be based on the height or size of clusters in the data. For example, different tables of an indoor scene can be clustered using horizontality and distance. Then, objects are projected on the tables planes and clustered in 2D to form object segments [RBMB09, GMLB12].

In some of the primitive detection methods, interactive techniques are used to drive the segmentation prior to fitting. The user can be asked to assign labels to regions [OVWK14, WGC99] or drive the segmentation using strokes [CZS*13, SXZ*12, SAG*13].

Finally, many segmentation methods have been developed when using meshes [Sha08], usually by adapting known image processing algorithms to 3D data. In particular, data-driven methods [XKH*16] that perform segmentation based on databases of labeled meshes are more and more used thanks to the availability of this segmented data. Using powerful descriptors, shapes can be segmented very efficiently and robustly using the diversity of the many segmented meshes.

Fitting Primitives to Segments. In order to compute the parameters of geometric primitives, a fitting method is finally applied to the individual detected clusters.

To do so, one common method is to use Principal Component Analysis to extract these parameters [WGC99, KLM*13]. By computing the covariance matrix of all points in the segment, an eigenanalysis allows recovering the primitive information. For instance, the normal of an optimal plane fitting the data is the eigenvector of the covariance matrix with the lowest corresponding eigenvalue. The covariance matrix for a set of N 3D points $\mathbf{X}_i = (x_i, y_i, z_i)_{i=1\ldots N}$ with their centroid $\bar{\mathbf{X}} = \frac{1}{N}\sum_{i=1\ldots N}\mathbf{X}_i$ is given by

$$\frac{1}{N}\sum_{i=1\ldots N}(\mathbf{X}_i - \bar{\mathbf{X}})(\mathbf{X}_i - \bar{\mathbf{X}})^T. \quad (1)$$

One can also apply the RANSAC algorithm (see Section 3.1) to the segments, which allows getting a faster and more accurate model as the segmented objects or parts often correspond to one primitive shape [RBMB09, GMLB12].

Finally, the primitive fitting problem can often be modeled as an unconstrained optimization problem. Thus, it can be solved using least-squares fitting by minimizing an energy representing the distance from the model to the data. Different energies can be used and are detailed in section 6.2. For instance Lukacs et al. [LMM98] provide per-primitive energies in the context of least-squares fitting, based on the efficient parameterization of primitive shapes brought by Marshall et al. [MLM01]. Depending on the complexity of the energy, the minimizing parameters can be estimated using standard linear system solvers such as Cholesky and QR matrix factorization or Singular Value Decomposition [GVL96]. Complex energies are minimized with iterative optimization methods like Newton's method or a Gradient descent [Avr76].

3.4 Assembling Primitives

After detecting primitive shapes with one or more of the techniques presented above, some methods apply different refining operations and assemble shapes together in order to generate more meaningful results.

Examples of Primitive Assembling Methods. Methods aiming at bounding objects, rooms or buildings by boxes or cuboids sometimes assemble detected planes using their boundaries [JX13, KHB*15, SHFH11]. In order to correctly group together the right planes, a generate-and-test strategy is often used. Candidate cuboids are generated from detected planes and activated based on the fitting error or penalty functions [SHFH11]. Such a strategy applied at different levels of detail leads to a hierarchical modeling of the data, which can be particularly useful in the context of urban modeling [LGZ*13, CLL11].

A few algorithms exploit geometric relations between detected shapes in order to obtain a more regular model. Global relations between parts are identified for mechanical objects [LWC*11] or buildings [ASF*13, MMBM15] under specific regularity assumptions.

4 Characterization

In this section, we enumerate the key characteristics that we use to describe and compare the methods. We propose a classification inspired by practical needs. It is based on characteristics that are important to applications and relate to the theoretical basis used by the individual methods. Table 4 is certainly one of the most important tools provided by this survey. It has been structured to help the user decide on the proper primitive recovery method to choose depending on their needs. This table lists the characteristics described in the following for all presented methods; Section 5.5 details this table.

4.1 Context

Detected Primitives. The first characteristic that distinguishes the presented methods is the type of primitives they aim to detect. Section 2.2 lists the most common primitives detected in three-dimensional environments. Figure 10 shows a classification of the presented methods in relation to the detected primitives.

Application Context. The context in which the input data is provided is directly linked to the application. It plays an important role in the choice of the inherent algorithm, parameters, priors and type of output information needed. The main input contexts are:

- indoor scenes: single room, building interiors, household scene;
- outdoor scenes: urban environment made of buildings, residential scene, natural scene;
- individual objects: regular, free form or organic shapes.

Figure 10 shows a classification of the presented methods in relation to their input context.

Specific Application Contexts. In man-made scenes for example, the primitive detection method shall make use of architectural rules and try to identify regular relationships [LWC*11, MMBM15], orthogonality or object support [SHKF12]. Although in natural scenes, the complexity of the vegetation requires very specific modeling primitives [LVHH06].

In the specific case of natural and organic shapes, the simplified model made of geometric primitives is an approximation of the original shape [CSAD04, WK05, TGB13]. The resulting set of primitives usually represent an over-segmentation of the object and given the method and heuristics used, the computed segments may or may not have semantic meaning. A simplification of better accuracy can be obtained with slightly more parameters using *Generalized Cylinders* [ZYH*15].

While the methods to segment indoor and outdoor scenes do not fundamentally differ, it is important to note that the design of algorithms is significantly impacted by the application context. As an example, indoor scenes are usually closed volumes [OVWK16] while urban and natural environments are open, hence are often partial datasets [XAZ12]. In addition, the type of input data usually differs given the context due to 3D capture devices e.g., RGB-D cameras that work well inside buildings, will not provide many data samples outside, where one might prefer using multi-view stereo datasets. Last, indoor scenes mostly contain relations of regularity between planar surfaces e.g., floor, walls and ceiling [MMBM15], while outdoor scenes are rarely regular, especially in natural environments [LVHH06].

Data. Both the type of input data and the core data structure are considered as characteristics. The input data type is obviously important in the application, while the core data structure helps determine the best algorithm to use. Section 2.1 presents the different types of three-dimensional data structures.

Detection Category. The category of the detection procedure used by a specific method corresponds to the theoretical foundation upon which the algorithm is built. This category is usually defined from the input data type and the application.

4.2 Properties

The following list of properties characterizes the methods and represents a tool to evaluate and compare them. It stems from the practical case where they are used and relates to accuracy, practicality, information and robustness. We also account for the amount of control that the user has to tune the output of the method.

4.2.1 Accuracy

Data Fidelity. The data fidelity value associated with a given method measures the accuracy of the modeling and represents the fidelity of the output model to the input data. This property allows distinguishing methods aiming at bounding objects and parts using primitive shapes (fidelity value 1) from methods aiming at simplifying them more faithfully (higher fidelity values). Table 2 explains in detail the three possible data fidelity values in the context of geometric primitive detection. Figure 11 shows a plot of the presented methods in relation to their data fidelity value.

Value	Characteristics
1	Approximating planes or bounding boxes
2	Planes fitting planar data only
3	Planes and primitives fitting all data

Table 2: ***Data fidelity scale****. A value of 1 corresponds to a set of planes or planar patches approximating the data. Typically, the approximation term here is related to an average error between the model and captured objects that represents more than about 5% of the bounding box of objects, depending on the application (see Section 6 about error metrics). Bounding boxes and bounding cylinders are also associated with this value of 1. A value of 2 corresponds to a set of planes fitting planar data, while non-planar data is not modeled. Starting from a data fidelity of 2, the average error between the model and the captured objects should be below 5% of their bounding box. A value of 3 corresponds to a set of primitives of all sorts modeling all parts of the data.*

Abstraction Level. The level of abstraction represents the simplicity of the output produced by a method. Here, we propose a measure of complexity which is based on the number of elements of the model and their relations. A model with many elements will be more complex, thus less abstract, than a model with a few related elements. This property allows distinguishing methods aiming at partitioning surfaces (abstraction level 1) from methods that segment objects into meaningful parts (abstraction level 3). This distinction is particularly important as the former requires stitching the detected primitive shapes together in order to acquire a complete decomposition [LWC*11, ASF*13]. Note that the stitching of primitives together appears naturally in some algorithms [CSAD04]. In the right part of Figure 1, the top row corresponds to abstraction level 1, the middle row is abstraction level 2 and the bottom row is the highest level of abstraction, level 3. Table 3 explains in detail the four possible abstraction levels in the context of geometric primitive detection. Figure 11 shows a plot of the presented methods in relation to their abstraction level.

Level	Characteristics
0	Raw point cloud
1	Primitive patches
2	Full primitives
3	Assembled primitives

Table 3: ***Abstraction scale****. Level 0 is given for information and more clarity. Level 1 corresponds to an unorganized set of small primitives covering parts of objects. Level 2 corresponds to a set of primitives covering all of the object and representing meaningful parts of it. Level 3 corresponds to an organized set of primitives corresponding to full objects or object parts with semantic meaning. In addition, spatial relations between primitives may also be present at level 3.*

4.2.2 Practicality

Timing. In the scope of this survey, the evaluation of the timing performance of a primitive detection method is based on the type of

data processed. Indeed, methods that will be considered real-time require a few milliseconds to one second to process their input data, for a dataset of average size in relation to all datasets targeted by the method. The decision of considering a method as real-time or not is based on the results advertised by the methods themselves, hence depending on the hardware available when the method was developed, which makes difficult a comparison based on the real-time property.

Examples of Timings. Methods that process multiple RGB-D frames per second [AEH15, ZXTZ15, HSSM14, TGRC13, CSM12, HHRB11] are considered real-time compared to methods that require several seconds to process a single frame [BFF15, CLW*14, JX13, SXZ*12]. When processing point clouds, some non real-time methods require several seconds to process 5000 points [MMBM15, MPM*14, LGZ*13, LWC*11, SHFH11] when others can process tens of thousands of points in less than a second [SWK07]. Several methods [OVWK16, CLL11, CC08] need only a few seconds to process hundreds of thousands of points, which means that they could process 5000 points datasets in less than a second. Although, they are not real-time considering the type of input data that they target, which are aligned point clouds representing a full scene. Reconstructed meshes are usually heavy datasets and most methods processing them are not real-time. For instance, methods that require several seconds or even minutes to process a single shape are not real-time [ZYH*15, YWLY12]. On the other hand, Thiery et al [TGB13] process 6000 vertex meshes in 300 milliseconds, which is considered real-time in this survey.

Scalability. The scalability of a method reflects its behavior when the amount of data increases, with bigger data sets or when a camera moves around an environment for example. In particular, in the scope of this survey, the focus is set on the good performance in terms of accuracy, speed and memory consumption.

Examples of Scaliblity. Methods based on Region Growing techniques are not very scalable in terms of timing, due to the inherent costly iterative behavior over the whole data set. On a different point, parameter spaces need to be quantized, which often limits their accuracy and scalability in terms of memory consumption. In contrast, stochastic methods perform well when increasing input data size, as they tend to localize interesting areas of the data and focus on those, instead of the full data set.

User Assistance. In some cases, the user's assistance can be requested. It can be optional to either oversee a decision of the system [ZXTZ15] or improve the reconstruction after the detection stage has been performed [ASF*13, KLM*13, WO02]. The user may also be required to perform tasks in order to guide the detection, such as assigning labels to regions [OVWK14, WGC99], labeling the data as a prior for a learning phase [LGZ*13], or even driving interactively the segmentation [CZS*13, SXZ*12, SAG*13]. In all cases, a graphical user interface is required in order for the user to perform their task properly. Note that in robotics applications, holding and moving the camera around the scene is not considered as an assistance, as the user is not involved in the actual scene reconstruction.

Examples of User Assistance. Shao et al. [SXZ*12] ask the user to perform strokes on color images when the automatic segmentation is not satisfactory. Similarly, *3-Sweep* [CZS*13] needs the user to draw initial coarse strokes on object boundaries and draw additional strokes following the object, for the algorithm to understand where to fit the model. Prior to the reconstruction algorithm of Ochmann et al. [OVWK14], a manual intervention is required in order to assign RGB-D scans to the different rooms. The algorithm of Lin et al. [LGZ*13] requires a manual labeling step prior to applying learning techniques for automatic classification. Through a user interface, Whitaker et al. [WGC99] ask the user to manually label matching planes in two 3D views. In the context of indoor scene reconstruction, Zhang et al. [ZXTZ15] use their graphical interface to allow the user to validate or invalidate object segmentation hypotheses. Several other systems [ASF*13, KLM*13, WO02] propose interfaces to provide high level information about the scene, allowing the user to limit the spatial extent of primitives and specify spatial relations between them.

Learning Phase. Several methods require an offline learning phase prior to the detection, in order to acquire the parameters of the different classes of objects that will be recognized in the scene. Usually, the learning process is applied to a set of previously labeled data, often manually, which requires the user to directly assist in the detection process.

Examples of Learning Phases. Two types of input data are used for training: manually annotated images [KHB*15] or point clouds [MKRVG15, LVHH06, LGZ*13] and automatically generated images [SXZ*12] and point clouds [CLW*14]. From this training data, descriptors are computed based on geometric properties such as position, orientation, size, planarity and saliency features, spin images [JH99] or appearance properties such as color or smoothness and consistency terms. Using these descriptors, different models can be learned such as Random Forests [SXZ*12, MKRVG15], Adaboost models [LGZ*13], Conditional Random Fields (CRF) with Support Vector Machines [KHB*15] or Gaussian Mixture Model with Expectation-Maximization [LVHH06]. These models are used to classify data as semantic objects such as houses, street lights, cars, trees, windows, walls, balconies, doors [MKRVG15, LGZ*13, LVHH06] or match known object meshes and find their orientation in the data [CLW*14, SXZ*12]. Khan et al. [KHB*15] learn the CRF model parameters that later allow image labeling and validation of primitive hypotheses.

Intuitive Tuning. In data fitting, the user may need to tune the processing depending on the input data or the needs in the output. For example, to model an indoor scene, the user might prefer a few meaningful shapes, even though the fidelity to the data is lower. On the other hand, to model a complex object, a larger number of shapes, offering a better accuracy, might be preferred. Therefore, one of the properties of a detection method is the ability to control the output in an intuitive manner. This is characterized by few parameters that the user can easily identify and for which impact can easily be perceived in the output. As for every algorithm, the goal is to find a suitable compromise between speed of the processing and quantity and quality of the output. In the context

of this survey, this is modeled by data fidelity and abstraction level values. In order to set these parameters for their input data and application, users are sometimes referred to the default values given in the paper [LM12] or software. To tune the parameters and better suit their needs, a good method is to estimate a value and then refine it with empirical trial-and-error iterations.

Examples of Control Parameters. We divide control parameters into two main categories. The first set of parameters are related to the output of the algorithm and are usually the more intuitive ones. Among them, control over the quality of the output is given by thresholds on distance [DDSD03, VLA15, MMBM15, LA13], angle deviation [OLA16, VLA15] between inliers and primitive shapes or confidence values [CLW*14]. To control the quantity of the output, the number of primitives to detect [TGB13, WK05, CSAD04] or the minimum surface area [VLA15] are sometimes required. On a higher level, the constrained spatial relations between shapes may also be provided [MMBM15]. Sometimes, these parameters can be set automatically given the extent of the data in space [SWK07]. A second family of parameters influence directly the algorithm and are more abstract. Spatial parameters such as the resolution of a voxel grid [FCSS09b] or parameter space [LO15] or the radius of a neighborhood [VLA15, LM12] are usually quite intuitive. On the other hand, algorithm-specific values such as the number of iterations [GMLB12], optimization weights [ASF*13, JX13, SXZ*12, WPM*12] or primitive initialization method [CLL11] do not make much sense to the user. Such parameters often have noticeable influence over the output given by the algorithm, as they allow controlling the compromise between accuracy and simplicity. Thus, they are quite easy to tune empirically in spite of their unintuitive nature.

Temporal Consistency. The consistency of the model through time is important when modeling the surroundings. This is a key property when capturing dynamic environments where objects or people move around, or when the camera itself is moving. Ideally, the methods that best handle time model the changes in the different observations as part of the representation that is built. This model should be updated to follow the changes in the observed scene and is therefore evolving and dynamic itself. For a static scene, the model should be stable, thus should not differ when generated from acquisitions done at different times. This property is very relevant for acquisition of scenes or objects using sequences of RGB-D images, but implies complex temporal cross-mappings for full meshes or point clouds.

4.2.3 Information

Semantics. Some techniques offer semantic information in addition to geometric primitives. Most of these methods rely on a prior classification model, which might be precomputed or given as output of a learning phase. Some approaches however infer semantic information from local geometry, such as local saliency features [LVHH06], but this information is usually limited.

Examples of Semantics. While some methods deduce semantic information by differentiating object parts [KLM*13] or full object meshes [SXZ*12], other differentiate indoor [OVWK14, OVWK16] or outdoor [LGZ*13] parts of houses, such as walls, doors and floors. In the context of city modeling, methods tend to model different parts of cities such as building facades, balconies, chimneys, trees or cars [VLA15, MKRVG15, LM12].

Needs Extra Information. Several methods require additional information for the detection to succeed. This can take the form of local attributes for all samples of the dataset or meta data that are global to the whole set. Extra attributes to the 3D data can be needed to assist the processing and make it faster or more robust. The most instrumental ones are color, which provides structural information, and local geometry attributes such as normals, curvature or neighboring sample distribution. The organization of 3D data as a 2D image (sensor topology), available when using depth maps, also provides useful information to speed up the processing.

Examples of Extra Information. Some methods make use of meta data to specialize the processing for a given context, which usually take the form of additional input parameters. For instance, Monszpart et al. [MMBM15] need prior information on the possible type of relations between objects, given by the angles between the primitives. In addition, most urban modeling methods assume regular arrangements between buildings and between the elements that form them. Methods that aim to model indoors sometimes try to match CAD models to detected objects. These CAD meshes are usually given to the system or defined by an online repository address. Some other methods make assumptions regarding the semantic scope of the application, given by the type of environment being observed, from household rooms to offices. In the context of robotics and scene reconstruction, some methods require as input the individual positions and orientations of cameras corresponding to the input set of acquired depth images. There can also be a requirement for the size and number of geometric objects to be detected. For the segmentation of objects, Cohen-Steiner et al. [CSAD04] as well as Wu et al. [WK05] take as input a fixed number of segments to be identified in the input data.

Provides Meta Data. In addition to the explanation of 3D data in terms of geometry, some methods output extra meta information to provide insight into other aspects of the whole dataset. For instance, house modeling methods sometimes output a graph of relative positions and orientations of detected primitives. This allows for spatial reasoning over the objects present in the scene (more details in Section 7.2). In the special case of urban building modeling, this usually takes the form of architectural rules between facade elements. When detecting objects, some methods output polygonal meshes for these objects, and they can also give matching information about similar objects. Sometimes, semantic information is also provided after the processing, to get insight into the type of objects being discovered and the type of environment being observed.

4.2.4 Robustness

Robustness to Noise. The quality of acquired 3D data depends on the performance of the acquisition device. Unfortunately, given the quality of the device used, the data often contains a certain amount

of noise. As this is a common issue, several methods try to handle the analysis while taking into account these artifacts. However, some fail to do so, for example with the use of local image features [CZS*13]. In these cases, robustness to noise quickly becomes a crucial factor in the development of a shape analysis technique. In particular, the robustness of a system will be estimated given the stability of its detection accuracy when the input data is disturbed with noise.

Examples of Robustness to Noise. Methods that best handle noisy data are based on robust theoretical paradigms such as the Hough transform [WO02] or RANSAC [SWK07]. The latter explicitly shows its robustness by corrupting input data with up to 20% of Gaussian noise and correctly reconstructing the primitive. While some methods apply a smoothing step as preprocessing on the data in order to reduce the amount of noise [KHB*15, CLW*14], other methods make use of structural priors to regularize the data [MPM*14, MMBM15, OVWK14]. Finally, Zhang et al. [ZXTZ15] create a global model which is therefore less sensitive to noise.

Robustness to Incomplete Data. When capturing 3D data, the completeness of the output depends on the acquisition procedure and the resulting information may contain holes and missing parts. This issue occurs often when trying to recover and abstract a shape. Although, the use of parametric models such as simple geometric primitives may allow reconstructing parts of objects that were not captured by the camera. The robustness of a method against incomplete data is measured by the completeness of shapes even when only parts of it were captured. A simple example is the recovery of a full table when it is poorly sampled and might not be fully captured. This occurs often in practice because of the orientation of a sensor which is mostly parallel to the horizontal surface of the table. Missing parts of objects in data captured using cameras can also be due to occlusions by closer objects, or surfaces that do not reflect light as expected by the sensor.

Examples of Robustness to Incomplete Data. While some methods fail to handle missing data [OVWK16, CZS*13], robust ones have been developed, whose theoretical background allows reconstructing missing parts [BFF15, VLA15]. Region growing based methods do not perform well when data is missing, as they only exploit existing data without trying to create a global model [TGRC13, SXZ*12, XZZ*11, LMM98]. This kind of global model, usually made of planes, actually helps recover unseen areas [MMBM15, ZXTZ15, OVWK14, JX13]. The use of priors on the scene also helps guess missing parts [MPM*14]. Other methods that handle missing data well include multiscale methods [AP10] and multiple view reconstruction [FCSS09b].

Robustness to Outliers. When detecting primitives in 3D data, the discrimination between points belonging to the model and points not belonging to any primitive is essential, and is taken into account in our classification. Another aspect of outliers is the artifacts created during the acquisition. These data points should not belong to any primitives as they do not represent samples of the observed objects and should be discarded as outliers.

Examples of Robustness to Outliers. In practice, some methods address outliers by filtering the input data [OVWK16, KLM*13, SXZ*12, LVHH06] or specifically modeling them [KHB*15, ZXTZ15]. Others are built upon theoretical paradigms known to be robust to outliers, such as RANSAC-based methods [SWK07, JX13, WPM*12, RBM*07], *Least Median of Squares* (LMS [RL05]) used by [ASF*13] or robust descriptors [MKRVG15, VLA15, MPM*14, LGZ*13]. In their study, Schnabel et al. [SWK07] show that their RANSAC-based method can handle up to 95% outliers.

5 Methods and Applications

We now analyze specific algorithms and applications. They are sorted by detected primitives (Section 2.2), then by application scope (Section 4.1). The main detection categories (Section 3) used for each context are discussed. A visual compendium of the presented methods is shown in Table 7 at the end of this document, to help the reader to quickly identify the methods. Table 4 summarizes all presented methods and their characteristics and is detailed in Section 5.5.

5.1 Planes

Indoor Scenes. Methods discussed below aim at detecting planes in the context of indoor scenes, either represented by unorganized point clouds or organized point clouds under the form of depth images.

The most used technique to detect planes in indoor scenes is to grow regions from given seed positions and stop the propagation using heuristics linked to plane detection. Starting from unorganized 3D point clouds, a first group of algorithms segments the data by growing regions using planar heuristics [OLA16, XZZ*11, MPM*14, MMBM15]. This initial primitive-based segmentation is then used for plane matching and registration of different views [XAZ12], matching object parts and joint segmentation of similar objects [MPM*14] or creation of a global regular model based on user defined priors [MMBM15, OLA16]. Although slower than stochastic methods, these region growing based algorithms output high quality and consistent models of the input point clouds. Making use of the image structure of depth data (sensor topology), which simplifies the search for neighbors, a second group of algorithms applies connected components techniques to depth maps [SMGKD14, FTK14, TGRC13, LLL*12, ZXTZ15, AEH15]. In addition, Shao et al. [SXZ*12] allow the user to draw strokes on the current image to assist the segmentation process. Applications include plane matching for view registration [LLL*12, ZXTZ15, SMGKD14] or CAD model matching for realistic scene reconstruction [SXZ*12]. The use of region growing methods when working with depth maps is motivated by the speed-up allowed by the image structure of the data, without which they would be too slow to be usable. Region growing based methods lead to models with higher quality and consistency, although they are subject to noise which is less important in indoor scenes than e.g. in outdoor environments.

Following a different paradigm, some methods exploit the stochastic nature of RANSAC to detect planes in three dimensional

data. In the robotics community, Simultaneous Localization And Mapping (SLAM) systems detect and match RANSAC detected planes to build a map of the observed scene [WS06, TRIC12, TJRF13, Kae15, ERAB15]. By designing a RANSAC-based plane detector as detailed in Section 3.1, some algorithms create a segmentation of the observed scene. Inferring spatial relations between objects using spatial information [RBM*07], hierarchical segmentation [SHKF12] or object graphs [OVWK14], leads to a consistent and human understandable model of the room. Other methods aim at simplifying the representation of the environment to automatically build floor plans of buildings [BV12, OVWK16] or create a high level representation of indoor scenes, made of CAD models [CLW*14]. The use of RANSAC-based methods allows faster processing which is critical especially in robotics applications. The refinement of the model through time improves the lesser quality and consistent results brought by a stochastic paradigm.

The Hough transform has been extended to three dimensions and plane detection in indoor environments [HSSM14, WO02, LO15]. Hough-based methods work well in indoor environments mainly made of planar surfaces, as only a few peaks will appear in the parameter space, and give stable results leading to a consistent reconstruction. A simpler clustering of the parameter space can also be performed (see Section 3.2.0.0.1). Similarly to the Hough transform, clustering the parameter space works well in indoor scenes, as shown in the results of [HHRB11]. Although, these simpler methods might be less robust than elaborate Hough-based systems.

Classical image segmentation techniques are also used to build models of indoor scenes, such as Whitaker et al. [WGC99] who apply a watershed algorithm on captured range data. The user is then asked to match planes between different views, and the corresponding images can be registered in the same space by minimizing the distance of 3D points to the equations of two corresponding planes. Although the segmentation is not based on planar heuristics, the context of indoor scenes made of large planar surfaces allows this method to perform well, using a solid, well known image processing technique.

Outdoor Scenes. Several methods have been developed to fit planes to outdoor elements such as building facades or walls and ceilings of houses.

In unorganized point clouds, planes are mostly detected with region growing techniques [Sch04, LGZ*13]. In order to build high-level models of houses in residential areas, Lin et al. [LGZ*13] add a semantic labeling step allowing the detection of house parts where RANSAC-detected planes can be hierarchically assembled. This allows the system to be robust to missing data, as the adjacency of planes and discovery of their intersections helps recover the full house structure. These local-to-global approaches are well suited to outdoor scene segmentation, as objects to detect are usually far apart from each other in the scene, e.g. the distinction between different houses.

In the context of urban scene reconstruction, normal-based clustering algorithms are also used and work well as buildings tend to have a regular structure made of many planar surfaces [FCSS09a, CC08].

Lafarge et al. [LA13] exploit the intersections between RANSAC planes to build a model of urban buildings containing both planar elements and 3D points. Arikan et al. [ASF*13] also use the intersections between planes, but add an interactive step where the user can refine poorly modeled regions. For this specific application context, the use of stochastic methods such as RANSAC might prevent all actual planes to be detected, resulting in incomplete models.

In order to automatically build 3D models of buildings, Vosselman et al. [VD01] use ground plans to pre-segment aerial images and apply a Hough-based plane detector coupled with a region growing step. The pre-segmentation step using available ground plans reduces the sensitivity of the Hough Transform to outliers and allows using a limited parameter space.

Individual Objects. Objects are usually made of more complex shapes than planes, hence only few methods have been developed in that direction, with the goal of simplifying meshes.

Billboard clouds [DDSD03] use a 3D Hough space to detect billboards and render the original shape in a similar visual quality but using very few primitives. The control given by the resolution of the Hough space allows its use in a variety of applications. *Variational shape approximation* [CSAD04] propose an automatic clustering method inspired by Lloyd's algorithm [Llo82] that requires as input a fixed number of planar patches to detect. Although this method leads to faithful and consistent results, the need for the specification of the number of patches, while giving control over the output, decreases its usability in automatic segmentation applications.

5.2 Bounding Boxes and Cuboids

Indoor Scenes. Bounding boxes and cuboids can be useful in the context of indoor scenes to build an occupancy model of the environment applied to e.g., autonomous robot navigation or object tracking.

Based on a stochastic framework, Bagautdinov et al. [BFF15] identify objects, especially persons, standing on the floor of indoor rooms. The use of local statistics is particularly robust in the presence of occlusions and can detect people even if only parts of them are seen by the camera.

Methods that detect planes at object boundaries and assemble them [JX13, KHB*15], associated with a robust candidate generation and activation method, are an efficient way to model objects by cuboids because not all planar faces of objects have to be seen.

Outdoor Scenes. In outdoor environments, bounding boxes can be useful to detect and track persons and cars for autonomous navigation or modeling parts of buildings made of cuboids.

In the context of autonomous navigation based on color images, Carr et al. [CSM12] create occupancy maps to bound vehicles and pedestrians in the scene by cuboids. The method runs in real-time

and from multiple views to track vehicles or people. Although, the real-time performance is made possible by the need for a prior registration step between the different views and is quite sensitive to the type of data that needs to be detected, thus lacks genericity.

In order to create high-level models of buildings from 3D scans, Martinovic et al. [MKRVG15] fit boxes to connected components of the building. Shen et al. [SHFH11] assemble RANSAC detected planes following horizontal splitting of the facades. Both of these methods acquire a meaningful representation of building exteriors based on architectural priors and are well fitted to the modeling of regular facades particularly present in old buildings.

Individual Objects. Kim et al. [KLM*13] build templates for 3D shapes. The deformation of an initial set of parts representing an object leads to multiple templates modeling the different types of the same object class. Object parts are bound by cuboids detected through object-wise region growing, which leads to a consistent segmentation across object instances and eases the matching between datasets.

5.3 Spheres, Cylinders, Cones

This section describes methods developed to detect spheres, cylinders and cones. However, most methods detect planes as well.

RANSAC-based algorithms have also been developed to detect simple surfaces of revolution such as spheres, cylinders and cones. Methods such as *Efficient RANSAC* [SWK07] or *multiBaySAC* [KL15], described in details in section 3.1, give stochastic answers to the weaknesses of the RANSAC paradigm. In particular, they offer an important speed-up over the original RANSAC implementation. Hence, they are suitable to many applications that require modeling by planes or simple revolution surfaces.

Indoor Scenes. Modeling objects in indoor environments using spheres, cylinders or cones can be used either for reverse engineering or automation of household robots.

Inspired by the Hough based plane detection, Rabbani et al. [RVDH05] detect cylinders in 3D point clouds to mainly model indoor industrial setups. One of the main advantages of the Hough transform is its robustness to outliers and missing parts of objects. Later, the cylinders detected using this Hough transform have been matched between different views and their parameters integrated into a formulation to estimate the motion between the views [RDvdHV07].

In the development of automatic household robots, detecting and reconstructing objects has been carried out using segmentation methods followed by primitive fitting. This leads to a simple model of the observed scene [RBMB09, GMLB12], mostly composed of tables and the objects upon them. The final object model is hybrid, made of both simple geometric primitives and meshes modeling non-simple parts of the objects such as pan handles. This gives a simple and light representation of objects while keeping a faithful model. For this specific application and context, these methods efficiently segment observed objects by projecting them on planar tables, thus reducing the dimensionality of the problem.

Outdoor Scenes Spheres, cylinders and cones are well suited models for outdoor elements such as trees or cables and urban elements such as building columns or domes. It also allows reconstructing modern urban elements such as spherical buildings.

Usually, cities and outdoor environments are easier to segment than indoor scenes, as building and natural elements have more distinctive features than objects. Therefore, applying a segmentation step prior to fitting leads to better results. The resulting primitives can be used for semantic interpretation and lead to automated robot navigation in natural environment [LVHH06] or creation of a global lightweight model of a city [LM12].

Chen et al. [CLL11] exploit the RANSAC primitive detection of Schnabel et al. [SWK07] to decompose an input point cloud and refine it to build a high-level textured model of buildings. The assembly of primitives is performed hierarchically which allows for easy decomposition of the buildings in parts. The use of a general primitive detection algorithm such as RANSAC allows recovering all kinds of regularly-shaped buildings such as e.g. half-spheres for domes, spheres for spherical buildings or cylinders for towers, in addition to traditional box-like buildings.

In order to ease the segmentation and fitting process, Wang et al. [WT04] developed a semi-automated technique to model buildings from color aerial images. The user is asked to select a primitive type and approximately fit it to the observed building part, further refined automatically. Although this method has a strong requirement for user interaction, it achieves a faithful reconstruction of buildings from solely aerial views.

Individual Objects. The most frequent use of revolution surfaces such as spheres, cylinders and cones are for the modeling of individual objects. They can be mechanical and made exclusively of this type of shapes, or organic shapes that can be approximated.

Primitive-based region growing methods are favored to model parts of these objects as those are usually made of one connected component [LMM98, ZYH*15, TGB13, AFS06, AP10, BSG*11, GG04]. Both simple mechanical parts [LMM98] and more complex or organic objects [TGB13, ZYH*15] can be divided into parts that are easily differentiable by geometric primitive heuristics, hence the meaningful segmentation performed by these methods.

Lloyd based methods also perform well with individual objects because clusters will naturally tend to segment object parts. These methods, usually developed for triangle meshes, perform iterations of triangle assignment and primitive fitting to the clusters [WK05, YWLY12]. These extensions of *Variational Shape Approximation* [CSAD04] give more accurate results with fewer but more complex primitives.

The regularization of RANSAC-detected shapes [SWK07] by Li et al. [LWC*11] makes RANSAC-based modeling of mechanical parts very efficient and accurate, especially in the context of reverse engineering of CAD models. Although, the use of such stochastic methods may lead to non-connected parts and require further processing in the space of the shape.

In order to build a 3D model of objects viewed in a single RGB image, Chen et al. [CZS*13] require the assistance of the user who draws strokes over the different parts of the objects. These

strokes, coupled with image processing information such as edges, allow defining object boundaries modeled by warped cylinders and cuboids. It is then possible to interactively modify the 3D properties of the objects such as position, rotation and orientation, and recover a nearly photorealistic image thanks to inpainting techniques. In addition to fitting generalized primitives to strokes of a drawing, Shtof et al. [SAG*13] use semantic classification of the strokes to automatically snap the primitives and regularize their spatial relations. The result is a globally regular primitive modeling of the input drawing. Obviously, these interactive methods are not suitable for automatic batch processing of datasets, which limits them to the modeling of specific images given the need of the user.

5.4 Ellipsoids, Tori, Parallelepipeds

Outdoor Scenes. Complex shapes such as ellipsoids and parallelepipeds allow modeling organic objects in urban [VLA15] and natural [LVHH06] environments such as vegetation or rocks. The analysis of their parameters allows semantic interpretation and classification of the observed elements. As these natural elements rarely have a clear boundary, the methods often tend to bound the observations instead of modeling their exact surface.

Individual Objects. Organically-shaped meshes are best modeled by complex shapes such as ellipsoids [SS05]. Even though the results are quite different from the original model, this method allows very light modeling of complex meshes.

Tori are simple shapes but are mostly seen and used in industrial environments. Full tori can be detected in mechanical parts [SWK07], while partial ones model blends in regular objects [AP10] or elbows in pipes [RDvdHV07].

5.5 Summary Table

Table 4 summarizes all of the described methods to detect simple geometric primitives in 3D data and lists the characteristics presented in section 4 for all of them. Methods are ordered by inverse publication year and first author's last name. For the methods having an available implementation, the reader is referred to Section 6.4.

An interactive web app, allowing one to reorder methods according to a given characteristic, is provided as supplemental material.

The detection categories correspond to the underlying theoretical method used for the detection of primitives, described in detail in Section 3. The theoretical foundation listed in the table corresponds to the method directly applied to the raw data in order to find primitives, whether or not it allows computation of the primitive parameters. Some methods may be based on several theoretical paradigms, and for those only the first applied method is listed. For example, RANSAC refinement steps applied after primitive growing are not listed in the table, although they are described with the full method in all the above sections.

In summary, the listed detection categories are as follows:

- Stochastic: RANSAC, local statistics;
- Parameter space;
- Clustering: primitive growing, automatic clustering, segmentation + fitting;
- Other methods: user-assisted.

6 Metrics and Evaluation

This section aims at giving insights into ways to evaluate simple geometric detection methods, as well as metrics used to model the error in these methods. It also lists available implementations and datasets from some of the aforementioned articles.

6.1 Evaluation Methodology

In order to evaluate the quality of a modeling instance made of simple geometric primitives, different metrics can be used depending on the application and the performance objective, including:

- fitting error, detailed in section 6.2;
- processing time measured in milliseconds or number of processed frames per second;
- simplicity of the model and overdetection: number of primitives.

Some metrics, such as segmentation correctness, need ground truth information to be computed. This usually requires prior manual and user-assisted work on the data, which includes:

- segmentation and spatial consistency: objects are correctly separated by primitives and modeled by one instance each;
- camera poses (in the context of scene analysis).

6.2 Evaluation Metrics

In order to evaluate the quality of an output model made of simple geometric primitives compared to the input data, different metrics have been used to estimate the error made with the detected set of primitives by measuring the distance to the model, or fitting error:

- the simplest metric is the sum of squared distances from points to the their corresponding primitives. For primitives $S_i, i \in [0,N]$ gathering inliers $\mathbf{P}^i_j, j \in [0,M]$, the fitting error is
$$\varepsilon = \sum_{i=0}^{N} \sum_{j=0}^{M} \left\| \mathbf{P}^i_j - proj(\mathbf{P}^i_j, S_i) \right\|^2$$
with $proj(\mathbf{P}^i_j, S_i)$ modeling the projection, i.e. the closest point, of point P^i_j on its corresponding primitive shape S_i;

- the Hausdorff distance [CRS98] defined for two sets of points $\mathbf{a} \in A$ and $\mathbf{b} \in B$ is the highest distance among all points $\mathbf{a}$ to the corresponding closest point of B, considering $d()$ as a given real distance function:
$$H_{AB} = max_{\mathbf{a} \in A} \{ min_{\mathbf{b} \in B} \ d(\mathbf{a}, \mathbf{b}) \} \ .$$

6.3 Processing Metrics

The following metrics are used to drive algorithms, although their values do not make much sense to evaluate of the quality of the output.

- the *quadric error metric*, introduced by Garland et al. [GH97] in order to simplify meshes. By summing squared point-to-plane

Table 4: ***Comparison of presented algorithms****, details in Section 5.5. Methods are ordered by inverse publication year and first author's last name. — Not applicable / Information not available in paper. ○ Moderate / Optional. ● Good / Required. RGB image. Depth image. RGB-D image. ∴ Point Cloud. △ Mesh. □ Voxel. ∗ Sequence or Set of images.*

Method		Primitives							Context	Input	Data	Detection Category	Accuracy		Practicality						Information			Robustness			
		Planes	Boxes	Cuboids	Spheres	Cylinders	Cones	Other					Data Fidelity	Abstraction	Real-time	Scalability	Needs User	Learning	Tuning	Temp. Consist.	Semantics	Needs Info	Provides Meta	Noise	Missing Data	Outliers	Code
Walls Layout	[OVWK16]	✓							building interiors	∴	∴	RANSAC	2	3		●			●	—	○			○	○	●	
Regular Planar Modeling	[OLA16]	✓							building interiors	∴	∴	primitive growing	3	3		●			●	—				○	○	○	
Parallel RANSAC	[AEH15]	✓							indoor scene		∴	primitive growing	2	2	●	●			●							●	
Occupancy Maps	[BFF15]		✓						indoor scene			local statistics	1	3		—				—				●	●	●	
Planar RGB-D SLAM	[ERAB15]	✓							indoor scene	∗	∴	RANSAC	2	2	●	●				●			●	●	●	●	
Quaternion Representation	[Kae15]	✓							indoor scene	∗	∴	RANSAC	2	2	●	●				●			●	●	●	●	
Bayes SAmple Consensus	[KL15]	✓			✓	✓			any	∴	∴	RANSAC	3	2	●	○				—				○	●	●	
Boxes around Objects	[KHB*15]		✓						indoor scene			primitive growing	1	3		—		●	●	—		●	●	●	○	●	
3D Kernel Hough Transform	[LO15]	✓							indoor scene		∴	parameter space	3	3	●	●			●	—				●	●	●	●
3D All The Way	[MKRVG15]		✓						urban buildings	∗	∴	segmentation + fitting	2	3		●		●	○	○	●	●	●	○	○	●	
RAPter	[MMBM15]	✓							building interiors	∴	∴	primitive growing	2	3		●			●	—				●	●	●	●
Level of Detail	[VLA15]	✓		✓				✓	urban buildings	△	△	segmentation + fitting	3	3		●			○	—	●		○	●	○	●	
Labeled KinectFusion	[ZXTZ15]	✓							indoor scene	∗	□	primitive growing	3	2	●	●	○		○	●				●	●	○	
Generalized Cylinder	[ZYH*15]					✓			individual objects	△	△	primitive growing	3	2		—			○	—					○	○	●
Semantic Modeling	[CLW*14]	✓							indoor scene	∗	∴	RANSAC	3	2		—		●	●	—	●	●	○	●		○	
Agglomerative Clustering	[FTK14]	✓							indoor scene			primitive growing	3	2	●	●			●	●				●		●	●
Planar Hough Transform	[HSSM14]	✓							indoor scene	∗	∴	parameter space	2	3	○	○			○	●				●	○	○	
Cluttered Indoor Scans	[MPM*14]	✓							building interiors	∴	∴	primitive growing	3	2		●			○	—		○	○	●	●	○	●
Hierarchical Building Descriptions	[OVWK14]	✓		✓					building interiors	∴	∴	RANSAC	3	2		●	●		○	—	○	○	●	●	●	○	
Dense Planar SLAM	[SMGKD14]	✓							indoor scene	∗	∴	primitive growing	2	2	●	●			○	●			●	●	○	●	
O-Snap	[ASF*13]	✓							urban buildings	∴	∴	RANSAC	3	3	●	○	○			—				●	●	●	
3-Sweep	[CZS*13]			✓		✓			individual objects			user-assisted	3	2	●		●		●	—						●	
Fitting Cuboids	[JX13]			✓					indoor scene			primitive growing	1	2		—				—		●		○	○	●	●
Learning Object Templates	[KLM*13]		✓						individual objects	△ ∗	∴	segmentation + fitting	1	2	—	○	○			—	○		○	●	○	●	●
Point Set Structuring	[LA13]	✓							urban buildings	∴	∴	RANSAC	2	2		●			●	—				●	●	●	
Semantic Learning	[LGZ*13]	✓	✓	✓					residential scene	∴	∴	primitive growing	3	3		●	●	●	●	—	●		○	●	●	●	
Geosemantic Snapping	[SAG*13]			✓	✓	✓	✓		individual objects			user-assisted	3	3	●		●		●	—			●	●		●	●
Point-plane SLAM	[TJRF13]	✓							indoor scene	∗	∴	RANSAC	2	1		●				●		●	●	●		●	
Sphere Meshes	[TGB13]	✓			✓				individual objects	△	△	primitive growing	2	3	○	●			●	—						○	●
2D-structured Point Cloud	[TGRC13]	✓							indoor scene		∴	primitive growing	2	2	●	○			○	○		●				○	
Indoor Robot Navigation	[BV12]	✓							building interiors	∗	∴	RANSAC	1	1	●	●				●				●	●	●	●
Monocular Occupancy Maps	[CSM12]			✓		✓			any			local statistics	1	3	●	○	○		○	●		●		●		●	
Objects On Table	[GMLB12]			✓		✓			household scene		∴	segmentation + fitting	3	3	—				●	—				●	●	●	
Hybrid City Representation	[LM12]	✓			✓	✓	✓		urban buildings	∴	∴	segmentation + fitting	3	2		●			○	—	●			●		●	
Indoor Plane Mapping	[LLL*12]	✓							indoor scene	∗		primitive growing	1	1	●	●				●				●		●	
Interactive Semantic Modeling	[SXZ*12]	✓							indoor scene	∗		primitive growing	2	2				●	●	○	●	●		●		●	
Object Support	[SHKF12]	✓							indoor scene			RANSAC	1	1						—		●	●		●		●
Planar Surface SLAM	[TRIC12]	✓							indoor scene	∗	∴	RANSAC	2	2	—	○				●				○	○	●	
Contracting Segments	[WPM*12]	✓			✓	✓			any	∴	∴	primitive growing	2	1		○			○	—				○		●	
Outdoor Planar SLAM	[XAZ12]	✓							urban buildings	∗	∴	primitive growing	2	1	—	●				●		●		●		●	
Quadric Surface Fitting	[YWLY12]	✓			✓	✓		✓	individual objects	△	△	automatic clustering	3	1		○			●	—				●	○		
CAD Model Recovery	[BSG*11]	✓			✓	✓	✓		individual objects	△	△	primitive growing	3	2		○				—				●			
Plane Filtering	[BV11]	✓							indoor scene	∗	∴	RANSAC	1	2	●					●				●		●	●
Algebraic Templates	[CLL11]	✓			✓	✓			urban buildings	∴	∴	RANSAC	3	3		○			●	—				●	●	●	
Cluster Normal Space	[HHRB11]	✓							indoor scene		∴	parameter space	1	2	●	○				—		○		●	○	●	
GlobFit	[LWC*11]	✓			✓	✓	✓		individual objects	∴	∴	RANSAC	3	3		○				—			●	●	●	●	●
Facade Partitioning	[SHFH11]			✓					urban buildings	∴	∴	RANSAC	2	3		○			●	—			○	●	●	●	
Plane Detection for SLAM	[XZZ*11]	✓							indoor scene	∗	∴	primitive growing	2	2						—						○	●
Hierarchical Modeling	[AP10]	✓			✓	✓	✓	✓	individual objects		∴	primitive growing	3	2	○	○			○	—			●	○	●	○	●
Manhattan World Stereo	[FCSS09a]	✓							urban buildings	∗	∴	parameter space	2	2		○			○	○					○		
Volumetric Integration	[FCSS09b]	✓							indoor scene	∗	□	parameter space	2	1		○			●	○			●	●	○	○	
Hybrid Object Model	[RBMB09]	✓			✓	✓	✓		household scene	∴	∴	segmentation + fitting	3	2	—	○				—				○	○	●	
Architectural Modeling	[CC08]	✓							urban buildings	∴	∴	parameter space	2	3	○	●				—				●		○	
Primitive-based registration	[RDvdHV07]	✓			✓	✓		✓	indoor scene		∴	parameter space	2	2	—	○				●				○	○	●	
3D Object Maps	[RBM*07]	✓							indoor scene	∗	∴	RANSAC	2	1	—				●			○	○	●	●	●	
Fast RANSAC	[SWK07]	✓			✓	✓	✓	✓	individual objects	∴	∴	RANSAC	3	2	●	○			○	—				●	●	●	●
Hierarchical Segmentation	[AFS06]	✓			✓	✓			individual objects	△	△	primitive growing	2	2	○	○			●	—				●	○	●	●
Outdoor Robot Navigation	[LVHH06]	✓				✓		✓	natural scene	∴	∴	segmentation + fitting	2	3		●		●		●	●		●	●		●	
Patch RANSAC	[WS06]	✓							indoor scene	∗	△	RANSAC	1	1	○	●				●				●		●	
Cylindric Hough Transform	[RVDH05]					✓			indoor scene		∴	parameter space	3	2	—	○			●	—		●		○	●	●	
Ellipsoidal Modeling	[SS05]							✓	organic shapes	△	△	automatic clustering	2	2	—	○			●	—		●				○	
Hybrid VSA	[WK05]	✓			✓	✓			individual objects	△	△	automatic clustering	2	2		○			●	—		●				○	
VSA	[CSAD04]	✓							individual objects	△	△	automatic clustering	2	1		●			●	—		●				○	
Local Slippage Analysis	[GG04]	✓			✓	✓	✓		individual objects	∴	∴	primitive growing	2	2	—	—			○	—				○		○	
Tensor Voting	[Sch04]	✓							urban buildings	∴	∴	primitive growing	1	2	—					—				●		●	
Assisted Model-Image Fitting	[WT04]	✓	✓			✓	✓		urban buildings			user-assisted	1	2				●		●				○	●	●	
Billboard Clouds	[DDSD03]	✓							individual objects	△	△	parameter space	2	1					●	—					○	○	●
Hough-based Reconstruction	[WO02]	✓							indoor scene	∗	∴	parameter space	1	2		●	○		○	—				●	●	●	
Grow and Merge	[MLM01]	✓				✓	✓		individual objects		∴	primitive growing	2	3	—	○				—				○		●	
Hough-based House Modeling	[VD01]	✓							residential scene			parameter space	2	2		○				—	○			●		○	
Plane-based Registration	[WGC99]	✓							indoor scene	∗	∴	segmentation + fitting	1	2				●		●				●			
Least-Squares Fitting	[LMM98]				✓	✓	✓	✓	individual objects		∴	primitive growing	2	2	—					—						○	

Method		Language	Link
3D Kernel Hough Transform	[LO15]	C++	link
RAPter	[MMBM15]	C++	link
Generalized Cylinder	[ZYH*15]	C++	link
Agglomerative Clustering	[FTK14]	C++	link
Cluttered Indoor Scans	[MPM*14]	C++	link
Fitting Cuboids	[JX13]	Matlab	link
Learning Object Templates	[KLM*13]	C++	link
Geosemantic Snapping	[SAG*13]	C#	link
Sphere Meshes	[TGB13]	C++	link
Indoor Robot Navigation	[BV12]	C++	link
Object Support	[SHKF12]	Matlab	link
Plane Filtering	[BV11]	C++	link
GlobFit	[LWC*11]	C++	link
Plane Detection for SLAM	[XZZ*11]	C++	link
Fast RANSAC (CGAL)	[SWK07]	C++	link
Fast RANSAC (original)	[SWK07]	C++	link
Hierarchical Segmentation	[AFS06]	C++	link
Billboard Clouds	[DDSD03]	C++	link
Variants of RANSAC	(PCL)	C++	link

Table 5: ***Available source code*** *(links accessed February 09, 2018)*

Name or Reference	Type of Data	Link
[LO15]	3D point clouds	link
RAPter [MMBM15]	Point cloud + Detected Primitives	link
[ZYH*15]	3D models (OBJ) + Parameters + Detected Primitives	link
[CLW*14]	RGB-D images + Generated Model	link
[MPM*14]	Point cloud + Normals	link
[KLM*13]	3D models + Ground truth	link
SUN3D [XOT13]	Sequence of RGB-D images + Camera poses + Segmentation	link
[SXZ*12]	RGB-D images + Matched 3D models	link
NYU Depth Dataset V2 [SHKF12]	Sequence of RGB-D images + Segmentation	link
GlobFit [LWC*11]	Point cloud from individual objects + Detected primitives	link
[SWK07]	Point cloud from individual objects + Detected primitives	link

Table 6: ***Available datasets*** *(links accessed February 09, 2018)*

distances, a quadratic form appears and allows efficient evaluation of the error at any point in space. For a vertex $\mathbf{v}$ and N planes $P_i, i \in [1,N]$ with normals $\mathbf{p}_i, i \in [1,N]$:

$$\varepsilon = \sum_{i=1}^{N} \mathrm{dist}(\mathbf{v},\mathbf{p}_i)^2 = \sum_{i=1}^{N} (\mathbf{p}_i^T \mathbf{v})^2$$
$$= \sum_{i=1}^{N} \mathbf{v}^T \mathbf{p}_i \mathbf{p}_i^T \mathbf{v} = \mathbf{v}^T (\sum_{i=1}^{N} \mathbf{p}_i \mathbf{p}_i^T) \mathbf{v} = \mathbf{v}^T Q \mathbf{v} \,.$$

In the scope of the original work [GH97], summing the Q matrices associated with the two vertices of an edge allows evaluating the error produced by the collapse of this edge. This provides a global ordering of edges to collapse for progressive mesh simplification. In the field of geometric primitive detection, Yan et al. [YWLY12] perform Lloyd-like iterations based on the quadric fitting error on an input triangle mesh;

- an extension of the quadric error metric, called the *spherical quadric error metric (SQEM)* [TGB13] allows to iteratively collapse edges to spheres, with potential null radii, progressively moving from a surface to a volume representation as the model is simplified. The SQEM represents the distance from a sphere to an oriented plane and is minimized to identify the best sphere approximation for a set of triangles, with the resulting mesh of spheres connected by edges and triangles being called a *Sphere-Mesh*. This is instrumental for extreme approximation, shape editing and, through its later extensions, animated mesh analysis [TGBE16] and hand recognition [TPT16].

6.4 Available Implementations

Implementation for some of the presented methods are available online. They are listed in Table 5. Links are only usable in the web version of this survey.

6.5 Datasets

Some authors provide the data used in their work in order to reproduce the results or test it against other methods. Articles have also been published to present a benchmark on 3D data and gather a certain amount of data along with ground truth information. Available datasets are listed in Table 6. Again, links are only usable in the web version of this survey.

7 Discussion

7.1 Concluding Remarks

Common geometric shapes, such as planes, cuboids, spheres, cylinders, cones, tori, ellipsoids and parallelepipeds are the building blocks of most of the objects present in man-made environments and of some natural elements as well. Their simplicity makes them a perfect tool for the analysis of heavy and complex 3D data acquired from noisy 3D scanners, as they allow both reduction of the size of the data and complexity of the model for a computer.

For scene modeling in the context of robotics, a simple geometric primitive-based representation allows faster and more accurate processing for real-time applications and autonomous navigation. For the automatic reconstruction of objects or buildings, geometric primitives can help recover the regularity of the scanned items. In the area of computer graphics, shape processing can also make use of geometric primitives to simplify objects and apply simple algorithms for deformation or animation.

In this survey, we have described the principles and proposed a set of characteristics together with a classification for the most recent methods aiming at detecting such simple geometric primitives in captured 3D data. We categorized the detection of simple geometric primitives in 3D data such as depth images, point clouds or polygonal meshes using several well established theoretical foundations that make use of stochastic paradigms, parameter spaces or clustering and segmentation techniques.

In addition to discussing and comparing recent detection methods along several criteria, this survey provides a classification of the methods based on their applications. Characteristics such as the

type of input and output data, the context in which methods are presented and the performance of algorithms on several criteria allow easy identification of the ones that best suit an application's needs. In consequence, the different tables presented in the last part of the manuscript aim at being very instrumental and shall be seen as the main tools to exploit for further research.

We also list available implementations and datasets for some of the presented methods, for fast integration and testing within an existing framework.

7.2 Towards Spatial Reasoning

Spatial reasoning enriches data in terms of space and organization through structural information. In the case of 3D space modeling, spatial reasoning is possible by acquiring qualitative and quantitative knowledge of spatial locations in the observed scene. In the particular context of this survey, these are represented by the positions and orientations of detected objects or parts, modeled by simple geometric primitives, with relations to each other.

Several recent methods have looked into quantitatively describing these relations, in order to infer information about the scene structure. They can be represented by a graph of objects where the edges model rigid transformation matrices.

Li et al. [LWC*11] define a graph of geometric relations between parts of objects modeled as simple primitives. Initially complex because of the noise, this graph is simplified by merging nodes and a regular model can be obtained. The simplification is based on a limited number of relations of coplanarity, coaxiality and orthogonality between object parts.

To model relations between objects in a closed room, which can be particularly useful for household robots, Rusu et al. [RBM*07] define relations between detected objects with 3D rigid transformation matrices. In the same context, Silberman et al. [SHKF12] hierarchically segment the scene into objects and infer adjacency relations between them. The method allows building a graph of objects supporting each other.

In order to model full building interiors, Ochmann et al. [OVWK14] detect doors and windows to create a graph of connectivity between rooms. Monszpart et al. [MMBM15] assume a regular structure between walls and floors in the building and find regular arrangements of planes to model that structure. This creates a limited number of geometric relations between parts of the building.

As a direct follow-up to *Efficient RANSAC* [SWK07], Schnabel et al. [SWWK08] use the detected primitives to build a topology graph of object and object parts within 3D point clouds. The analysis of this graph leads to the recognition and semantic interpretation of objects in different types of scans.

Although used in the methods described above, Spatial Reasoning remains largely under-exploited, in particular with relation to the ground fitting framework (e.g., E.M.) which could embed such a notion in its core behavior.

7.3 Research Challenges

The problem of simple geometric primitive detection in captured 3D data raises numerous challenges in the context of modern applications. Although consumer depth cameras represent a great opportunity for many applications, they still raise many issues as their price range implies a lower quality in terms of noise, temporal consistency and missing data. The more general inlier/outlier decision is also existing with this type of data. Many solutions have been proposed to fix the issues due to noise and outliers or even missing data, but they usually imply a lower performance for the solution to stay generic enough. Indoor scene modeling methods based on streams of depth data seem to perform generally well, as the repetition of observations of a closed environment allows building a noise-free and consistent model through time. Most methods aim at modeling man-made environments using planes or even model curved objects with this primitive shape, because of its simplicity and the fact that it can be easily identified with known geometric heuristics such as normal orientations. Fewer methods detect more complex primitives in order to build a more reliable model of the data which allows even lighter representations for a similar quality. Therefore, future research challenges lean towards the improvement of results in terms of completeness and consistency of the model. In particular, completeness can take the form of more complex primitives, although they need to stay generic and not data-specific. Meanwhile, overall performance and compatibility with real time constraints remain a key enabler for future applications.

Interpretation The first challenge we identify when detecting geometric primitives in 3D data is to give more meaning to the detected primitives, to get more information beyond a list of 3D shapes. In order to have better interpretation and make sense of the resulting objects, several research paths could be explored.

First, semantic classification has already been used in several cases, but usually as a prior for segmentation to discriminate objects or parts instead of rising conclusions in the output.

Second, beyond the extraction of per-primitive semantics, a global consistency check is often missing in state-of-the-art methods, with two instances of the same semantic object being potentially fit with different primitive sets.

Third, the amount of variability of primitives detected in a scene, as well as their uncertainty level, are, so far, not explicitly provided on the output channel of the detection systems. Instead of a static primitive, new systems could be built to provide parametric primitives with restricted parameter spaces. Allowing the user to explore the space of possible fitted primitives could be achieved interactively through specifically modulated ranges of parameters and interfaces. A few methods were developed in that direction, in order to fit generalized primitives to 2D images. 3-Sweep [CZS*13] ask the user to draw strokes along the axes of the primitives, while Shtof et al. [SAG*13] automatically fit a selected primitive to parts of drawings. In both cases, an advanced user interface is required and the processing is not fully automatic. On the other hand, the Generalized Cylinder Decomposition [ZYH*15] automatically segments a given mesh model into a set of warped cylinders by growing regions. This type of primitives allows modeling faithfully

trivial objects from our daily lives that cannot be constructed only with the simpler primitives, while keeping a rather simple set of parameters.

To some extent, the spatial relationships between the detected primitives could offer an interpretation of the scene which goes beyond its sole geometry, including functional behavior, mechanical constraints and visibility cues. While primitive detection may often be seen as the first step of such higher level analysis, the whole process can also be designed as an interleaved loop, where the spatial relationships help better optimize the primitives and the primitive placements help better infer the relationships.

Improvement of the Processing Quite a number of methods still require long preprocessing, preventing them from being exploited in real time or on large data sets. To that end, the development of primitive detection procedures specifically designed for fine-grained parallel execution would allow to harvest modern GPU horsepower. In that direction, Oesau et al. [OLA16] recently proposed an efficient implementation of their primitive growing building modeling method. By applying parallel operations of region growing and plane fitting to separate data segments, the whole processing time could be reduced.

Similarly, parameter space methods have a significant memory footprint that prevents using them e.g. on embedded systems. Designing compressed representations for these spaces, while supporting efficient random-access capabilities, would be key to their further use in such contexts.

While most methods take raw data as input, one can think about the high level data interpretation brought by the fitted geometric primitives as information to be introduced back into the low-level capture device and help tailor the raw sampling process itself.

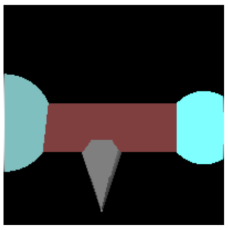
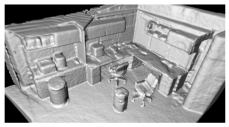
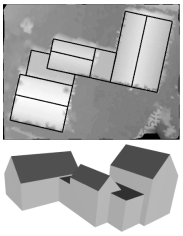
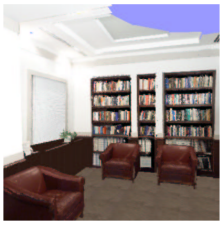
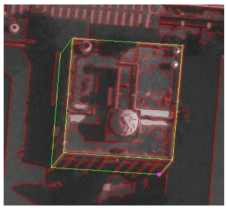
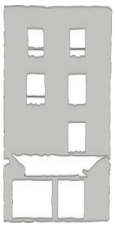
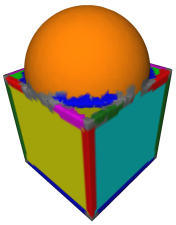
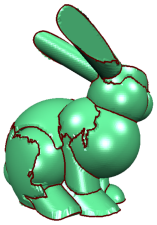
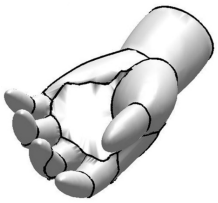
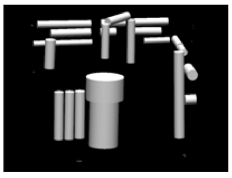
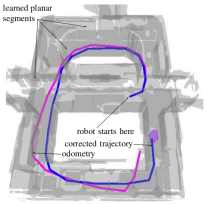
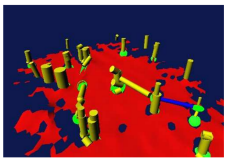
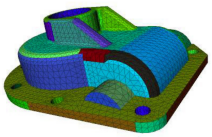
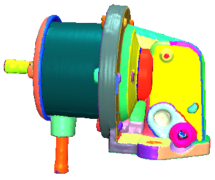
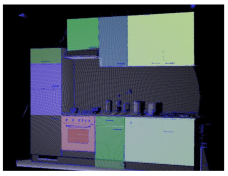
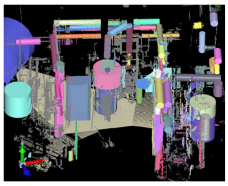
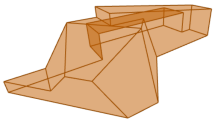
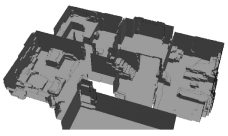
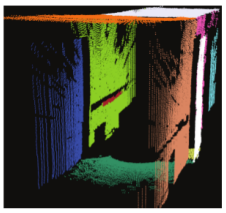
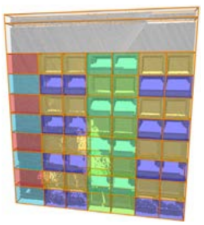
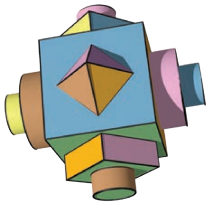
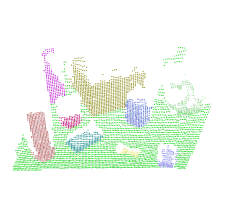
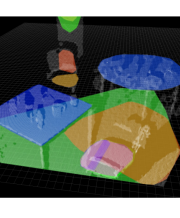
Also, with the rise of 3D+time data sets, an important shape analysis problem will be to detect "simple primitives with simple motions" in complex scenes. Again, the objective is the explanation of such scenes at higher levels, prior to more advanced reasoning.

Although a few methods already moved in that direction [TGB13, AFS06, AP10], high level primitive recognition appears to be a multiscale problem. It therefore implies a multiscale output describing complex scenes with simple primitives which have themselves a multiscale description and hierarchical relationships. A number of research directions can be designed around this idea.

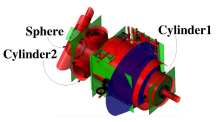
Connection to Deep Learning There are several ways to envision connections with the rising *deep learning* methodology. At first, deep neural networks may be used for detecting and fitting primitives, using a similar architecture to what current deep recognition systems use [KSH12], where convolution layers help extract advanced features from raw data and dense layers make possible reconstructing primitive placements and parameters. But simple primitives sets can also be seen as media to vectorize complex 3D scenes when it comes to learning higher level features, offering a compact representation for input/output stream of neural networks that may impact favorably time and memory consumption [WSK*15]. Last, interestingly, the geometry of a deep neural network is indeed complex and hard to analyze. Simple primitive detection algorithms might then be used to simplify inception or drive neural nets processing e.g., compression, decimation and visualization, by modeling the neural net itself, raising fundamental questions regarding its spatial embedding.

Acknowledgements This work is partially supported by the French National Research Agengy (ANR) under grant ANR 16-LCV2-0009-01 ALLEGORI and by BPI France, under grant PAPAYA.

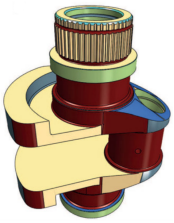
Table 7: ***Visual compendium of presented algorithms***. *Images courtesy of below cited references.*

					
Least-Squares Fitting [LMM98]	Plane-based Registration [WGC99]	Hough-based House Modeling [VD01]	Grow and Merge [MLM01]	Hough-based Reconstruction [WO02]	Billboard Clouds [DDSD03]
					
Assisted Model-Image Fitting [WT04]	Tensor Voting [Sch04]	Local Slippage Analysis [GG04]	VSA [CSAD04]	Hybrid VSA [WK05]	Ellipsoidal Modeling [SS05]
					
Cylindric Hough Transform [RVDH05]	Patch RANSAC [WS06]	Outdoor Robot Navigation [LVHH06]	Hierarchical Segmentation [AFS06]	Fast RANSAC [SWK07]	3D Object Maps [RBM*07]
					
Primitive-based registration [RDvdHV07]	Architectural Modeling [CC08]	Hybrid Object Model [RBMB09]	Volumetric Integration [FCSS09b]	Manhattan World Stereo [FCSS09a]	Hierarchical Modeling [AP10]
					
Plane Detection for SLAM [XZZ*11]	Facade Partitioning [SHFH11]	GlobFit [LWC*11]	Cluster Normal Space [HHRB11]	Algebraic Templates [CLL11]	Plane Filtering [BV11]

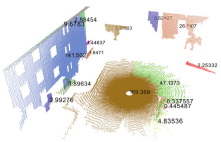
(continued on next page)



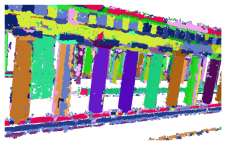
CAD Model Recovery [BSG*11]



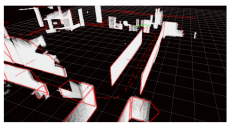
Quadric Surface Fitting [YWLY12]



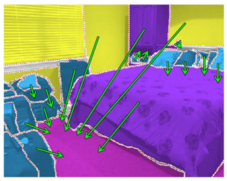
Outdoor Planar SLAM [XAZ12]



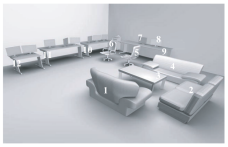
Contracting Segments [WPM*12]



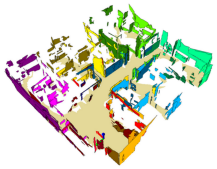
Planar Surface SLAM [TRIC12]



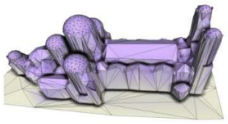
Object Support [SHKF12]



Interactive Semantic Modeling [SXZ*12]

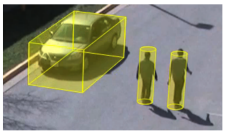


Indoor Plane Mapping [LLL*12]

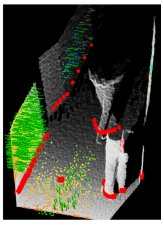


Hybrid City Representation [LM12]

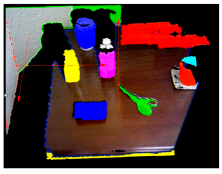
Objects On Table [GMLB12]



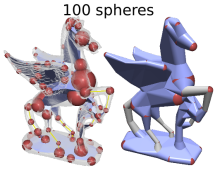
Monocular Occupancy Maps [CSM12]



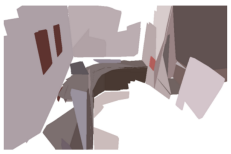
Indoor Robot Navigation [BV12]



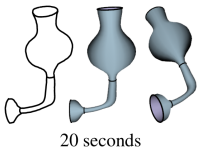
2D-structured Point Cloud [TGRC13]



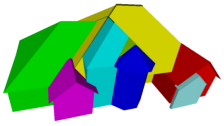
Sphere Meshes [TGB13]



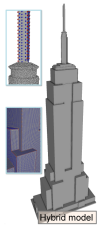
Point-plane SLAM [TJRF13]



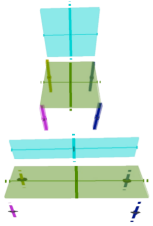
Geosemantic Snapping [SAG*13]



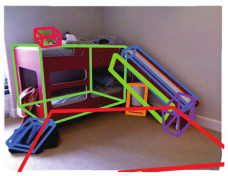
Semantic Learning [LGZ*13]



Point Set Structuring [LA13]



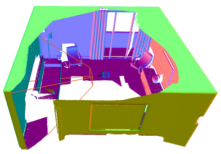
Learning Object Templates [KLM*13]



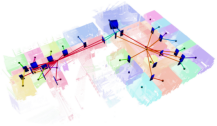
Fitting Cuboids [JX13]

3-Sweep [CZS*13]

O-Snap [ASF*13]

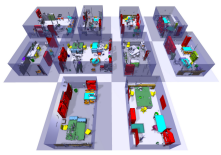


Dense Planar SLAM [SMGKD14]

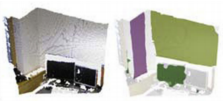


Hierarchical Building Descriptions [OVWK14]

(continued on next page)



Cluttered Indoor Scans [MPM*14]



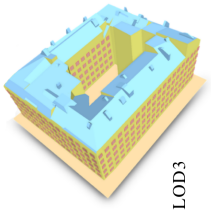
Planar Hough Transform [HSSM14]

Agglomerative Clustering [FTK14]

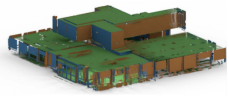
Semantic Modeling [CLW*14]

Generalized Cylinder [ZYH*15]

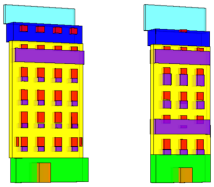
Labeled KinectFusion [ZXTZ15]



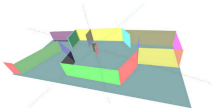
Level of Detail [VLA15]



RAPter [MMBM15]

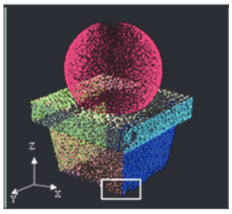


3D All The Way [MKRVG15]



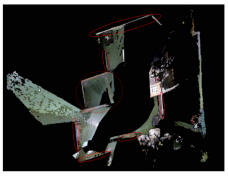
3D Kernel Hough Transform [LO15]

Boxes around Objects [KHB*15]

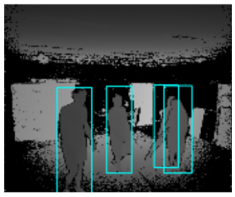


Bayes SAmple Consensus [KL15]

Quaternion Representation [Kae15]

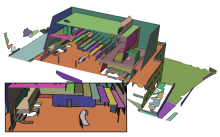


Planar RGB-D SLAM [ERAB15]

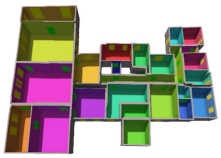


Occupancy Maps [BFF15]

Parallel RANSAC [AEH15]



Regular Planar Modeling [OLA16]



Walls Layout [OVWK16]

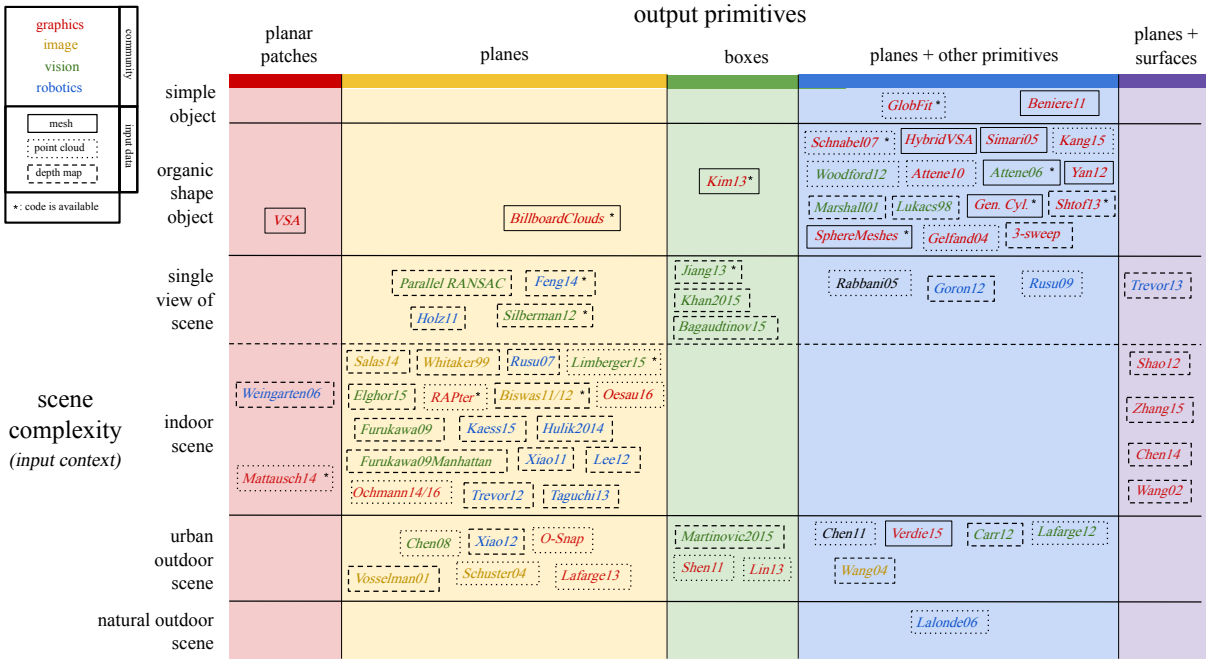


Figure 10: ***Plot of input context / output primitive type for all methods*** *(see Section 2).*

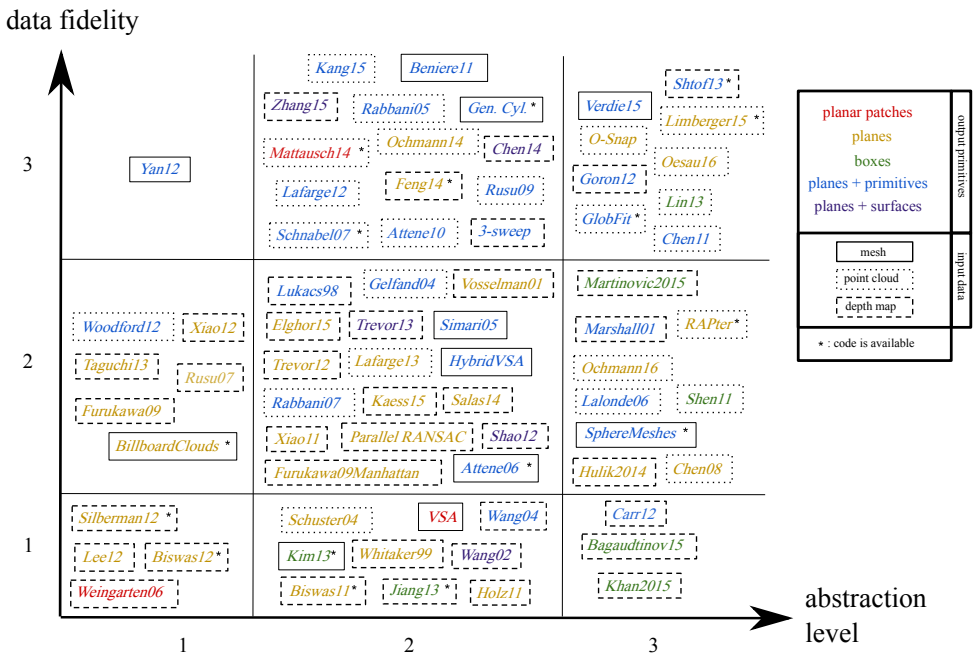


Figure 11: ***Plot of data fidelity values / abstraction levels for all methods*** *(see Section 4.2)*

References

[AB73] AGIN G. J., BINFORD T. O.: Computer description of curved objects. In *Proceedings of the 3rd international joint conference on Artificial intelligence* (1973), Morgan Kaufmann Publishers Inc., pp. 629–640. 2

[AB99] AMENTA N., BERN M.: Surface reconstruction by voronoi filtering. *Discrete & Computational Geometry 22*, 4 (1999), 481–504. 4

[ABCO*03] ALEXA M., BEHR J., COHEN-OR D., FLEISHMAN S., LEVIN D., SILVA C. T.: Computing and rendering point set surfaces. *Visualization and Computer Graphics, IEEE Transactions on 9*, 1 (March 2003), 3–15. 4

[ACK01] AMENTA N., CHOI S., KOLLURI R. K.: The power crust. *Proceedings of the sixth ACM symposium on Solid modeling and applications* (June 2001), 249–266. 4

[AEH15] ALEHDAGHI M., ESFAHANI M. A., HARATI A.: Parallel ransac: Speeding up plane extraction in rgbd image sequences using gpu. *Computer and Knowledge Engineering (ICCKE)* (October 2015), 295–300. 9, 13, 15, 19, 25

[AFS06] ATTENE M., FALCIDIENO B., SPAGNUOLO M.: Hierarchical mesh segmentation based on fitting primitives. *The Visual Computer 22*, 3 (March 2006), 181–193. 9, 17, 19, 20, 22, 23

[And79] ANDREW A. M.: Another efficient algorithm for convex hulls in two dimensions. *Information Processing Letters 9*, 5 (1979), 216–219. 5

[AP10] ATTENE M., PATANÈ G.: Hierarchical structure recovery of point-sampled surfaces. *Computer Graphics Forum 29*, 6 (September 2010), 1905–1920. 5, 9, 15, 17, 18, 19, 22, 23

[ASF*13] ARIKAN M., SCHWÄRZLER M., FLÖRY S., WIMMER M., MAIERHOFER S.: O-snap: Optimization-based snapping for modeling architecture. *ACM SIGGRAPH 32*, 6 (November 2013), 6:1–6:15. 2, 6, 11, 12, 13, 14, 15, 16, 19, 24

[Avr76] AVRIEL M.: Nonlinear programming: analysis and methods. *Prentice-Hall series in automatic computation* (1976). 11

[Bal81] BALLARD D. H.: Generalizing the hough transform to detect arbitrary shapes. *Pattern recognition 13*, 2 (1981), 111–122. 7

[BFF15] BAGAUTDINOV T., FLEURET F., FUA P.: Probability occupancy maps for occluded depth images. *Computer Vision and Pattern Recognition* (June 2015). 6, 13, 15, 16, 19, 25

[BGZ16] BUSÉ L., GALLIGO A., ZHANG J.: Extraction of cylinders and cones from minimal point sets. *Graphical Models 86* (2016), 1–12. 6

[BL79] BEUCHER S., LANTUEJOUL C.: Use of watersheds in contour detection. *International Workshop on Image Processing: Real-time Edge and Motion Detection/Estimation, Rennes, France.* (September 1979). 11

[BSG*11] BÉNIÈRE R., SUBSOL G., GESQUIÈRE G., LE BRETON F., PUECH W.: Recovering primitives in 3D cad meshes. *IS&T/SPIE Electronic Imaging* (2011), 78640R–78640R. 9, 17, 19, 24

[BTS*14] BERGER M., TAGLIASACCHI A., SEVERSKY L., ALLIEZ P., LEVINE J., SHARF A., SILVA C.: State of the art in surface reconstruction from point clouds. *EUROGRAPHICS star reports* (April 2014), 161–185. 3, 4

[BV11] BISWAS J., VELOSO M.: Fast sampling plane filtering, polygon construction and merging from depth images. *Robotics: Science and Systems Conference (RSS)* (June 2011). 5, 19, 20, 23

[BV12] BISWAS J., VELOSO M.: Depth camera based indoor mobile robot localization and navigation. *Robotics and Automation (ICRA)* (May 2012), 1697–1702. 16, 19, 20, 24

[CC08] CHEN J., CHEN B.: Architectural modeling from sparsely scanned range data. *International Journal of Computer Vision 78*, 2-3 (July 2008), 223–236. 8, 13, 16, 19, 23

[CGF09] CHEN X., GOLOVINSKIY A., FUNKHOUSER T.: A benchmark for 3D mesh segmentation. *ACM Transactions on Graphics (Proc. SIGGRAPH) 28*, 3 (August 2009). 3

[CLL11] CHEN J.-Y., LAI H.-J., LIN C.-H.: Point cloud modeling using algebraic template. *International Journal of Innovative Computing, Information and Control 7*, 4 (April 2011), 1521–1532. 11, 13, 14, 17, 19, 23

[CLW*14] CHEN K., LAI Y.-K., WU Y.-X., MARTIN R., HU S.-M.: Automatic semantic modeling of indoor scenes from low-quality rgb-d data using contextual information. *ACM Transactions on Graphics 33*, 6 (November 2014), 208:1–208:12. 13, 14, 15, 16, 19, 20, 25

[CM02] COMANICIU D., MEER P.: Mean shift: A robust approach toward feature space analysis. *Pattern Analysis and Machine Intelligence, IEEE Transactions on 24*, 5 (May 2002), 603–619. 10

[CM05] CHUM O., MATAS J.: Matching with prosac-progressive sample consensus. *Computer Vision and Pattern Recognition* (June 2005), 220–226. 6

[CRS98] CIGNONI P., ROCCHINI C., SCOPIGNO R.: Metro: Measuring error on simplified surfaces. *Computer Graphics Forum 17*, 2 (August 1998), 167–174. 18

[CSAD04] COHEN-STEINER D., ALLIEZ P., DESBRUN M.: Variational shape approximation. *ACM Transactions on Graphics (TOG) 23*, 3 (August 2004), 905–914. 2, 10, 12, 14, 16, 17, 19, 23

[CSM12] CARR P., SHEIKH Y., MATTHEWS I.: Monocular object detection using 3d geometric primitives. *ECCV* (October 2012), 864–878. 5, 6, 13, 16, 19, 24

[CZS*13] CHEN T., ZHU Z., SHAMIR A., HU S.-M., COHEN-OR D.: 3-sweep: Extracting editable objects from a single photo. *ACM Transactions on Graphics (TOG) 32*, 6 (November 2013), 195. 11, 13, 15, 17, 19, 21, 24

[DDSD03] DÉCORET X., DURAND F., SILLION F. X., DORSEY J.: Billboard clouds for extreme model simplification. *ACM Transactions on Graphics (TOG) 22*, 3 (August 2003), 689–696. 2, 7, 14, 16, 19, 20, 23

[DH72] DUDA R. O., HART P. E.: Use of the hough transformation to detect lines and curves in pictures. *Communications of the ACM 15*, 1 (January 1972), 11–15. 7

[DMPT01] DEVILLERS O., MOURRAIN B., PREPARATA F., TREBUCHET P.: On circular cylinders by four or five points in space. *INRIA* (2001). 6

[Ebr15] EBRAHIM M.: 3d laser scanners' techniques overview. *International Journal of Science and Research (IJSR) 4* (10 2015), 5–611. 1

[ERAB15] ELGHOR H. E., ROUSSEL D., ABABSA F., BOUYAKHF E. H.: Planes detection for robust localization and mapping in rgb-d slam systems. *3DV* (October 2015), 452–459. 16, 19, 25

[FB81] FISCHLER M. A., BOLLES R. C.: Random sample consensus: a paradigm for model fitting with applications to image analysis and automated cartography. *Communications of the ACM 24*, 6 (June 1981), 381–395. 6

[FCSS09a] FURUKAWA Y., CURLESS B., SEITZ S. M., SZELISKI R.: Manhattan-world stereo. *CVPR* (June 2009), 1422–1429. 8, 16, 19, 23

[FCSS09b] FURUKAWA Y., CURLESS B., SEITZ S. M., SZELISKI R.: Reconstructing building interiors from images. *ICCV* (September 2009), 80–87. 3, 14, 15, 19, 23

[FH75] FUKUNAGA K., HOSTETLER L. D.: The estimation of the gradient of a density function, with applications in pattern recognition. *Information Theory, IEEE Transactions on 21*, 1 (January 1975), 32–40. 10

[FH83] FAUGERAS O. D., HEBERT M.: A 3-d recognition and positioning algorithm using geometrical matching between primitive surfaces. In *Proceedings of the Eighth international joint conference on Artificial intelligence-Volume 2* (1983), Morgan Kaufmann Publishers Inc., pp. 996–1002. 2

[FO08] FERNANDES L. A., OLIVEIRA M. M.: Real-time line detection through an improved hough transform voting scheme. *Pattern recognition 41*, 1 (2008), 299–314. 7

[Fol96] FOLEY J.: 12.7 Constructive Solid Geometry. *Computer Graphics: Principles and Practice, Addison-Wesley systems programming series* (1996), 557–558. 4

[FS96] FAROUKI R. A., SVERRISSON R.: Approximation of rolling-ball blends for free-form parametric surfaces. *Computer-Aided Design 28*, 11 (1996), 871–878. 4

[FTK14] FENG C., TAGUCHI Y., KAMAT V. R.: Fast plane extraction in organized point clouds using agglomerative hierarchical clustering. *ICRA* (June 2014), 6218–6225. 5, 9, 15, 19, 20, 25

[GG04] GELFAND N., GUIBAS L. J.: Shape segmentation using local slippage analysis. *Eurographics, symposium on Geometry processing* (July 2004), 214–223. 9, 17, 19, 23

[GH97] GARLAND M., HECKBERT P. S.: Surface simplification using quadric error metrics. *Proceedings of the 24th annual conference on Computer graphics and interactive techniques* (August 1997), 209–216. 18, 20

[GMLB12] GORON L. C., MARTON Z.-C., LAZEA G., BEETZ M.: Robustly segmenting cylindrical and box-like objects in cluttered scenes using depth cameras. *Proceedings of ROBOTIK 2012* (May 2012), 1–6. 2, 11, 14, 17, 19, 24

[GVL96] GOLUB G., VAN LOAN C.: Matrix computations. *Johns Hopkins Studies in the Mathematical Sciences* (1996). 11

[HDD*92] HOPPE H., DEROSE T., DUCHAMP T., MCDONALD J., STUETZLE W.: Surface reconstruction from unorganized points. *Computer Graphics and Applications 26*, 2 (March 1992). 4

[HHNM80] HAKALA D., HILLYARD R., NOURSE B., MALRAISON P.: Natural quadrics in mechanical design. *Autofact West 1* (1980), 363–378. 4

[HHRB11] HOLZ D., HOLZER S., RUSU R. B., BEHNKE S.: Real-time plane segmentation using rgb-d cameras. *RoboCup 2011* (July 2011), 306–317. 5, 8, 13, 16, 19, 23

[Hou62] HOUGH P. V. C.: Method and means for recognizing complex patterns. *US Patent 3,069,654* (December 1962). 7

[HP82] HEBERT M., PONCE J.: A new method for segmenting 3-d scenes into primitives. In *Proceedings of the 6th International Conference on Pattern Recognition* (Munich, West Germany, October 1982), pp. 836– 838. 2

[HSSM14] HULIK R., SPANEL M., SMRZ P., MATERNA Z.: Continuous plane detection in point-cloud data based on 3d hough transform. *Journal of visual communication and image representation 25*, 1 (January 2014), 86–97. 7, 13, 16, 19, 25

[JH99] JOHNSON A. E., HEBERT M.: Using spin images for efficient object recognition in cluttered 3d scenes. *IEEE Transactions on pattern analysis and machine intelligence 21*, 5 (May 1999), 433–449. 13

[JX13] JIANG H., XIAO J.: A linear approach to matching cuboids in rgbd images. *Computer Vision and Pattern Recognition (CVPR)* (June 2013), 2171–2178. 11, 13, 14, 15, 16, 19, 20, 24

[Kae15] KAESS M.: Simultaneous localization and mapping with infinite planes. *ICRA* (May 2015). 16, 19, 25

[KBH06] KAZHDAN M., BOLITHO M., HOPPE H.: Poisson surface reconstruction. *Proceedings of the fourth Eurographics symposium on Geometry processing 7* (June 2006). 4

[KEB91] KIRYATI N., ELDAR Y., BRUCKSTEIN A. M.: A probabilistic hough transform. *Pattern recognition 24*, 4 (1991), 303–316. 7

[KHB*15] KHAN S. H., HE X., BENNAMOUN M., SOHEL F., TOGNERI R.: Separating objects and clutter in indoor scenes. *Computer Vision and Pattern Recognition* (June 2015). 9, 11, 13, 15, 16, 19, 25

[KL15] KANG Z., LI Z.: Primitive fitting based on the efficient multibaysac algorithm. *PloS one 10*, 3 (2015), e0117341. 6, 17, 19, 25

[KLM*13] KIM V. G., LI W., MITRA N. J., CHAUDHURI S., DIVERDI S., FUNKHOUSER T.: Learning part-based templates from large collections of 3d shapes. *Transactions on Graphics (Proc. of SIGGRAPH) 32* (November 2013). 10, 11, 13, 14, 15, 17, 19, 20, 24

[KSH12] KRIZHEVSKY A., SUTSKEVER I., HINTON G. E.: Imagenet classification with deep convolutional neural networks. In *Advances in Neural Information Processing Systems 25*. 2012, pp. 1097–1105. 22

[LA13] LAFARGE F., ALLIEZ P.: Surface reconstruction through point set structuring. *EUROGRAPHICS 32*, 2pt2 (May 2013), 225–234. 6, 14, 16, 19, 24

[LGZ*13] LIN H., GAO J., ZHOU Y., LU G., YE M., ZHANG C., LIU L., YANG R.: Semantic decomposition and reconstruction of residential scenes from lidar data. *ACM Transactions on Graphics, (Proc. of SIGGRAPH) 32*, 4 (November 2013). 9, 11, 13, 14, 15, 16, 19, 24

[LLL*12] LEE T.-K., LIM S., LEE S., AN S., OH S.-Y.: Indoor mapping using planes extracted from noisy rgb-d sensors. *Intelligent Robots and Systems (IROS)* (October 2012), 1727–1733. 9, 15, 19, 24

[LLLM86] LI H., LAVIN M. A., LE MASTER R. J.: Fast hough transform: A hierarchical approach. *Computer Vision, Graphics, and Image Processing 36*, 2-3 (1986), 139–161. 7

[Llo82] LLOYD S. P.: Least squares quantization in pcm. *Information Theory, IEEE Transactions on 28*, 2 (March 1982), 129–137. 9, 16

[LM12] LAFARGE F., MALLET C.: Creating large-scale city models from 3d-point clouds: a robust approach with hybrid representation. *International journal of computer vision 99*, 1 (August 2012), 69–85. 5, 10, 14, 17, 19, 24

[LMM98] LUKÁCS G., MARTIN R., MARSHALL D.: Faithful least-squares fitting of spheres, cylinders, cones and tori for reliable segmentation. *ECCV* (June 1998), 671–686. 2, 9, 11, 15, 17, 19, 23

[LO15] LIMBERGER F. A., OLIVEIRA M. M.: Real-time detection of planar regions in unorganized point clouds. *Pattern Recognition 48*, 6 (2015), 2043–2053. 7, 14, 16, 19, 20, 25

[Lon98] LONCARIC S.: A survey of shape analysis techniques. *Pattern recognition 31*, 8 (August 1998), 983–1001. 3

[LPRM02] LÉVY B., PETITJEAN S., RAY N., MAILLOT J.: Least squares conformal maps for automatic texture atlas generation. *ACM Transactions on Graphics (TOG) 21*, 3 (July 2002), 362–371. 2

[LVHH06] LALONDE J.-F., VANDAPEL N., HUBER D., HEBERT M.: Natural terrain classification using three-dimensional ladar data for ground robot mobility. *Journal of Field Robotics 23*, 10 (November 2006), 839 – 861. 10, 12, 13, 14, 15, 17, 18, 19, 23

[LWC*11] LI Y., WU X., CHRYSANTHOU Y., SHARF A., COHEN-OR D., MITRA N. J.: Globfit: Consistently fitting primitives by discovering global relations. *ACM Transactions on Graphics 30*, 4 (July 2011), 52:1–52:12. 2, 6, 11, 12, 13, 17, 19, 20, 21, 23

[Mac67] MACQUEEN J.: Some methods for classification and analysis of multivariate observations. *Proceedings of the fifth Berkeley symposium on mathematical statistics and probability 1*, 14 (1967), 281–297. 9

[MC04] MATAS J., CHUM O.: Randomized ransac with t d, d test. *Image and vision computing 22*, 10 (September 2004), 837–842. 6

[MKRVG15] MARTINOVIC A., KNOPP J., RIEMENSCHNEIDER H., VAN GOOL L.: 3d all the way: Semantic segmentation of urban scenes from start to end in 3d. *Computer Vision and Pattern Recognition* (June 2015). 10, 13, 14, 15, 17, 19, 25

[MLM01] MARSHALL D., LUKACS G., MARTIN R.: Robust segmentation of primitives from range data in the presence of geometric degeneracy. *PAMI 23*, 3 (March 2001), 304–314. 11, 19, 23

[MMBM15] MONSZPART A., MELLADO N., BROSTOW G., MITRA N.: RAPter: Rebuilding man-made scenes with regular arrangements of planes. *ACM SIGGRAPH* (August 2015). 9, 11, 12, 13, 14, 15, 19, 20, 21, 25

[MPM*14] MATTAUSCH O., PANOZZO D., MURA C., SORKINE-HORNUNG O., PAJAROLA R.: Object detection and classification from large-scale cluttered indoor scans. *Computer Graphics Forum 33*, 2 (July 2014), 11–21. 9, 13, 15, 19, 20, 25

[OBA*03] OHTAKE Y., BELYAEV A., ALEXA M., TURK G., SEIDEL H.-P.: Multi-level partition of unity implicits. *ACM SIGGRAPH 2003* (July 2003), 173. 4

[OLA16] OESAU S., LAFARGE F., ALLIEZ P.: Planar shape detection and regularization in tandem. *Computer Graphics Forum 35*, 1 (2016), 203–215. 9, 14, 15, 19, 22, 25

[OVWK14] OCHMANN S., VOCK R., WESSEL R., KLEIN R.: Towards the extraction of hierarchical building descriptions from 3d indoor scans. *EUROGRAPHICS Workshop on 3D Object Retrieval* (April 2014). 11, 13, 14, 15, 16, 19, 21, 24

[OVWK16] OCHMANN S., VOCK R., WESSEL R., KLEIN R.: Automatic reconstruction of parametric building models from indoor point clouds. *Computers & Graphics 54* (February 2016), 94–103. 12, 13, 14, 15, 16, 19, 25

[PBAC75] POPPLESTONE R. J., BROWN C. M., AMBLER A. P., CRAWFORD G. F.: Forming models of plane-and-cylinder faceted bodies from light stripes. In *Proceedings of the 4th International Joint Conference on Artificial Intelligence - Volume 1* (1975), Morgan Kaufmann Publishers Inc., pp. 664–668. 2

[RBM*07] RUSU R. B., BLODOW N., MARTON Z., SOOS A., BEETZ M.: Towards 3d object maps for autonomous household robots. *Intelligent Robots and Systems (IROS)* (October 2007), 3191–3198. 15, 16, 19, 21, 23

[RBMB09] RUSU R. B., BLODOW N., MARTON Z. C., BEETZ M.: Close-range scene segmentation and reconstruction of 3d point cloud maps for mobile manipulation in domestic environments. *Intelligent Robots and Systems (IROS)* (October 2009), 1–6. 11, 17, 19, 23

[RDvdHV07] RABBANI T., DIJKMAN S., VAN DEN HEUVEL F., VOSSELMAN G.: An integrated approach for modelling and global registration of point clouds. *ISPRS journal of Photogrammetry and Remote Sensing 61*, 6 (February 2007), 355–370. 5, 17, 18, 19, 23

[Req80] REQUICHA A. G.: Representations for rigid solids: Theory, methods, and systems. *ACM Comput. Surv. 12*, 4 (December 1980), 437–464. 2

[RL05] ROUSSEEUW P., LEROY A.: Robust regression and outlier detection. *Wiley Series in Probability and Statistics* (2005). 15

[Rou84] ROUSSEEUW P.: Least median of squares regression. *Journal of the American Statistical Association 79*, 388 (January 1984), 871–880. 6

[RVDH05] RABBANI T., VAN DEN HEUVEL F.: Efficient hough transform for automatic detection of cylinders in point clouds. *ISPRS journal of Photogrammetry and Remote Sensing 3* (September 2005), 60–65. 8, 17, 19, 23

[RWS*06] REN Z., WANG R., SNYDER J., ZHOU K., LIU X., SUN B., SLOAN P.-P., BAO H., PENG Q., GUO B.: Real-time soft shadows in dynamic scenes using spherical harmonic exponentiation. *ACM Trans. Graph. 25*, 3 (2006). 3

[SAG*13] SHTOF A., AGATHOS A., GINGOLD Y., SHAMIR A., COHEN-OR D.: Geosemantic snapping for sketch-based modeling. *Computer Graphics Forum (Proc. EUROGRAPHICS) 32*, 2pt2 (May 2013), 245–253. 11, 13, 18, 19, 20, 21, 24

[Sch04] SCHUSTER H.-F.: Segmentation of lidar data using the tensor voting framework. *ISPRS 35*, B3 (July 2004), 1073–1078. 9, 16, 19, 23

[Sha08] SHAMIR A.: A survey on mesh segmentation techniques. *Computer Graphics Forum 27*, 6 (2008), 1539–1556. 11

[SHFH11] SHEN C.-H., HUANG S.-S., FU H., HU S.-M.: Adaptive partitioning of urban facades. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH ASIA 2011) 30*, 6 (December 2011), 184:1–184:9. 6, 11, 13, 17, 19, 23

[Shi72] SHIRAI Y.: Recognition of polyhedrons with a range finder. *Pattern Recognition 4*, 3 (1972), 243IN1245–244250. 2

[SHKF12] SILBERMAN N., HOIEM D., KOHLI P., FERGUS R.: Indoor segmentation and support inference from rgbd images. *ECCV* (October 2012). 12, 16, 19, 20, 21, 24

[SMGKD14] SALAS-MORENO R. F., GLOCKEN B., KELLY P. H., DAVISON A. J.: Dense planar slam. *ISMAR* (September 2014). 9, 15, 19, 24

[SS05] SIMARI P. D., SINGH K.: Extraction and remeshing of ellipsoidal representations from mesh data. *Proceedings of Graphics Interface 2005* (May 2005), 161–168. 18, 19, 23

[Ste91] STEPHENS R. S.: Probabilistic approach to the hough transform. *Image and vision computing 9*, 1 (1991), 66–71. 7

[SWK07] SCHNABEL R., WAHL R., KLEIN R.: Efficient ransac for point-cloud shape detection. *Computer Graphics Forum 26*, 2 (June 2007), 214–226. 2, 5, 6, 13, 14, 15, 17, 18, 19, 20, 21, 23

[SWWK08] SCHNABEL R., WESSEL R., WAHL R., KLEIN R.: Shape recognition in 3d point-clouds. *The 16-th International Conference in Central Europe on Computer Graphics, Visualization and Computer Vision 8* (2008). 6, 21

[SXZ*12] SHAO T., XU W., ZHOU K., WANG J., LI D., GUO B.: An interactive approach to semantic modeling of indoor scenes with an rgbd camera. *ACM Transactions on Graphics (TOG) 31*, 6 (November 2012), 136. 9, 11, 13, 14, 15, 19, 20, 24

[TGB13] THIERY J.-M., GUY E., BOUBEKEUR T.: Sphere-meshes: Shape approximation using spherical quadric error metrics. *ACM Transactions on Graphics, (Proc. of SIGGRAPH Asia) 32*, 6 (November 2013). 2, 9, 12, 13, 14, 17, 19, 20, 22, 24

[TGBE16] THIERY J.-M., GUY E., BOUBEKEUR T., EISEMANN E.: Animated mesh approximation with sphere-meshes. *ACM Trans. Graph. 35*, 3 (2016), 30:1–30:13. 3, 20

[TGRC13] TREVOR A. J., GEDIKLI S., RUSU R. B., CHRISTENSEN H. I.: Efficient organized point cloud segmentation with connected components. *Semantic Perception Mapping and Exploration (SPME)* (May 2013). 9, 13, 15, 19, 24

[TJRF13] TAGUCHI Y., JIAN Y.-D., RAMALINGAM S., FENG C.: Point-plane slam for hand-held 3d sensors. *Robotics and Automation (ICRA)* (May 2013), 5182–5189. 2, 16, 19, 24

[TPT16] TKACH A., PAULY M., TAGLIASACCHI A.: Sphere-meshes for real-time hand modeling and tracking. *ACM Trans. Graph. 35*, 6 (2016), 222:1–222:11. 3, 20

[TRIC12] TREVOR A. J., ROGERS III J. G., CHRISTENSEN H. I.: Planar surface slam with 3d and 2d sensors. *ICRA* (May 2012), 3041–3048. 16, 19, 24

[TZ00] TORR P. H., ZISSERMAN A.: Mlesac: A new robust estimator with application to estimating image geometry. *Computer Vision and Image Understanding 78*, 1 (April 2000), 138–156. 6

[VD01] VOSSELMAN G., DIJKMAN S.: 3d building model reconstruction from point clouds and ground plans. *International archives of photogrammetry remote sensing and spatial information sciences 34.3*, W4 (2001), 37–44. 16, 19, 23

[VLA15] VERDIE Y., LAFARGE F., ALLIEZ P.: Lod generation for urban scenes. *ACM Transactions On Graphics (TOG)* (2015). 10, 14, 15, 18, 19, 25

[WGC99] WHITAKER R. T., GREGOR J., CHEN P.: Indoor scene reconstruction from sets of noisy range image. *3-D Digital Imaging and Modeling (3DIM)* (October 1999), 348–357. 11, 13, 16, 19, 23

[WK05] WU J., KOBBELT L.: Structure recovery via hybrid variational surface approximation. *Computer Graphics Forum 24*, 3 (September 2005), 277–284. 2, 12, 14, 17, 19, 23

[WO02] WANG J., OLIVEIRA M. M.: Improved scene reconstruction from range images. *Computer Graphics Forum 21*, 3 (September 2002), 521–530. 7, 13, 15, 16, 19, 23

[WPM*12] WOODFORD O. J., PHAM M.-T., MAKI A., GHERARDI R., PERBET F., STENGER B.: Contraction moves for geometric model fitting. *ECCV* (October 2012), 181–194. 1, 2, 10, 14, 15, 19, 24

[WPM*14] WOODFORD O. J., PHAM M.-T., MAKI A., PERBET F., STENGER B.: Demisting the hough transform for 3d shape recognition and registration. *International Journal of Computer Vision 106*, 3 (February 2014), 332–341. 7

[WS06] WEINGARTEN J., SIEGWART R.: 3d slam using planar segments. *Intelligent Robots and Systems* (October 2006), 3062–3067. 16, 19, 23

[WSK*15] WU Z., SONG S., KHOSLA A., YU F., ZHANG L., TANG X., XIAO J.: 3d shapenets: A deep representation for volumetric shapes. In *2015 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)* (2015), pp. 1912–1920. 22

[WT04] WANG S., TSENG Y.: Semi-automated csg model-based building extraction from photogrammetric images. *XXth Congress of the International Society for Photogrammetry and Remote Sensing 24*, 3 (2004), 277–284. 17, 19, 23

[XAZ12] XIAO J., ADLER B., ZHANG H.: 3d point cloud registration based on planar surfaces. *Multisensor Fusion and Integration for Intelligent Systems (MFI)* (September 2012), 40–45. 12, 15, 19, 24

[XKH*16] XU K., KIM V. G., HUANG Q., MITRA N., KALOGERAKIS E.: Data-driven shape analysis and processing. *SIGGRAPH ASIA 2016 Courses*, 4 (2016). 11

[XOK90] XU L., OJA E., KULTANEN P.: A new curve detection method: randomized hough transform (rht). *Pattern recognition letters 11*, 5 (1990), 331–338. 7

[XOT13] XIAO J., OWENS A., TORRALBA A.: Sun3d: A database of big spaces reconstructed using sfm and object labels. *International Conference on Computer Vision (ICCV)* (December 2013), 1625–1632. 20

[XZZ*11] XIAO J., ZHANG J., ZHANG J., ZHANG H., HILDRE H. P.: Fast plane detection for slam from noisy range images in both structured and unstructured environments. *Mechatronics and Automation (ICMA)* (August 2011), 1768–1773. 9, 15, 19, 20, 23

[YWLY12] YAN D.-M., WANG W., LIU Y., YANG Z.: Variational mesh segmentation via quadric surface fitting. *Computer-Aided Design 44*, 11 (November 2012), 1072–1082. 5, 10, 13, 17, 19, 20, 24

[ZXTZ15] ZHANG Y., XU W., TONG Y., ZHOU K.: Online structure analysis for real-time indoor scene reconstruction. *ACM Transactions on Graphics (TOG) 34*, 5 (November 2015), 159. 9, 13, 15, 19, 25

[ZYH*15] ZHOU Y., YIN K., HUANG H., ZHANG H., GONG M., COHEN-OR D.: Generalized cylinder decomposition. *ACM Transactions on Graphics (TOG) 34*, 6 (November 2015), 171. 9, 12, 13, 17, 19, 20, 21, 25