Fast and Accurate Normal Estimation by Efficient 3d Edge Detection

Richard Bormann[1], Joshua Hampp[1], Martin Hägele[1], Markus Vincze[2]

***Abstract*— Accurate surface normal computation is one of the most basic and important tasks for 3d perception. While much progress has been made in speeding up normal estimation algorithms and improving their accuracy, a significant inaccuracy still remains even with modern implementations, which is the correct determination of surface normals close to non-differentiable surface edges. Current algorithms tend to amalgamate neighborhood points from independent surfaces yielding normals that neither fit well to the one nor the other surface. This paper introduces a fast and accurate 3d edge detection algorithm suitable to detect discontinuities both in depth and on surfaces with nearly 90% accuracy at rates beyond 30 Hz. Based on this method, we demonstrate how established normal estimation algorithms can be extended for edge-awareness. Additionally, a new edge-aware, fast, accurate, and robust normal estimation approach is described which exploits the data structures computed for 3d edge detection and estimates normals at 23 Hz. We assess the performance of all proposed methods and compare them with other state-of-the-art approaches.**

I. Introduction

Surface normals are the backbone of a myriad of 3d perception algorithms and acquiring them in a fast and accurate fashion is absolutely crucial. The most prominent normal computation algorithms today utilize a point neighborhood to estimate the surface normal as the third principal component of the point set, i.e. the surface normal represents the normal of the tangential plane to these points [1]. Although this method has been implemented very efficiently and delivers accurate results in most cases [2], [3] it has its drawbacks when it comes to surface discontinuities such as the edges of a box. In those cases the unrestricted selection of neighborhood points mistakenly mixes points from two or three different surfaces of the box to compute a normal which is the mean of all contributing surfaces and commonly far away from the correct result. In the macroscopic view this issue establishes as round-shaped normals at edges of objects or rounded normal transitions from a ground plane to an object. In effect, this error typically harms the performance of 3d perception algorithms, e.g. surface segmentation might not distinguish between individual surfaces, surface classification could assign wrong properties to an area or 3d object modeling might produce different models when the object is standing on a table or when it is held by a gripper.

This paper proposes a simple and efficient solution to the problem of inaccurate normal estimation at surface discontinuities. The basic idea is to select only points from the same surface to estimate a normal correctly. Surfaces are delineated by surface borders that need to be determined first. Let an individual surface be defined as a smooth manifold in the Euclidean 3d space. Hence, the borders between surfaces are non-differentiable sharp bends. We introduce a method that computes local gradients to detect surface borders at acute changes of local slope. The resulting 3d edge image is utilized to only select valid neighboring points from the same surface for normal computation. This algorithm attains high efficiency if the point cloud data is available in an ordered structure, e.g. in a depth image. Fig. 1 presents an example of the improved normal quality when the proposed algorithm is employed. The main contributions of this paper are:

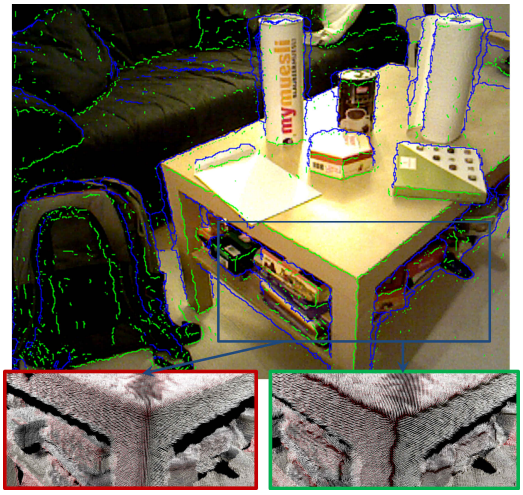


Fig. 1. 3d edge detection (blue lines=depth edges, green lines=surface discontinuities) enables accurate normal estimation near edges (green frame) instead of computing rounded normals (red frame).

1) An efficient 3d edge detection algorithm suitable to detect edges originating from both depth and surface discontinuities.
2) The extension of established accurate normal estimation methods especially to handle surface discontinuities through intelligent neighborhood point selection.
3) A highly efficient, accurate, dense and edge-aware normal estimation procedure based on data structures of the edge detection method.
4) A thorough evaluation of the developed algorithms.

The remainder of this paper is structured as follows. Sec. II provides a brief overview over related literature, followed by an explanation of the 3d edge detection algorithm in Sec. III and the approach for accurate normal estimation in Sec. IV. Subsequently, Sec. V is dedicated to the evaluation of the proposed methods before we conclude in Sec. VI.

[1] R. Bormann, J. Hampp, and M. Hägele are with the Robot and Assistive Systems Department, Fraunhofer IPA, 70569 Stuttgart, Germany. <first name>.<last name>@ipa.fraunhofer.de

[2] M. Vincze is with the Automation and Control Institute, Vienna University of Technology, 1040 Vienna, Austria.

II. Literature

Estimating point normals from range sensor data is a common exercise in the area of mobile robotics and the prerequisite for many 3d perception algorithms in this field [2]. According to its importance the robotics community has adopted numerous approaches which can be divided into optimization approaches, that minimize a cost functional and usually work with a tangential plane, and methods that average normals within a local neighborhood [1].

One of the most popular and classic optimization methods is the minimization of point distances to a common tangential plane via singular value decomposition (SVD) [4]–[6]. This approach can be improved w.r.t. speed and robustness by transforming the depth image into spherical coordinates [7]. A similar solution to the minimization problem is obtained from the analysis of local variance in the point coordinates, which results in the application of a principal component analysis (PCA) whose first two components span the tangential plane and the third yields the normal vector [2], [8]–[11]. The plane fitting can also be formulated as a Maximum Likelihood estimation problem [12]–[14], yielding an equivalent solution to the SVD or PCA based approaches. Other authors propose to estimate normals from fitting higher order geometric models to the local point neighborhood, e.g. quadric surfaces, which allow to estimate curvature at the same time [15]–[19]. The drawback of these methods is the need for more neighboring points to solve the higher dimensional linear equation system [1].

Examples for the averaging approach are [20]–[22] where simple normals are computed as cross-product between the vectors originating from a central point towards two neighboring points. Robust normals are then estimated as the neighborhood's average of these simple normals. Jin et al. [23] summarize multiple variants of this method which compute a weighted average of the simple normals, e.g. angle-weighted [24], area-weighted [25], centroid-weighted [25] and gravitational-weighted [25].

According the the analysis of [1], [12], [14], optimization methods generally attain higher accuracy and computation speed than averaging approaches at similar neighborhood sizes. However, the computational complexity of least squares optimization methods is still significant [26]. Major speed ups have been introduced to the automotive sector by the use of fast least squares and the operation on spherical range images in [7]. The robotics community exploited the organized point cloud structures of modern range sensors that facilitate the neighborhood search since 3d points can be accessed through a 2d index matrix [2]. The highly efficient normal estimation approach of [27] is based on the cross product of two tangential vectors within the point neighborhood. For the sake of robustness these tangential vectors are smoothed using integral images. Similarly, [3] computes integral images for smoothing the local depth coordinates or the local covariance matrix yielding a very fast yet accurate normal estimation algorithm.

However, scene discontinuities such as depth edges or surface discontinuities represent a significant source of error for most of the discussed normal estimation algorithms. The reason for this is the unconstrained usage of neighborhood points for smoothing, even beyond surface borders, which results in rounded normal directions around edges (see Fig. 5). In [3], this issue is addressed by detecting depth edges before normal computation and dynamic neighborhood radius selection based on distance to the next edge. The drawback of this conservative neighborhood selection is that there is a significant portion of points around edges for which no normal can be determined. Other, less efficient implementations select a more accurate neighborhood by checking every neighborhood point against the query point for a depth step [2], [22]. Another approach is the detection of surface discontinuities after normal estimation by searching for high curvature regions [2], [16]. Normals close to such edge regions need to re-computed with correct neighborhood support in a second pass. Direct, edge-aware neighborhood selection is proposed in [11], [28] where the best set of support points for the tangential plane is determined with a MLESAC technique [29]. The remaining inlier points from the neighborhood contribute weighted by distance to the covariance matrix. Given that the normal is determined via PCA eventually and considering this intricate support point selection scheme, this method can be assumed to be less efficient than integral image based approaches like [3], [27].

TABLE I

Properties of some popular normal estimation procedures: basic normal estimation of the PCL library [2], its edge-aware extension [11], cross-product based [22], and integral image based normal estimation [3].

Property	[2]	[11]	[22]	[3]	this
accuracy on continuous surfaces	high	high	high	high	high
acc. near depth discontinuities	low	high	high	high	high
acc. near surface discontinuities	low	high	low	low	high
computation speed	low	low	med	high	high
coverage (normal estim. density)	high	high	high	low	med
direct edge estimation without previous normal computation	no	no	depth	depth	yes

This paper introduces an efficient 3d edge estimation procedure for both depth edges and surface discontinuities in organized point clouds that, in contrast to the existing approaches, can be computed on the point data directly without previous normal estimation. We furthermore show how this edge knowledge can be considered efficiently for neighborhood selection with an averaging method [22] and an optimization method [3]. Eventually, we present a simple and highly efficient normal estimation algorithm based on pre-computed data of the 3d edge detection method. Tab. I compares the properties of some popular normal estimation approaches in robotics with the proposed algorithm.

III. Efficient 3d Edge Detection

This section explains our algorithm for the fast computation of 3d edges which might either originate from depth discontinuities, such as the transition from foreground objects to background, or from surface discontinuities, which are

located at non-smooth borders of two smooth surface areas, e.g. two adjacent planes of a box. The algorithm requires that the input point cloud is provided in an organized matrix structure as delivered by sensors like the Kinect.

A. Pre-Processing

At first, the organized point cloud of size $W \times H$ is rearranged as three image matrices X, Y, Z representing the x, y, and z coordinates of each point. Without loss of generality let x denote the horizontal direction from left to right, y the vertical direction from top to bottom and z the depth coordinate from the camera towards the scene. Then we compute the x-derivatives X_x and Z_x of X and Z as well as the y-derivatives Y_y and Z_y of Y and Z with a normalized 3x3 Sobel kernel. These derivatives represent the smoothed difference of neighboring pixels in x or y direction, respectively. For further noise reduction within the noisy z-dimension a Gaussian or a bilateral filter can be applied to the depth derivatives Z_x and Z_y.

B. Depth Discontinuities

The presence of a depth discontinuity is determined for each pixel $(u, v) \in [1, \ldots, W] \times [1, \ldots, H]$ in a similar way as for the Depth Change Indication Map in [3]. The minimal depth changes that can be measured by a typical depth sensing device increase quadratically with growing distance [3]. Hence, a distance dependent depth discontinuity threshold is computed

$$\tau(u, v) = \gamma \cdot Z(u, v)^2 \tag{1}$$

in order to create the depth discontinuity edge image

$$E_d(u, v) = \begin{cases} 1 & , \|Z_x(u, v)\| \geq \tau(u, v) \\ 1 & , \|Z_y(u, v)\| \geq \tau(u, v) \\ 0 & , \text{otherwise} \end{cases} \tag{2}$$

which is 1 at a depth edge and 0 elsewhere.

C. Surface Discontinuities

For the computation of surface discontinuities we need to find abrupt changes in local slope of depth measurements along the x and y directions. However, taking the values of Z_x and Z_y directly would not yield desirable results as the typical sensor noise would generate high slope variation between most neighboring pixels resulting in edge detections all over the image. Hence, local slope needs to be averaged in the vicinity of the query point. For example, in the x direction the average slope has to be estimated left $\bar{Z}_x^l(u, v)$ and right $\bar{Z}_x^r(u, v)$ of query point (u, v). An obvious way for robust average slope estimation would be a linear regression over the sampled surface points on the left and right side, respectively. This approach, however, turned out to be computationally too demanding. A more efficient solution is to average the pixel-wise slope measurements X_x and Z_x or Y_y and Z_y along the search direction left and right or above and below query pixel (u, v).

The number of slope measurements $\tilde{w}^s(u, v), s \in \{l, r, a, b\}$, that become averaged left (l), right (r), above (a), or below (b) the query pixel,

$$\tilde{w}^s(u, v) = \left[m(\varphi^*) \cdot Z(u, v) + n(\varphi^*) \right]_{w_{\min}}^{w_{\max}} \tag{3}$$

increases linearly with the depth $Z(u, v)$ of the query pixel in order to compensate for the increasing sensor noise at distance (see Sec. III-B). Our experiments showed that a linear mapping attained 1-2% better recall at similar precision compared to a fixed width or a quadratic mapping. The linear mapping is tweaked to yield $\tilde{w}^s = 5$ pixels at 0.5 m distance and φ^* pixels (parameter of algorithm) at 2 m distance. Width $\tilde{w}^s$ is, however, truncated at a lower limit $w_{\min}$ to ensure enough smoothing and at an upper limit $w_{\max}$. Moreover, $\tilde{w}^s$ is also limited by surrounding depth edges, e.g.

$$w^l(u, v) = \max_{k=1,\ldots,\tilde{w}^l} k, \quad \text{s.t.} \quad \sum_{i=1}^{k} E_d(u - i, v) \stackrel{!}{=} 0 \tag{4}$$

in order to exclude measurements from a different surface from the averaging. Please notice that increasing w^s with distance is only meant for reducing some noise artifacts of the sensor, not for adjusting the level of detectable detail since this happens fully automatically by the projective nature of the sensor that may cover a length of 2 cm at close distance with 10 pixels whereas 10 pixels may correspond to 10 cm far away. It its obvious that attempting to detect same levels of detail all over the image is infeasible with a projective sensor like a depth camera.

To speed up the slope averaging operation further, the respective values are not added up at each query but are prepared in advance by using customized integral images. For detecting vertical surface edges we prepare integral images $I[X_x]$ and $I[Z_x]$ in which a cell (u, v) stores the sum of all pixel values left of it in the same row, i.e.

$$I[X_x](u, v) = \sum_{i=0}^{u} X_x(u - i, v) \ , \tag{5}$$

$$I[Z_x](u, v) = \sum_{i=0}^{u} Z_x(u - i, v) \ . \tag{6}$$

Similarly, we deal with horizontal lines using integral images $I[Y_y]$ and $I[Z_y]$ that sum up the pixel values column-wise

$$I[Y_y](u, v) = \sum_{i=0}^{v} Y_y(u, v - i) \ , \tag{7}$$

$$I[Z_y](u, v) = \sum_{i=0}^{v} Z_y(u, v - i) \ . \tag{8}$$

Determining the average slope of any range of points within a row or column just requires two look-ups in the corresponding integral images. For the average differences of X_x and Z_x left and right of (u, v) in x direction we have

$$\bar{X}_x^l(u, v) = I[X_x](u - 1, v) - I[X_x](u - w^l(u, v), v) \ , \tag{9}$$

$$\bar{X}_x^r(u, v) = I[X_x](u + w^r(u, v), v) - I[X_x](u + 1, v) \ , \tag{10}$$

$$\bar{Z}_x^l(u, v) = I[Z_x](u - 1, v) - I[Z_x](u - w^l(u, v), v) \ , \tag{11}$$

$$\bar{Z}_x^r(u, v) = I[Z_x](u + w^r(u, v), v) - I[Z_x](u + 1, v) \ , \tag{12}$$

and for Y_y and Z_y above and below (u, v) in y direction

$$\bar{Y}_y^a(u,v) = I[Y_y](u, v-1) - I[Y_y](u, v - w^a(u,v)) \ , \quad (13)$$
$$\bar{Y}_y^b(u,v) = I[Y_y](u, v + w^b(u,v)) - I[Y_y](u, v+1) \ , \quad (14)$$
$$\bar{Z}_y^a(u,v) = I[Z_y](u, v-1) - I[Z_y](u, v - w^a(u,v)) \ , \quad (15)$$
$$\bar{Z}_y^b(u,v) = I[Z_y](u, v + w^b(u,v)) - I[Z_y](u, v+1) \ . \quad (16)$$

Following, the angle corresponding to a slope is computed with a fast approximation of the `atan2` function[1] which is always less than 0.3° off the correct value. Along the x direction we receive the left ϕ_x^l and right ϕ_x^r slope angles in the metric x, z coordinate system

$$\phi_x^l = \text{atan2}(-\bar{Z}_x^l(u,v), -\bar{X}_x^l(u,v)) \ , \quad (17)$$
$$\phi_x^r = \text{atan2}(\bar{Z}_x^r(u,v), \bar{X}_x^r(u,v)) \ , \quad (18)$$

and along the y direction we get the upper ϕ_y^a and lower ϕ_y^b slope angles in the metric y, z coordinate system

$$\phi_y^a = \text{atan2}(-\bar{Z}_y^a(u,v), -\bar{Y}_y^a(u,v)) \ , \quad (19)$$
$$\phi_y^b = \text{atan2}(\bar{Z}_y^b(u,v), \bar{Y}_y^b(u,v)) \ . \quad (20)$$

Finally, by evaluating the difference of neighboring slope angles around (u, v) we can assert whether the query point is located on a surface discontinuity. For such edge points one of the following inequalities holds

$$\left|\left|\phi_x^l - \phi_x^r\right| - \pi\right| > \theta \ , \quad (21)$$
$$\left|\left|\phi_y^a - \phi_y^b\right| - \pi\right| > \theta \ , \quad (22)$$

where θ represents the minimum detectable angle between the slopes of two separate surfaces. Please notice that we do not compare the surfaces' slopes directly because their difference or ratio is not constant under observation from different perspectives. The angle difference instead is always constant from all perspectives. Usually, a 3d surface discontinuity generates several neighboring positive responses on the angle criteria (21) and (22). To obtain sharp edges of one pixel width we only label the pixel with maximum angle difference within a row or column neighborhood of consecutive potential edge pixels. The final edge image E is then assembled of the depth edges E_d and the surface discontinuities.

The computations for the integral image and surface edge detection along the y-direction are further speed optimized by accessing memory on the transpose of Y_y and Z_y, i.e. in alignment with memory layout to fully exploit caching. Moreover, the main loop on surface discontinuity detection can be parallelized easily with OpenMP.

IV. Edge-Aware Normal Estimation

Edge image E covering depth and surface discontinuities now allows for the computation of very accurate normals based exclusively on points of the corresponding surface. In this section we present two ways how to pass on the edge information for correct neighborhood selection to the integral image based optimization approach of [3] as well as to the averaging approach of [22] without introducing significant overhead. Additionally, we present another edge-aware normal estimation procedure which exploits the already computed integral images on point differences from the edge detection algorithm.

[1] `https://gist.github.com/volkansalma/2972237`

A. Edge-Awareness by Point Cloud Modification

Integral image based normal estimation as proposed by [3] computes a Depth Change Indication Map $\mathcal{C}$ indexing depth edges in the first pass. We figured out that this procedure treats undetermined (NaN) depth values similar to depth edges in the following. Hence, we simply set all edge points of E to NaN in the point cloud which are then represented as edges in $\mathcal{C}$. The second pass generates the Final Smoothing Area Map $\mathcal{B}$ which obeys the distance to labeled edges in $\mathcal{C}$. $\mathcal{B}$ is used to estimate normals from an appropriate neighborhood which is not crossing edges.

B. Edge-Aware Neighborhood Selection

Given a query point q and its coordinates (u, v) in the ordered structure algorithms like [22] exploit the ordered structure by computing normals from a local $k \times k$ neighborhood $\mathcal{N}_k$ with point offsets $(n_i, n_j) \in [-r, \ldots, r] \times [-r, \ldots, r], k = 2r+1$. In order to compute the normal of q only with neighboring points that lie on the same surface we have to ensure that the line connecting q and a neighboring point is not intersected by an edge from E. However, checking this condition explicitly for each neighboring point is computationally very intense. We propose a significantly more efficient approach, which divides $\mathcal{N}_k$ into 8 angular sectors $\alpha_l, l \in \{1, \ldots, 8\}$ distributed around center q. In the beginning, we compute the Euclidean distance in pixel coordinates between center (u, v) and the closest edge pixel of E for each sector α_l by scanning E over $\mathcal{N}_k$ once completely. This yields a closest edge distance mapping $d(\alpha_l)$ for each sector α_l. For the sake of efficiency, the sector mapping $\alpha(n_i, n_j) : [-r, \ldots, r] \times [-r, \ldots, r] \mapsto \alpha_l$, which maps pixel offsets (n_i, n_j) of neighboring points to a sector α_l, is precomputed for the chosen neighborhood size k. Eventually, we can find the sub set $\mathcal{S} \subseteq \mathcal{N}_k$ of neighboring points $(n_i, n_j) \in \mathcal{N}_k$ that lie on the same surface as q by verifying $\|(n_i, n_j)\| < d(\alpha(n_i, n_j))$.

The actual normal estimation is realized via the cross product of two vectors pointing from query point q to two valid neighboring points $p_i, p_j \in \mathcal{S}$ similar to [23]. To facilitate a robust estimation numerous such cross products with consecutive neighbor points of $\mathcal{S}$ are averaged and normalized to the normal n_q at point q

$$n_q = \sum_{i=1}^{|\mathcal{S}|-1} (q - p_i) \times (q - p_{i+1}) \ . \quad (23)$$

C. Fast Edge-Aware Normal Estimation

After computing edge image E as described in Sec. III we have integral images $I[X_x], I[Z_x], I[Y_y], I[Z_y]$ on metric coordinate differences readily available. These can be used in a simple way for computing two smoothed tangential vectors at query point q. The smoothing ranges w^l, w^r, w^a, w^b are

determined w.r.t. to camera distance as in (3) and clamped by surface edges as in (4) but with E_d replaced by the final edge image E. Then we obtain the average differences

$$\bar{X}_x(u,v) = I[X_x](u+w^r,v) - I[X_x](u-w^l,v) \ , \quad (24)$$

$$\bar{Z}_x(u,v) = I[Z_x](u+w^r,v) - I[Z_x](u-w^l,v) \ , \quad (25)$$

$$\bar{Y}_y(u,v) = I[Y_y](u,v+w^b) - I[Y_y](u,v-w^a) \ , \quad (26)$$

$$\bar{Z}_y(u,v) = I[Z_y](u,v+w^b) - I[Z_y](u,v-w^a) \ , \quad (27)$$

which directly yield the two smoothed tangential vectors

$$\mathbf{t_1} = \left[\bar{X}_x, 0, \bar{Z}_x\right] \ , \quad \mathbf{t_2} = \left[0, \bar{Y}_y, \bar{Z}_y\right] \ . \quad (28)$$

The normal is finally computed as the cross product

$$\mathbf{n} = \mathbf{t_2} \times \mathbf{t_1} \ . \quad (29)$$

The re-use of existing data structures and efficient smoothing based on integral images renders this normal estimation approach highly efficient while respecting the presence of depth and surface discontinuities.

V. Evaluation

The evaluation assesses the quality and efficiency of the proposed 3d edge detection algorithm and the different approaches to edge-aware normal estimation. All quantitative experiments were conducted on the basis of 100 simulated scenes which were captured by a simulated RGB-D sensor.

A. Experimental Setup

For quantitative evaluation we recorded 100 different scenes with a simulated Kinect sensor from the Gazebo simulation environment. Every scene contained a ground floor, walls, a table, and several kinds of smaller objects, such as cans, cups, boxes, or cones. The number of smaller objects was chosen randomly between 10 and 50 for each scene and the placement was random as well resulting in very cluttered and complex scenes as they might occur in reality. All surfaces are color coded (green=plane, red=convex, yellow=concave, black=edge) in order make the data set also applicable to surface classification. Fig. 5 displays some exemplary scenes from this data set in the left column. The simulated RGB-D sensor delivered point cloud and color image data at 640×480 pixels resolution. Ground truth edges and normals could be computed in advance on the noise-free data of the simulated sensor. We multiplied the ground truth points with zero-mean normal distributed noise with varying standard deviation $\sigma \in \{0.0005, 0.001, 0.002, 0.005\}$ throughout the experiments to obtain noisy point cloud data $p_{\text{noisy}} = p \cdot N(0, \sigma)$ with increasing error at distance. All experiments were run on a two core mobile Intel i7 M640 CPU with 2.8 GHz and 6 GB RAM.

B. 3d Edge Detection

The 3d edge detection method was evaluated using the 100 simulated scenes w.r.t. to recall, i.e. the ratio of correctly detected edge pixels to ground truth edge pixels, and precision, defined by the ratio of correctly detected edge pixels to the number of detected edge pixels. A grid of the following parameters was searched completely for the optimal configuration of the algorithm:

- noise reduction (none, Gaussian filter, bilateral filter),
- noise reduction kernel size (3x3, 5x5, 7x7),
- line smoothing width at 2m distance, $\varphi^* \in \{10, 15, 20\}$,
- minimum detectable edge angle $\theta \in \{35°, 45°, 60°\}$.

TABLE II

Recall and precision on edge detection averaged over all tested noise levels and free configuration parameters.

		no filter	Gauss 3x3	Gauss 5x5	Gauss 7x7	bilateral 3x3	bilateral 5x5	bilateral 7x7
rec.	10	91.2	88.9	87.0	84.0	89.6	87.9	85.3
φ^*	15	90.4	88.0	86.0	82.9	88.8	86.9	84.2
	20	89.8	87.2	85.2	82.0	88.0	86.1	83.3
	35	90.9	88.7	86.8	83.7	89.4	87.6	85.0
θ	45	90.6	88.2	86.2	83.1	88.9	87.1	84.4
	60	89.9	87.3	85.3	82.0	88.1	86.2	83.4
prec.	10	84.7	86.7	85.8	80.3	86.3	86.5	83.5
φ^*	15	85.4	87.4	86.1	80.6	87.4	87.0	83.8
	20	85.6	87.6	86.2	81.2	87.6	87.1	84.0
	35	82.4	84.3	83.5	77.7	84.0	84.3	81.3
θ	45	85.3	87.4	86.1	80.6	87.4	87.0	83.7
	60	88.0	89.9	88.4	83.8	90.0	89.4	86.3

Table II compares all configurations based on recall and precision. The reported numbers are averaged over the five considered noise levels and the free parameters. The bilateral filter outperforms the Gaussian filter by a small margin of up to 1% recall-wise at a similar level of precision, however, at significantly higher computational costs. Using any noise filter yields 2% better recall and 2% worse precision but at heavily degrading performance under increasing noise. There is a clear trend indicating that a 3x3 smoothing kernel is preferable over larger kernels. Despite the little differences smaller smoothing line widths φ^* seem too be preferred slightly. The performance increases with the minimum detectable edge angle θ caused by a slow decrease in recall combined with a fast increase in precision. Fig. 2 displays some recall and precision graphs over the noise levels of input data indicating that Gaussian filtering obtains the best robustness at large noise levels. Based on closer inspection of the individual results and on tests with a real Kinect sensor we picked the following standard parameter set as the optimal setting: Gaussian noise reduction with a 3x3 kernel, $\varphi^* = 15$, and $\theta = 45°$, which represents a good compromise between robustness and accuracy.

Using this configuration, edge detection requires 27 ms using two cores with OpenMP and 35.6 ms on a single core, i.e. a processing rate of 37 Hz or 28 Hz, respectively. At a realistic noise level of $\sigma = 0.002$ the standard configuration achieves 88.1% recall and 88.0% precision. To put these numbers in context, the procedures of [2], [3], which only detect depth edges, yield 66.6% recall and 95.2% precision. Fig. 5 provides some exemplary 3d edge detection results at different noise levels.

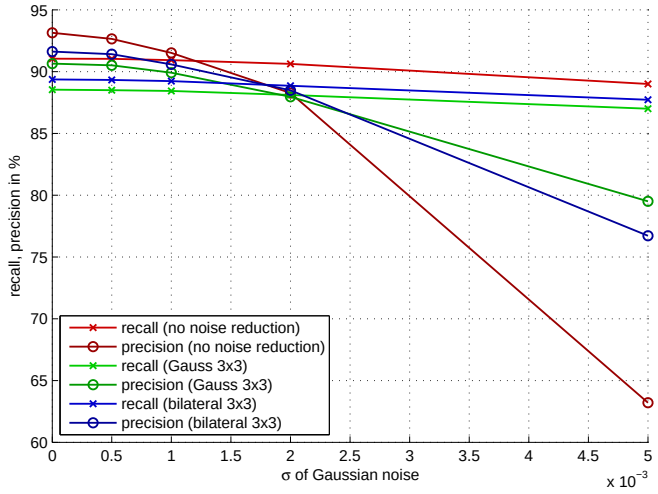


Fig. 2. Noise characteristic of precision and recall on 3d edge detection for 3 different noise reduction methods and fixed $\varphi^* = 15$, $\theta = 45$.

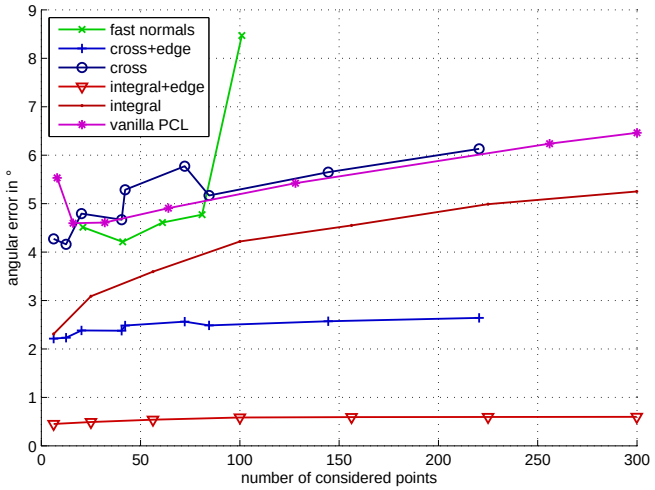


Fig. 3. Angular error at different neighborhood sizes measured on noise-free scenes for fast normal estimation (Sec. IV-C), for edge-aware/non-edge-aware normal estimation with cross product (Sec. IV-B/ [22]) or with integral images (Sec. IV-A/ [3]), and for vanilla PCL normal estimation [2].

C. Accurate Normal Estimation

Based on the previously discussed 3d edge detection algorithm, a cross-product based normal estimation method [22] and a depth edge-aware integral image based normal estimation approach [3] have been extended by complete edge-awareness. Fig. 3 compares the basic versions with the extensions in regard to average angular error of the estimated normals while varying the size of considered neighborhood. It can be observed that the edge-aware extension improves both methods substantially beyond half the original error on average. Furthermore, while the angular error increases with neighborhood size for the edge-free algorithms, it stays almost constant when the edge-aware extension is employed. This finding shows the positive effect on accuracy if normals are not smoothed over edges by points from different surfaces. The graph also indicates a solid normal estimation accuracy of less than $5°$ for the fast normal estimation introduced in this work if the smoothing region is chosen small enough. We tested all three flavors of the integral image method [3], namely smoothed covariance matrix, average 3d gradient, and average depth change. Measured over all noise levels, average 3d gradient surpassed the other two variants by far with respect to accuracy and robustness and is hence reported in the graphs.

Fig. 4 is dedicated to a more precise robustness analysis of the edge-aware normal estimation methods. The fast normal estimation, although starting at a relatively high level of angular error of $4.5°$ at low noise, is very robust and does not increase above $7°$ at heavy noise. Cross product and integral image based normal estimation are more sensitive to noise and raise by at least $7°$ between no noise and heavy noise.

Tab. III summarizes and compares several performance measures of the evaluated normal estimation algorithms at a realistic noise level of $\sigma = 0.002$. It shows that every methods has its strengths and weaknesses. The classic normal estimation of the PCL library and the cross product approach can estimate normals at virtually any point, however, achieve

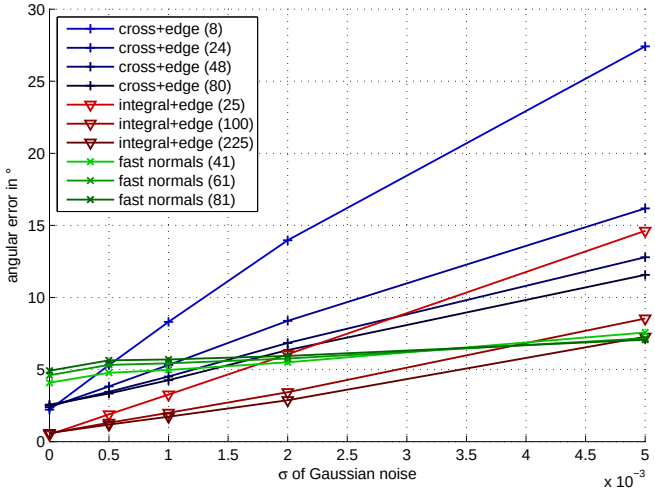


Fig. 4. Angular error at different noise levels measured for the edge-aware normal estimation with integral images (Sec. IV-A), with cross product (Sec. IV-B), and for the fast normal estimation (Sec. IV-C). The (maximum) number of considered neighborhood points is noted in brackets.

the lowest accuracy while claiming high computational efforts. In contrast, the extended version of integral image based normal estimation attains the highest accuracy with only $2.7°$ angular error at a low computation time of only 90 ms with one core, but at a very low coverage. The latter drawback might be mitigated by combination with an accurate and dense approach like the edge-aware cross product normal estimation, which is, however, rather slow when computed over the entire image. The most interesting algorithms for practical usage are fast edge-aware normal estimation and integral image based normal estimation which yield very high computation rates around 20 Hz. Whereas the latter method is slightly faster, fast edge-aware normal estimation achieves higher coverage and better accuracy, especially in the vicinity of surface discontinuities.

TABLE III

NORMAL ESTIMATION PERFORMANCE OF 6 DIFFERENT METHODS AT A POINT CLOUD NOISE LEVEL OF $\sigma = 0.002$. THE EDGE DETECTION WAS RUN WITH PARAMETERS 3X3 GAUSSIAN SMOOTHING, $\varphi^* = 15$, $\theta = 45$. NEIGHBORHOOD SIZES ARE REPORTED IN BRACKETS.

	angular error [°]	good normals [%]	cover-age [%]	time 1 core [ms]	time 2 cores [ms]
integral+edge (400)	2.7	96.8	84.5	89.9	80.6
integral (100)	6.2	92.2	93.7	50.8	n/a
cross+edge (80)	6.3	95.8	97.0	654.8	348.5
cross (80)	9.0	89.7	99.9	424.6	229.9
fast edge-aware (61)	5.8	92.7	94.3	62.1	43.4
vanilla PCL (128)	6.6	92.3	100.0	4535.2	2049.1

VI. CONCLUSION

This paper has introduced a novel method for detecting 3d edges caused by depth or surface discontinuities which is extremely fast to compute yet very accurate. Especially the novel ability to detect edges at surface discontinuities makes this algorithm suitable to guide the neighborhood selection of established normal computation approaches in order to improve their accuracy significantly. Additionally, we have proposed a novel fast edge-aware normal estimation method which re-uses data from edge detection and achieves a competitive performance with respect to quality and speed. The comprehensive evaluation compared numerous properties of popular normal estimation algorithms and their edge-aware extensions and is meant to provide guidance in selecting the appropriate method for different tasks.

The proposed algorithms are publicly available wrapped as a ROS package[2]. We intend to integrate the fast 3d edge detection and normal estimation methods into the PCL library in near future.

REFERENCES

[1] K. Klasing, D. Althoff, D. Wollherr, and M. Buss, "Comparison of surface normal estimation methods for range sensing applications," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, May 2009, pp. 3206–3211.

[2] R. Rusu and S. Cousins, "3d is here: Point Cloud Library (PCL)," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, May 2011, pp. 1–4.

[3] S. Holzer, R. Rusu, M. Dixon, S. Gedikli, and N. Navab, "Adaptive neighborhood selection for real-time surface normal estimation from organized point cloud data using integral images," in *Proceedings of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Oct. 2012, pp. 2684–2689.

[4] R. Hoffman and A. K. Jain, "Segmentation and Classification of Range Images," *IEEE Transactions on Pattern Analysis and Machine Intelligence (PAMI)*, vol. 9, no. 5, pp. 608–620, Sept. 1987.

[5] J. Huang and C.-H. Menq, "Automatic data segmentation for geometric feature extraction from unorganized 3-D coordinate points," *IEEE Transactions on Robotics and Automation*, vol. 17, no. 3, pp. 268–279, June 2001.

[6] N. J. Mitra, A. Nguyen, and L. Guibas, "Estimating surface normals in noisy point cloud data," *International Journal of Computational Geometry & Applications*, vol. 14, no. 4–5, pp. 261–276, Oct. 2004.

[2] https://github.com/ipa-rmb/cob_object_perception/tree/indigo_dev/cob_surface_classification

[7] H. Badino, D. Huber, Y. Park, and T. Kanade, "Fast and accurate computation of surface normals from range images," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, May 2011, pp. 3084–3091.

[8] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle, "Surface Reconstruction from Unorganized Points," in *Proc. of the Annual Conference on Computer Graphics and Interactive Techniques*, ser. SIGGRAPH '92. New York, NY, USA: ACM, 1992, pp. 71–78.

[9] M. Pauly, M. Gross, and L. Kobbelt, "Efficient simplification of point-sampled surfaces," in *IEEE Visualization*, Nov. 2002, pp. 163–170.

[10] J. Weingarten, G. Gruener, and R. Siegwart, "Probabilistic plane fitting in 3d and an application to robotic mapping," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, vol. 1, Apr. 2004, pp. 927–932.

[11] R. B. Rusu, Z. C. Marton, N. Blodow, M. Dolha, and M. Beetz, "Towards 3d Point cloud based object maps for household environments," *Robotics and Autonomous Systems*, vol. 56, no. 11, pp. 927–941, 2008.

[12] C. Wang, H. Tanahashi, H. Hirayu, Y. Niwa, and K. Yamamoto, "Comparison of local plane fitting methods for range data," in *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, vol. 1, 2001, pp. 663–669.

[13] K. Kanatani, *Statistical Optimization for Geometric Computation: Theory and Practice*. Courier Corporation, July 2005.

[14] K. Pathak, N. Vaskevicius, and A. Birk, "Uncertainty analysis for optimum plane extraction from noisy 3d range-sensor point-clouds," *Intelligent Service Robotics*, vol. 3, no. 1, pp. 37–48, Nov. 2009.

[15] M. Milroy, C. Bradley, and G. Vickers, "Segmentation of a wrap-around model using an active contour," *Computer-Aided Design*, vol. 29, no. 4, pp. 299–320, Apr. 1997.

[16] M. Yang and E. Lee, "Segmentation of measured point data using a parametric quadric surface approximation," *Computer-Aided Design*, vol. 31, no. 7, pp. 449–457, June 1999.

[17] M. Vanco, "A Direct Approach for the Segmentation of Unorganized Points and Recognition of Simple Algebraic Surfaces," Ph.D. dissertation, University of Technology Chemnitz, 2003.

[18] D. OuYang and H.-Y. Feng, "On the normal vector estimation for point cloud data from smooth surfaces," *Computer-Aided Design*, vol. 37, no. 10, pp. 1071–1079, Sept. 2005.

[19] T. K. Dey and J. Sun, "Normal and Feature Approximations from Noisy Point Clouds," in *FSTTCS 2006: Foundations of Software Technology and Theoretical Computer Science*, ser. Lecture Notes in Computer Science, S. Arun-Kumar and N. Garg, Eds. Springer Berlin Heidelberg, 2006, no. 4337, pp. 21–32.

[20] H. Gouraud, "Continuous Shading of Curved Surfaces," *IEEE Transactions on Computers*, vol. C-20, no. 6, pp. 623–629, June 1971.

[21] L. Spinello, R. Triebel, and R. Siegwart, "Multimodal detection and tracking of pedestrians in urban environments with explicit ground plane extraction," in *Proc. of the IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS)*, Sept. 2008, pp. 1823–1829.

[22] G. Arbeiter, S. Fuchs, J. Hampp, and R. Bormann, "Efficient segmentation and surface classification of range images," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, May 2014, pp. 5502–5509.

[23] S. Jin, R. R. Lewis, and D. West, "A comparison of algorithms for vertex normal computation," *The Visual Computer*, vol. 21, no. 1-2, pp. 71–82, Feb. 2005.

[24] G. Thürmer and C. A. Wüthrich, "Computing Vertex Normals from Polygonal Facets," *J. of Graphics Tools*, vol. 3, no. 1, pp. 43–46, 1998.

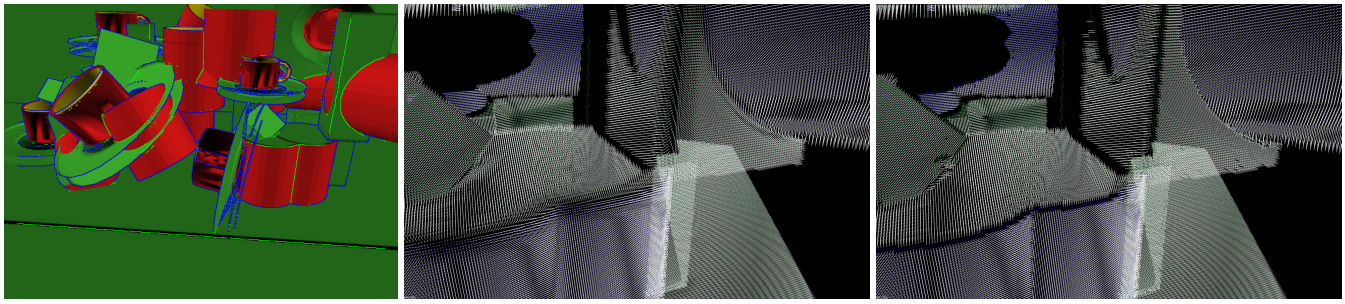
[25] N. Max, "Weights for Computing Vertex Normals from Facet Normals," *Journal of Graphics Tools*, vol. 4, no. 2, pp. 1–6, Mar. 1999.

[26] R. B. Rusu, "Semantic 3d object maps for everyday manipulation in human living environments," Ph.D. dissertation, Technische Universität München, 2009.

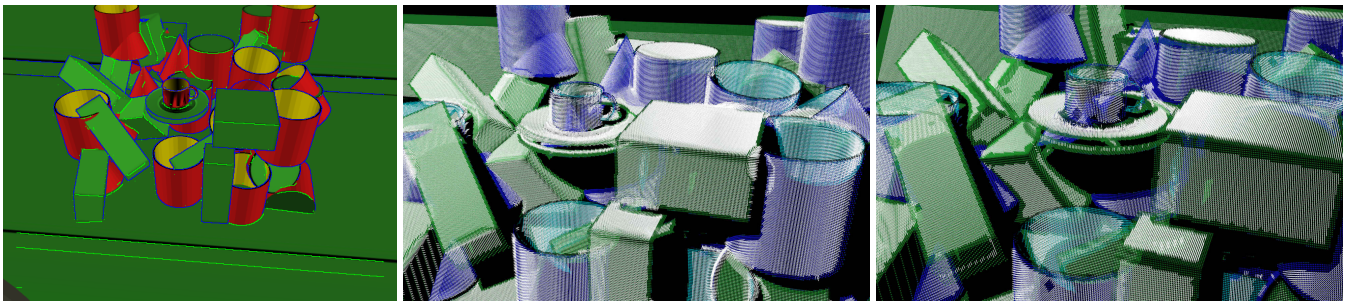
[27] D. Holz, S. Holzer, R. B. Rusu, and S. Behnke, "Real-Time Plane Segmentation Using RGB-D Cameras," in *RoboCup 2011: Robot Soccer World Cup XV*, ser. Lecture Notes in Computer Science, T. Röfer, N. M. Mayer, J. Savage, and U. Saranli, Eds. Springer Berlin Heidelberg, 2012, no. 7416, pp. 306–317.

[28] Z. Marton, R. Rusu, and M. Beetz, "On fast surface reconstruction methods for large and noisy point clouds," in *Proceedings of the IEEE International Conference on Robotics and Automation (ICRA)*, May 2009, pp. 3218–3223.

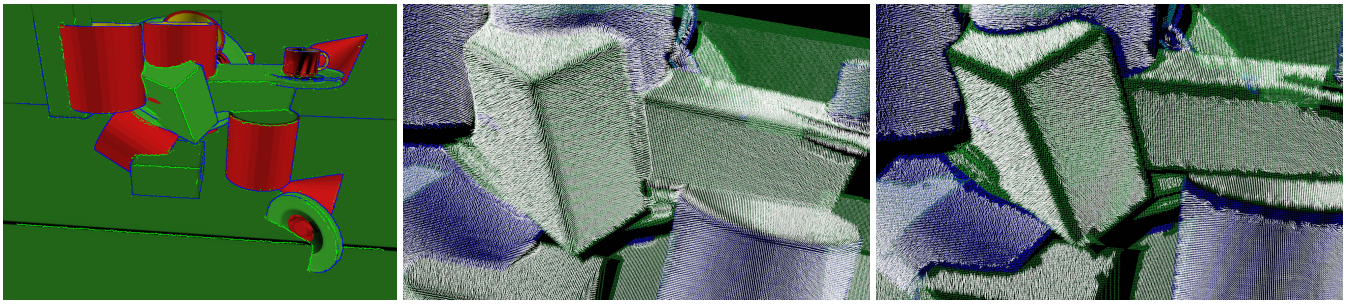
[29] P. H. S. Torr and A. Zisserman, "MLESAC: A New Robust Estimator with Application to Estimating Image Geometry," *Computer Vision and Image Understanding*, vol. 78, p. 2000, 2000.



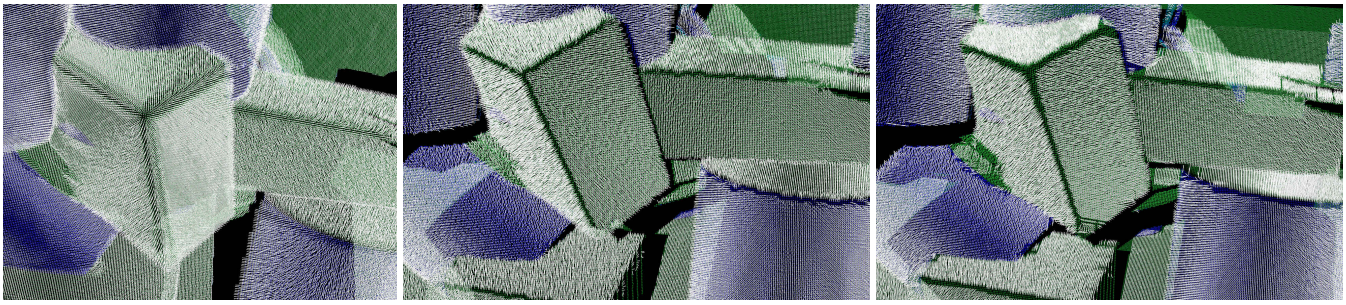
$\sigma = 0$, comparison of cross product based normal estimation without and with edge-awareness



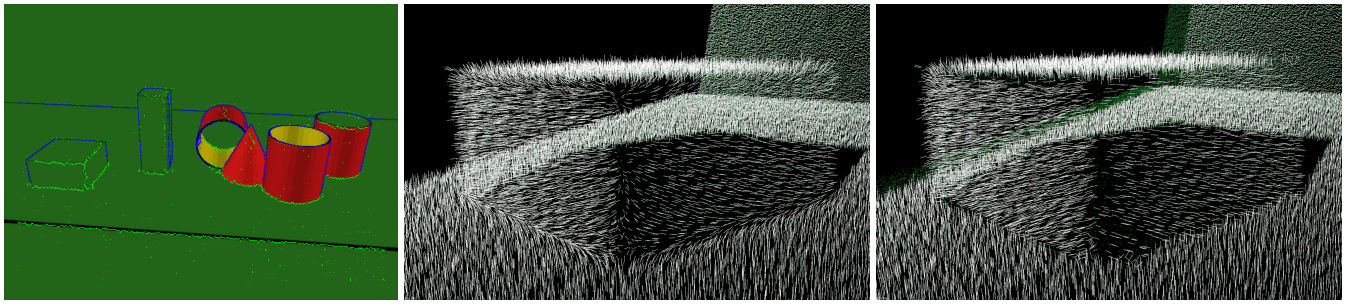
$\sigma = 0$, comparison of integral image based normal estimation without and with edge-awareness



$\sigma = 0.002$, comparison of integral image based normal estimation without and with edge-awareness



$\sigma = 0.002$, comparison of vanilla PCL, edge-aware cross product based, and fast edge-aware normal estimation



$\sigma = 0.005$, comparison of vanilla PCL and fast edge-aware normal estimation

Fig. 5. Exemplary 3d edge detection results (blue lines=depth edge, green lines=surface discontinuities) and classic normal estimation methods compared with their edge-aware counterparts at various noise levels.