Local Hough Transform for 3D Primitive Detection

Bertram Drost
MVTec Software GmbH
Munich, Germany
drost@mvtec.com

Slobodan Ilic
Siemens AG
Munich, Germany
slobodan.ilic@siemens.com

Abstract

Detecting primitive geometric shapes, such as cylinders, planes and spheres, in 3D point clouds is an important building block for many high-level vision tasks. One approach for this detection is the Hough Transform, where features vote for parameters that explain them. However, as the voting space grows exponentially with the number of parameters, a full voting scheme quickly becomes impractical. Solutions in the literature, such as decomposing the global voting space, often degrade the robustness w.r.t. noise, clutter or multiple primitive instances.

We instead propose a local Hough Transform, which votes on sub-manifolds of the original parameter space. For this, only those parameters are considered that align a given scene point with the primitive. The voting then recovers the locally best fitting primitive. This local detection scheme is embedded in a coarse-to-fine detection pipeline, which refines the found candidates and removes duplicates.

The evaluation shows high robustness against clutter and noise, as well as competitive results w.r.t. prior art.

1. Introduction

The segmentation of geometric primitives plays an important role in scene understanding, robotics, reverse engineering and other applications. In general, it helps to build a high-level description of the scene by decomposing a potentially large set of 3D points into a small set of primitives, which can be processed on a higher, more abstract level. We propose a novel approach for detecting and segmenting certain 3D primitives – namely planes, spheres and cylinders – from 3D point clouds.

Most approaches for primitive segmentation are based on oversegmentation with region growing, RANSAC or the Hough Transform. While the Hough Transform has promising properties, such as deterministic runtime and robustness against noise, it has several drawbacks when applying it to primitives in 3D. First, the size of the parameter or voting space grows exponentially with the number of parameters. For example, cylinders have four parameters for their position and one for the radius. Large voting spaces, however, require more memory, are slower to process as usually more votes are cast, and have an increased sensitivity due to binning. Second, it is difficult to uniformly sample the directional components of primitives such as planes and cylinders, as there is no proper uniform segmentation of the unit sphere in 3D for arbitrary sampling sizes. Attempts to avoid those problems typically lead to problems with large scenes where a lot of clutter is present, or with multiple object instances.

We introduce a voting scheme for geometric primitives that circumvents all of the problems of the classical Hough Transform for high-dimensional spaces. The approach is based on the local voting scheme of Drost *et al*. [3], which is able to detect arbitrary, rigid free-form objects in 3D point clouds. We tailor that detection scheme to geometric primitives by using their intrinsic symmetries. This allows several fundamental optimizations, making the detection faster, more robust and more generic. By using a local parameter space, the hough space is small – at most 2 dimensions – and more robust w.r.t. binning. Also, the directional component is implicit in the scene normals and requires no discretization. Finally, since our approach is local, it is highly robust even with large amounts of clutter and with multiple shape instances.

In the remainder of this paper, we first given an overview over related work. We then introduce the local voting scheme of Drost *et al*. [3] and describe our adaptions of the method to symmetric shapes. We then embed the voting scheme into a detection pipeline that refines the results and removes duplicates. Finally, we present evaluation results over several datasets.

2. Related Work

Many approaches can roughly be classified into region growing, Hough transform and RANSAC-based approaches. Since the literature on this topic is vast, we present only a few selected works.

Region Growing Methods based on region growing start with an (over-)segmentation of the scene – sometimes even up to individual points – and combine spatially neighboring segments if they describe a consistent primitive. Different kinds of segmentation, neighborhood metrics of the segments and models to be fit into the segments exist.

Mrwald *et al*. [9] fit B-Splines of first or second order into adjacent segments, fusing the segments if the fit of the fused segments is better than for the two separate segments. Using B-Splines, their approach is able to segment several types of geometric primitives. Gotardo *et al*. [5] extract edges and curvature information from range images to segment the image, and fit planes into each segment. Kim and Ahn [8] proposed a region growing algorithm that uses curvature analysis of local patches to automatically select the primitive type and parameters. Holz *et al*. [6] proposed a real-time segmentation of range scans using integral images for fast normal estimation, and clustering and segmentation in normal space.

RANSAC RANSAC-based approaches randomly sample a set of points from the scene and compute a primitive that contains all points. The hypothesis is evaluated by checking it against all other scee points. Multiple hypotheses are randomly created, and the best one is taken. Schnabel *et al*. [13] proposed a RANSAC based approach that is highly robust, but requires up to a minute for several million points. Oehler *et al*. [10] proposed a coarse-to-fine RANSAC segmentation of planes from 3D data. Tarsha-Kurdi *et al*. [14] compared several approaches and designed a RANSAC-based approach that integrates domain-specific knowledge of the shape sizes. They concluded that RANSAC is inferior to the hough transform.

Voting Finally, methods based on the Hough Transform recover the parameters of primitives using a voting scheme. The parameter space is discretized into cells, and the detected features vote for parameters that explains them. Several authors proposed Hough transform for detecting planes in bird-eye LIDAR-Scans of urban environments [15, 11]. Rabbani and Van den Heuvel proposed a Hough-based approach for cylinders [12].

Bormann *et al*. [2] gives a comprehensive review of Hough-based approaches for detecting planes in 3D point clouds. They describe the difficulty of finding a good Hough space, which is a compromise between accuracy, runtime, storage space, and robustness. In the particular case of planes and cylinders, one needs a subdivision of the space of normal directions, *i.e*., the the surface of a unit sphere in 3D, that is uniform in the sense that each cell covers the same area on the sphere. Since no such subdivision exists in general, several approximations were proposed. Bormann *et al*. propose a new one, based on an adapted sampling of the polar coordinates. Note that our approach circumvents all of the issues raised by [2].

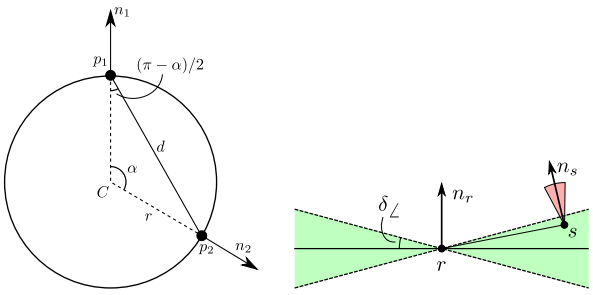


Figure 1: **Left**: Point pair feature for spheres. Given a sphere with center C and radius r, the normals of two points p_1 and p_2 on the sphere form the angle α. **Right**: Illustration of the influence of angular noise on point pairs of a plane. Given a reference point r and its normal n_r, a scene point s is considered to be "on" the plane defined by r and n_r, if $|\measuredangle(s-r, n_r) - \pi/2| < \delta_\angle$, *i.e*., if it is on a plane that is within the tolerance $\delta_\angle$ of the normal vector. The direction of the normal n_s of s must also be within that tolerance.

Others Ahn *et al*. [1] proposed a semi-manual approach for primitive segmentation, where the user select the primitive type and one point of the primitive. The primitive is initialized by fitting it into a neighborhood of the point selected by the user. While robust, this approach is not suitable for fully automatic segmentation.

3. Local Hough Transform

3.1. Point Pairs

The voting scheme of [3] and our proposed detection scheme uses pairs of oriented 3D points as basic features. Each such pair is described using a *point pair feature*

$$F(p_1, p_2) = (|d|, \measuredangle(n_1, d), \measuredangle(n_2, d), \measuredangle(n_1, n_2)) \qquad (1)$$

where $d = p_1 - p_2$.

For the detection, point pairs are extracted from the scene and matched against the model. In [3], the model point pairs were discretized and stored in a hash table to allow this matching in constant time. However, that approach introduces several issues: Since features are discretized, mismatches can happen at the sampling boundaries; an off-line phase is required that builds the hash table; a static hash table does not allow shape parameters, such as a radius, or shapes with a-priori unknown size, such as large planes or long cylinders; and finally, it does not allow an explicit model of the expected noise in point positions and normal directions.

We instead use the implicit nature of primitives to match point pairs and to extract possible shape parameters. In the following, δ_p is the expected approximate noise in the position of the points, and $\delta_\measuredangle$ is the expected approximate angular error of the normal vectors.

Plane Ideally, all point pairs on a plane have parallel normal vectors and an angle of $90°$ between the difference vector of the two points and the normal vectors. The corresponding point pair features are thus

$$F(p1, p2) = (|d|, \pi/2, \pi/2, 0) \tag{2}$$

Since planes are infinitely large, $|d|$ can take any non-negative value.

In practice, both the point coordinates and the directions of the normal vectors will be affected by noise. We ignore the effects of position noise on angles, since the distance between points is typically much larger than the noise on their position. As illustrated in Fig. 1, all point pair features on a plane can then be described as

$$\begin{aligned} F(p1, p2) \in \mathbb{R} \times & [\pi/2 - \delta_\measuredangle, \pi/2 + \delta_\measuredangle] \times \\ & [\pi/2 - \delta_\measuredangle, \pi/2 + \delta_\measuredangle] \times \\ & [0, 2\delta_\measuredangle] \end{aligned} \tag{3}$$

Sphere As visualized in Fig. 1, given two points p_1, p_2 on the surface of a sphere of radius r, the point pair can be described based on the angle α formed by the sphere's center and the two points:

$$F(p_1, p_2) = (|d|, \measuredangle(n_1, d), \measuredangle(n_2, d), \measuredangle(n_1, n_2)) \tag{4}$$

$$= (2r \sin(\alpha/2), (\pi - \alpha)/2, (\pi + \alpha)/2, \alpha) \tag{5}$$

For the estimation of how noise affects the feature, we assume that the three angles of the feature are affected only by $\delta_\measuredangle$, while the distance is affected only by δ_d. We write α and $|d|$ for the measured values from the feature components F_1 and F_4, and $\overline{\alpha}$ and $\overline{|d|}$ for the underlying exact ground truth values. Since both normals might be off by an angle of up to $\delta_\measuredangle$, we deduce that

$$\overline{\alpha} \in [\max(0, \alpha - 2\delta_n), \alpha + 2\delta_n] \tag{6}$$

$$\overline{d} \in [\max(0, |d| - \delta_p), |d| + \delta_p] \tag{7}$$

From this, we find the range of possible radii as

$$r \in \left[\frac{|d| - \delta_p}{2 \sin((\alpha + 2\delta_n)/2)}, \frac{|d| + \delta_p}{2 \sin(\max(0, \alpha - 2\delta_n)/2)} \right] \tag{8}$$

Note that for $\alpha < 2\delta_n$, the upper boundary for r can be infinity. This corresponds to two points with almost parallel normal vectors which could lie on a sphere with an infinitely large radius.

The remaining two values of the feature, the angles between the normal vectors and the difference vector, are not used for aboves computation. However, their value is checked against the possible range of values, and the point pair is discarded as not being on a sphere if one of those two checks fails.

Cylinder While an implicit model for point pairs on a cylinder is possible, such a model is rather complex and it is computationally expensive to extract the voting parameters α and r from a given point pair. Instead, we precreate a model that maps given point pairs to the corresponding voting parameters.

Let r be the cylinder's radius, l the distance of the two points along the cylinder's main axis, α the angle between the two points when projected along the cylinder's axis (similar to Fig. 1) and d the distance of the two points if projected along the cylinder's axis. The feature vector then is

$$\begin{aligned} F(p_1, p_2) = (\sqrt{d^2 + l^2}, & \measuredangle((1, 0, 0)^T, \\ & (r(1 - \cos\alpha), r \sin\alpha, l)^T), \pi - F_2, \alpha) \end{aligned} \tag{9}$$

Note that the three angles of the features do not depend on the radius, while the length component scales linear with the radius. On a cylinder with radius r, two points will form the point pair $F^r(p_1, p_2)$. Scaling the complete system by $1/r$ leads to the feature

$$F^1(p_1, p_2) = (F_1^r/r, F_2^r, F_3^r, F_4^r) \tag{10}$$

We pre-compute the list of possible features, using a cylinder with radius 1 and some maximum length. Due to the symmetries of a cylinder, we can fix a single point p_1 from that cylinder and pair it with all other points p_2. The complexity of the cylinder model generation is then $O(n)$, compared to $O(n^2)$ for a non-symmetric, free-form model. In the online phase, those features are matched with the observed features using only the scale-invariant angular components F_2, F_3 and F_4. For each matching feature, we compute the radius as

$$r = F_1^r / F_1^1 \tag{11}$$

The matching point pair is discarded if the radius is outside the range of allowed radii.

While cylinders can have arbitrary lengths, we must decide for a finite length when sampling the prototype cylinder of radius 1. However, note that for very large ratios d/r (*i.e.*, for points very far away from each other w.r.t. the radius of the cylinder), the feature converges towards

$$F(p_1, p_2) = (|p_1 - p_2|, \pi/2, \pi/2, \alpha) \tag{12}$$

Since we use only the angles for feature lookup, all point pairs for which $|p_1 - p_2|$ exceeds a certain threshold will fall into the same bins. We thus add a set of special bins, for which $F_2 = F_3 = \pi/2$ and $F_4 \in [0, \pi]$. If a scene feature falls into one of those bins, a vote is cast for all allowed radii.

3.2. Local Parameter Space

The size and complexity of the space $SE(3)$ of rigid 3D transformations make it difficult to directly use it for a voting space. Instead, [3] proposed to vote on sub-manifolds of that space. Each manifold is generated by fixing a reference point $r \in S$ in the scene, and by allowing only those transformations $T \in SE(3)$ which align r with a model point and its normal, *i.e.*, which transform the model onto r. Each such manifold is parametrized naturally using the model point $m \in M$ which is aligned with r, and the rotation angle α which rotates the model around the reference point and its normal vector. Since the transformations are defined and parametrized locally w.r.t. r, they are called the *local parameter space*. Fig. 1 summarizes the dimensions of the parameter spaces.

Reducing the space While (m, α) is good for parametrizing the transformation of an arbitrary free-form object, it is an overparameterization for geometric primitives. Planes and spheres with fixed radius are symmetric w.r.t. rotations around normal vectors, and they look identical from each point on their surface. Their local parameter space thus requires *no* parameters. In other words, a single 3D point and its normal fully defines the plane and the fixed-radius sphere it is on. For cylinders with fixed radius, the position of the corresponding point on their surface is irrelevant, however, they are not symmetric w.r.t. rotations around normal vectors. The local parameter space for cylinders with fixed radius is thus one-dimensional and contains only α.

Extending the space Above's analysis is valid for spheres and cylinders with fixed radius. However, in many applications, that radius might vary within some predefined range. We thus *extend* the local parameter space by certain non-rigid shape parameters of the primitives, namely their radius. Note that fundamentally, extensions of the local parameter space by additional non-rigid shape parameters – such as the scale of the object – are possible even for free-form objects. However, since the parameter space grows exponentially with the number of parameters, it quickly becomes too large for practical applications. Additional shape parameters are thus especially suited for symmetric objects, for which the local parameter space is already reduced as described above.

Table 1: Summary of proposed parameter space dimensions.

	Number of parameters		
Shape	**Rigid**	**Local**	**Shape**
Free-form	6	3	0
Plane	3	0	0
Sphere	3	0	1 (radius)
Cylinder	4	1	1 (radius)

3.3. Voting Scheme

The voting scheme performs the Hough Transform on the local parameter space [3]. Given a reference point $r \in S$ from the scene, it iterates over all other points $s \in S$ and computes the point pair feature $F(r, s)$. The primitive's local parameters and (for cylinders and spheres) shape parameters are extracted, and a vote is cast for each possible match.

Since the voting is local, it is performed for multiple reference points r. For this, the reference points are sampled uniformly from the scene, using the smallest expected size of the primitive as sampling distance. This ensures that at least one reference point ends up on the surface of the primitive.

4. Detection Pipeline

The local hough transform only generates a set of candidates for the primitives. Due to noise in the input data and sampling during voting, the candidates will only be approximate and need to be refined. Furthermore, multiple candidates can be generated that represent the same underlying primitive. To counter both effects, the candidates are refined, and a non-maximum suppresion is performed to remove similar candidates.

4.1. Segmentation

For planes and cylinders, which are potentially unbounded, an additional and optional segmentation step can be performed prior to the refinement. The segmentation is responsible for removing far-away parts of the scene which would lie on the primitive, if it was unbounded, but which one would not consider to be on the primitive in practice. The segmentation is required since the voting does not give exact bounding informations about the primitive.

First, all points that are on (or close to) the primitive are detected. The thresholds are determined from the expected noise in the point positions and normal directions, as well as the expected inaccuracies due to the voting. The resulting points are segmented into connected components, where a user-defined threshold is used as segmentation distance. Only the component which contains the reference

point from the voting scheme is then used for the further refinement of the primitive.

4.2. Refinement

The results of the voting scheme will inherently be noisy due to noise in the input data and sampling of the parameter and feature spaces. We employ an iterative re-weighted least squares approach to refine the candidates, *i.e.*, to minimize the weighted sum of squared distances from the primitive to the observed scene data. The target energy we want to minimize is then

$$E(p) = \sum_{s \in S} w_s d_{\text{primitive}}(p, s)^2 \tag{13}$$

where p are the primitive's parameters and $d_{\text{primitive}}(p, s)$ is the distance between scene point s and the surface of the primitive. Following [4] we can re-write this energy as $E(p) = \sum_{s \in S} E_s(p)^2$ and define the vector of residuals $e(p) = \{E_s(p)\}_{s \in S}$ such that $E(p) = |e(p)|^2$. Then,

$$E(p) = e^T e \tag{14}$$

$$\nabla E(p) = 2(\nabla e)^T e \tag{15}$$

gives us the Jacobian matrix $J = \nabla e$. To solve for the update, we use the iterative Gauss-Newton method. Given some initial approximation p_k of the parameters, an update step is defined using the Jacobian as

$$(J^T J) d_k = -J^T e \tag{16}$$

$$p_{k+1} = p_k + d_k \tag{17}$$

The whole optimization is iterated, recomputing the weights w_s inbetween using the Tukey Biweight function as

$$w_c = w(d) = \begin{cases} (1 - d^2/d_{max}^2)^2 & \text{if } d < d_{max} \\ 0 & \text{else} \end{cases} \tag{18}$$

d_{max} is the maximum distance between a scene point and the primitive. It is initialized based on the approximated noise. After each iteration, it is updated by approximating the mean of the distances using the median absolute deviation $\sigma \approx 1.4826\ median(D)$.

Planes are parametrized using a point c on their surface, and their normal n. **Spheres** are parametrized using their center c and their radius r. **Cylinders** require a somewhat more complicated parametrization to avoid discontinuities in the parameter space as well as overparameterizations of the update step. Each cylinder is parametrized as $p = (T, r)$, $T \in SE(3)$, $r > 0$. T is a rigid 3D transformation that maps the cylinder's main axis onto the z-axis. To be numerically more robust, we choose T such that the cylinder encloses the origin. The distance of a point s from the cylinder is then

$$d_{\text{cylinder}}(p, s) = \left(|Ts - Ts \cdot (0, 0, 1)^T| - r\right) \tag{19}$$

Note that (T, r) is an overparametrization of the cylinder. To be numerically more robust, in particular to avoid overparametrization of the update, we parametrize the update δ_p as

$$\delta_p = (\delta_r, \delta_{tx}, \delta_{ty}, \delta_{rx}, \delta_{ry}) \tag{20}$$

where δ_r encodes the change in the cylinder's radius, δ_{tx} and δ_{ty} encode the change in the cylinder's position, orthogonal to its main axis, and δ_{rx} and δ_{ry} encode the change in the cylinder's main axis by tilting it. The tilt of the main axis is performed by interpreting $(\delta_{rx}, \delta_{ry}, 0)^T$ as Rodruigez parameters of the corresponding rotation, which effectively tilts the z-axis. The update δ_T is thus composed of the translation $(\delta_{tx}, \delta_{ty}, 0)^T$ and the described rotation. Given current cylinder parameters $p = (T, r)$ and an update δp, the new parameters $p' = (T', r')$ are computed as

$$r' = r - \delta_r \tag{21}$$

$$T' = \delta_T^{-1} T \tag{22}$$

4.3. Non-Maximum Suppression

The Non-Maximum Suppression (NMS) removes duplicate candidates that are too similar by keeping only the one with the highest score. This step serves two purposes: First, it avoids returning many identical primitives. Since the voting scheme is local to some reference point, multiple reference points that lie on the same target shape can create similar candidates. NMS removes those duplicates. Second, it improves the overall performance of the detection pipeline, since less candidates need to be processed in subsequent steps.

The NMS is performed by first sorting all candidates by their score. The candidates are then processed from best to worst; for each one, all similar candidates that have a lower score are removed. To speed up the search, the primitives are indexed using the direction of the main axis (cylinders) or normal vector (planes), or using the center points (sphere).

4.4. Detection Pipeline

First, a non-maximum suppression is applied to the candidates from the voting step. The candidates are then refined twice in a coarse-to-fine manner. For the first refinement, a subsampled set of all scene points is used, while the second refinement uses all scene points. A sparse cloud is enough to refine the candidate with a good accuracy, but is faster than when using the full point cloud. Since the candidate is more accurate, less iterations of the more expensive

full-cloud refinement are required. After the voting and after each refinement step, the non-maximum suppression is used to remove duplicate candidates.

5. Theoretical Comparison with RANSAC

The proposed method can be seen as a hybrid between a global voting scheme and RANSAC. While RANSAC selects *multiple* random points, enough to fit the target primitive, the proposed method selects only a *single* point, the reference point. The primitive parameters are deduced from that single point using the voting scheme, which is deterministic. By selecting the reference points through uniform sampling of the scene, their selection can be seen as quasi-deterministic. Contrary to RANSAC, the method is thus non-random and has a more deterministic runtime. Additionally, since only a single point on the object must be selected, the method has a lower complexity when the ratio of inlier points in the scene is low.

The expected overall complexity of the method is approximately linear w.r.t. the surface size of the scene. If an upper bound on the size of the primitive is known, the number of processed point per reference point is constant since the scene is sampled uniformly. The number of reference points is linear w.r.t. the surface size of the scene.

6. Evaluation

6.1. Refinement

In a first set of experiments, we evaluated the proposed refinement algorithms for all three primitives. The tests were done on synthetic data with known ground truth to evaluate the accuracy and basin of convergence. Depending on the primitive, we varied the number of close-range clutter points (outliners), the noise in the data, the noise in the initial parameters of the primitive and the amount of visibility of the primitive.

To make the measurements somewhat comparable and independent of the overall scale of the data, we measure distance-based values relative to the size of the primitives. Two relative sizes are used: The maximum diameter of the primitive, and the radius of spheres and cylinders. In particular, the gaussian noise σ which is applied to the scene point positions, the initialization error and the average distance of points on the GT primitive to the refined primitive is given relative to the diameter of the primitive, and the error of radius r (cylinder, sphere) and of the sphere center c is given relative to the radius of the corresponding primitive.

Fig. 2 shows the results. Overall, the refinement of all three primitives is highly robust against close-range clutter and copes well even with larger amounts of noise. The basin of convergence is well within the expected inaccuracy of the voting scheme, making the refinement a good extension of the proposed voting scheme. For spheres and cylinders that are only barely visible (10% visibility or less), the refinement becomes less stable, especially in presence of many outliners or noise. However, those situations represent an ill-posed problem, as the remaining part of the primitives becomes more and more planar, making a robust estimation of the radius difficult.

Table 2: Results on the SegComp ABW Dataset [7] (Results from [10] and [5].)

approach	correct	over	under	missed	noise
USF	12.7 (83.5%)	0.2	0.1	2.1	1.2
WSU	9.7 (63.8%)	0.5	0.2	4.5	2.2
UB	12.8 (84.2%)	0.5	0.1	1.7	2.1
UE	13.4 (88.1%)	0.4	0.2	1.1	0.8
OU	9.8 (64.4%)	0.2	0.4	4.4	3.2
PPU	6.8 (44.7%)	0.1	2.1	3.4	2.0
UA	4.9 (32.2%)	0.3	2.2	3.6	3.2
UFPR	13.0 (85.5%)	0.5	0.1	1.6	1.4
MRPS	11.1 (73.0%)	0.2	0.7	2.2	0.8
ours	12.3 (80.7%)	0.2	0.8	2.6	0.0

6.2. Detection - Quantitative

ABW SecComp Dataset We evaluated the plane detector on the SegComp ABW Dataset [7], a database of 30 scenes taken with a laser scanner, and with known ground truth. Fig. 2 shows the results of our detector, compared to prior art. Note that we have zero noise, indicating that all of our detected planes corresponded to a real GT plane (no false positives). Even though our approach has a slightly lower detection rate than the top methods, we found that the missed planes had a very high inclination w.r.t. to the camera, sometimes consisting of only 5-10 disconnected pixels. The normal estimator failed for points on those planes, leading to them being not detected.

Synthetic Dataset We also performed an evaluation of the primitive detection on some 1000 artifical scenes with known ground truth. We varied the distribution and number of primitives and occlusion. Gaussian noise was fixed to $\sigma = 0.01\%$ of the primitive diameters. Fig. 4 shows the resulting detection rate. All well-behaving primitives, which were not too much occluded or tiltet against the camera, were detected.

6.3. Detection - Qualitative

We finally performed a large number of qualitative experiments on real-world data. Scenes were acquired with different sensors, including Kinect-like structured light sensors (Fig. 5), time-of-flight sensors and stereo setups (Fig. 3). Note that the only parameters that were varied are the expected radius range for cylinders and spheres, or the expected minimum and maximum size for planes.

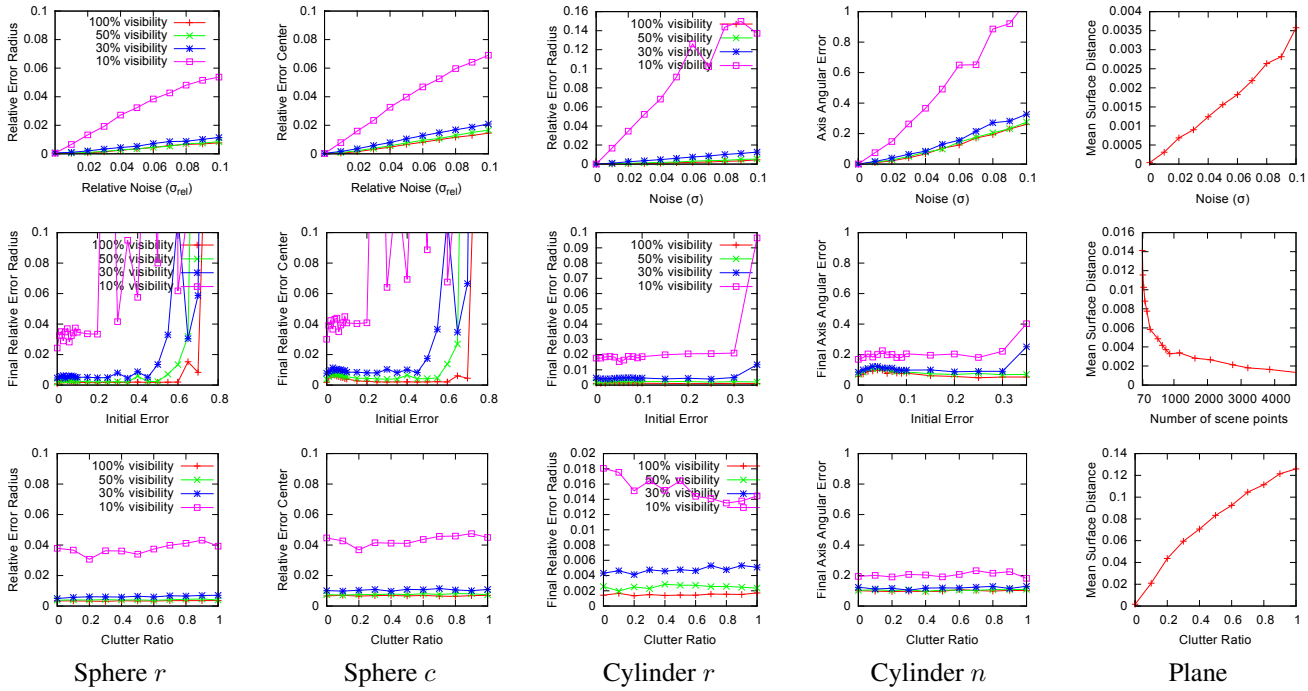


Figure 2: Accuracy of the refinement on synthetic data. Each graph shows results aggregated over several thousand runs. More details and discussion can be found in the text. Overal, the refinement is reasonably robust against noise, initialization error, and clutter.

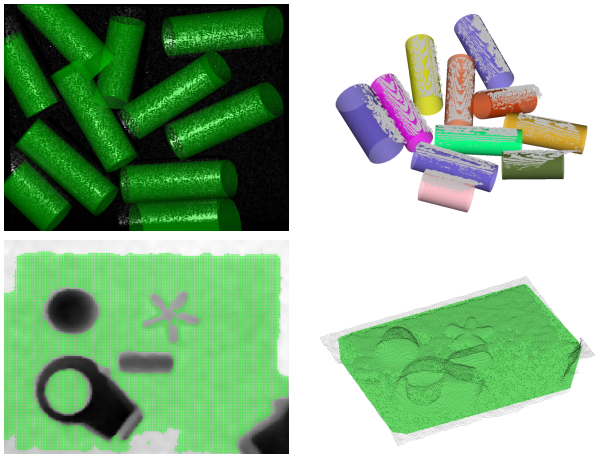


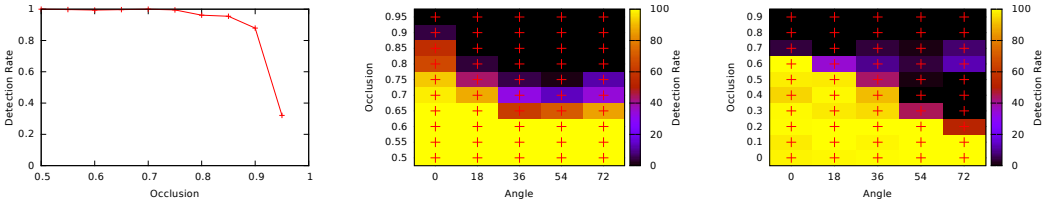
Figure 3: Real-world example for detecting cylinders and planes in industrial setups.

While cylinders are detected robustly, the determination of the exact length of the cylinders can be tricky. In some cases, the cylinder is assumed to be slightly longer than it should be if its end is touching a different object. Other than that, we found that all primitives were detected robustly, despite occlusions, clutter and noise.

7. Conclusion and Discussion

This paper presents a new Hough-Transformation for primitive detection in 3D point clouds. By using a local parametrization of the primitives, the method avoids many of the problems of a global Hough Space. The voting is integrated into a detection pipeline that removes duplicates and refines the results in a coarse-to-fine manner further.

The evaluation of refinement the detection on a large number of synthetic and real datasets shows that both are highly robust against large amounts of noise and clutter. Comparison of the plane detector against prior art shows competitive results. We believe that the deterministic runtime and the robustness against noise makes the voting scheme a good candidate against RANSAC- and oversegmentation-based approaches.



Spheres Cylinders Planes

Figure 4: Detection rates over several thousand synthetic scenes. Note that occlusion was added from the center outwards, leading to failurs if more than 70% of cylinders were detected. Also, at most half the cylinder was visible. For cylinders and planes, detection rate is also plottet against the tilt angle.

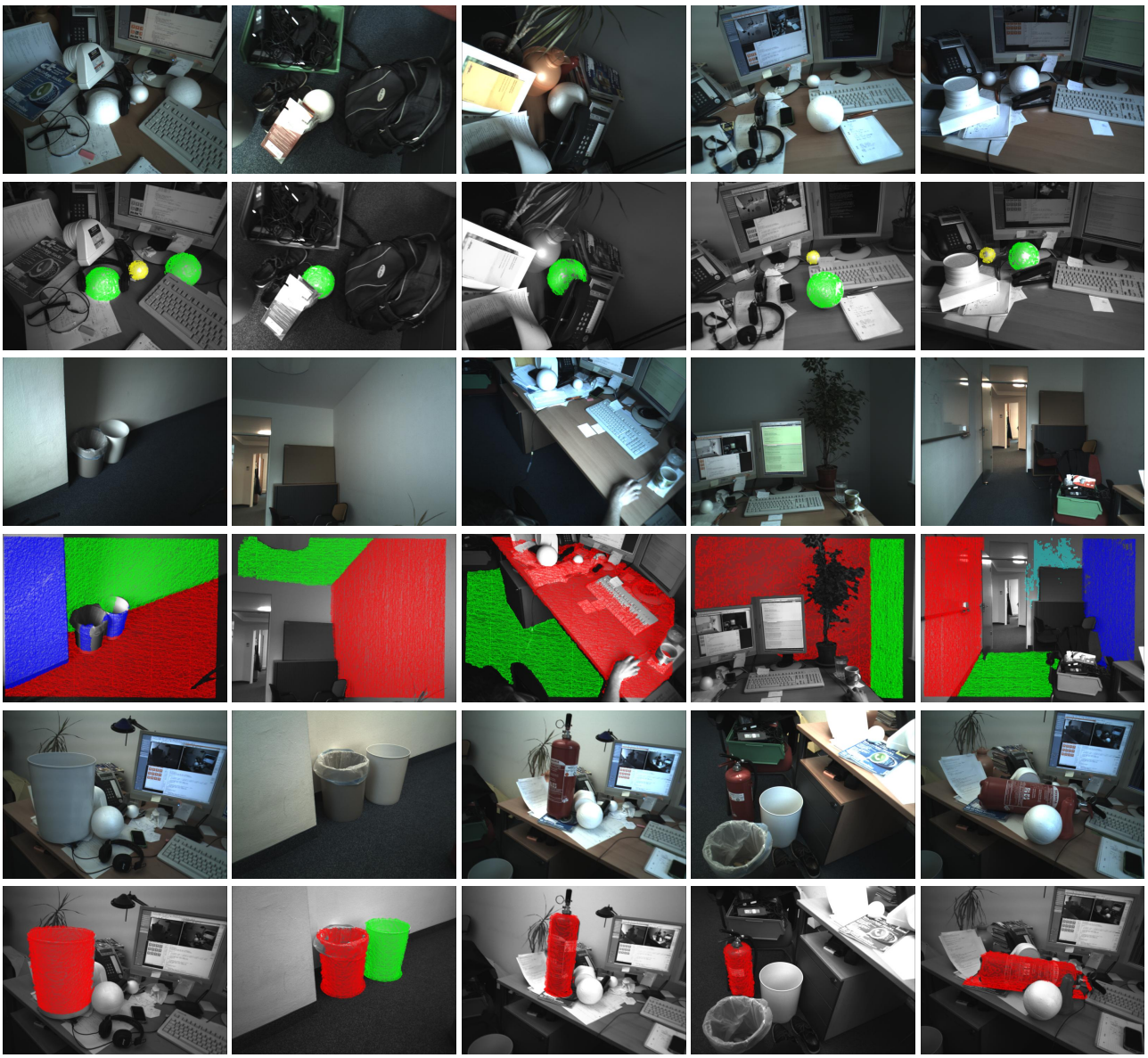


Figure 5: Some real-world examples for detecting different primitives. The examples show heavy clutter, occlusion, multiple instances of the target radius, and shaped of different size. Note that only the depth image was used for detection and that the RGB image is used only for visualization. Rows 1–2 show spheres, rows 3–4 planes and rows 5–6 cylinders.

References

[1] S. J. Ahn, I. Effenberger, S. Roth-Koch, and E. Westkämper. Geometric segmentation and object recognition in unordered and incomplete point cloud. In *Pattern Recognition*, pages 450–457. Springer, 2003. 2

[2] D. Borrmann, J. Elseberg, K. Lingemann, and A. Nüchter. The 3d hough transform for plane detection in point clouds: A review and a new accumulator design. *3D Research*, 2(2):1–13, 2011. 2

[3] B. Drost, M. Ulrich, N. Navab, and S. Ilic. Model globally, match locally: Efficient and robust 3D object recognition. In *Computer Vision and Pattern Recognition (CVPR), 2010 IEEE Conference on*, pages 998–1005. IEEE, 2010. 1, 2, 4

[4] A. W. Fitzgibbon. Robust registration of 2D and 3D point sets. *Image and Vision Computing*, 21(13-14):1145–1153, 2003. 5

[5] P. F. Gotardo, O. R. P. Bellon, and L. Silva. Range image segmentation by surface extraction using an improved robust estimator. In *Computer Vision and Pattern Recognition, 2003. Proceedings. 2003 IEEE Computer Society Conference on*, volume 2, pages II–33. IEEE, 2003. 2, 6

[6] D. Holz, S. Holzer, R. B. Rusu, and S. Behnke. Real-time plane segmentation using rgb-d cameras. In *RoboCup 2011: Robot Soccer World Cup XV*, pages 306–317. Springer, 2012. 2

[7] A. Hoover, G. Jean-Baptiste, X. Jiang, P. J. Flynn, H. Bunke, D. B. Goldgof, K. Bowyer, D. W. Eggert, A. Fitzgibbon, and R. B. Fisher. An experimental comparison of range image segmentation algorithms. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 18(7):673–689, 1996. 6

[8] S. I. Kim and S. J. Ahn. Extraction of geometric primitives from point cloud data. *Energy*, 2:0, 2005. 2

[9] T. Morwald, A. Richtsfeld, J. Prankl, M. Zillich, and M. Vincze. Geometric data abstraction using b-splines for range image segmentation. In *Robotics and Automation (ICRA), 2013 IEEE International Conference on*, pages 148–153. IEEE, 2013. 2

[10] B. Oehler, J. Stueckler, J. Welle, D. Schulz, and S. Behnke. Efficient multi-resolution plane segmentation of 3d point clouds. In *Intelligent Robotics and Applications*, pages 145–156. Springer, 2011. 2, 6

[11] J. Overby, L. Bodum, E. Kjems, and P. Iisoe. Automatic 3d building reconstruction from airborne laser scanning and cadastral data using hough transform. *International Archives of Photogrammetry and Remote Sensing*, 35(B3):296–301, 2004. 2

[12] T. Rabbani and F. V. D. Heuvel. Efficient hough transform for automatic detection of cylinders in point clouds. In *Proceedings of the 11th Annual Conference of the Advanced School for Computing and Imaging (ASCI05)*, volume 3, pages 60–65, 2005. 2

[13] R. Schnabel, R. Wahl, and R. Klein. Efficient RANSAC for point-cloud shape detection. In *Computer Graphics Forum*, volume 26, pages 214–226. Citeseer, 2007. 2

[14] F. Tarsha-Kurdi, T. Landes, and P. Grussenmeyer. Hough-transform and extended ransac algorithms for automatic detection of 3d building roof planes from lidar data. In *ISPRS Workshop on Laser Scanning 2007 and SilviLaser 2007*, volume 36, pages 407–412, 2007. 2

[15] G. Vosselman and S. Dijkman. 3d building model reconstruction from point clouds and ground plans. *International Archives of Photogrammetry Remote Sensing and Spatial Information Sciences*, 34(3/W4):37–44, 2001. 2