2012 IEEE/RSJ International Conference on
Intelligent Robots and Systems
October 7-12, 2012. Vilamoura, Algarve, Portugal

Adaptive Neighborhood Selection for Real-Time Surface Normal Estimation from Organized Point Cloud Data Using Integral Images

S. Holzer[1,2] and R. B. Rusu[2] and M. Dixon[2] and S. Gedikli[2] and N. Navab[1]

***Abstract*— In this paper we present two real-time methods for estimating surface normals from organized point cloud data. The proposed algorithms use integral images to perform highly efficient border- and depth-dependent smoothing and covariance estimation. We show that this approach makes it possible to obtain robust surface normals from large point clouds at high frame rates and therefore, can be used in real-time computer vision algorithms that make use of Kinect-like data.**

I. Introduction

The recent development of a new class of affordable depth sensors, like the Kinect, has been of great interest to the robotics community. These new sensors are able to simultaneously capture high-resolution color and depth images at high frame rates. When the camera's intrinsic calibration parameters are known, these depth images can be converted into *organized point clouds* (i.e., clouds of 3D points sampled from a regular 2D grid), which are useful in a wide array of important robotics applications, such as 3D registration and object recognition.

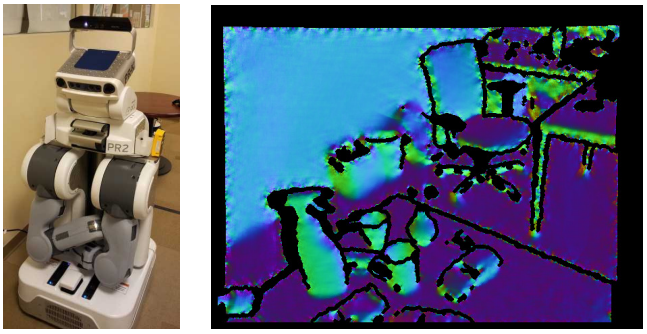
Besides depth, surface normal orientation is one of the most discrimative information that can be obtained from point clouds and is therefore optimally suited to be used in object detection approaches. However, state-of-the-art normal estimation algorithms are often slow when operating on large and/or noisy point clouds. Thus, for robotics applications that require real-time performance, fast normal estimation is essential.

In this work, we present two algorithms for estimating surface normals in organized point clouds. Both methods employ an adaptive window size to analyze local surfaces, which allows us to effectively handle depth-dependent sensor noise and to avoid common artifacts caused by depth discontinuities. Typically, multi-scale algorithms come with increased computational costs, especially when large window sizes are used; however, because of the point clouds' inherent grid structure, we are able to use integral images to perform the necessary computation in constant time, independent of the window size.

II. Related Work

Normal estimation methods can be divided into two different categories: *averaging* and *optimization-based* methods [1]. *Averaging methods* [1], [2], [3], [4], [5], [6] compute the normals at a certain point as a (weighted) average of point data within a certain neighborhood. Examples for these neighboring points can be the nearest neighbors, all points within a certain region of interest, or points that are in some other way connected to the point of interest, e.g. neighbors in a triangulated mesh. The weights, which define how strong the influence of particular information taken from the local neighborhood is, can be computed in different ways. Common ways of computing the weighted mean include weighting all points equally [2], weighting by angle [5], weighting by sine and edge length reciprocals [4], weighting by areas of adjacent triangles [4], weighting by edge length reciprocals [4], and weighting by square root of edge length reciprocals [4]. In [6], Holz *et al*. presented how integral images can be used in the efficient averaging for normal estimation process. For a more detailed overview of methods based on averaging please refer to [3].

Optimization-based methods usually try to fit geometric primitives, e.g. a plane, into the local neighborhood of the point of interest or penalize other criteria, e.g. the angle between the estimated normal vector and tangential vectors. If the optimization is formulated as a linear problem in matrix-vector notation, the desired minimization can be directly obtained from the result of a singular value decomposition (SVD) or a principal component analysis (PCA) [1]. Amongst others, existing methods try to fit planes [7], [8], [9], [10], maximize the angle between the tangential vectors and the normal vector [11], or try to estimate not only the orientation of the tangent plane but also the curvature [12], [13], [14]. A more detailed comparison of these methods can be found in [1].



(a) (b)

Fig. 1. **(a)** PR2 robot with a Kinect depth sensor mounted on its head. **(b)** Color-coded surface normals estimated using the proposed approaches.

*This work was supported by Willow Garage Inc.

[1]S. Holzer and N. Navab are with the Department of Computer Science, Technische Universität München, 85748 Garching bei München, Germany {holzers, navab}@in.tum.de

[2]S. Holzer, R. B. Rusu, M. Dixon and S. Gedikli are with Willow Garage Inc., Menlo Park, CA 94025, USA {holzers, rusu, mdixon, gedikli}@willowgarage.com

III. Normal Estimation

In the following we will describe the different steps we use to compute surface normals from organized point clouds based on integral images. First, we will describe the basic principle of integral images. Then we introduce the pre-processing step used to estimate the neighborhood size for each pixel. Finally, we introduce two different approaches for computing surface normals: a method based on averaging as well as an optimization-based method. Both techniques make use of integral images to improve processing speed. The first approach (see Sec. III-C) uses a single integral image for depth and border aware smoothing of the depth map, while the second approach (see Sec. III-D) computes covariance matrices using integral images and obtains the normals from the covariance matrices.

A. Integral Images

An integral image $\mathcal{I}_\mathcal{O}$ corresponding to an image $\mathcal{O}$ makes it possible to compute the sum of all values of $\mathcal{O}$ within a certain rectangular region $\mathcal{R}_{(m_s,n_s)\rightarrow(m_e,n_e)}$ (see Fig. 2) by accessing only four data elements in memory. This not only makes the computation very efficient but also makes the computational costs independent of the size of the rectangle. We will use this property later for smoothing since it allows us to use smoothing areas which differ in size for every point. To be able to compute the area sum of an image $\mathcal{O}$ each pixel element $(m,n)^\top$ in the integral image $\mathcal{I}_\mathcal{O}$ is defined as the sum of all elements which are inside of the rectangular area between $\mathcal{O}(0,0)$ and $\mathcal{O}(m,n)$:

$$\mathcal{I}_\mathcal{O}(m,n) = \sum_{i=0}^{m}\sum_{j=0}^{n}\mathcal{O}(i,j). \tag{1}$$

This can be efficiently computed iteratively as

$$\begin{aligned}\mathcal{I}_\mathcal{O}(m,n) \quad = \quad & \mathcal{O}(m,n) \\ & + \mathcal{I}_\mathcal{O}(m-1,n) \\ & + \mathcal{I}_\mathcal{O}(m,n-1) \\ & - \mathcal{I}_\mathcal{O}(m-1,n-1),\end{aligned} \tag{2}$$

where $\mathcal{I}_\mathcal{O}(u,v) = 0$ if (u,v) is not in the domain of $\mathcal{O}$. Therefore, a single pass over the input image is sufficient to compute the corresponding integral image. The average value within a region can then be computed as

$$\begin{aligned}S(\mathcal{I}_\mathcal{O},m,n,r) = \frac{1}{4r^2}\cdot \quad (\quad & \mathcal{I}_\mathcal{O}(m+r,n+r) \\ & - \mathcal{I}_\mathcal{O}(m-r,n+r) \\ & - \mathcal{I}_\mathcal{O}(m+r,n-r) \\ & + \mathcal{I}_\mathcal{O}(m-r,n-r) \quad),\end{aligned} \tag{3}$$

where $(m,n)^\top$ defines the center and r the inner radius of the rectangular region.

B. Smoothing

Data obtained from a 3D sensor typically contains noise. The standard approach for reducing the effect of noise is to use smoothing. A big advantage of using integral images for averaging is hereby that its processing speed is independent of the size of the smoothing area since we always need the same amount of memory accesses. Therefore, we can use varying smoothing sizes depending on the characteristics of the considered point and its neighborhood.

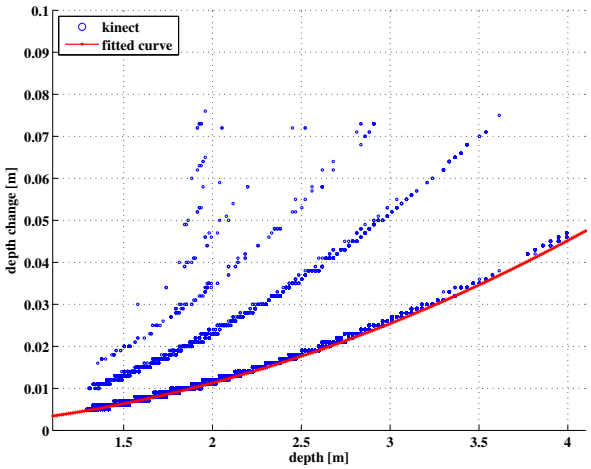


Fig. 3. Visualization of depth changes (blue circles) depending on the depth at which they occur. The minimal depth changes are fitted by a parabola (red line).

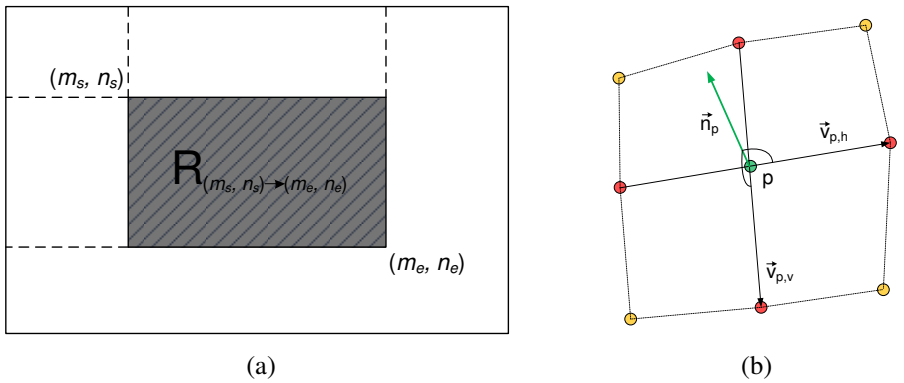
In the following we will use two different indicators for estimating the size of the smoothing area for a certain point of interest. The first indicator is the depth of the point of interest, since the noise usually depends on the depth of the perceived data, i.e. if data is acquired at a far distance then it has a worse signal-to-noise ratio than data that is acquired at close distances. However, if the size of the smoothing area is determined based only on the depth of the point of interest we would also smooth over object borders. This would merge information across two distinct surfaces and lead to incorrect normal estimates. In order to prevent this, we make the size of the smoothing area also dependent on large depth changes which are likely to be object borders. Both indicators are then combined into a single Smoothing Area Map which defines the smoothing area for each point of the organized point cloud.

1) Depth-Dependent Smoothing Area Map: Fig. 3 shows the minimal depth change (blue circles) that can occure at a specific depth. As shown, the value gets bigger with increasing depth. The red curve shows that the relationship between depth and minimum depth change can be described using the function

$$f_{DC}(d) = \alpha \cdot d^2, \tag{4}$$

where d is a depth value. For the device used in our experiments we estimated $\alpha = 0.0028$. Based on this observation it is clear that the smoothing area has to get bigger with increasing depth in a similar way. Therefore, we define the response of the depth-dependent *Smoothing Area Map* $\mathcal{B}(m,n)$:

$$\mathcal{B}(m,n) = \beta \cdot f_{DC}\left(\mathcal{D}\left(m,n\right)\right), \tag{5}$$



(a) (b)

Fig. 2. **(a)** The sum of a 2D region can be efficiently computed from an integral image by accessing only four data elements in memory which correspond to the for corners of the rectangular region. **(b)** Estimating a surface normal as cross-product of the vectors between the horizontal and vertical neighbors of the point of interest.

where β is a user specified value to control the increase of the smoothing area size and $\mathcal{D}(m,n)$ is the depth value at the image point $(m,n)^\top$.

2) Depth Change Indication Map: The simplest solution for computing the *Depth Change Indication Map* would be to apply a threshold on the first derivative of the depth map. However, as we just saw, the minimum possible depth change depends on the depth and therefore, a simple threshold would only be valid at a specific depth. Instead, we create a binary *Depth Change Indication Map* $\mathcal{C}$ by using a depth change detection threshold $t_{DC}(d)$ which is dependent on the actual distance:

$$t_{DC}(d) = \gamma \cdot f_{DC}(d), \tag{6}$$

where γ is a scale factor which defines how sensitive the depth change detection will be. Assuming that $\delta\mathcal{D}_x(m,n) = \mathcal{D}(m+1,n) - \mathcal{D}(m,n)$ and $\delta\mathcal{D}_y(m,n) = \mathcal{D}(m,n+1) - \mathcal{D}(m,n)$, the depth change indication map $\mathcal{C}$ is then computed as

$$\mathcal{C}(m,n) = \begin{cases} 1 & \begin{cases} \text{if } \|\delta\mathcal{D}_x(m,n)\| \geq t_{DC}(\mathcal{D}(m,n)) \\ \text{or } \|\delta\mathcal{D}_y(m,n)\| \geq t_{DC}(\mathcal{D}(m,n)) \end{cases} \\ 0 & \text{otherwise.} \end{cases} \tag{7}$$

3) Final Smoothing Area Map: Using the depth-dependent *Smoothing Area Map* $\mathcal{B}$ and the *Depth Change Indication Map* $\mathcal{C}$ we then compute the final *Smoothing Area Map* $\mathcal{R}$. For this, we first compute the distance transform map [15] corresponding to $\mathcal{C}$, which gives us for each point $(m,n)^\top$ the 2D distance to the next depth change as $\mathcal{T}(m,n)$. $\mathcal{R}$ is then computed as

$$\mathcal{R}(m,n) = \min(\mathcal{B}(m,n), \frac{\mathcal{T}(m,n)}{\sqrt{2}}), \tag{8}$$

where $\min(a,b)$ returns the minimum of a and b. The distance values $\mathcal{T}(m,n)$ are divided by the square root of 2 since we do not use circular but rectangular smoothing areas.

C. Normal Estimation based on Smoothed Depth Changes

A standard way of estimating the surface normal $\vec{n}_p$ at a point p at image location $(m,n)^\top$ is to compute the 3D vector $\vec{v}_{p,h}$ between the left and right neighbor as well as the vector $\vec{v}_{p,v}$ between the upper and lower neighbor of p and then computing the cross-product between these two vectors:

$$\vec{n}_p = \vec{v}_{p,h} \times \vec{v}_{p,v} \tag{9}$$

Due to the noise characteristics of depth sensors, *e.g.* the Kinect, this would lead to noisy normals. Smoothing the depth data before computing the normals is a common approach to reduce the influence of noise. However, smoothing with a fixed window size also smoothes over object boundaries, which leads to undesired artifacts. Therefore, we use the smoothing area map described in Sec. III-B.3 to prevent from smoothing over object boundaries and efficiently compute the vectors $\vec{v}_{p,h}$ and $\vec{v}_{p,v}$ as:

$$\vec{v}_{p,h,x} = \frac{\mathcal{P}_x(m+r,n) - \mathcal{P}_x(m-r,n)}{2}, \tag{10}$$

$$\vec{v}_{p,h,y} = \frac{\mathcal{P}_y(m+r,n) - \mathcal{P}_y(m-r,n)}{2}, \tag{11}$$

$$\vec{v}_{p,h,z} = \frac{S(\mathcal{I}_{\mathcal{P}_z}, m+1, n, r-1)}{2} - \frac{S(\mathcal{I}_{\mathcal{P}_z}, m-1, n, r-1)}{2} \tag{12}$$

$$\vec{v}_{p,v,x} = \frac{\mathcal{P}_x(m,n+r) - \mathcal{P}_x(m,n-r)}{2}, \tag{13}$$

$$\vec{v}_{p,v,y} = \frac{\mathcal{P}_y(m,n+r) - \mathcal{P}_y(m,n-r)}{2}, \tag{14}$$

$$\vec{v}_{p,v,z} = \frac{S(\mathcal{I}_{\mathcal{P}_z}, m, n+1, r-1)}{2} - \frac{S(\mathcal{I}_{\mathcal{P}_z}, m, n-1, r-1)}{2}, \tag{15}$$

where $\mathcal{P}_x$, $\mathcal{P}_y$, and $\mathcal{P}_z$ are two-dimensional maps storing the x-, y-, and z-coordinates of the organized point cloud, $\mathcal{I}_{\mathcal{P}_z}$ is the integral image of the z-components of the point cloud, and $r = \mathcal{R}(m,n)$. The normals are then computed using Eq. (9).

D. Normal Estimation based on Covariance Matrices

Our second method is an optimization-based method, where we estimate surface normals by trying to fit a plane

into the local neighborhood $\mathcal{N}_p$ of the point of interest p. This is done by computing the eigenvectors of the corresponding covariance matrix $\mathcal{C}_p$. The size of the neighborhood is estimated using the smoothing area map as described in Sec. III-B.3. Neighboring points are commonly found by performing a nearest neighbor search or selecting all points within a certain distance. This, however, is an expensive operation and therefore, we make use of the method described by Porikli and Tuzel [16] for efficient computation of covariance matrices using integral images. For this we have to compute nine integral images, where three of them, namely $\mathcal{I}_{\mathcal{P}_x}$, $\mathcal{I}_{\mathcal{P}_y}$, and $\mathcal{I}_{\mathcal{P}_z}$, are for the x-, y- and z-coordinates of the points of the point cloud and the remaining six are for all possible combinations of the point-coordinates: $\mathcal{I}_{\mathcal{P}_{xx}}$, $\mathcal{I}_{\mathcal{P}_{xy}}$, $\mathcal{I}_{\mathcal{P}_{xz}}$, $\mathcal{I}_{\mathcal{P}_{yy}}$, $\mathcal{I}_{\mathcal{P}_{yz}}$, $\mathcal{I}_{\mathcal{P}_{zz}}$, where $\mathcal{I}_{\mathcal{P}_{ab}}$ is the element-wise multiplication of $\mathcal{I}_{\mathcal{P}_a}$ and $\mathcal{I}_{\mathcal{P}_b}$. The covariance matrix $\mathcal{C}_p$ for a point p at $(m,n)^\top$ can then be computed as:

$$\mathcal{C}_p = \begin{pmatrix} c_{xx} & c_{xy} & c_{xz} \\ c_{yx} & c_{yy} & c_{yz} \\ c_{zx} & c_{zy} & c_{zz} \end{pmatrix} - \begin{pmatrix} c_x \\ c_y \\ c_z \end{pmatrix} \begin{pmatrix} c_x \\ c_y \\ c_z \end{pmatrix}^\top, \quad (16)$$

with

$$\begin{aligned} c_{xx} &= \mathcal{S}(\mathcal{I}_{\mathcal{P}_{xx}}, m, n, \mathcal{R}(m,n)), & (17)\\ c_{xy} = c_{yx} &= \mathcal{S}(\mathcal{I}_{\mathcal{P}_{xy}}, m, n, \mathcal{R}(m,n)), & (18)\\ c_{xz} = c_{zx} &= \mathcal{S}(\mathcal{I}_{\mathcal{P}_{xz}}, m, n, \mathcal{R}(m,n)), & (19)\\ c_{yy} &= \mathcal{S}(\mathcal{I}_{\mathcal{P}_{yy}}, m, n, \mathcal{R}(m,n)), & (20)\\ c_{yz} = c_{zy} &= \mathcal{S}(\mathcal{I}_{\mathcal{P}_{yz}}, m, n, \mathcal{R}(m,n)), & (21)\\ c_{zz} &= \mathcal{S}(\mathcal{I}_{\mathcal{P}_{zz}}, m, n, \mathcal{R}(m,n)), & (22) \end{aligned}$$

and

$$\begin{aligned} c_x &= \mathcal{S}(\mathcal{I}_{\mathcal{P}_x}, m, n, \mathcal{R}(m,n)), & (23)\\ c_y &= \mathcal{S}(\mathcal{I}_{\mathcal{P}_y}, m, n, \mathcal{R}(m,n)), & (24)\\ c_z &= \mathcal{S}(\mathcal{I}_{\mathcal{P}_z}, m, n, \mathcal{R}(m,n)). & (25) \end{aligned}$$

Finally, we compute the normal $\vec{n}_p$ from the covariance matrix $\mathcal{C}_p$ as the eigenvector which corresponds to the smallest eigenvalue. Although this method is computationally more expensive than the method described in Sec. III-C it has the advantage that the eigenvalues of the covariance matrix can be directly used to get information about the planarity of the neighborhood around the point p at image location $(m,n)^\top$. This can be used for plane fitting as well as efficient edge or corner detection.

IV. Results

In this section we show how the presented surface normal estimation methods perform under various conditions and compare them to a state-of-the-art kNN-based implementation [17]. All experiments are performed on a standard laptop with a 2.26 GHz Intel(R) Core(TM)2 Quad CPU and 4 GB of RAM, where only one core is used for the computations. An open-source implementation of our approach is available in the Point Cloud Library (PCL)[1] [18].

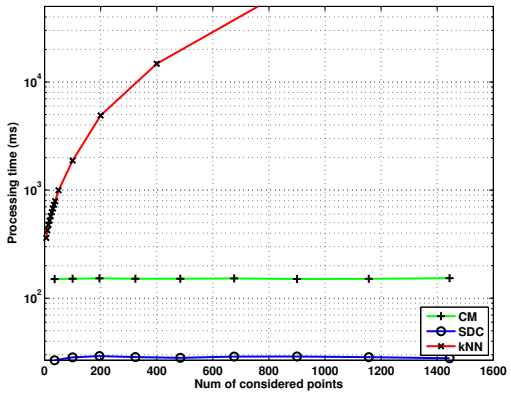


Fig. 4. Processing time of different normal estimation methods with respect to the size of the smoothing area. Note that the processing time is given in log-scale.

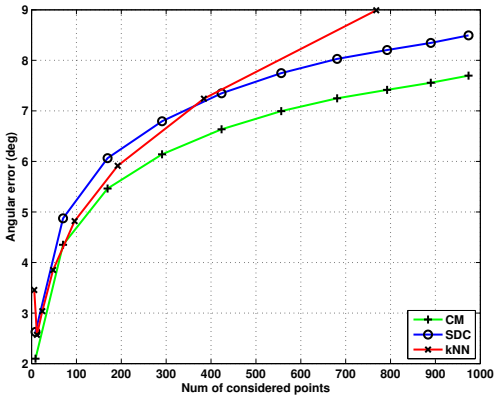


Fig. 5. Normal estimation error for different normal estimation methods with respect to the size of the smoothing area.

A. Processing Speed

Fig. 4 shows the processing time with respect to the number of points considered for smoothing for the approaches introduced in Sec. III-C (SDC), Sec. III-D (CM), and a state-of-the-art kNN-based approach (kNN) of Rusu *et al.* [17], which uses the k-nearest neighboring points for normal estimation. While the necessary processing time is constant for our approaches the processing time of *et al.* [17] increases with the number of considered neighboring points. The experiment is done using synthetic data by rendering a mesh model into a depth map of size 640×480 which is then converted into a point cloud of 307200 points. The average processing time for CM is approximately 151 ms and for SDC approximately 28 ms. Therefore, by using the SDC method we can obtain surface normals for full-resolution Kinect-data at around 35 Hz.

B. Normal Estimation Error

In Fig. 5 we compare different normal estimation methods with respect to the angular error between the ground truth

[1] http://www.pointclouds.org

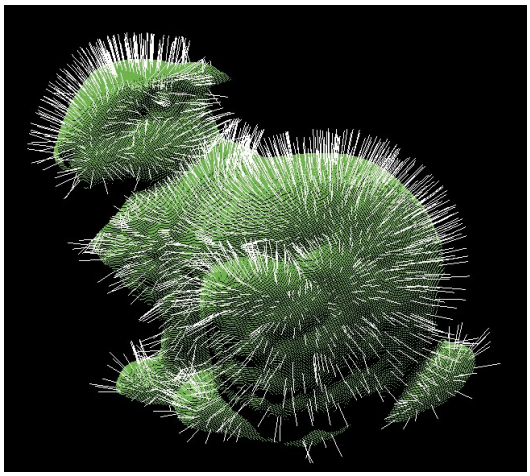


Fig. 6. Surface normals estimated from a partial view of the Stanford bunny.

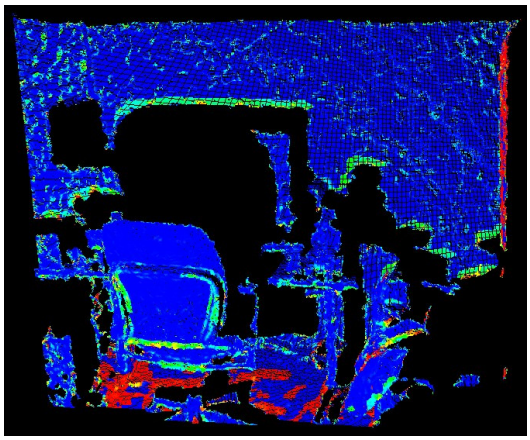


Fig. 7. Visualization of the difference between the normals estimated using our covariance matrix approach and the approach of [17]. For every point in the point cloud we visualize the dot-product between the normals estimated with the different methods, where blue color corresponds to a small and red corresponds to a large difference between the normals.

normal vectors and the estimated ones. As in Sec. IV-A this experiment is done using synthetic data without added noise. While the proposed methods (CM, SDC) behave similarly for different numbers of considered neighbors, the error obtained by the kNN-based approach increases faster when larger neighborhoods are used. An explanation for this is that the kNN-based approach uses a fixed number of neighbors independent of the presence of possible object borders. The proposed methods, on the other hand, adapt the number of considered points according to the specific depth as described in Sec. III-B.3.

Fig. 8 addresses the sensitivity to noise. As we see, our approach based on smoothed depth changes (SDC) has significantly better abilities to handle noise than our approach based on covariance matrices (CM) or the kNN-based approach, which both use the covariance matrix to estimate surface normals.

C. Qualitative Results

In Fig. 9 we show qualitative results of three different scenes. The first row shows the data which we get from a Kinect sensor, that is a color image and a depth map. The second row shows resulting normals that we get without depth-dependent smoothing (left) and with depth-dependent smoothing (right). The normals are color-coded, where each vector component is represented by a different color channel. It is easily visible that the depth-dependent smoothing helps to dramatically reduce the noise in the resulting normal vectors. Fig. 6 shows surface normals estimated from a point cloud of a partial view of the Stanford bunny. In Fig. 7 we visualize the difference between the normals obtained using the method of [17] and our covariance matrix based method.

V. Conclusion

In this paper we presented new methods for fast and robust estimation of surface normals from organized point cloud data. The use of integral images makes it possible to adapt the considered neighborhood size according to depth and object borders without any additional cost in terms of processing speed. We demonstrated that this way of normal estimation enables dense normal estimation at high frame rates and therefore, makes it possible to integrate surface normal information into vision based applications which need to run at a reasonable speed. However, the proposed approach shows two weaknesses: the lack of ability to compute normals for points very close to a depth change and the fact that it smoothes over edges where no depth change is present. Although not considered within this paper, the first problem can be easily addressed by using a different normal estimation method for close to a depth change. For the edge-smoothing problem, a two-pass approach can be applied where in the second pass areas with high variations in surface normal orientations are treated as object borders similar to high changes in depth.

References

[1] K. Klasing and D. Althoff and D. Wollherr and M. Buss, Comparison of surface normal estimation methods for range sensing applications, IEEE International Conference on Robotics and Automation (ICRA), 2009, Kobe, Japan.

[2] H. Gouraud, Continuous Shading of Curved Surfaces, IEEE Trans. Comput., June 1971.

[3] S. Jin and R. R. Lewis and D. West, A comparison of algorithms for vertex normal computation, The Visual Computer, 2005.

[4] N. Max, Weights for computing vertex normals from facet normals, J. Graph. Tools, March 1999.

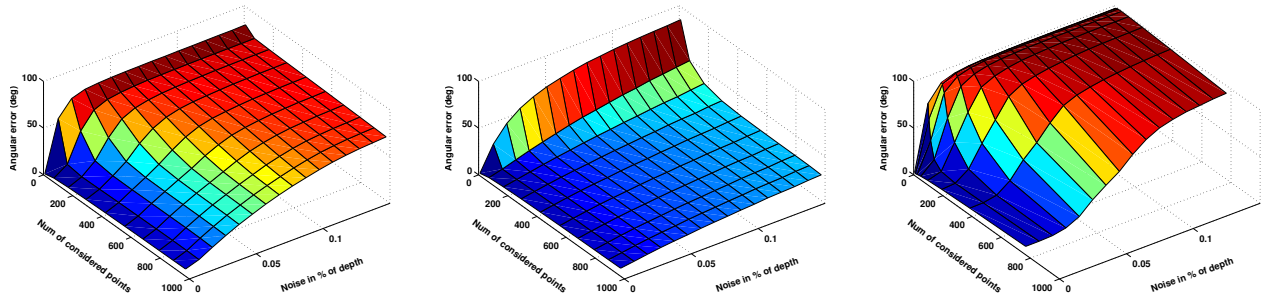
[5] G. Thürmer and C. A. Wüthrich, Computing vertex normals from polygonal facets, J. Graph. Tools, March 1998.

[6] D. Holz and S. Holzer and R. B. Rusu and S. Behnke, Real-Time Plane Segmentation using RGB-D Cameras, Proceedings of the 15th RoboCup International Symposium, July 2011, Istanbul, Turkey.

[7] 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, 2001.

[8] H. Hoppe and T. Derose and T. Duchamp and J. J. Mcdonald and W. Stuetzle, Surface reconstruction from unorganized points, Annual Conference on Computer Graphics, 1992.

[9] C. Wang and H. Tanahashi and H. Hirayu and Y. Niwa and K. Yamamoto, Comparison of Local Plane Fitting Methods for Range Data, Computer Vision and Pattern Recognition, 2001.



(a) CM (b) SDC (c) kNN

Fig. 8. Normal estimation error for different normal estimation methods with respect to the size of the smoothing area and the noise in the z-component.

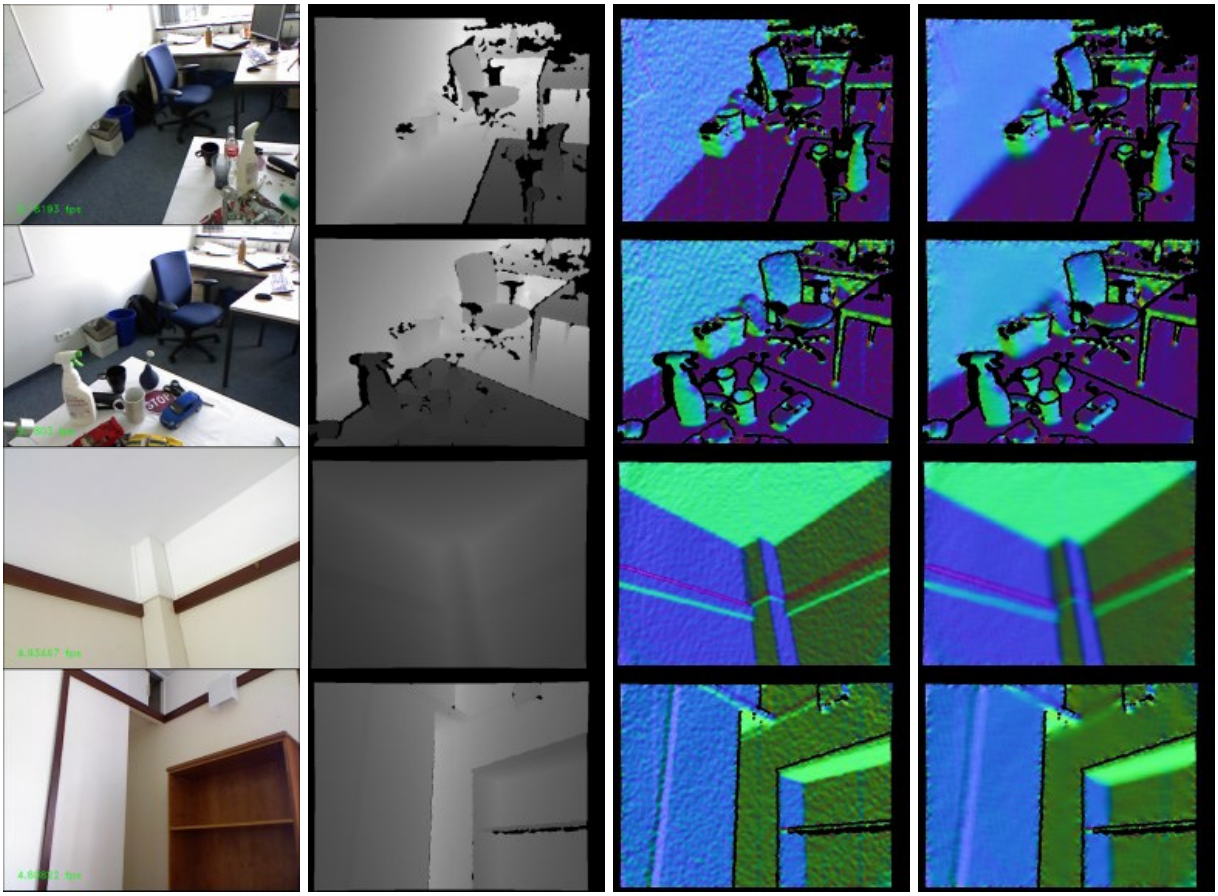


Fig. 9. Comparison of normal estimation with fixed and variable smoothing area size for real scenes. The first two columns show color images together with their corresponding depth maps. The third column shows normals estimated using a smoothing area of fixed size and the fourth column shows results using a variable smoothing size following our proposed approach. The normals are color-coded, where each vector component is represented by a different color channel.

[10] K. Kanatani, Statistical Optimization for Geometric Computation: Theory and Practice, 1996.

[11] M. Gopi and S. Krishnan and C. T. Silva, Surface Reconstruction Based on Lower Dimensional Localized Delaunay Triangulation, Computer Graphics Forum, 2000.

[12] M. Yang and E. Lee, Segmentation of measured point data using a parametric quadric surface approximation, Computer-aided Design, 1999.

[13] D. Ouyang and H. Feng, On the normal vector estimation for point cloud data from smooth surfaces, Computer-aided Design, 2005.

[14] M. Vanco, A Direct Approach for the Segmentation of Unorganized Points and Recognition of Simple Algebraic Surfaces, PhD thesis, University of Technology Chemnitz, 2003.

[15] G. Borgefors, Distance Transformations in Digital Images, Computer Vision, Graphics, and Image Processing, 1986.

[16] F. Porikli and O. Tuzel, Fast Construction of Covariance Matrices for Arbitrary Size Image Windows, IEEE International Conference on Image Processing, 2006.

[17] R. B. Rusu and Z. C. Marton and N. Blodow and M. Dolha and M. Beetz, Towards 3D Point cloud based object maps for household environments, Robot. Auton. Syst., 2008.

[18] R. B. Rusu and S. Cousins, 3D is here: Point Cloud Library (PCL), IEEE International Conference on Robotics and Automation (ICRA), 2011.