2011 IEEE International Conference on Robotics and Automation
Shanghai International Conference Center
May 9-13, 2011, Shanghai, China

Fast and Accurate Computation of Surface Normals from Range Images

H. Badino, D. Huber, Y. Park and T. Kanade

Abstract— **The fast and accurate computation of surface normals from a point cloud is a critical step for many 3D robotics and automotive problems, including terrain estimation, mapping, navigation, object segmentation, and object recognition. To obtain the tangent plane to the surface at a point, the traditional approach applies total least squares to its small neighborhood. However, least squares becomes computationally very expensive when applied to the millions of measurements per second that current range sensors can generate. We reformulate the traditional least squares solution to allow the fast computation of surface normals, and propose a new approach that obtains the normals by calculating the derivatives of the surface from a spherical range image. Furthermore, we show that the traditional least squares problem is very sensitive to range noise and must be normalized to obtain accurate results. Experimental results with synthetic and real data demonstrate that our proposed method is not only more efficient by up to two orders of magnitude, but provides better accuracy than the traditional least squares for practical neighborhood sizes.**

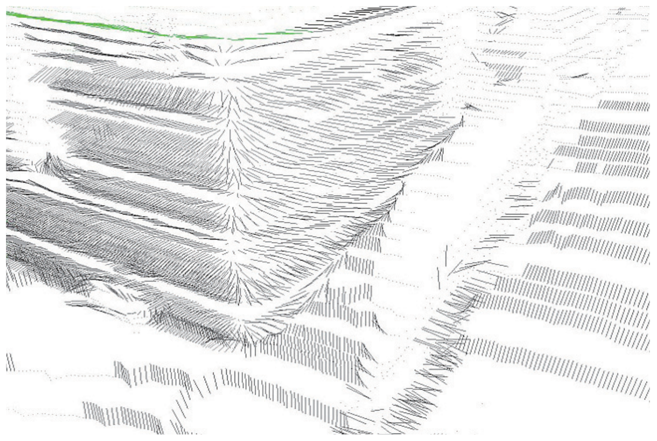


Fig. 1. Normals obtained from noisy sparse 3D data. The data corresponds to the marked region of the spherical range image shown in Figure 6c.

I. Introduction

For most robotics and automotive problems, computational power is a limited resource, which must be distributed wisely. Surprisingly, the efficient computation of normals from a point cloud is a topic that has had very little attention in the literature considering that surface normals are used for a large number of problems in the computer vision domain, such as terrain estimation [4], mapping [24], navigation [25], object segmentation [10], and object recognition [9].

In the robotics and automotive domain, the environment is usually measured as the sensors move small displacements between views, allowing the use of range images to store and organize the data [10]. In this paper, we use Spherical Range Images (SRI) to efficiently compute surface normals. Figure 1 shows an example of the extracted surface normals from noisy sparse data obtained from a 3D imager.

The most commonly used method for obtaining surface normals from point clouds is total linear least squares [11] (least squares, hereafter, for brevity), since it is relatively cheap to compute and easy to implement. However, least squares becomes computationally very expensive when applied to the millions of measurements per second that current range sensors such as stereo or high-definition LIDAR can generate.

In this paper, we make three main contributions towards the efficient and accurate computation of surface normals. First, we propose two new approaches for computing the normals with linear least squares. For this purpose, the traditional least squares loss function is reformulated in such a way, that box-filtering techniques can be applied to minimize the number of operations required. The new algorithms present speed up factors of up to two orders of magnitude over the traditional method. Second, we propose a novel method for solving the normals by computing derivatives of the surface defined by the spherical range image. The new method is not only computationally very efficient, but also more accurate than least squares for small window sizes. Third, we show that without a proper normalization, the traditional least squares fails to compute accurate normal estimates. This problem is solved by applying a non-isotropic scaling to the data matrix.

H. Badino, D Huber and T. Kanade are with the Robotics Institute, Carnegie Mellon University, Pittsburgh, PA, USA. Y. Park is with the Agency for Defence Development, Daejeon, Korea.

II. Related Work

The computation of surface normals has been widely addressed by surface reconstruction approaches, where the 3D points are usually unorganized. The surface can be reconstructed up to the first order with the normal vectors at the measured points. In [11], the computation of the tangent plane at each point with least squares was first proposed. The least squares solution was also addressed by [18], where the authors evaluated the effects of neighborhood size, measurement noise, and surface curvature when estimating planes with least squares and derived error bounds for the estimated normals. An alternative solution to the normal computation is based on the construction of Voronoi diagrams [1]. In this approach, the normals can be estimated from the center of large polar balls. Since the initial approaches to the normal estimation problem using polar balls assumed noise free data

[20], the method was adapted to the case of noisy point clouds [5], [7]. Both, the least squares and the polar balls methods were compared in accuracy and computation time in [6]. The experiments showed that the polar ball method is more robust to noise but that least squares is more accurate for low noise levels. It also showed that least squares is by far the fastest method for the computation of the normals.

In the robotics and automotive domain, the points are usually organized in graphs or range images, and the computation of surface normals focuses almost exclusively on least squares solutions [9], [10], [16], [21], [22], [24], [28]. Alternative methods for computing normals from organized point clouds rely on averaging the normals of adjacent triangles [2], [13], [19], [26], but the obtained normals are sensitive to noise and to the proper construction of the graph. Least squares and some of these averaging methods were compared in quality and computation time in [13], [21], [27]. In all these experiments, least squares shows the best results in terms of accuracy and speed.

III. Definitions

In this section, we define the variables, transformation equations, and the range image that are used in the next sections.

A. Spherical Coordinates

A coordinate vector in the spherical coordinate system is defined as $\boldsymbol{m} = (r, \theta, \phi)^T$, where r is the range, θ the azimuth, and ϕ the elevation component. The coordinates are constrained so that $r \geq 0$, $-\pi < \theta \leq \pi$, and $-\pi/2 < \phi \leq \pi/2$.

A point in Cartesian coordinates is represented by $\boldsymbol{p} = (x, y, z)^T$. The transformation from the spherical to the Cartesian coordinate system is given by

$$\boldsymbol{p} = r\boldsymbol{v} \tag{1}$$

where $\boldsymbol{v}$ is a unit direction vector defined as:

$$\boldsymbol{v} = \begin{bmatrix} \sin\theta\cos\phi \\ \sin\phi \\ \cos\theta\cos\phi \end{bmatrix} \tag{2}$$

The transformation from Cartesian to spherical coordinate system is given by

$$\boldsymbol{m} = \begin{bmatrix} r \\ \theta \\ \phi \end{bmatrix} = \begin{bmatrix} \sqrt{x^2+y^2+z^2} \\ \arctan(x/z) \\ \arcsin\left(y/\sqrt{x^2+y^2+z^2}\right) \end{bmatrix}. \tag{3}$$

Given a set of n measurements $\boldsymbol{m}_i$ of a scene observed from a single viewpoint, we seek to estimate the unit surface normals $\boldsymbol{n}_i$ at positions $\boldsymbol{p}_i$. The next sections are dedicated to solving this problem.

B. Spherical Range Images

A Spherical Range Image (SRI) is a function $s(\theta, \phi)$ in which the domain (θ, ϕ) represents the azimuth and elevation components, and the codomain $s()$ is a real value that represents a distance or range. The three components of the image – azimuth (column), elevation (row) and range (image value) – define the coordinate vector of a 3D point, i.e., $(\theta, \phi, r)^T$ with $r = s(\theta, \phi)$. In the actual implementation of the SRI, the domain is discretized and the codomain is approximated by a floating point variable. In this way, an SRI stores an image as observed from a single viewpoint, providing a 2.5D representation of the scene.

Range images are widely used for data processing and visualization purposes [4], [9], [10], [19], [27]. Figure 3 shows some examples of synthetic SRIs with their corresponding 3D model, and Figure 6c shows an example of a real SRI obtained from a Velodyne LIDAR.

C. Procedure

As noted by [20], most methods for computing normals follow three main steps: 1) identify neighbors, 2) estimate the normal vector using the neighbors, and 3) define the direction of the obtained normal. For the first step we use a small rectangular window of size $k = w \times h$ around the point to be estimated. The second step corresponds to the proper estimation of the normals using the extracted neighbors points. The next two sections deal with this step, and Section VI compares the accuracies and computation times. The third step is required to be able to identify the insides and outsides of complex objects. When estimating normals of surfaces observed from a single viewpoint, a globally consistent normal orientation is obtained by constraining the dot product $\boldsymbol{p}^T\boldsymbol{n}$ to be negative (i.e., if the obtained normal is such that $\boldsymbol{p}^T\boldsymbol{n} > 0$, then $\boldsymbol{n}$ is negated).

IV. Least Squares Approaches

This section presents three different least squares formulations to the surface normal estimation problem.

A. Traditional Total Least Squares

In the simplest case, least squares is formulated to find the plane parameters that optimally fit some small area of the surface [11], [18]. A plane is defined by the equation $n_x x + n_y y + n_z z - d = 0$, where $(x, y, z)^T$ lies on the plane and (n_x, n_y, n_z, d) are the sought plane parameters. Given a subset of k 3D points $\boldsymbol{p}_i$, $i = 1, 2, ..., k$ of the surface, least squares finds the optimal normal vector $\boldsymbol{n} = (n_x, n_y, n_z)^T$ and scalar d that minimizes

$$e = \sum_{i=1}^{k} \left(\boldsymbol{p}_i^T\boldsymbol{n} - d\right)^2 \quad \text{subject to} \quad |\boldsymbol{n}| = 1. \tag{4}$$

The closed form solution to Equation 4 for $\boldsymbol{n}$ is given by finding the eigenvector corresponding to the smallest eigenvalue of the sample covariance matrix $\boldsymbol{M} = 1/k \sum_{i=1}^{k} (\boldsymbol{p}_i - \bar{\boldsymbol{p}})(\boldsymbol{p}_i - \bar{\boldsymbol{p}})^T$ with $\bar{\boldsymbol{p}} = 1/k \sum_{i=1}^{k} \boldsymbol{p}_i$. The component d is found as the normal distance to the plane, i.e., $d = \bar{\boldsymbol{p}}^T\boldsymbol{n}$. This is the most widely used method in the literature [10], [11], [14], [16], [18], [21], [22], [23], [24], [28].

The computational complexity of the implementation of the above solution is linear in the number of neighbors k. The main problem with the above algorithm is that the computation of the matrix $\boldsymbol{M}$, and its corresponding eigen analysis, makes the algorithm extremely slow for practical image sizes. Table Ia shows the minimum number of operations required for this method, which we will call "traditional least squares."

B. Normalized Least Squares

The traditional least squares assumes independent identically distributed noise. Nevertheless, the noise is caused mainly due to range measurement error, which propagates linearly in the viewing direction $\boldsymbol{v}$ of the 3D vector $\boldsymbol{p}$ (see Equation 1). In consequence, obtaining the eigenvectors directly from $\boldsymbol{M}$ produces bad results. The solution to this problem is to normalize the coordinates of the 3D points so that the three principal moments of the $\boldsymbol{M}$ becomes all equal to unity while ensuring a good conditioning of the data matrix. This is analogous to the conditioning method proposed by Hartley [8], but in a different domain. Before computing the eigenvalues, the matrix $\boldsymbol{M}$ is normalized by Cholesky factorization, i.e.:

$$\boldsymbol{M'} = \boldsymbol{K}^{-1}\boldsymbol{M}\boldsymbol{K}^{-T} \tag{5}$$

where $\boldsymbol{K}$ is a lower triangular matrix such that $\sum_{i=1}^{k} \boldsymbol{p}_i\boldsymbol{p}_i^T = \boldsymbol{K}\boldsymbol{K}^T$. The eigenvectors are then obtained from the normalized matrix $\boldsymbol{M'}$. Table Ib shows the total number of operations required for the normalized least squares. As it can be seen, the Cholesky factorization does not depend on the number of neighbors k, but it adds a significant overhead to the traditional least squares.

(a) TRADITIONAL LEAST SQUARES

Operation	Mult.	Add.
Calculation of $\boldsymbol{p}_i$	3	0
Calculation of $\boldsymbol{M}$	$5 \times k + 7$	$9 \times (k-1) + 6$
Eigen analysis	66	38
Total	$5 \times k + 76$	$9 \times (k-1) + 44$

(b) NORMALIZED LEAST SQUARES

Operation	Mult.	Add.
Traditional LS	$5 \times k + 76$	$9 \times k + 35$
3x3 Cholesky Fact.	7	4
Inversion of $\boldsymbol{K}$	11	1
Scaling	18×4	18×4
Total	$5 \times k + 166$	$9 \times k + 112$

(c) UNCONSTRAINED LEAST SQUARES

Operation	Mult.	Add.
Calculation of $\boldsymbol{p}_i$	3	0
Product $\boldsymbol{p}_i\boldsymbol{p}_i^T$	6	0
Box-filt. for $\widetilde{\boldsymbol{M}}$	0	4×6
Box-filt. for $\tilde{\boldsymbol{b}}$	0	4×3
Inversion of $\widetilde{\boldsymbol{M}}$	24	10
Equation 7	9	6
Total	42	52

(d) FAST LEAST SQUARES

Operation	Mult.	Add.
Box-filtering for $\hat{\boldsymbol{b}}$	0	4×3
Calculation of $\boldsymbol{v}_i/r_i$	3	0
Equation 10	9	2×3
Total	12	18

(e) SRI DERIVATIVE METHOD

Operation	Mult.	Add.
Prewitt $k \times k$	$k-2$	k
Gauss 3×3	9	8
Equation 13	11	6
Total	$18 + k$	$14 + k$

TABLE I

NUMBER OF MULTIPLICATIONS AND ADDITIONS REQUIRED FOR ONE NORMAL COMPUTATION USING k NEIGHBOR POINTS.

C. Unconstrained Least Squares

We present now a computationally efficient solution that does not require eigen-analysis and the corresponding factorization. We first divide both parts of the loss function in Equation 4 by d^2 and eliminate the constraint on the normal vector to obtain:

$$\tilde{e} = \sum_{i=1}^{k} \left(\boldsymbol{p}_i^T\tilde{\boldsymbol{n}} - 1\right)^2 \tag{6}$$

where $\tilde{\boldsymbol{n}}$ is the sought normal vector, defined up to a scale factor. Note that this formulation degenerates for planes passing through the origin (for $d = 0$), but this situation cannot happen for range images. We will show in the experimental results that, although this formulation is not theoretically optimal, it is more accurate than the traditional least squares in practical real situations. The closed form solution for $\tilde{\boldsymbol{n}}$ is given by

$$\tilde{\boldsymbol{n}} = \widetilde{\boldsymbol{M}}^{-1}\tilde{\boldsymbol{b}} \tag{7}$$

where $\widetilde{\boldsymbol{M}} = \sum_{i=1}^{k} \boldsymbol{p}_i\boldsymbol{p}_i^T$ and $\tilde{\boldsymbol{b}} = \sum_{i=1}^{k} \boldsymbol{p}_i$. The final unit normal is found by normalization, i.e., $\boldsymbol{n} = \tilde{\boldsymbol{n}}/|\tilde{\boldsymbol{n}}|$.

The matrices $\widetilde{\boldsymbol{M}}$ and $\tilde{\boldsymbol{b}}$ can be computed efficiently. First, an image I that contains the outer products is built, i.e., $I(l,m) = \boldsymbol{p}_{l,m}\boldsymbol{p}_{l,m}^T$, where $\boldsymbol{p}_{l,m}$ is the 3D point corresponding to image location (l,m). Then, box-filtering is applied to obtain a new image with the sums of the outer products within the window (Fig. 2). The same procedure is applied to obtain an image of vectors for $\tilde{\boldsymbol{b}}$. Box-filtering is a well-known technique in computer vision [17]. This technique not only minimizes the number of operations required, but is also computationally independent of the window size.

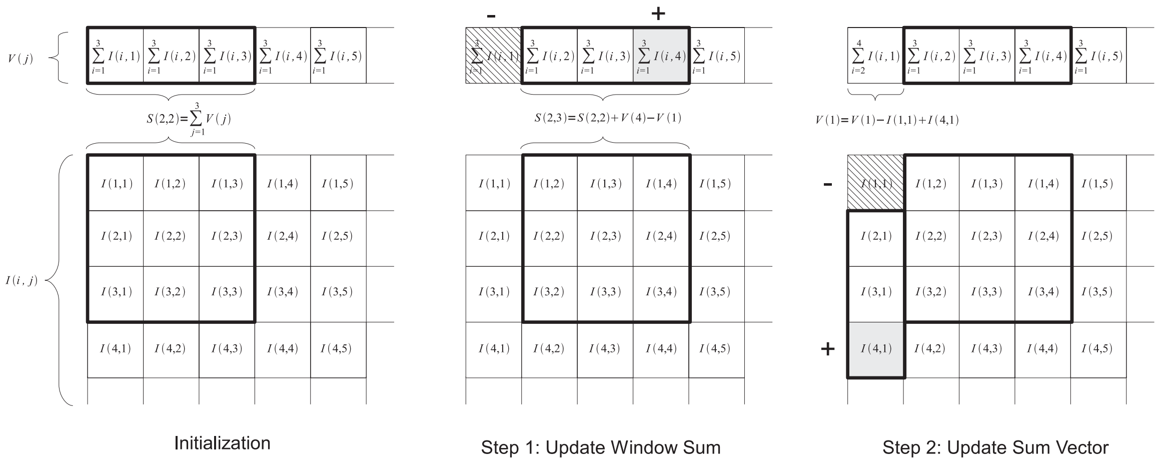
Most of the computational power required for the above method is spent in the summation and inversion of matrices, which, although independent of k, represents a considerable computational burden (see Table Ic)[1].

D. Fast Approximate Least Squares

In this section, we simplify the loss function to completely avoid the computation of the matrix $\widetilde{\boldsymbol{M}}^{-1}$ every time. By multiplying and dividing the right hand side of Equation 6 by r_i^2 we obtain:

$$\tilde{e} = \sum_{i=1}^{k} r_i^2 \left(\boldsymbol{v}_i^T\tilde{\boldsymbol{n}} - r_i^{-1}\right)^2 \tag{8}$$

[1]Observe that Cholesky factorization could be use to solve Eq. 7, but it would require more operations than the single matrix inversion of $\widetilde{\boldsymbol{M}}$.



$V(j)$
$S(2,2)=\sum_{j=1}^{3} V(j)$
$S(2,3)=S(2,2)+V(4)-V(1)$
$V(1)=V(1)-I(1,1)+I(4,1)$
$I(i,j)$
-
+
Initialization
Step 1: Update Window Sum
Step 2: Update Sum Vector

Fig. 2. Box-filtering technique. The box-filtering technique consists in three main steps. An initialization, where a vector of column sums is built and the initial sum within the window is calculated. After initialization, 2 steps are performed iterating from top to bottom and from left to right. The first step consists in updating the previous window sum by a subtraction and addition of vector sums. The second step consist in updating the vector sum that is not longer required in the current row by an addition and subtraction of original image components. Observe that, independent of the window size, two additions and two subtractions are required to compute the total sum within the window. There is an overhead in the initialization that is linearly dependent on k, but it becomes negligible for large image size n and small window size k. Observe that the codomain of the image can be scalars, vectors or matrices.

with $\boldsymbol{v}_i$ as defined in Equation 2. Since all points p_i are in a small neighborhood, all r_i are similar. Dropping the r_i^2 from the above equation leads us to the following approximate formulation of the loss function:

$$\hat{e} = \sum_{i=1}^{k} \left(\boldsymbol{v}_i^T \hat{\boldsymbol{n}} - r_i^{-1} \right)^2 \tag{9}$$

whose solution for $\hat{\boldsymbol{n}}$ is given by

$$\hat{\boldsymbol{n}} = \widehat{\boldsymbol{M}}^{-1} \hat{\boldsymbol{b}} \tag{10}$$

with $\widehat{\boldsymbol{M}} = \sum_{i=1}^{k} \boldsymbol{v}_i \boldsymbol{v}_i^T$ and $\hat{\boldsymbol{b}} = \sum_{i=1}^{k} \boldsymbol{v}_i / r_i$. In this new, approximate formulation, the matrix $\widehat{\boldsymbol{M}}^{-1}$ is independent of the ranges (i.e., the measurements), and depends only on the image parameters, so that it can be precomputed. The vector $\hat{\boldsymbol{b}}$ is obtained using the same box-filtering technique described above. This simplification greatly reduces the computational requirements and is independent of the window size k (see Table Id). Observe that this method works directly with spherical coordinates, not requiring the pre-computation of 3D points in Cartesian coordinates as in the three previous formulations. A similar formulation was presented in [9].

V. SRI Derivative Approach

We propose now a method, which obtains the normals by performing calculations directly in the spherical space. Instead of fitting a plane to obtain the normal vector, we compute the normal directly from the surface defined by the SRI.

A. Derivation of the *del* Operator

We will first demonstrate how transform the Cartesian *del* operator to spherical coordinates. In the next section, we will apply the resulting operator to the SRI to obtain the normal vector. The *del* operator in the Cartesian coordinate system is given by

$$\nabla \equiv \hat{\boldsymbol{x}} \frac{\partial}{\partial x} + \hat{\boldsymbol{y}} \frac{\partial}{\partial y} + \hat{\boldsymbol{z}} \frac{\partial}{\partial z} \tag{11}$$

where $\hat{\boldsymbol{x}}$, $\hat{\boldsymbol{y}}$, and $\hat{\boldsymbol{z}}$ are the unit vectors in the respective coordinate directions. Applying the chain rule to the partial derivatives in 11 gives

$$\begin{aligned}
\frac{\partial}{\partial x} &= \frac{\partial}{\partial r}\frac{\partial r}{\partial x} + \frac{\partial}{\partial \theta}\frac{\partial \theta}{\partial x} + \frac{\partial}{\partial \phi}\frac{\partial \phi}{\partial x} \\
\frac{\partial}{\partial y} &= \frac{\partial}{\partial r}\frac{\partial r}{\partial y} + \frac{\partial}{\partial \theta}\frac{\partial \theta}{\partial y} + \frac{\partial}{\partial \phi}\frac{\partial \phi}{\partial y} \\
\frac{\partial}{\partial z} &= \frac{\partial}{\partial r}\frac{\partial r}{\partial z} + \frac{\partial}{\partial \theta}\frac{\partial \theta}{\partial z} + \frac{\partial}{\partial \phi}\frac{\partial \phi}{\partial z}.
\end{aligned}$$

We apply first the above partial derivatives to Equation 3 and substitute the results into 11 to obtain:

$$\begin{aligned}
\nabla \equiv\ & \hat{\boldsymbol{x}} \left(\frac{\partial}{\partial r} \sin\theta\cos\phi + \frac{\partial}{\partial \theta} \frac{\cos\theta}{r\cos\phi} - \frac{\partial}{\partial \phi} \frac{\sin\theta\sin\phi}{r} \right) + \\
& \hat{\boldsymbol{y}} \left(\frac{\partial}{\partial r} \sin\phi + \frac{\partial}{\partial \phi} \frac{\cos\phi}{r} \right) + \\
& \hat{\boldsymbol{z}} \left(\frac{\partial}{\partial r} \cos\theta\cos\phi - \frac{\partial}{\partial \theta} \frac{\sin\theta}{r\cos\phi} - \frac{\partial}{\partial \phi} \frac{\cos\theta\sin\phi}{r} \right)
\end{aligned}$$

The expressions in parentheses define a rotation of a vector so that the previous equation can be expressed as

$$\nabla \equiv \begin{bmatrix} \hat{\boldsymbol{z}} & \hat{\boldsymbol{x}} & \hat{\boldsymbol{y}} \end{bmatrix} \boldsymbol{R}_{\theta,\phi} \begin{bmatrix} \partial/\partial r \\ \frac{1}{r\cos\phi}\,\partial/\partial\theta \\ \frac{1}{r}\,\partial/\partial\phi \end{bmatrix} \tag{12}$$

where

$$\boldsymbol{R}_{\theta,\phi} = \begin{bmatrix} \cos\theta & -\sin\theta & 0 \\ \sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\phi & 0 & -\sin\phi \\ 0 & 1 & 0 \\ \sin\phi & 0 & \cos\phi \end{bmatrix}.$$

Equation 12 expresses the Cartesian *del* operator as a function of variables and derivatives in spherical coordinates.

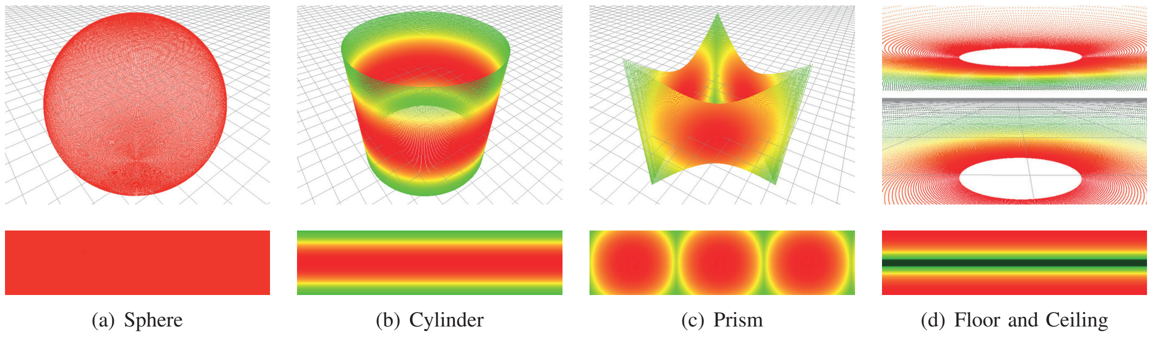


Fig. 3. Synthetic Data sets. The top images show the modeled object of each data set. The grids have a size of 2x2 meters. The sensor is located at the center of the objects and the color encodes the range. The bottom images show the Spherical Range Image for the corresponding top image.

B. Obtaining the Surface Normals

The SRI implies a functional dependence of the range on the azimuth and elevation components, i.e., $r = s(\theta, \phi)$. The normal vectors of a surface can be obtained by computing the derivatives of the function defining the surface. For this, we apply the del operator of Equation 12 to the function $s()$ to obtain

$$\boldsymbol{n} = \nabla s(\theta, \phi) = \hat{\boldsymbol{R}}_{\theta,\phi} \begin{bmatrix} 1 \\ \frac{1}{r\cos\phi}\partial r/\partial\theta \\ \frac{1}{r}\partial r/\partial\phi \end{bmatrix} \tag{13}$$

where $\hat{\boldsymbol{R}}_{\theta,\phi} = \left[\begin{array}{ccc} \hat{\boldsymbol{z}} & \hat{\boldsymbol{x}} & \hat{\boldsymbol{y}} \end{array}\right] \boldsymbol{R}_{\theta,\phi}$. Observe that $\hat{\boldsymbol{R}}_{\theta,\phi}$ is independent of the ranges and can be precomputed and stored in a look-up table.

The partial derivatives are obtained by applying standard image processing convolution kernels on the SRI, as is usually done for edge extraction in images [3]: the image is pre-filtered with a Gaussian 3×3 mask, and then a Prewitt operator is applied. The Prewitt computational complexity depends linearly on the number of neighbors k (see Table Ie), but it can be implemented very efficiently for practical window sizes, as we demonstrate in the next section.

VI. Experimental Results

We have conducted experiments with simulated and real data in order to evaluate the accuracy of each method under different types of objects and window sizes.

A. Synthetic Data

Figure 3 shows four synthetic SRIs with corresponding 3D models that were used for the evaluation. Four data sets were generated:

- Sphere: a centered sphere with a radius of 10 m.
- Cylinder: an open cylinder with planar radius of 10 m and height of 20 m.
- Prism: a uniform triangular open prism of 22 m maximal height.
- Floor and Ceiling: two parallel planes emulating a floor and a ceiling, each at a normal distance of 2 m from the sensor.

The Sphere data set covers the full field of view with an angular resolution of approximately $0.5°$ for the azimuth and elevation components forming a range image size of 750×375. The last three data sets cover a field of view of $360° \times 86°$ with the same angular resolution and an image size of 750×175. The lower part of Figure 3 shows the corresponding SRIs for each object. The parameter configuration was chosen to simulate a typical LIDAR sensor. The data sets are available online [12].

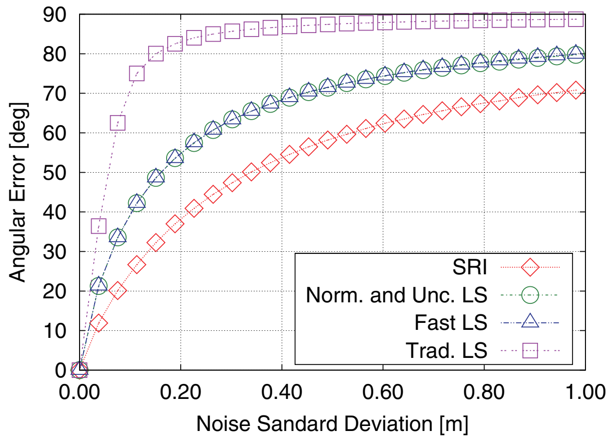
The traditional LS and the three new proposed methods were used to estimate the normals under different levels of noise and window sizes. The evaluation was performed obtaining the average angular error in the estimation of the normals. The angular error is defined as $e_i = \arccos(\boldsymbol{n}_i^T \bar{\boldsymbol{n}}_i)$ where $\boldsymbol{n}_i$ is the estimated normal and $\bar{\boldsymbol{n}}_i$ is the known truth normal to the surface at point $\boldsymbol{p}_i$. The average error in degrees is then $\frac{180}{\pi n}\sum_{j=1}^{n} e_i$. Furthermore, 30 trials were averaged to obtain the final error.

In the experiments, Gaussian noise with σ up to 1.0 meter was added to the SRIs. The magnitude of the noise might appear large for standard LIDAR sensors, but it is the typical level for distant points measured with stereo systems. Choosing a wide noise interval allows us to analyze the accuracy of the estimation at all levels of noise.

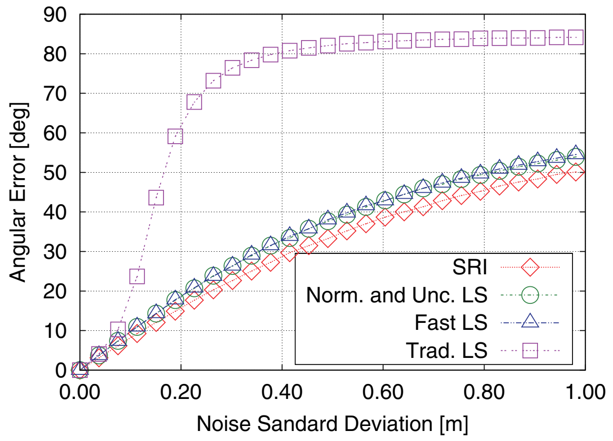
Figures 4a to 4d show the results when window sizes are varied from 3×3 to 9×9. The obtained results were consistent between all data sets and window sizes. Because of space limitations, only the most representative plot for each data set and window size is shown. The results for the normalized LS and the unconstrained LS were exactly equivalent. For readability, only one curve is shown for both approaches.

The most remarkable result obtained from the plots of Figure 4 is that the traditional least squares performs badly for all objects and window sizes. The normalized least squares provides better results by just normalizing the data matrix before the eigen-analysis, as addressed in Section IV-B. The fast least squares shows a very similar estimation error to the unconstrained LS, confirming that the elimination of the ranges from Equation 8 is reasonable. Only a very small difference can be observed between those two curves at very large noise levels. As expected, the difference is less marked on the Sphere data set, where the assumption of constant ranges within the mask does actually hold.

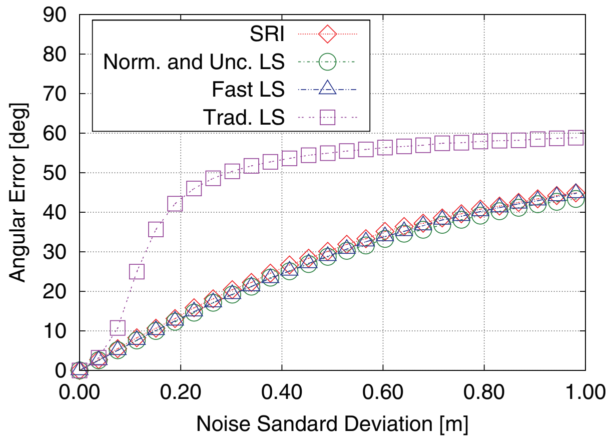
The SRI derivative approach performs better than all LS approaches for small window sizes. For large window sizes,



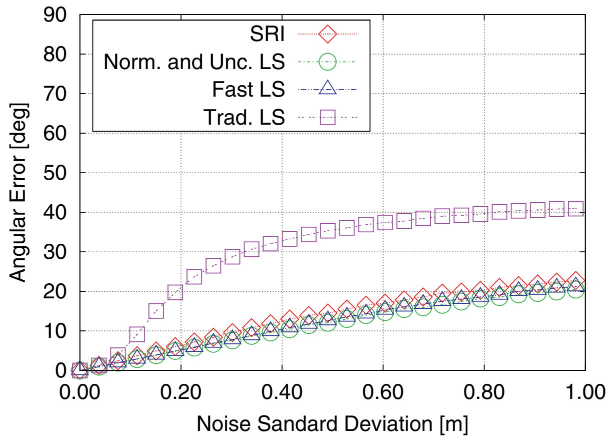
(a) Sphere 3×3



(b) Cylinder 5×5



(c) Prism 7×7



(d) Floor and ceiling 9×9

Fig. 4. Accuracy results with synthetic data for different window sizes.

the new proposed LS formulations perform better. The reason for this is that the derivative is a local property of the surface and will not be estimated with much greater accuracy as the window size increases. This leads to a limited improvement of the normal estimate when using larger window sizes. Least squares, on the other hand, fully benefits from more measurements, increasing the estimation accuracy.

Figure 5 shows the results for all methods, data sets and window sizes for the noise level $\sigma = 0.2$ m. The four charts show that the results for all five methods are consistent for all objects and window sizes.

As a general guideline, we can conclude that SRI derivative method is the best for small window sizes. For large window sizes the least squares methods produce better results at the expense of a higher computation time.

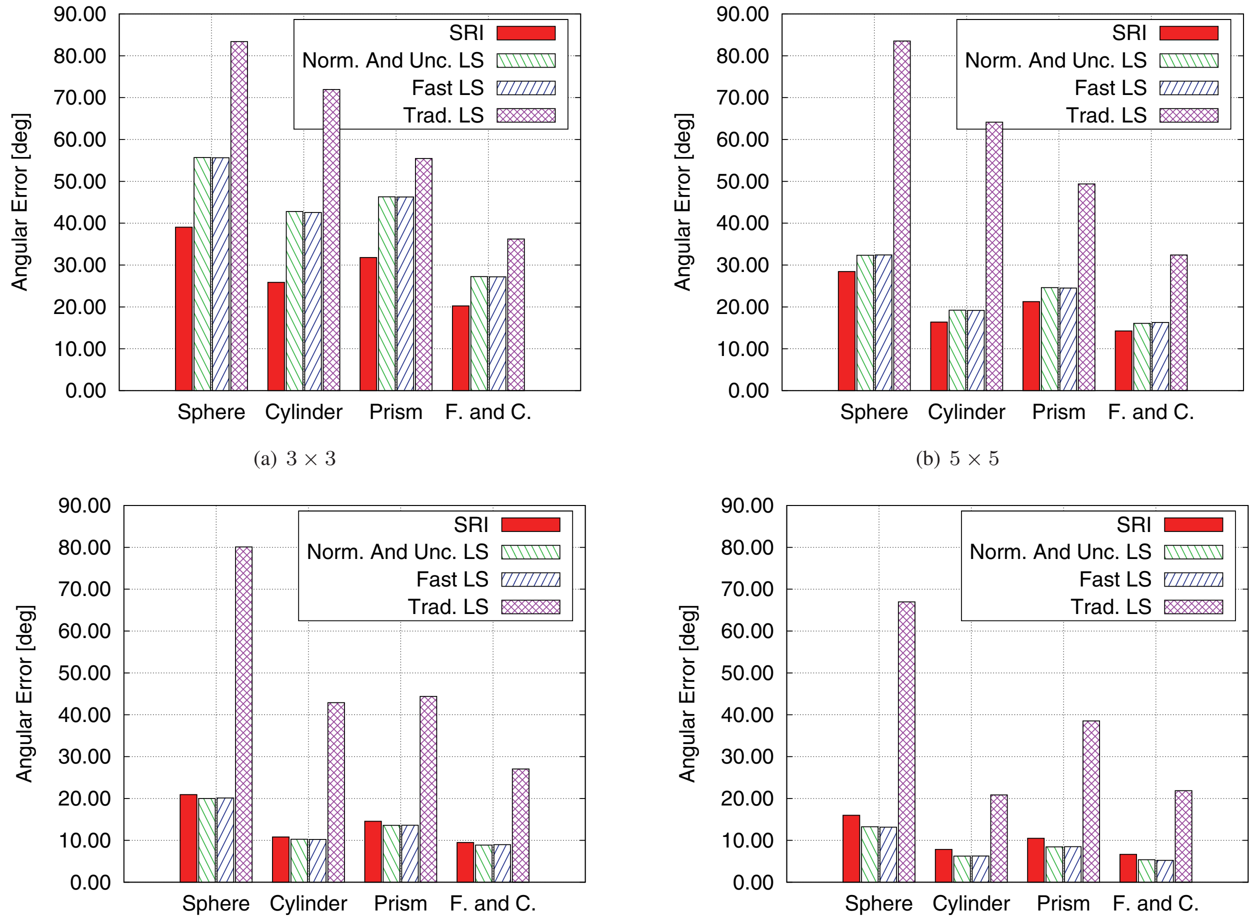
B. Computation Times

Table II shows the obtained computation times for each method. For the test, each algorithm was implemented in C++ using OpenMP and executed 50 times with the Cylinder data set on an Intel Core 2 Duo 2.66GHz CPU. The times are shown in milliseconds and they correspond to the average time for the whole range image. As predicted in Table I, the

M. Size	Methods	Time (ms) $\pm \sigma$	SUF
3×3	Trad. LS	192.25 ± 5.92	1
	Norm. LS	227.52 ± 4.04	0.85
	Unc. LS	18.87 ± 0.18	10.19
	Fast LS	7.02 ± 0.05	27.39
	SRI	3.66 ± 0.01	52.53
5×5	Trad. LS	195.00 ± 3.04	1
	Norm. LS	237.2 ± 4.04	0.82
	Unc. LS	19.14 ± 0.16	10.18
	Fast LS	7.04 ± 0.03	27.70
	SRI	$3.71 \pm < 0.01$	52.56
7×7	Trad. LS	436.12 ± 7.2	1
	Norm. LS	483.5 ± 3.1	0.9
	Unc. LS	19.14 ± 0.14	15.57
	Fast LS	7.07 ± 0.03	42.14
	SRI	4.33 ± 0.01	68.81
9×9	Trad. LS	436.12 ± 2.58	1
	Norm. LS	489.96 ± 4.8	0.89
	Unc. LS	19.35 ± 0.16	22.54
	Fast LS	7.18 ± 0.12	60.74
	SRI	4.40 ± 0.01	99.12

TABLE II

ACTUAL COMPUTATION TIMES WITH SPEED UP FACTORS (SUF) OVER THE TRADITIONAL LS.



(a) 3×3 (b) 5×5 (c) 7×7 (d) 9×9

Fig. 5. Estimation errors with the synthetic data sets for $\sigma = 0.2m$.

Method	*Angular Difference*
SRI & Unc. LS	$3.71° \pm 1.51°$
SRI & Fast LS	$3.75° \pm 1.51°$
Unc. & Fast LS	$0.39° \pm 0.11°$

TABLE III

AVERAGE ANGULAR DIFFERENCES FOR THE REAL DATA SEQUENCE.

computation time for the fast and unconstrained LS does not increase with the window size. Observe that according to Table I, the fast LS method should be faster than the SRI derivative method. However, Table I shows theoretical values of an optimal implementation and does not count for overhead in the form of indexing operations and memory accesses. The SRI derivative method is the fastest to compute and provides a speed up factor of up to two orders of magnitude with respect to the traditional least squares.

C. Results with Real Data

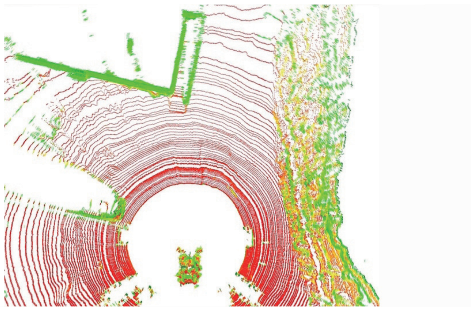
We have also conducted experiments with real data obtained from a Velodyne HDL-64E High Definition LIDAR scanner [15] mounted on the top of a vehicle. An SRI is defined with the sensor specifications, and the sensor measurements are registered in the range image as they are acquired. The registration of the sensor measurements in the SRI leads to a sparse image, since not every image position is occupied with a measurement. A linear interpolation step is used to fill those holes. To avoid interpolating image locations at depth discontinuities, an empty image position is only filled if the ranges to be interpolated are proportional up to a scale. An example of the resulting SRI for a real scenario is shown in Figure 6c.

All methods were tested with 6 sequences of data acquired on different scenarios. Figure 1 shows a snapshot of the normals obtained with the SRI derivative method on one of the sequences containing more than 23 million measurements. The sequence was acquired as the vehicle was traveling in rough terrain containing some human-made structures. Table III shows the average angular differences between the proposed methods for the whole sequence, from where it can be seen that all three methods provide consistent normal estimates.

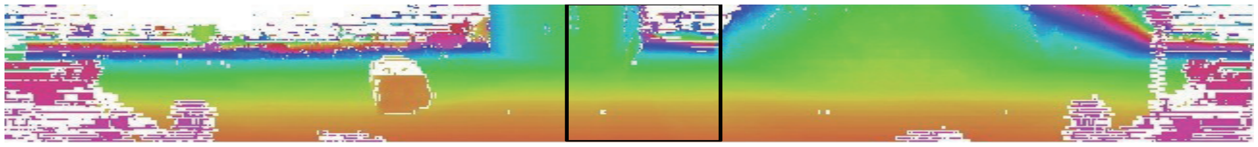
VII. CONCLUSIONS

Based on the experiments performed in this paper we can draw the following conclusions when computing surface normals with range images. First, the traditional least

(a) Visual snapshot



(b) Bird's eye view. red: vertical, green: horizontal.



(c) Spherical range image: the color encodes the range. The size of the image is 3300×100 covering a field of view of $360^\circ \times 30^\circ$ with angular resolution of 0.11×0.2 deg/px. The rectangle corresponds to the snapshot of Figure (a) and the normal estimates of Figure 1.

Fig. 6. Experiment results with real data

squares is very sensitive to noise and does not perform accurately without a proper normalization. This problem is easily solved by an appropriate scaling of the data matrix. Most remarkably, the literature on normal estimation usually does not specify if such a normalization step is performed, even though this is a critical step for the accurate estimation of surface normals. Second, the proposed unconstrained least squares and the fast approximate least squares show the same accuracy level as the normalized version, while being up to 17 times faster. Third and last, the new proposed SRI derivative approach is the fastest and more accurate method for small window sizes. It is also clearly the best option for large window sizes when time constraints prevail over accuracy requirements.

References

[1] N. Amenta and M. Bern. Surface reconstruction by Voronoi filtering. *Discrete and Computational Geometry*, 22:481–504, 1999.

[2] M. Bosse and R. Zlot. Map matching and data association for large-scale two-dimensional laser scan-based SLAM. *International Journal of Robotics Research*, 27(6):667–691, 2008.

[3] J. Canny. A computational approach to edge detection. *Transactions on Pattern Analysis and Machine Intelligence*, 8(6):679–698, 1986.

[4] C. Castejón, B. L. Boada, D. Blanco, and L. Moreno. Traversable region modeling for outdoor navigation. *Journal of Intelligent and Robotic Systems*, 43(2-4):175–216, 2005.

[5] T. K. Dey and S. Goswami. Provable surface reconstruction from noisy samples. *Computational Geometry: Theory and Applications*, 35(1):124–141, 2006.

[6] T. K. Dey, G. Li, and J. Sun. Normal estimation for point clouds: a comparison study for a Voronoi based method. In *Point-Based Graphics, Eurographics/IEEE VGTC Symposium*, pages 39–46, 2005.

[7] T. K. Dey and J. Sun. Normal and feature approximations from noisy point clouds. In *Foundations of Software Technology and Theoretical Computer Science*, pages 21–32, 2006.

[8] R. I. Hartley. In defense of the 8-point algorithm. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 19:580–593, 1997.

[9] M. Hebert and T. Kanade. Outdoor scene analysis using range data. In *International Conference on Robotics and Automation*, pages 1426–1432, 1986.

[10] R. Hoffman and A. K. Jain. Segmentation and classification of range images. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 9:608–620, September 1987.

[11] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle. Surface reconstruction from unorganized points. *SIGGRAPH Comput. Graph.*, 26:71–78, July 1992.

[12] http://www.cs.cmu.edu/vmr/datasets/normals, 2011.

[13] K. Klasing, D. Althoff, D. Wollherr, and M. Buss. Comparison of surface normal estimation methods for range sensing applications. In *Conference on Robotics and Automation*, pages 1977–1982, 2009.

[14] L. Kobbelt and M. Botsch. A survey of point-based techniques in computer graphics. *Journal of Computer and Graphics*, 28(6):801–814, 2004.

[15] http://www.velodyne.com/lidar/products/manual/hdl-64e, 2011.

[16] Z. C. Marton, . B. Rusu, and M. Beetz. On fast surface reconstruction methods for large and noisy point clouds. In *International Conference on Robotics and Automation*, pages 2829–2834, 2009.

[17] M. J. McDonnell. Box-filtering techniques. *Computer Graphics and Image Processing*, 17(1):65–70, September 1981.

[18] N. J. Mitra, A. Nguyen, and L. Guibas. Estimating surface normals in noisy point cloud data. *International Journal of Computational Geometry & Applications*, 14(4 & 5):261–276, 2004.

[19] F. Moosmann, O. Pink, and C. Stiller. Segmentation of 3D lidar data in non-flat urban environments using a local convexity criterion. In *Intelligent Vehicles Symposium*, June 2009.

[20] D. OuYang and H. Y. Feng. On the normal vector estimation for point cloud data from smooth surfaces. *Computed Aided Design*, pages 1071–1079, 2005.

[21] K. Pathak, N. Vaskevicius, and A. Birk. Uncertainty analysis for optimum plane extraction from noisy 3D range-sensor point-clouds. *Intelligent Service Robotics*, 3(1):37–48, 2009.

[22] R. B. Rusu, N. Blodow, Z. C. M., and M. Beetz. Aligning point cloud views using persistent feature histograms. In *International Conference on Intelligent Robots and Systems*, 2008.

[23] O. Schall, A. Belyaev, and H.-P. Seidel. Robust filtering of noisy scattered point data. In M. Pauly and M. Zwicker, editors, *IEEE/Eurographics Symposium on Point-Based Graphics*, pages 71–77, Stony Brook, New York, USA, 2005.

[24] A. Segal, D. Haehnel, and S. Thrun. Generalized ICP. In *Robotics: Science and Systems*, Seattle, USA, June 2009.

[25] H. Sekkati and S. Negahdaripour. 3-D motion estimation for positioning from 2-d acoustic video imagery. In *Iberian conference on Pattern Recognition and Image Analysis*, pages 80–88, 2007.

[26] L. Spinello, R. Triebel, and R. Siegwart. Multimodal detection and tracking of pedestrians in urban environments with explicit ground plane extraction. In *International Conference on Intelligent Robots and Systems*, pages 1823–1829, 2008.

[27] C. Wang, H. Tanahashi, H. Hirayu, Y. Niwa, and K. Yamamoto. Comparison of local plane fitting methods for range data. *Conference on Computer Vision and Pattern Recognition*, 1:663–669, 2001.

[28] J. W. Weingarten, G. Gruener, and A. Dorf. Probabilistic plane fitting in 3D and an application to robotic mapping. In *International Conference on Robotics and Automation*, pages 927–932, 2004.