JACOBS
UNIVERSITY

School of Engineering and Science

Bachelor's Thesis

Fast and Accurate Computation of Surface Normals

Vladimir Komsiyski

May 2012

Supervisor: Prof. Dr. Andreas Nüchter

Abstract

Computation of surface normal vectors is a vital part of almost every visualization problem as well as object representation ones. Thus, fast and accurate estimation is needed in the area of artificial intelligence and robotics as all automotive problems such as robot navigation, terrain estimation, object segmentation and object recognition, rely on it. Normal vectors are widely used in 3D visualizations and computer simulations for lighting computation, bump mapping and other problems. Obtaining the surface normals of a point cloud is typically done by applying the total least squares method to a small neighborhood. However, when the number of points gets significantly large, this method becomes rather slow and another approach must be tackled. We introduce two new methods for normal estimation, both using spherical range images. The first one suggests obtaining the normals by calculating the derivatives of the image surface. The second one optimizes the least squares method using the spherical range images. We compare these methods in terms of accuracy, time performance, and robustness to range noise, especially when millions of points are present in the cloud and time is an issue.

Contents

Chapter 1

Introduction

1.1 Surface Normals and Point Clouds

A surface normal at a certain point is a vector perpendicular to the tangential plane to the surface at that point. Normals basically define the orientation of the surface at all points and specify surface's local texture.

Points are obtained by scanning the environment with a sensor or a set of sensors measuring the distances to the nearest objects in multiple directions. Range (the distance from the sensor to the points) can be stored as well as the coordinates of each point. A single scan can produce billions of points that can be used to reconstruct the environment and it is only natural to use these points in order to compute the corresponding normals to the surfaces of interest. Figure 1.1 shows a capture of a part of a typical point cloud.

Figure 1.1: A point cloud obtained by scanning a room with four people.

1.2 Scientific Contribution

For almost all automotive problems time performance is crucial and measures must be taken to minimize the computation time without loss of accuracy. Surface normals are needed in many of these problems problems, namely terrain mapping, navigation, object recognition, object segmentation.

Recently, more attention has been paid to efficient surface normals computation as the point clouds get bigger and sensors may produce millions of points per second [2]. As faster computation is required when huge amounts of data are present and real-time results are desired, a novel approach is required. A reasonable solution is offered by the SRI methods, which we present and implement. Its potential benefits for the problems, in which real-time computation of big amounts of data is required, are obvious.

1.3 Outline and Motivation for Research

We will use Spherical Range Images (SRI) for the computation of the normals. SRI encode the coordinates and the range of the points and store them in a 2D image, as described more thoroughly in A.3. An example of an SRI image is shown in Figure 1.2. Usually total linear least squares method is used for the normal computation. It is easy to implement and reasonably computationally cheap, but as the number of measurements grow to billions it becomes very expensive. Here we will introduce the SRI derivative approach [2] and compare it to the optimized for spherical range images least squares method.

Figure 1.2: An example SRI image. The 360 degree view allows to present the whole point cloud. The elevation range here is 100 degrees but can also be adjusted to show the whole span of the cloud or just a desired part. Note that in this image only ranges upto 10 m are shown in order for the image to be perceivable.

Chapter 2

State of the Art

2.1 Related Work

The computation of surface normals is closely related to the computation of surfaces from a point cloud. There are various ways for doing this. One of them is using Voronoi diagrams and obtaining the normals from the center of large polar balls [1, 14]. This approach has been adapted also for noisy point clouds [5].

Still, the problems of computation of surface planes and normals have been addressed mainly by using the least squares method [6]. Also, the effects of used neighborhood size, shape of the estimated surfaces, and noise on the results have been investigated and evaluated. Error bounds for the estimated normals have been derived when using this approach [9]. Comparison between the least squares and Voronoi diagrams methods has been made and it was revealed that the least squares is much faster and more accurate when there is little or none noise in the measurements, but the polar balls method is more robust to noise [4].

When addressing robotics and automotive problems and highly unorganized point clouds are considered, the points are usually presented in range images and the computations are usually done by using the least squares method [6, 8, 10]. Alternative approaches suggest organizing the points in triangular meshes and averaging the normals of adjacent triangles, but the results highly depend on the construction on the graph and are very sensitive to noise [7]. Some of these methods have been compared to the least squares and the results show that least squares approach is the fastest and most accurate [7, 10, 12].

2.2 SRI Derivative Approach

The SRI derivative method has been presented and tested against different versions and optimizations of the least squares method [2]. The results on simple synthetic shapes, such as sphere, cylinder and prism, are very thorough and show that the SRI derivative approach is to be the preferable, better performing one. Little investigation on real data has been done though, and also little noise and choice on neighborhood have been considered.

We will also compare the currently presented SRI derivative approach to the to be optimized with SRI images least squares method as it is the best one by far and will present the results

for big point clouds from real scans for both methods in terms of both speed and accuracy. Since actual normals for real scans are unknown and accuracy cannot be computed, synthetic point clouds will be used for the tests. Following the already accepted way of investigating surface normals computation methods, we will show that the SRI derivative method is faster on both synthetically generated point clouds and real data from 3D scans, and performing with acceptable average angular error.

Chapter 3

Methods for Normal Estimation

The two methods to be implemented are discussed here. They will be integrated in the 3D Toolkit and its tools. The goal is to achieve optimal performance on the provided point clouds as well as several synthetic point clouds representing implicit shapes, such as sphere and cube, and to make a comparison between the two methods.

The implementations will be exclusively in C++, starting with the SRI creation from the point clouds, proceeding with the actual computation of the normals and eventually visualizing the results. The first steps towards implementation have already been taken, namely obtaining the range images from point cloud data, as can be seen in Figure 1.2 and Figure A.1.

3.1 Total Least Squares Method

Widely investigated and used, the least squares method is relatively easy to derive and implement. The main idea is to find the best fitting plane through a small neighborhood of the point of interest and to take a vector perpendicular to it. The best fitting plane is determined by minimizing the squares of the distances from the points in interest to it. Given a point with coordinates $(x, y, z)^T$ lying on the plane and a set of k points in the neighborhood, least squares method yields a normal vector $\mathbf{n} = (n_x, n_y, n_z)^T$, fulfilling

$$e = \sum_{i=1}^{k} (\mathbf{p}_i^T \mathbf{n} - d)^2 \text{ is minimum subject to } |\mathbf{n}| = 1. \tag{3.1}$$

It must be noted that the best fitting plane is given by the equation:

$$n_x x + n_y y + n_z z - d = 0$$

The solution of Equation (3.1) for $\mathbf{n}$ is given by the smallest eigenvector of the matrix

$$\boldsymbol{M} = \frac{1}{k} \sum_{i=1}^{k} (\mathbf{p}_i - \overline{\mathbf{p}})(\mathbf{p}_i - \overline{\mathbf{p}})^T \text{ where } \overline{\mathbf{p}} = \frac{1}{k} \sum_{i=1}^{k} \mathbf{p}_i$$

This solution for the normal vector is linear on the number of neighboring points k taken into consideration. The main drawback is the computation of the matrix $\boldsymbol{M}$ and its smallest eigenvector, which performs in poor time when tackled with real-time data.

The main drawbacks of this method are the search of the k-nearest neighbors, the eigen-analysis, and the matrix operations (addition, inversion) and in this form it is extremely slow for practical image sizes. We will cope with the nearest neighbor search (NNS) using the Spherical Range Images, taking the neighboring pixels as best neighbor candidates. Thus, after having generated the SRI, the NNS is done in constant instead of linear time, yielding linear runtime for the normal calculation instead of quadratic.

Optimizations of the least squares method suggest Cholesky factorization of the matrix $\boldsymbol{M}$ before computing the eigenvalues, adding, however, significant overhead to the computation. Another approach would be to reduce the error function, thus resulting in faster implementations, but also less accurate.

An approximation [2] is presented and used here as it is much faster, although less accurate than the traditional approach. Dividing Equation (3.1) by d^2 yields a simplified loss function and further division by the squared range ρ_i^2 (considering that the neighborhood is small and therefore the ranges are similar and can be assumed equal - this is the key approximation made) yields:

$$e = \sum_{i=1}^{k} ((\rho^{-1}\mathbf{v}_i)^T \widehat{\mathbf{n}} - \rho_i^{-1})^2 \tag{3.2}$$

$$\mathbf{v}_i = \begin{bmatrix} \cos\theta_i \sin\phi_i \\ \sin\theta_i \sin\phi_i \\ \cos\phi_i \end{bmatrix} \tag{3.3}$$

This suggests a solution for $\widehat{\mathbf{n}}$ as:

$$\widehat{\mathbf{n}} = \widehat{\boldsymbol{M}}^{-1} b \text{ with } \widehat{\boldsymbol{M}} = \sum_{i=1}^{k} \mathbf{v}_i \mathbf{v}_i^T,\ b = \sum_{i=1}^{k} \frac{\mathbf{v}_i}{\rho_i} \tag{3.4}$$

This way the computation of eigenvalues is avoided and the matrix $\widehat{\boldsymbol{M}}$ can be precomputed for the desired image coordinates as it does not depend on the range. Also, this approximation works directly on spherical coordinates and no overhead from coordinate system conversion is present.

3.2 SRI Derivative Method

The tangential surface and therefore its normal vector can be obtained by simply taking the derivative of the surface function in the point of interest. Having into account that a SRI defines the relationship between the range and angular components of the scene, we can combine Equation (A.4) and Equation (A.5) in order to obtain

$$\mathbf{n} = \nabla\rho = \nabla s(\theta, \phi) \tag{3.5}$$

As a differentiation operator for spherical coordinates (defined in A.1) was needed, we derived the ∇ (*del*) operator for such a coordinate system as seen in A.2. The result is a vector $\mathbf{n}$ expressed directly in Cartesian corrdinates. Since s is basically the range as a function of the two angles (the coordinates of the pixel), its derivative with respect to ρ is 1. Therefore, using Equations (3.5) and (A.7)

$$\mathbf{n} = \begin{bmatrix} \cos\theta\sin\phi - \dfrac{\sin\theta}{\rho\sin\phi}\dfrac{\partial\rho}{\partial\theta} + \dfrac{\cos\theta\cos\phi}{\rho}\dfrac{\partial\rho}{\partial\phi} \\ \sin\theta\sin\phi - \dfrac{\cos\theta}{\rho\sin\phi}\dfrac{\partial\rho}{\partial\theta} + \dfrac{\sin\theta\cos\phi}{\rho}\dfrac{\partial\rho}{\partial\phi} \\ \cos\phi - \dfrac{\sin\phi}{\rho}\dfrac{\partial\rho}{\partial\phi} \end{bmatrix} \tag{3.6}$$

The computation of the derivatives is also done on the SRI using the modified Scharr operator, described in A.4. More weight is added to the derivative by multiplying it by $2f$ where f is as defined in Equation (4.1). This way the resultant resolution of the SRI is taken into account when computing the derivatives. The factor was chosen empirically.

Chapter 4

Implementation

4.1 SRI Generation

The major difficulty with both approaches is the choice of resolution of the SRI. To achieve consistency in the results, the same SRI for the each point cloud data set are used for the normal calculation via both SRI Derivative and the optimized LS method. For the SRI Derivative method the biggest source of error is the derivatives computation, and the choice of different image resolution will inevitably yield different derivatives.

In this work a dynamic estimation of the resolution for the SRI was implemented, taking a factor

$$f = round\left(\sqrt{\frac{N}{150000}}\right) \tag{4.1}$$

Here N is the total number of points present in the scan. After this estimation, resolution of 360×100 pixels is multiplied by f. Since all scans had angular spans of 360 degrees of azimuth and 100 degrees of elevation, a factor f effectively means that each pixel from the image corresponds to $1/f$ degrees of the field of view for both angular coordinates. Using larger factor can produce SRI with "gaps", or empty pixels, inside of them, resulting from the absence of a point in the cloud with the corresponding coordinates. Smaller factor on the other hand will produce a less accurate image.

After having precomputed the SRI, the derivatives computation and the NNS are done in constant time. The runtime of the both methods are still linear on the number of normals (which is the same as the number of pixels of SRI, namely $36000 \times f^2$), but the SRI Derivative method is noticeably faster than the LS approach, since instead of all matrix operations, two simple image operations are performed.

It must be noted that not all points from the point cloud have their normal vectors computed this way. For an N-big point cloud, the number of computed normals is approximately $0.24N$, which should be enough for estimation of the rest of the normals. Additionally, one could increase the factor used to produce more normals.

4.2 Matrix Operations

For all matrix operations needed for the optimized LS method the `newmat` C++ matrix library, already integrated in 3DTK, is used.

4.3 Nearest Neighbors Choice

For the LS method the k selected neighbors can be simply the neighboring pixels from the SRI. In this work $k = 4$ is used, taking the pixels on the left, right, above and below as nearest neighbors. Better accuracy could be achieved by taking more neighbors for the exchange of increasing the runtime.

For the SRI method suitable neighbors for the derivatives estimation must be chosen. How this is done here is explained in more detail in A.4.

4.4 Normal Visualization

Another problem faced here is the proper visualization of normal vectors.

One possible approach is to use the RGB color model to represent the three dimensions of the vector with each axis modeled by one of the three base colors and interpolating between them to obtain all possible directions in the 3D space. The resulting normal distribution for a spherical point cloud data (the corresponding SRI can be seen in Figure 4.1) is shown on Figure 4.2.

Another visualization approach is using OpenGL and 3DTK to produce an interactive 3D view of the normals. A capture of the output generated from the same data is shown on Figure 4.3. Another scene can be seen in detail on Figures 4.4 - 4.6.

Figure 4.1: A 360×100 degrees Spherical Range Image representing spherical point cloud data. As expected, there is only one grey value throughout the image, since all points of the sphere are at equal distance from the origin.

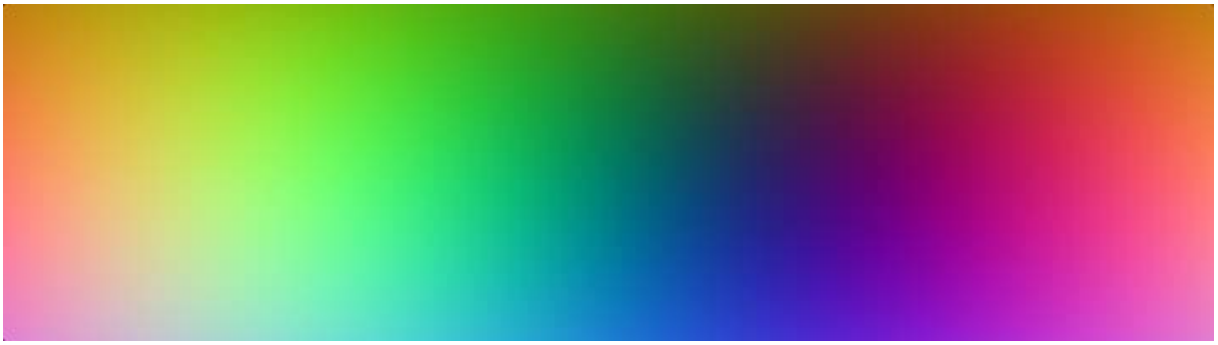


Figure 4.2: A 360×100 degrees image encoding the normal vectors generated from SRI on Figure 4.1 corresponding to spherical point cloud data.

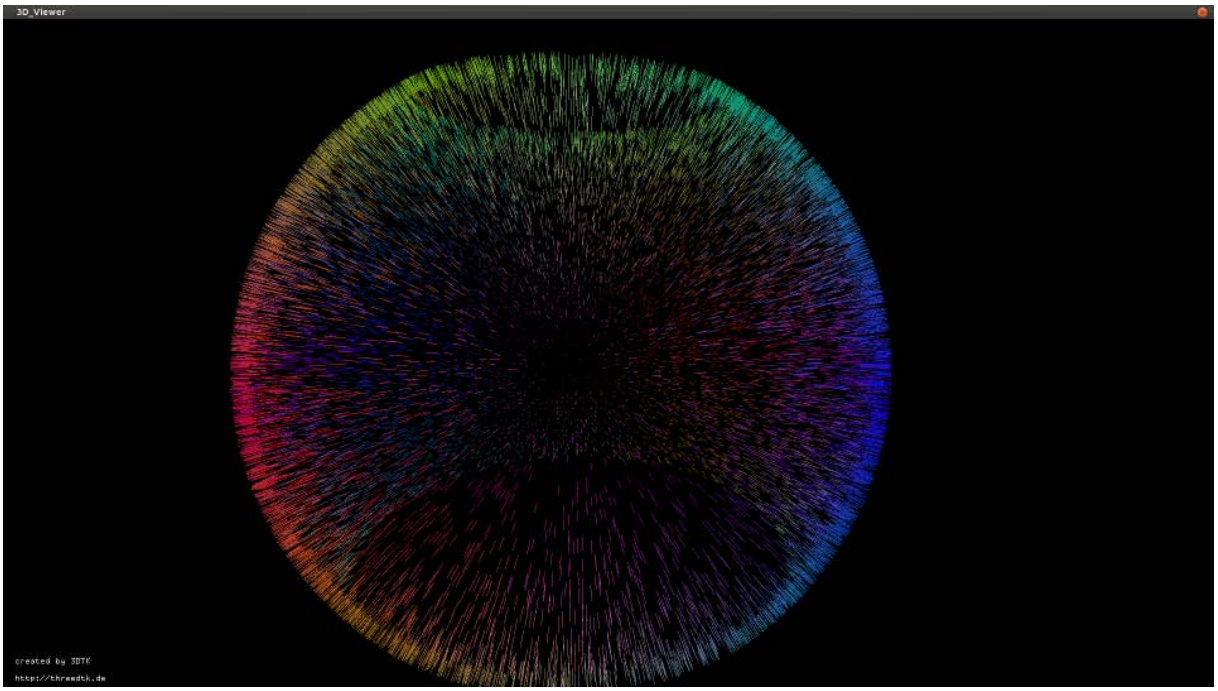


Figure 4.3: A 3D view of the normals of a sphere, also generated from the SRI on Figure 4.1. Note that here orientation of the normals does not affect the color, as it does on Figure 4.2 (meaning that the normal vectors $\mathbf{n}$ and $-\mathbf{n}$ have the same color). The 100 degrees elevation range can also be noticed.

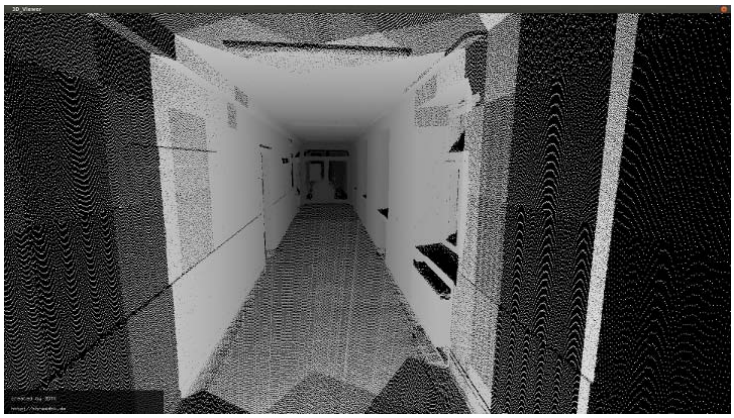


Figure 4.4: A capture of a point cloud obtained by scanning a corridor.

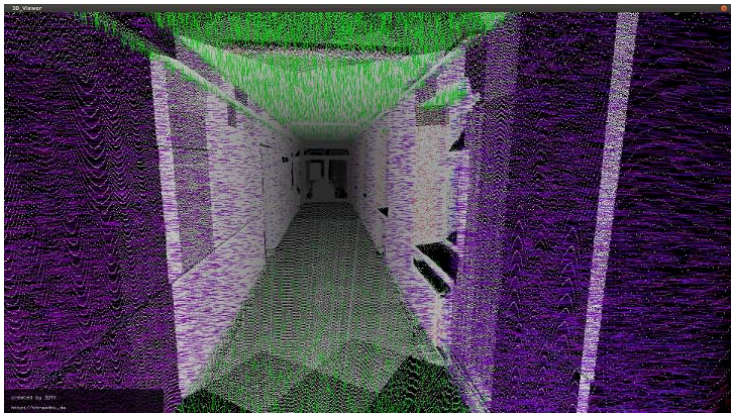


Figure 4.5: The point cloud from Figure 4.4 with the computed normal vectors.

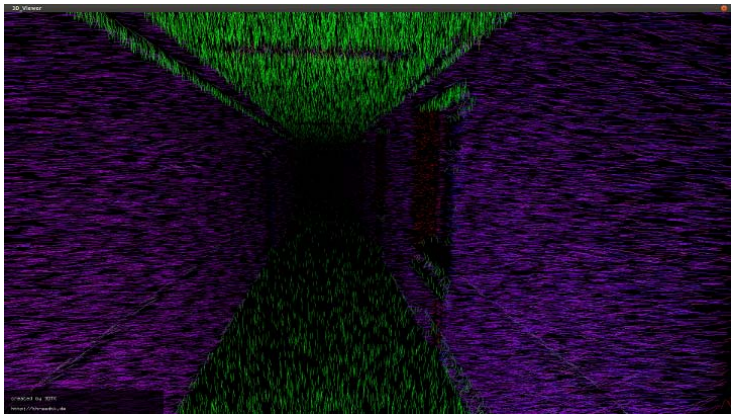


Figure 4.6: Only the normal vectors from Figure 4.5.

Chapter 5

Testing and Evaluation

Having implemented coordinate and image conversion, both normal calculation methods, and normal visualization, we conducted series of tests to evaluate the performance on real point-cloud data as well as on artificially created synthetic data consisting of implicit shapes such as sphere and cube. Also, different point cloud resolutions and window sizes will be investigated to ensure consistency.

The 3D Toolkit software was extended for our purposes, computing and visualizing the normal vectors directly on the point cloud, as well as generating a number of files helpful for evaluation - a SRI image, a normal distribution image, and a file storing each of the used points with its corresponding normal. The RXP data format was used exclusively for the real scans.

5.1 Accuracy

The error of the calculated normal is defined as the angle between it and the actual normal:

$$e = \arccos(\mathbf{n}_i^T \overline{\mathbf{n}}_i)$$

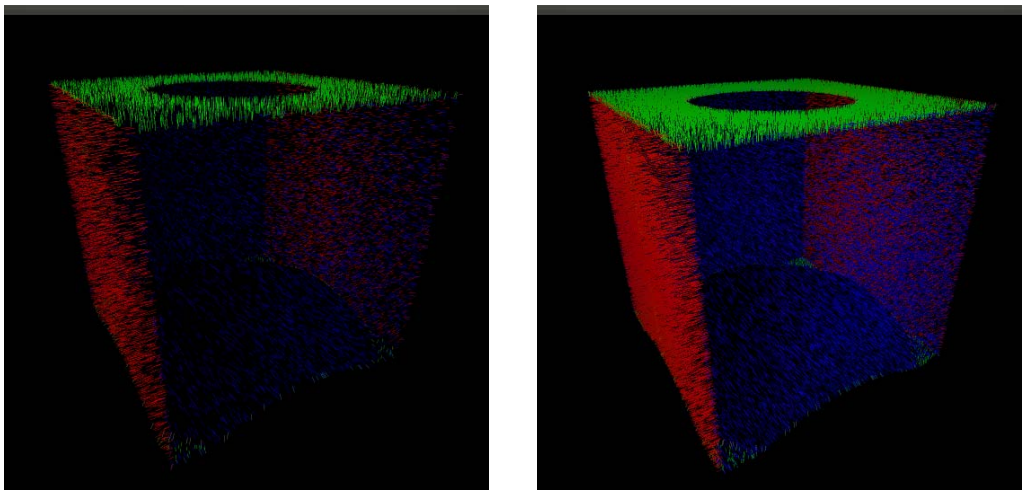
Here $\mathbf{n}_i$ is the calculated normal by any of the methods and $\overline{\mathbf{n}}_i$ is the known actual normal vector at a point p_i of the point cloud. The averaged error of a method on a single data set consisting of N points is then

$$\overline{e} = \frac{1}{n} \sum_{i=1}^{n} e_i$$

Since the normal vectors for the implicit shapes are known, accuracy of the methods can be easily measured. For obtaining difference images used for accuracy estimation a number of additional programs were used.

5.1.1 Synthetic Data

Cubical and spherical point clouds with different resolutions (number of points present in the cloud) were used for the tests. Example output for a cube with different resolutions can be seen and compared on Figure 5.1.



(a) **(b)**

Figure 5.1: 3D views of the normals of a cube. Again the orientation of the normals is not considered. The elevation range is again 100 degrees. Both images show 10% of the computed normals, whose number is 144000 for (a) and 576000 for (b).

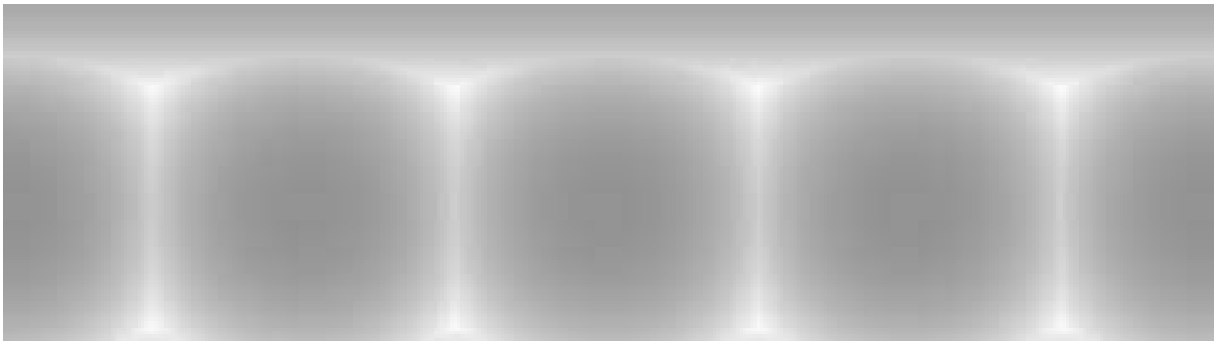


Figure 5.2: A 360×100 degrees Spherical Range Image representing cubical point cloud data.

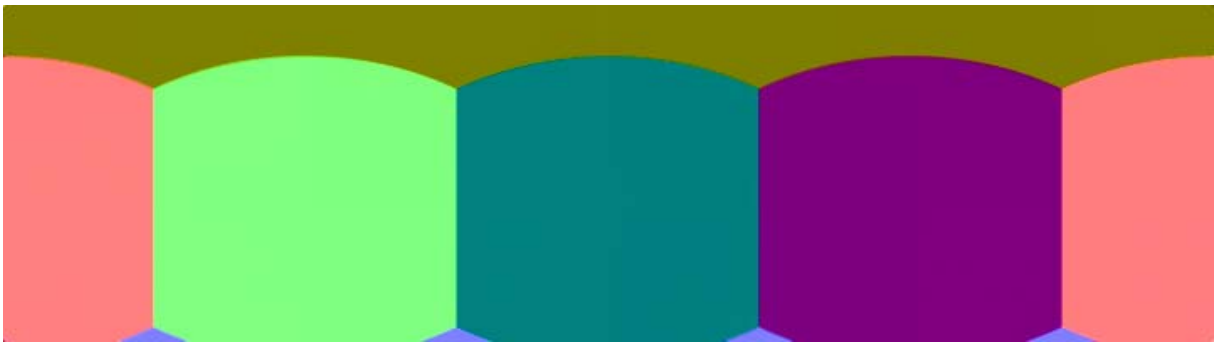


Figure 5.3: A 360×100 degrees image encoding the normal vectors generated from SRI on Figure 5.1 corresponding to cubical point cloud data.

For the spherical data, the output images can be seen on Figures 4.1 and 4.2. On Figures 5.2 and 5.3 similar are shown for the cubical data.

The output normals of the two methods were compared to the precomputed actual ones. Also, average angular difference of the normals computed via the two approaches was computed for the cubical point clouds.

For the spherical data sets both approaches performs very well, regardless of the resolution. This is easily explained by the fact that the derivatives computed on the SRI of the sphere are always 0, thus eliminating the major source of error for this method. Also, having a smooth spherical surface yields always correct neighbors for the LS computation (since they are taken from the image). Both approaches converge to an average angular error of 0.0046 ± 0.0007 degrees with average angular difference of 0.001 degrees between them, effectively meaning that not only do they produce small errors, but also the normals computed by each of them are practically the same. This verifies that both approaches calculate the normal vectors correctly for the spherical data sets.

For the cubical data sets the results are shown on Figure 5.4. It can be seen that LS method outperforms the SRI one, mostly because of the error with the derivative computation. Still, the average error of both approaches is significantly bigger than the one for the spherical data. This is due to the edges of the cube - there the derivatives are computed wrongly and as neighbors are taken points with much different ranges, which is in contrast with our initial assumption that the ranges of neighboring pixels are almost the same, the basis of the LS approximation. The average angular difference between the normals computed by the two methods is 2.72 ± 0.09 degrees, which is in the same range as the SRI errors. It can also be said that like with spherical data, here the resolution of the SRI does not really affect the accuracy.

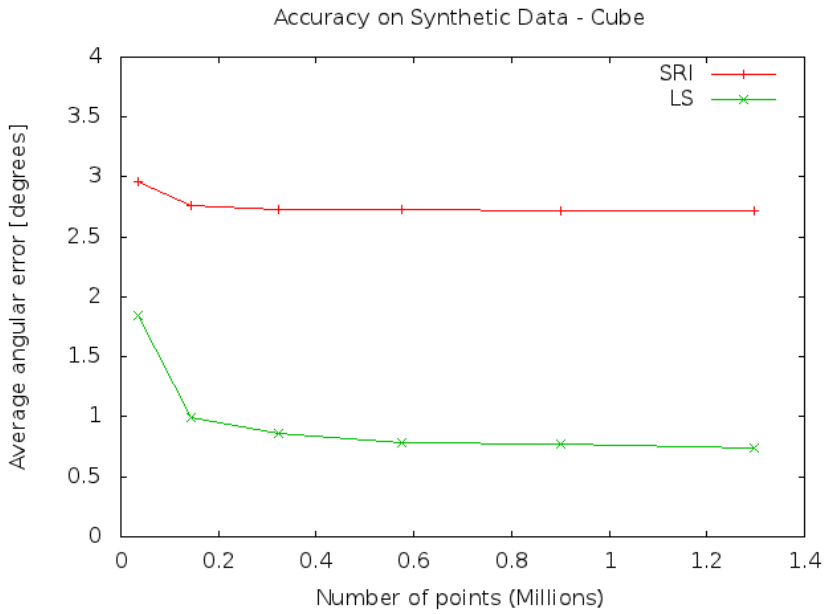


Figure 5.4: Accuracy for cubical point clouds.

5.1.2 Real Data

For testing with real scan data, 11 already taken scans of the campus of Jacobs University Bremen were used. They contained between 15.7 and 22.2 millions of points each, thus giving a sound idea of the performance on reasonably sized point clouds. In addition, 2 more scans were taken indoors - one with 773,758 points, and the other one with 5,416,894 points. Since the actual normal vectors of the scenes cannot be computed, only the average angular difference between the normal vectors obtained via the two methods was computed for each scan.

The indoor scans showed satisfying accuracy, giving average angular difference of 5.2 ± 1.5 degrees. The error comes mainly from the discontinuities (or edges) in the images as can be seen on Figures 5.6 and 5.8. This is due to the wrong computation of derivative in the SRI case and the nearest neighbor approximation in the LS case. Still, both approaches give similar output on continuous surfaces. Here again the accuracy does not depend on the resolution, although the actual error remains unknown and this the angular differences are calculated from different point clouds. Both approaches perform reasonably well, giving similar output and small differences, on continuous surfaces, produce noticeable errors on any discontinuities like edges, and therefore would produce larger average error on scans/images with many small objects and details. For plane detection though, the error is acceptable.

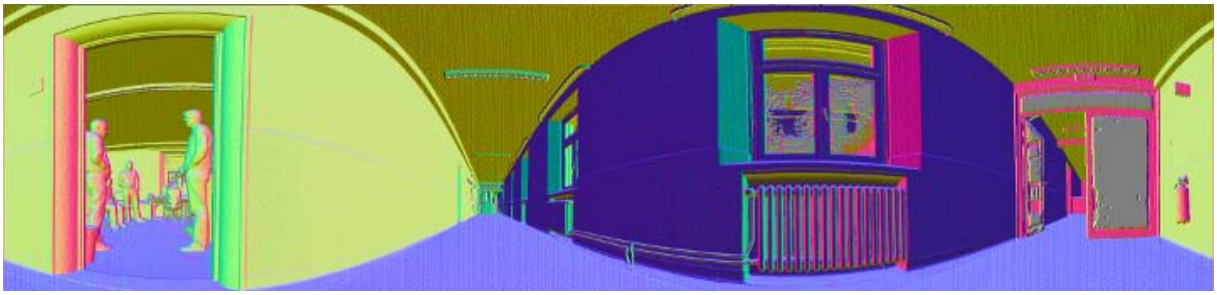


Figure 5.5: The normal vectors computed on the SRI from Figure 1.2 using the SRI Derivative method. The optimized LS method produces output with no noticeable differences.

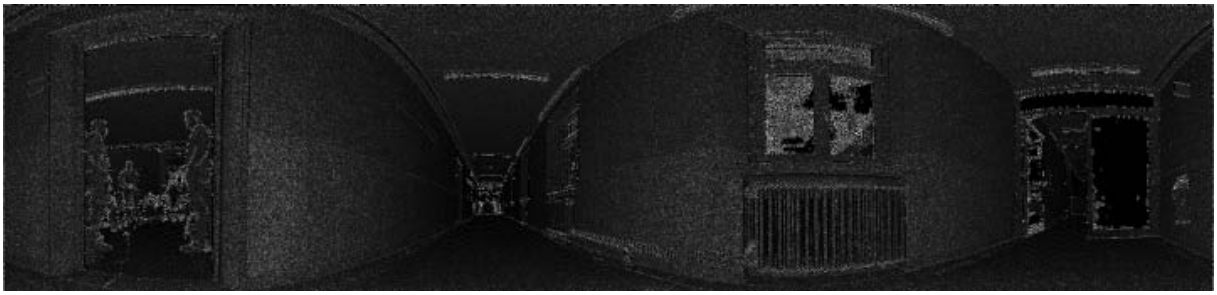


Figure 5.6: Angular differences between the two approaches for the data from Figures 1.2 and 5.5. Greater error corresponds to greater grey value and it can clearly be seen that the edges in the scene are the major error source.

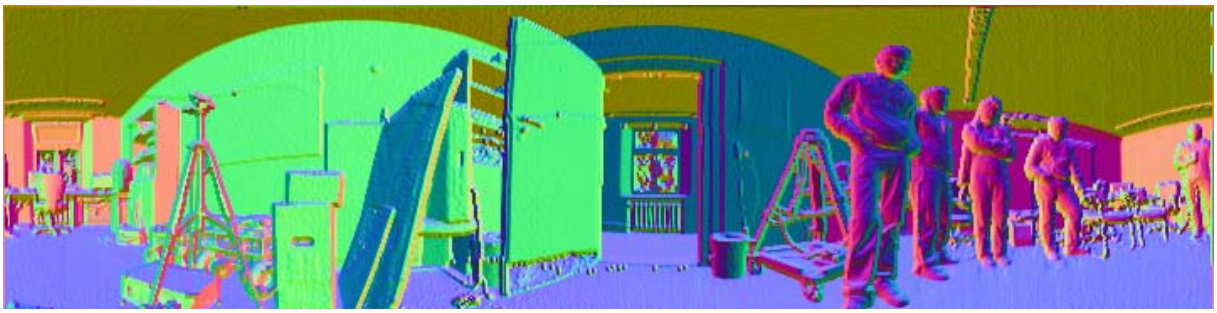


Figure 5.7: The normal vectors computed from the point cloud from Figure 1.1 using the optimized LS method.

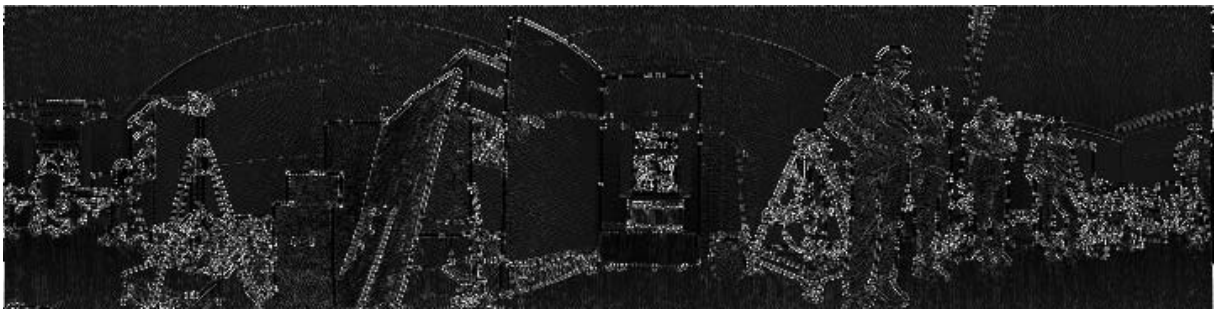


Figure 5.8: Angular differences between the two approaches for the data from Figures 1.1 and 5.7. The greater grey value on the edges here is clearer than on Figure 5.6 as the resolution of this scan is poorer and the errors are bigger.

For the outdoor scans (Figures 5.9, 5.10, 5.12, 5.13) the same observations hold. However, the accuracy dropped significantly to 9.5 ± 3.5 degrees. This big error is mostly because of the presence of many trees on the scans. The leaves contain much detail and their small sizes makes the normal estimation unreliable in the local neighborhood. The small sizes also suggest many closely put edges and therefore discontinuities on the SRI. Since the normal vectors of leaves of trees are practically insignificant for almost any problem, we removed the trees from the images as shown on Figures 5.11 and 5.15 and rerun the accuracy checks. The result was a improvement of the angular difference between the two methods of almost 45% reaching 5.6 ± 2.2 degrees coming close to the measured difference for the indoor scans. The slight raise is explainable by the fact that the scenes are slightly more detailed and contain more discontinuities. Also, the outdoor scans contain 3 - 4 times more points than the bigger indoor scan and 20 - 30 times more points than the smaller one.

With this we can conclude that the difference (assuming reasonably continuous objects on the scene) approaches an upper limit of around 6 degrees. This speculation is intended as a rough estimation of the final error of the computed normals and is needed because no other means of error calculation for real point cloud data are available.

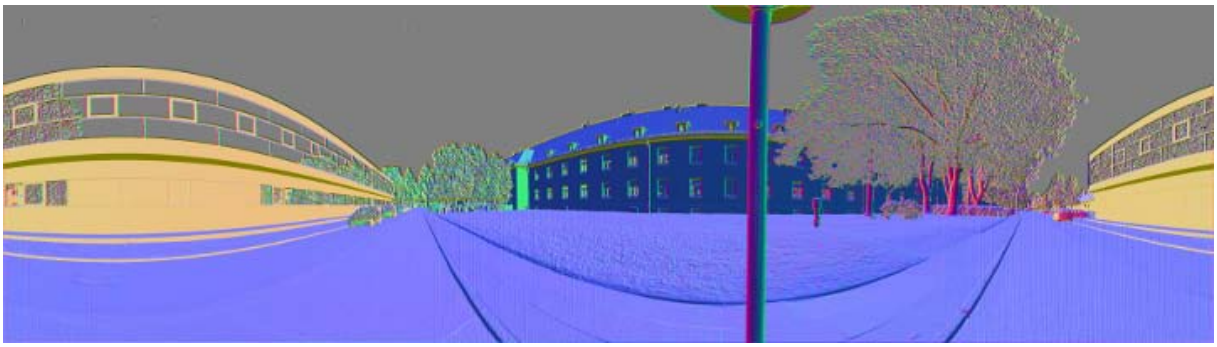


Figure 5.9: The normal vectors computed from a point cloud taken outdoors computed via the SRI Derivative method.

Figure 5.10: The difference image on Figure 5.9 and the output from the LS approach. Clearly, the biggest error comes from the trees and in this case - from the trees' reflections in the windows. This is a scanner imperfection.

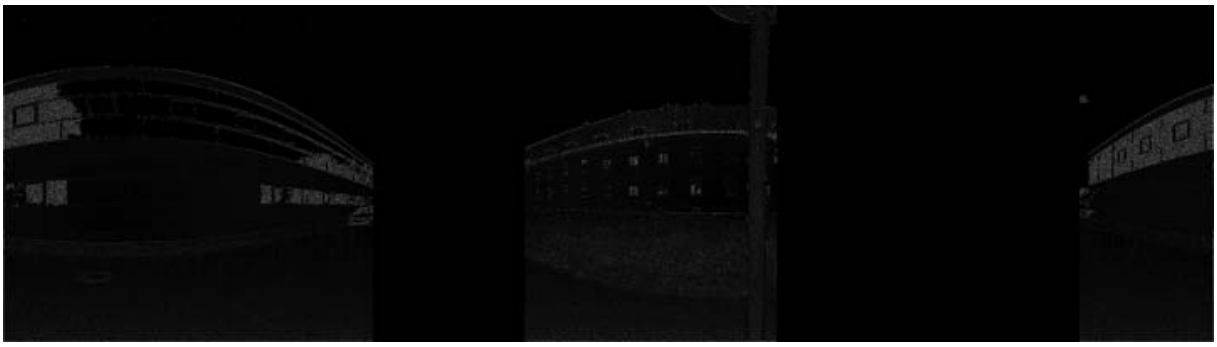


Figure 5.11: The image from Figure 5.10 with the tree areas removed. This results in reduction of the number of calculated normals of about 35%, but in exchange the accuracy can be better evaluated.

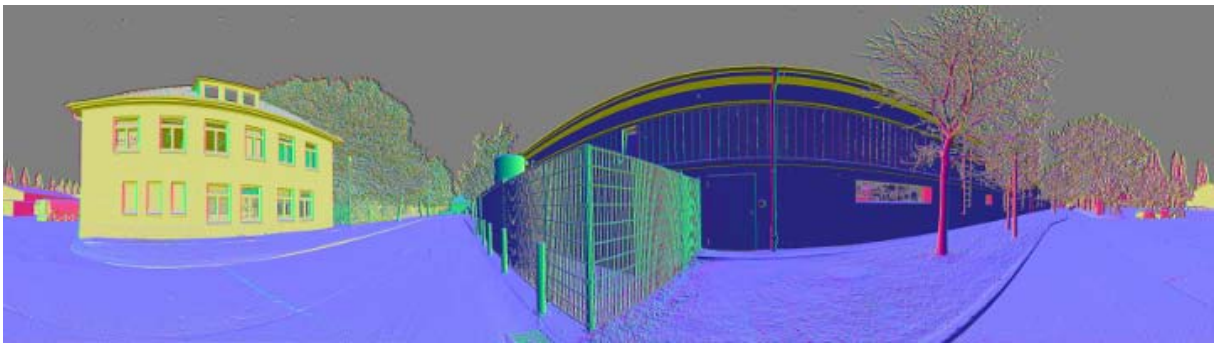


Figure 5.12: Another normal vector output from an outdoor scan. It is produced via the optimized LS method.

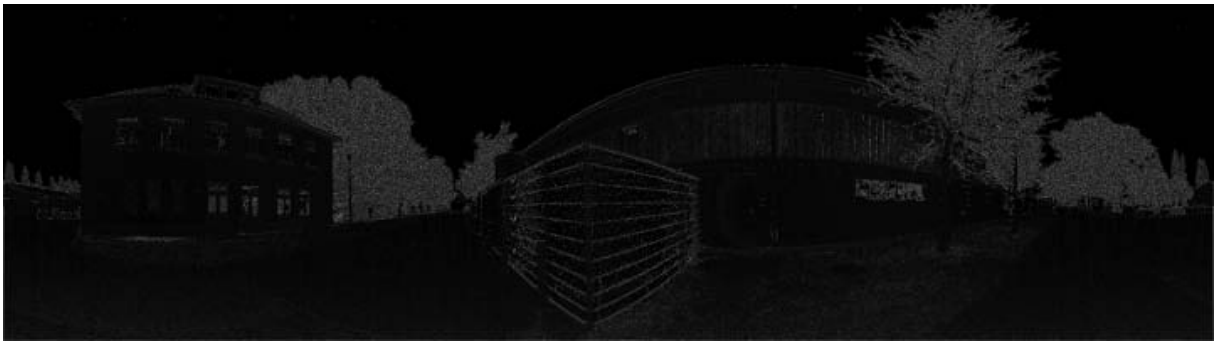


Figure 5.13: Angular differences between the two approaches for the data from Figures 5.12 and the same output computed via the SRI method. Again it can be seen that the image areas containing the trees have the greatest gray value and therefore the biggest angular difference.

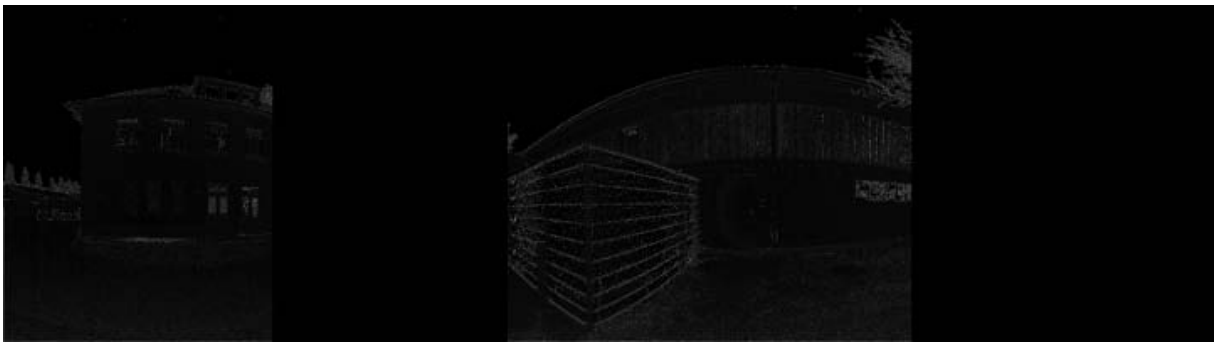


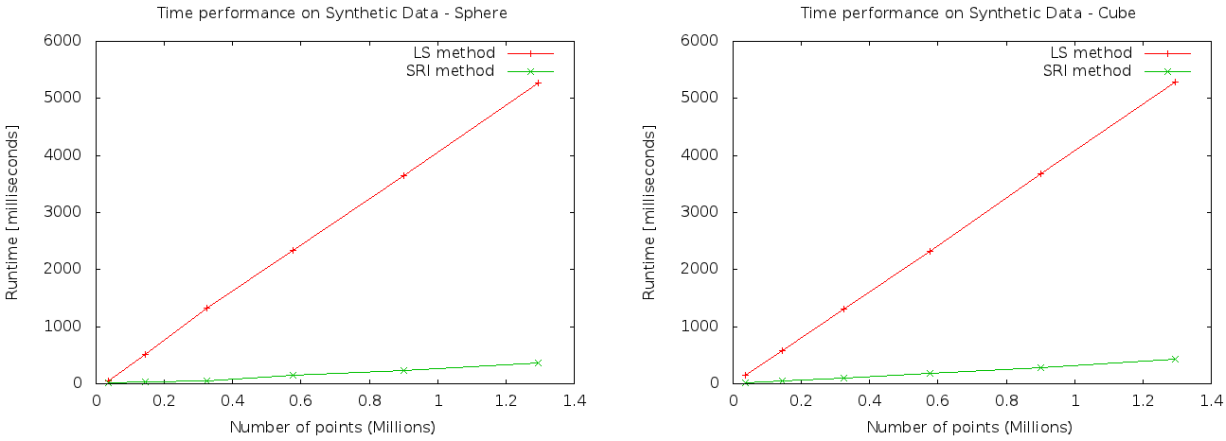
Figure 5.14: The image from 5.13 with the trees removed. About 45% of the data is ignored this way.

5.2 Runtime

The measurement of time performance of the algorithms is trivial. All tests were performed with a quad-core Intel Core i7 2675QM @ 2.2 GHz processor.

5.2.1 Synthetic Data

Again cubical and spherical point clouds with different resolutions were used. For each of the two methods and each resolution of both shapes the algorithms were run 10 times and the runtimes were averaged. The results are shown on Figure 5.15.



(a) **(b)**

Figure 5.15: Time performance of the two methods for spherical point clouds with different resolutions (a) and cubical ones (b).

The linear dependence on the number of points is obvious and expected. Both data sets run for almost identical times, so the nature of the point cloud and images does not affect the runtime.

From the collected data it can be shown that the SRI Derivative method performs in about 13 times less time than the optimized LS method on average. Considering that the latter is by far the fastest Least Squares method, the conclusion is that the newly investigated SRI Derivative approach outperforms any previously proposed normal estimation method significantly.

5.2.2 Real Data

The same data sets used for accuracy estimation were used to determine the time performance of the methods. Although scans with mostly the same number of points were used, the number of calculated normals differs among them, since the sky occupies a significant part of the SRI, therefore reducing the total number of points whose normals are to be computed. This allows for more measurements. The results are shown on Figure 5.16.

Again linearity is observed and the SRI Derivative approach outperforms the LS method 13 times and the slopes of the two functions are the same as the ones from the synthetic data. This

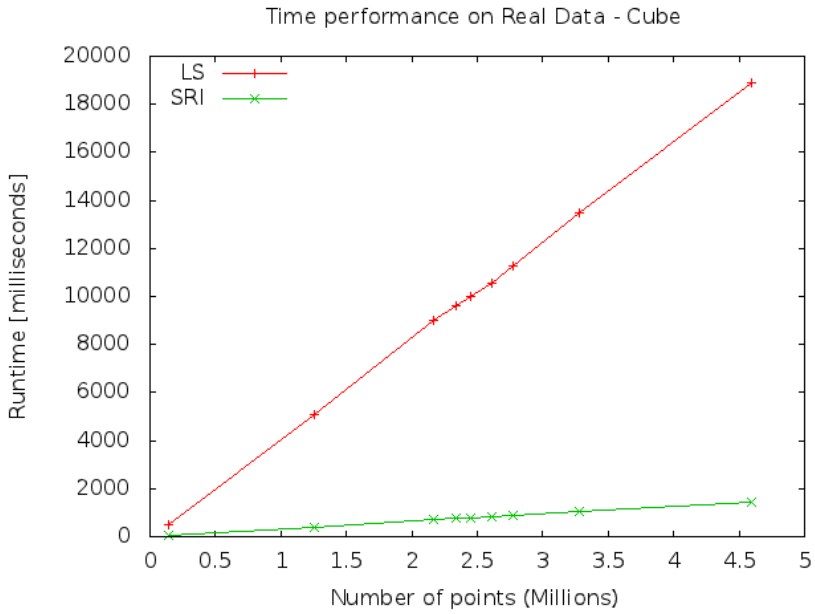


Figure 5.16: Time performance for real point cloud scans.

confirms that neither the normal distribution nor the point cloud resolution affects the runtime of either of the methods.

Chapter 6

Conclusions

In the process of investigating the problem of real-time accurate surface normal computation, tackled as point cloud data sets grow with the improvement of current scanning sensors, we aim in achieving better performance in terms of time and accuracy revisiting and improving existing and widely used methods as well as newly proposed ones. Upon successful completion of the tests we have two working models for estimating the normals and the means for distinguishing the benefits and drawbacks of each of them, thus providing different solutions for a variety of computer vision problems, optimization of lighting computation algorithms, terrain estimation, object recognition and segmentation.

Upon successful completion of the tests, two working methods for normal calculation that perform better than the previously investigated ones in terms of runtime are present. A clear distinction between them should be also available in order for one to be able to select the best suiting method for the problem in hand, depending on the specific needs and goals - either good time performance preferred over accuracy, or the other way around, or some trade-off between them. The proposed methods are easily extensible and easy to implement.

The optimized for Spherical Range Images Least Squares approach performs faster than any other LS implementation, reducing the runtime to linear by searching for and accessing nearest neighbors in constant time after the SRI generation. The computation of the image itself is also done in linear on the total number of points present in the scan time. The Spherical Range Image Derivative approach outperforms this method 13 times in terms of speed. We can assume that the latter is less accurate, based on the results on synthetic point cloud data representing a cube. Still, both approaches show satisfying accuracy when only an estimation of the normal vectors is needed and speed is crucial (as in real-time computations and scene generations). We can speculate on average angular error of the resulting normals of less than 10%, provided the scene contains reasonable number of edges and small details.

Further work on this topic may include processing of two or more different scans of the same scene, or just changing the point of origin for generation the SRI. This would allow for matching the calculated normals later and reduce the final error, sacrificing speed for accuracy this way. For a local neighborhood, the error can be reduced by taking a bigger resolution of the image representing the point cloud in the area of interest. Also, the error of the SRI Derivative method can be further reduced by taking a better estimate of the partial derivatives

at the points on the edges and adjusting them for the resolution of the problem in hand, for example also generating more SRIs on the same point cloud, but optimizing the ranges instead of the normals. This would allow for better derivative and nearest neighbor estimates and one can use more points whose normal vectors are computed. Again, this will inevitably increase the runtime, but depending on the tackled problem and its requirements it may be a suitable approach.

We extend the state of art by implementing newly proposed and improved old solutions to real-world data sets and test their consistency, reliability and robustness. The comparison between the proposed methods provides means of selection of the best possible method for the problem in hand, depending on its parameters - scale, time constraints and accuracy requirements.

Appendix A

Mathematical Background

A.1 Coordinates

A vector in Cartesian coordinate system is defined as a tuple $\mathbf{v} = (x, y, z)^T$ with the three real numbers representing the coordinates on the respective axes. In spherical coordinates instead of three axes, other three real number are used - two angles and the distance to the origin (range). The choice of notation differs among different works. We use the one suggested in [13] and shown on Figure A.1.

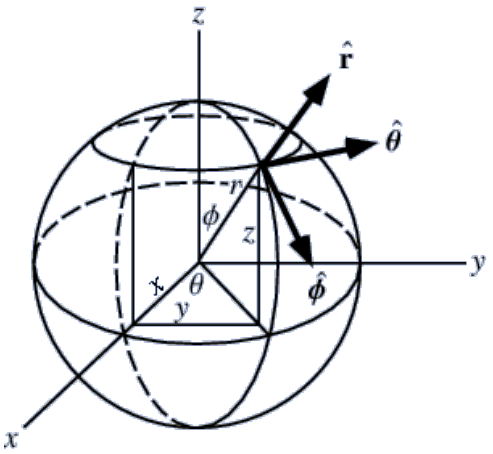


Figure A.1: Spherical Coordinates. Figure taken from [13].

Thus, a representation of a vector in spherical coordinates is $\mathbf{m} = (\rho, \theta, \phi)^T$, Here $\theta \in 0; 2\pi]$ is the azimuth or the angle in the plane of the observer between the projection of point of interest and the direction of the viewpoint of the observer. $\phi \in [0; \pi]$ is the elevation component, or the angle between the plane of the observer and the vector pointing to the point of interest. The range $\rho \geq 0$ is the distance from the origin to the point.

The transformation of a vector from spherical to Cartesian coordinates is done as shown in Equation (A.1) and the transformation from Cartesian coordinate system to spherical coordinate

system is given by Equation (A.2).

$$\mathbf{v} = \rho \begin{bmatrix} \cos\theta \sin\phi \\ \sin\theta \sin\phi \\ \cos\phi \end{bmatrix} \tag{A.1}$$

$$\mathbf{m} = \begin{bmatrix} \rho \\ \theta \\ \phi \end{bmatrix} = \begin{bmatrix} \sqrt{x^2 + y^2 + z^2} \\ \arctan(y/x) \\ \arcsin(z/\rho) \end{bmatrix} \tag{A.2}$$

A.2 The ∇ Operator

The *del* operator (noted ∇) is defined as partial derivative operators in three dimensions in Cartesian coordinate system:

$$\nabla \equiv \vec{x}\frac{\partial}{\partial x} + \vec{y}\frac{\partial}{\partial y} + \vec{z}\frac{\partial}{\partial z} \tag{A.3}$$

Here $\vec{x}, \vec{y}, \vec{z}$ are the unit vectors in the three coordinate axes. We need this operator in spherical coordinates in order to be able to work directly with them, but the result must be back in Cartesian coordinates. This way there is no overhead due to coordinate conversion. Note that this makes the operator derived here different from ∇ in spherical coordinates, which returns spherical coordinates again. Our needs require an operator that computes the Cartesian partial derivatives in spherical coordinates. To express partial derivatives with respect to Cartesian axes in terms of partial derivatives of the spherical coordinates:

$$\begin{bmatrix} x \\ y \\ z \end{bmatrix} = \begin{bmatrix} \rho\cos\theta\sin\phi \\ \rho\sin\theta\sin\phi \\ \rho\cos\phi \end{bmatrix} \tag{A.4}$$

$$\begin{aligned} \begin{bmatrix} dx \\ dy \\ dz \end{bmatrix} &= \begin{bmatrix} \cos\theta\sin\phi d\rho - \rho\sin\theta\sin\phi d\theta + \rho\cos\theta\cos\phi d\phi \\ \sin\theta\sin\phi d\rho + \rho\cos\theta\sin\phi d\theta + \rho\sin\theta\cos\phi d\phi \\ \cos\phi d\rho - \rho\sin\phi d\phi \end{bmatrix} \\ &= \begin{bmatrix} \cos\theta\sin\phi & -\rho\sin\theta\sin\phi & \rho\cos\theta\cos\phi \\ \sin\theta\sin\phi & \rho\cos\theta\sin\phi & \rho\sin\theta\cos\phi \\ \cos\phi & 0 & -\rho\sin\phi \end{bmatrix} \begin{bmatrix} dx \\ dy \\ dz \end{bmatrix} \end{aligned} \tag{A.5}$$

After coordinate conversion:

$$\begin{bmatrix} d\rho \\ d\theta \\ d\phi \end{bmatrix} = \begin{bmatrix} \cos\theta\sin\phi & \sin\theta\sin\phi & \cos\phi \\ -\dfrac{\sin\theta}{\rho\sin\phi} & \dfrac{\cos\theta}{\rho\sin\phi} & 0 \\ \dfrac{\cos\theta\cos\phi}{\rho} & \dfrac{\sin\theta\cos\phi}{\rho} & -\dfrac{\sin\phi}{\rho} \end{bmatrix} \begin{bmatrix} dx \\ dy \\ dz \end{bmatrix} \tag{A.6}$$

And finally ([13]):

$$\begin{aligned} \nabla \equiv\ & \vec{x}\Big(\cos\theta\cos\phi\frac{\partial}{\partial\rho} - \frac{\sin\theta}{\rho\sin\phi}\frac{\partial}{\partial\theta} + \frac{\cos\theta\cos\phi}{\rho}\frac{\partial}{\partial\phi}\Big) + \\ & \vec{y}\Big(\sin\theta\cos\phi\frac{\partial}{\partial\rho} + \frac{\cos\theta}{\rho\sin\phi}\frac{\partial}{\partial\theta} + \frac{\sin\theta\cos\phi}{\rho}\frac{\partial}{\partial\phi}\Big) + \\ & \vec{z}\Big(\cos\phi\frac{\partial}{\partial\rho} - \frac{\sin\phi}{\rho}\frac{\partial}{\partial\phi}\Big) \end{aligned} \tag{A.7}$$

A.3 Spherical Range Image (SRI)

A Spherical Range Image (SRI) provides a point cloud representation in 2D image form in the following way: the value of the image at column θ and row ϕ is ρ, where $(\rho, \theta, \phi)^T$ represents a 3D point in spherical coordinates. Considering the image as a value function $s(\theta, \phi)$ of the coordinates, we can state:

$$\rho = s(\theta, \phi) \tag{A.8}$$

For our purposes we need a descrete approximation of the function and pixel gray scale values for the range. For visualization the values are rounded to the nearest integer number but for the computations floating point numbers are used.

A SRI provides a visual representation of the environment as seen from a single viewpoint defined by the choice of origin of the spherical coordinate system. These images are widely used in various visualization and data processing problems. Example SRIs can be seen on Figures 1.2, 4.1, 5.2.

A.4 The Scharr Operator

When computing the partial derivatives on the SRI needed for the normal estimation, we use the Scharr operator. It is an alternative of the Sobel operator.

The Sobel operator is used in image processing particularly for edge-detection [3]. Effectively, it computes the gradient of the image intensity at each point, or the derivative of the function

describing the image (with coordinates and intensity) with respect to the intensity. In the case of SRI, the result gives the partial derivative of this function with respect to the range.

The computation of the gradient $\mathbf{G}$ of an image $\mathbf{A}$ requires the combination of the gradients in both dimensions $\mathbf{G}_x$ and $\mathbf{G}_y$.

$$\mathbf{G} = \sqrt{\mathbf{G}_x^2 + \mathbf{G}_y^2} \text{ with } \mathbf{G}_x = \begin{bmatrix} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{bmatrix} \otimes \mathbf{A} \text{ and } \mathbf{G}_y = \begin{bmatrix} -1 & -2 & -1 \\ 0 & 0 & 0 \\ 1 & 2 & 1 \end{bmatrix} \otimes \mathbf{A} \tag{A.9}$$

Here the directions of the x- and y-coordinates are assumed to be increasing in the right and down directions respectively and the symbol $\otimes$ denotes 2-dimensional convolution operation.

H. Scharr optimized the kernels with respect to minimizing the weighted mean squared angular error in Fourier domain [11]. These kernels are really derivative kernels rather than just keeping symmetry constraints. This is the reason they are used in this work. However, we need only the single-dimensional gradients, corresponding to the partial derivatives of the x and y coordinates of the image, and respectively to θ and ϕ on the SRI. Also, a slight modification is made on the original Scharr kernels by multiplication by 2, giving more weight to the derivatives. The resulting kernels are as follows:

$$\mathbf{K}_\theta = \begin{bmatrix} -6 & 0 & 6 \\ -20 & 0 & 20 \\ -6 & 0 & 6 \end{bmatrix} \text{ and } \mathbf{K}_\phi = \begin{bmatrix} -6 & -20 & -6 \\ 0 & 0 & 0 \\ 6 & 20 & 6 \end{bmatrix} \tag{A.10}$$

Also, further weight is added to the derivative by multiplying it by the scan factor defined in Equation (4.1). This way the resultant resolution of the SRI is taken into account when computing the derivatives.

Bibliography

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

[2] H. Badino, D. Huber, Y. Park, and T. Kanade. Fast and Accurate Computation of Surface Normals from Range Images. *IEEE International Conference on Robotics and Automation*, 11:3084 – 3091, May 2011.

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

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

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

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

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

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

[9] 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.

[10] 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.

[11] Hanno Scharr. Optimal Operators in Digital Image Processing. *Dissertation*, 2000.

[12] 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.

[13] Eric W. Weisstein. Spherical Coordinates. From MathWorld - A Wolfram Web Resource.

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

Proclamation

Hereby I confirm that I wrote this thesis independently and that I have not made use of any other resources or means than those indicated.

Bremen, May 2012