2014 IEEE/RSJ International Conference on
Intelligent Robots and Systems (IROS 2014)
September 14-18, 2014, Chicago, IL, USA

A Quantitative Evaluation of Surface Normal Estimation in Point Clouds

Krzysztof Jordan[1] and Philippos Mordohai[1]

Abstract— **We revisit a well-studied problem in the analysis of range data: surface normal estimation for a set of unorganized points. Surface normal estimation has been well-studied initially due to its theoretical appeal and more recently due to its many practical applications. The latter cover several aspects of range data analysis from plane or surface fitting to segmentation, object detection and scene analysis. Following the vast majority of the literature, we also focus our attention on techniques that operate in small neighborhoods around the point whose normal is to be estimated. We pay close attention to aspects of the implementation, such as the use of weights and normalization, that have not been studied in detail in the past. We perform quantitative evaluation on a diverse set of point clouds derived from 3D meshes, which allows us to obtain accurate ground truth.**

I. INTRODUCTION

Point clouds are increasingly becoming more prevalent as a form of data. They are directly acquired by range sensors such as LIDAR or depth cameras based on structured light. While range images can be analyzed relying partially on image processing techniques, if more than one range image is available, these techniques become inapplicable in general. Point clouds are often directly sufficient for tasks such as obstacle avoidance, but require further processing for higher level tasks. The most common estimation on point cloud inputs is per-point normal estimation, since surface normals are useful for the segmentation of range data [1], [2] and for computing shape descriptors [3], [4], [5], [6], [7].

While surface normal estimation is typically treated as a tool that does not require any attention in terms of algorithm design, similar to a mean filter for instance, normals are often estimated sub-optimally even by state of the art systems. The differences in accuracy due to these choices may be small, and the impact on overall system performance may also be small, but there is no reason to include a sub-optimal estimator in a system. It is possible that an increase in error of a few degrees in surface normal estimation will not affect the results of a sophisticated segmentation algorithm, but why should one use a worse estimator when better options are available at the same or similar computational cost?

Along the lines of most surface normal estimation methods from point clouds [8], [9], all algorithms in this paper operate in local neighborhoods around each point. They aim at estimating the normal at a point based on minimizing the fitting error of planar or quadratic surfaces passing through each point of the neighborhood, or by estimating a 2D subspace that is tangent at the point of interest based on pairwise point relationships. Even though at least three points would be required to infer a plane, the vector from each point of the neighborhood to the *anchor* point of the plane must lie in the desired plane. Thus, pairwise operations provide constraints that can be aggregated to lead to complete solutions. Techniques that operate on triplets of points have also been reported in the literature, but are out of scope of this paper because they either require Delaunay triangulation as pre-processing or they consider all combinations of three points and are computationally expensive.

In this paper, we extend a recent study by Klasing et al. [10] in two ways: by augmenting the set of methods evaluated and by performing the evaluation on a much larger, diverse set of 3D shapes. We keep four of the methods proposed in [10] (Section III-A) and augment the set by considering three modifications: (i) using the reference point, instead of the neighborhood mean, as the anchor point of the plane, (ii) normalizing the vectors formed by points in the neighborhood and the anchor point, and (iii) applying weights to the contributions of neighboring points that decay with distance.

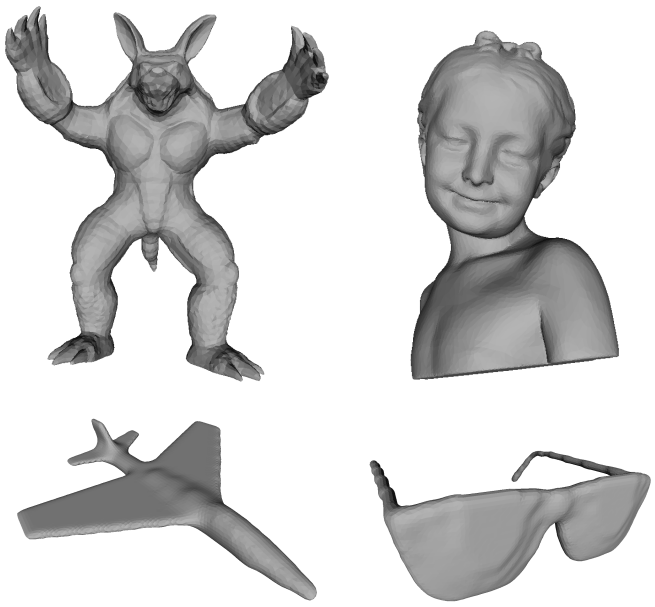


Fig. 1. Screenshots of the armadillo, girl, airplane and glasses models that are used as inputs in our experiments. The models are displayed as meshes, but only the vertices are actually used in the computation.

[1]Department of Computer Science, Stevens Institute of Technology, Hoboken, NJ 07030, USA
{kjordan1, Philippos.Mordohai}@stevens.edu
This work was supported in part by a Google Research Award.

We evaluate all methods on a set of 40 3D models made available online by Dutagaci et al. [11] (Fig. 1). The models are in the form of meshes, which allow the computation of surface normals with much higher accuracy than what is possible using just the point cloud as input. Most of the surfaces have regions of high curvature and are not just smooth planar or quadratic patches. Only the point clouds are provided as input to the various method in the experiments and the accuracy of the estimated surface normals is compared to the ground truth. As expected, we observe non-trivial errors even without additional noise. This loss of accuracy is inevitable due to the absence of connectivity information. While this may appear as an unfair comparison, the estimation of normals of the original continuous surfaces, which are approximated by the point clouds, is precisely the problem we tackle.

II. RELATED WORK

The prevailing approach for estimating surface normals from unorganized point clouds was introduced in the classic paper by Hoppe et al. [8] and is based on total least squares (TLS) minimization. This approach has been adopted by numerous authors without modifications. Among these publications, the paper by Mitra et al. [9] contains an in depth analysis of the effects of neighborhood size, curvature, sampling density, and noise in normal estimation using the TLS approach. We study different aspects of the computation, focusing on the objective function itself, but refer readers to [9] for a rigorous analysis of the TLS approach including error bounds on the estimated normals.

An early comparative study on surface normal estimation for range images was published by Wang et al. [12]. While the methods evaluated and the authors' observation remain relevant, we adopt the notation and terminology of the more recent work by Klasing et al. [10]. Details on the latter can be found in Section III and are not repeated here. Badino et al. [13] also evaluated different techniques surface normal estimation from points recently, but their focus was on range images exploiting the inherent neighborhood structure to further accelerate computation.

Tombari et al. [7] also solve the standard total least squares formulation, but apply weights the linearly decay with distance on the contribution of each neighboring point. For efficiency, they chose to not compute the centroid of the region, centering the computation on the reference point itself. We evaluate both of these modifications to the TLS formulation, but we use Gaussian instead of linear weights.

III. METHODS

In this section, we present the methods that are evaluated in Section IV. We begin with the methods presented by Klasing et al. [10] and proceed by introducing the modifications that are the focus of our study. We adopt the notation of [10] to allow direct comparisons and only consider methods they classify as "optimization-based", as opposed to "averaging" which require Delaunay triangulation of the point cloud.

The input point cloud is a set of n points $P = \{p_1, p_2, ..., p_n\}$, $p_i \in \mathbb{R}^3$. We are interested in the estimation of the surface normal for a point in the point cloud, which we refer to as the *reference point*. The reference point is denoted by $p_i = [p_{ix}, p_{iy}, p_iz]^T$ and its normal by $n_i = [n_{ix}, n_{iy}, n_{iz}]^T$. Only points in the neighborhood of p_i, which is denoted by $Q_i = \{q_{i1}, q_{i2}, ..., q_{ik}\}$, $q_{ij} \in P$, $q_{ij} \neq p_i$, are used for the estimation. The number of points k included in the neighborhood is a key parameter for all methods. Standard nearest neighbor searches based on k-d trees are used to find the neighbors of each point. Following [10], we define the *data matrix*, the *neighbor matrix* and the *augmented neighbor matrix* as follows:

$$\mathbf{P} = [p_1, p_2, ..., p_n]^T \quad (1)$$

$$\mathbf{Q_i} = [q_{i1}, q_{i2}, ..., q_{ik}]^T \quad (2)$$

$$\mathbf{Q_i^+} = [p_i, q_{i1}, q_{i2}, ..., q_{ik}]^T. \quad (3)$$

The augmented neighbor matrix $\mathbf{Q_i^+}$ contains the reference point p_i in addition to its neighbors.

A. Objective Functions for Normal Estimation

As Klasing et al. [10] remark, seemingly different criteria, such as minimizing the fitting error of a local plane passing through $\mathbf{Q_i^+}$ or maximizing the angle between the surface normal at p_i and the vectors formed by p_i and its neighbors, result in very similar objective functions to be optimized. In all cases, the desired solution is the singular vector associated with the minimal singular value of an appropriate matrix.

PlaneSVD: This method [12] fits a local plane $S_i(x, y, z) = n_{ix}x + n_{iy}y + n_{iz}z + d$ to the points in $\mathbf{Q_i^+}$. The objective function is:

$$J_1(b_i) = || \ [\mathbf{Q_i^+} \ \mathbf{1}_{k+1}] \ b_i||_2, \quad (4)$$

where $\mathbf{1}_{k+1}$ is column vector of $k+1$ ones and $b_i = [n_i^T \ d]^T$. The minimizer of (4) is the singular vector of $[\mathbf{Q_i^+} \ \mathbf{1}_{k+1}]$ associated with the smallest singular value. The first three elements of this vector are normalized and assigned to n_i.

PlanePCA: PlaneSVD minimizes the fitting error of the local plane in the neigborhood of p_i. A different criterion is to find the direction of minimum variance in the neighborhood Q_i^+. If the points in Q_i^+ formed a perfect plane, the direction of minimum variance would be the normal to this plane and the variance would be 0. In the presence of noise or non-planar surfaces, the direction of minimum variance is the best approximation to the normal. To obtain the objective function (TLS) we subtract the empirical mean of the data matrix [8], [14], [9]:

$$J_2(n_i) = || \ [\mathbf{Q_i^+} \ - \ \bar{\mathbf{Q}}_\mathbf{i}^+] \ n_i||_2, \quad (5)$$

where $\bar{\mathbf{Q}}_\mathbf{i}^+$ is a matrix containing the mean vector $\bar{q}_i^+ = \frac{1}{k+1}(p_i + \sum_{j=1}^{k} q_{ij})$ in every row. The minimizer of (5) is the singular vector of $[\mathbf{Q_i^+} \ - \ \bar{\mathbf{Q}}_\mathbf{i}^+]$ associated with the smallest singular value. The name PlanePCA is justified because this computation is equivalent to taking the principal component with the smallest variance after performing Principal

Component Analysis (PCA) on $\mathbf{Q_i^+}$. On the other hand, in PlaneSVD the data are not centered by subtracting the mean.

This is also equivalent to forming the empirical covariance matrix of the points in $\mathbf{Q_i^+}$, computing its eigendecomposition and setting n_i equal to the eigenvector associated with the minimum eigenvalue [15], [12].

VectorSVD: An alternative formulation for inferring the surface normal at p_i is to seek the vector that maximizes the angle between itself and the vectors from p_i to q_{ij}, which should be in the two-dimensional tangent subspace of p_i. The difference with the previous methods is that VectorSVD *forces the estimated plane to pass through the reference point* p_i. The maximization of angles can be formulated as the minimization of the inner products between n_i and the vectors from p_i to q_{ij}.

$$J_3(n_i) = || \begin{bmatrix} (q_{i1} - p_i)^T \\ (q_{i2} - p_i)^T \\ \dots \\ (q_{ik} - p_i)^T \end{bmatrix} n_i||_2. \quad (6)$$

The singular vector associated with the minimum singular value of the matrix above is the minimizer of this objective and the desired normal.

Following Klasing et al. [10], we omit the VectorPCA method due to its similarity to PlanePCA.

QuadSVD: This is the only quadratic method evaluated in this paper. QuadSVD [16], [17] assumes that the surface is composed of small quadratic patches, instead of small planar patches as all other methods in this paper do. Specifically, a surface in the form of $S_i(x, y, z) = c_1x^2 + c_2y^2 + c_3z^2 + c_4xy + c_5xz + c_6yz + c_7x + c_8y + c_9z + c_{10}$ is fitted to to the points in $\mathbf{Q_i^+}$. The objective function is:

$$J_4(c_i) = ||\mathbf{R_i}c_i||_2, \quad (7)$$

where c_i is the vector of coefficients and each row of $\mathbf{R_i}$ contains the linear and quadratic terms corresponding to a point in $\mathbf{Q_i^+}$, that is, each row of $\mathbf{R_i}$ contains: $[q_{ijx}^2 \ q_{ijy}^2 \ q_{ijx}^2 \ q_{ijx}q_{ijxy} \ q_{ijx}q_{ijz} \ q_{ijy}q_{ijz} \ q_{ijx} \ q_{ijy} \ q_{ijz} \ 1]$ with the first row corresponding to p_i. The final normal n_i is computed by evaluating the gradient of S_i at p_i.

B. Modifications

In this section, we introduce three modifications that are sometimes applied in the literature to improve the robustness of surface normal estimation. To the best of our knowledge, however, a quantitative analysis of their effects and a set of recommendations for practitioners have not been published before. We begin with an equivalent formulation of VectorSVD, but not centered on the reference point, as the baseline here. For each neighborhood $\mathbf{Q_i}$, we form a 3×3 square matrix $\mathbf{M_i}$ as follows:

$$\mathbf{M_i} = \sum_{q_{ij} \in \mathbf{Q_i}} (q_{ij} - \bar{q}_i^+)(q_{ij} - \bar{q}_i^+)^T, \quad (8)$$

where $\bar{q}_i^+$ is the mean of the neighborhood including the reference point. The eigenvector of $\mathbf{M_i}$ corresponding the minimum eigenvalue minimizes the following criterion and is taken as the normal at p_i.

$$J_5(n_i) = ||\mathbf{M_i}n_i||_2. \quad (9)$$

Anchoring on the reference point: A variation that has already appeared in the methods of Section III-A is the use of either the neighborhood mean or the reference point itself as the anchor point through which the plane is assumed to pass. While in many cases, especially for large values of k, this choice should not make a large difference, it is worth investigating the advantages and disadvantages of either choice. The conventional TLS approach [8], [15], [9], termed PlanePCA here, uses the neighborhood mean as the anchor, while VectorSVD uses the reference point as the anchor.

Conceptually, if the data are assumed noise-free, using the reference point should lead to more precise estimates. Conversely, if the data are assumed to be corrupted by noise, the neighborhood mean, may be a better proxy for the anchor point than p_i. In terms of processing time, avoiding the computation of the neighborhood mean saves a number of operations that is linear in k.

$$\mathbf{M_i^{(R)}} = \sum_{q_{ij} \in \mathbf{Q_i}} (q_{ij} - p_i)(q_{ij} - p_i)^T. \quad (10)$$

Normalized vectors: An observation that can be made from (8) and all preceding objective functions is that vectors formed using neighbors closer to the anchor point are shorter than vector formed using the furthest points in $\mathbf{Q_i}$. This is not a desirable property in general, since it violates the assumption that the surface is only locally planar. The second modification of (8) we examine is by normalizing the vectors in the outer products that form $\mathbf{M_i}$:

$$\mathbf{M_i^{(N)}} = \sum_{q_{ij} \in \mathbf{Q_i}} \frac{(q_{ij} - \bar{q}_i^+)(q_{ij} - \bar{q}_i^+)^T}{||q_{ij} - \bar{q}_i^+||_2^2}, \quad (11)$$

in case the plane is anchored at the neighborhood mean, or

$$\mathbf{M_i^{(NR)}} = \sum_{q_{ij} \in \mathbf{Q_i}} \frac{(q_{ij} - p_i)(q_{ij} - p_i)^T}{||q_{ij} - p_i||_2^2}, \quad (12)$$

in case the plane is anchored at the reference point.

As a result of normalization, all the matrices contributing to the sum are rank-1 with their non-zero eigenvalue equal to 1.

Weighted contributions: For similar reasons to the above modification and in order to reduce the effects of points that were included in $\mathbf{Q_i}$ despite being far from p_i, several authors [18], [7] have applied weights to the outer products as they are summed to form $\mathbf{M_i}$. The weights decrease with distance to make the effects of distant points smaller.

$$\mathbf{M_i^{(W)}} = \sum_{q_{ij} \in \mathbf{Q_i}} e^{-\frac{||q_{ij} - \bar{q}_i^+||_2^2}{2\sigma^2}} (q_{ij} - \bar{q}_i^+)(q_{ij} - \bar{q}_i^+)^T. \quad (13)$$

The distances used in the weight computation above are for the case the plane is anchored at the neighborhood mean. The computation of $\mathbf{M_i^{(WR)}}$ is analogous when the plane is anchored on p_i.

In this paper, we define the weights as shown in (13), but other alternatives exist. For example, Tombari et al. [7] choose a linear decay with distance. In most cases, an additional parameter is required for specifying the weights.

We set σ equal to the average distance from each point in the point cloud to its k^{th} neighbor. σ remains fixed for a point cloud and assigns non-trivial weights to all points in most neighborhoods, as long as they are not outliers, very far from the anchor point. Making σ a function of the distance to a certain nearest neighbor allows us to use the same procedure for setting it, regardless of scale and number of points in the input point cloud. We did not attempt to quantify the effects of σ in this paper.

Notation: Having three binary choices leads to a total of eight methods. We use N for normalization, W for weights and R for reference point to denote which modifications are active. For example, the method with all modifications active is denoted by NWR and computes the normal as the eigenvector corresponding to the minimum eigenvalue of the following matrix.

$$\mathbf{M_i^{(NWR)}} = \sum_{q_{ij} \in \mathbf{Q_i}} e^{-\frac{||q_{ij}-p_i||_2^2}{2\sigma^2}} \frac{(q_{ij}-p_i)(q_{ij}-p_i)^T}{||q_{ij}-p_i||_2^2}. \tag{14}$$

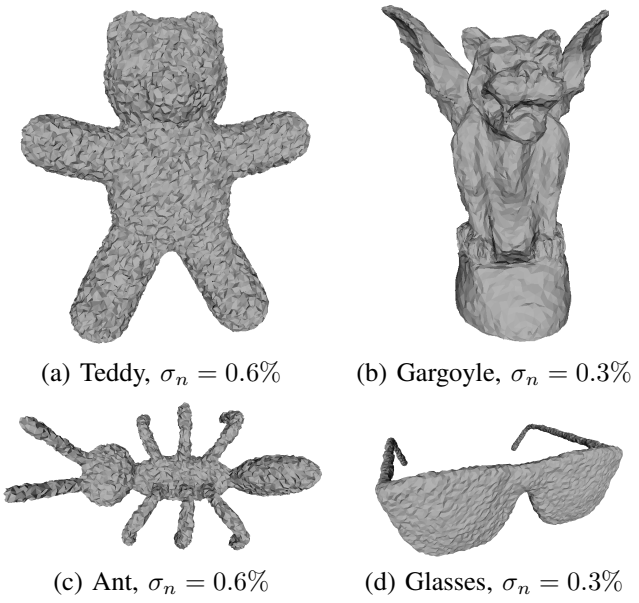
Note that all methods in Section III-B operate in neighborhoods that have not been augmented by the reference point p_i, except for the computation of the neighborhood mean. We made this choice because we do not assume that $p_i\bar{q}_i^+$ lies on local plane. When none of the modifications are active, the method is denoted by $base$ and is similar to the PlanePCA algorithm up to the exclusion of the reference point from the neighborhood. R is identical to VectorSVD, since p_i is used as the anchor.

IV. EXPERIMENTAL RESULTS

In this section, we describe the data and ground truth generation for our experiments followed by quantitative results.

A. Experimental Setup

We performed experiments on 40 3D meshes, made available online by Dutagaci et al. [11]. Some examples are shown in Fig. 1. These are more challenging than the data used by previous studies [10], [13] that consisted of smooth surfaces with fewer discontinuities. The average number of vertices per model is 8,548, for a grand total of 341,909 vertices. Each point cloud is centered and scaled so that the diagonal of its bounding box is equal to one unit of distance. It should be pointed out that the vertices are not uniformly sampled. Instead, they are denser in areas with more details. We decided against re-sampling the meshes, since non-uniform density is a common challenge for normal estimation in practice. We only use the mesh connectivity information to generate ground truth, but provide only the point clouds as input to the normal estimation methods of Section III.



(a) Teddy, $\sigma_n = 0.6\%$ (b) Gargoyle, $\sigma_n = 0.3\%$

(c) Ant, $\sigma_n = 0.6\%$ (d) Glasses, $\sigma_n = 0.3\%$

Fig. 2. Screenshots of meshes corrupted by noise with σ_n given as a fraction of the diagonal of the bounding box of the entire mesh. Only the vertices are used for surface normal estimation.

Ground truth normal estimation: We exploit mesh connectivity information to compute the ground truth as the weighted average of the surface normals of all triangles incident at a vertex. We choose to weigh the normal of each triangle according to the angle under which the triangle is incident at the vertex of interest [19]. (Weighing the normals according to the areas of the triangles [20], [19] produces estimates within 1° of the angle-weighted ones on average. Either weighting scheme could have been used.)

We estimated surface normals for all vertices in all 40 meshes with k ranging from 10 to 50 in steps of 10. We report the *average error per point in degrees*, dividing by the total number of points. Since we are only interested in surface normal estimation accuracy over a large set of points, this error metric is more relevant than averaging the error per mesh and then averaging the per-mesh errors.

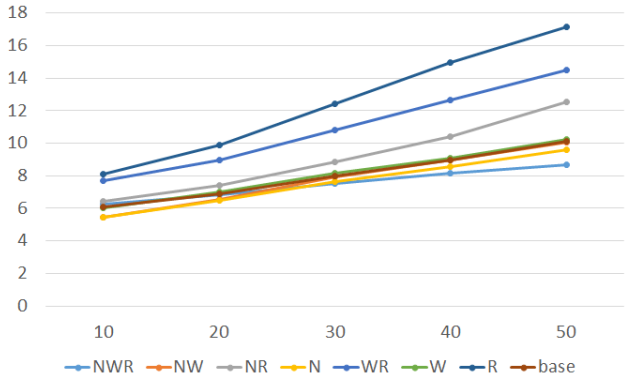
Additive zero-mean Gaussian noise was added to the points. We set the standard deviation of the noise σ_n to 0.15%, 0.3%, 0.45% and 0.6% of the diagonal and repeat the experiments as above. Increasing σ_n even further resulted in extremely noisy models. Some examples of the effects of noise can be seen in Fig. 2. *The ground truth normals of the noise-free meshes were used as ground truth for these experiments as well.* Otherwise, the experiments would be equivalent to the noise-free case on deformed inputs.

B. Quantitative Results

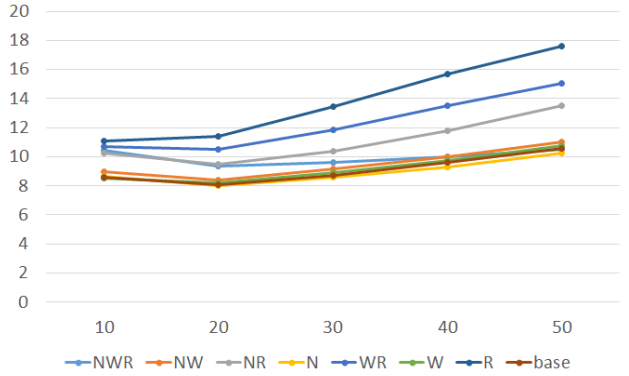
We begin by evaluating the effects of the three modifications presented in Section III-B, before comparing the most

effective among them to the methods of Section III-A.

Figure 3 shows the average orientation error per point in degrees as a function of k for the noise-free case, as well as for additive noise with $\sigma_n = 0.3\%$.



(a) Noise-free inputs



(b) Additive noise $\sigma_n = 0.3\%$

Fig. 3. Average orientation error per point in degrees over all 3D models as a function of k for the methods of Section III-B. The lowest overall error is achieved by N on the noise-free data, while NWR shows greater stability as k increases. NWR shows similar behavior on noisy data as well. The lowest error is achieved by N for $k \geq 20$ for all values of σ_n.

Effects of anchor point selection: One of the most consistent findings among all combinations of noise, normalization and use of weights is that anchoring the plane on the neighborhood mean leads to higher accuracy. Table I shows the average error in degrees for the noise-free case comparing methods with different choices for the anchor point side by side. The middle column shows error values when the local planes are anchored at the reference point (R on) and the right column the same errors when the planes are anchored on the neighborhood mean (R off). Results for all noise levels are very similar and have been omitted.

Effects of normalization: A second consistent finding in all our experiments is that normalizing the vectors used in the formulation of the objective functions is beneficial since it does not give more weight to distant points. Table II shows a similar comparison with that of Table I focusing on the effects of setting N on or off. The results below are for the noise free case.

Effects of weights: The conclusion from these experiments is less clear. The use of weights improves accuracy

TABLE I

AVERAGE ORIENTATION ERROR PER POINT IN DEGREES OVER ALL 3D MODELS FOR $k = 10$.

	R on	R off
NW	6.28	5.46
N	6.43	5.43
W	7.67	6.00
base	8.10	6.08

TABLE II

AVERAGE ORIENTATION ERROR PER POINT IN DEGREES OVER ALL 3D MODELS FOR $k = 10$.

	N on	N off
WR	6.28	7.67
W	5.46	6.00
R	6.43	8.10
base	5.43	6.08

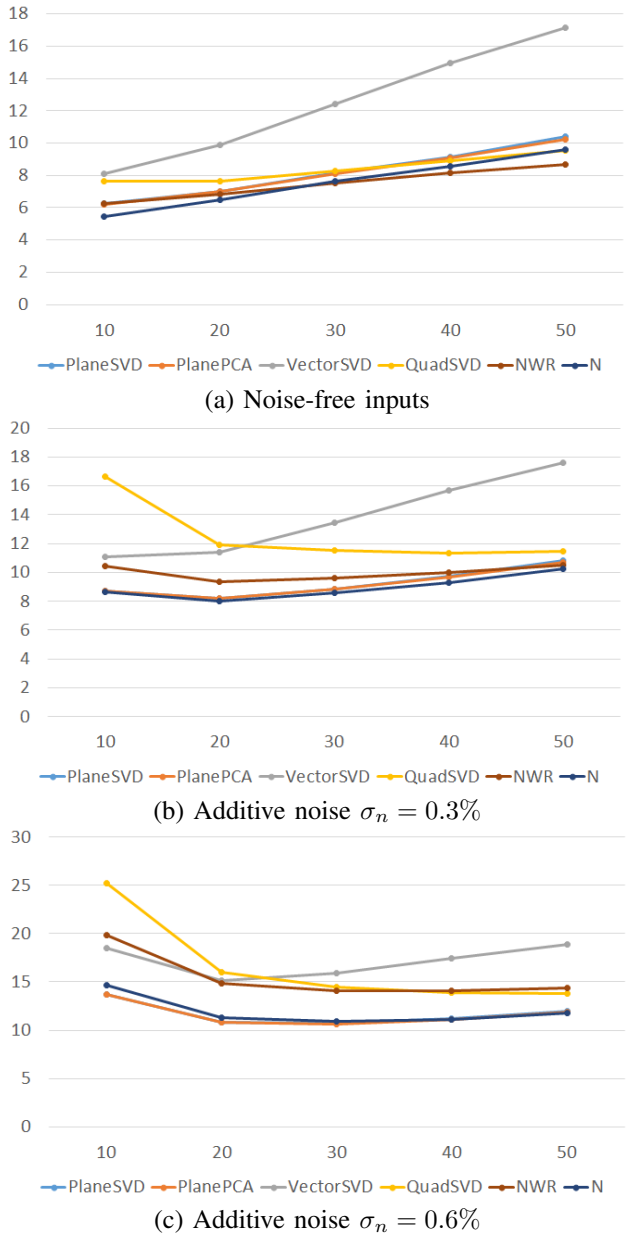
when the reference point is used as anchor. On the other hand, it is detrimental when the neighborhood mean is used as anchor. As noise levels increase, the use of weights has a slight positive impact on accuracy. It is possible that adapting σ in (13) appropriately may reveal more positive effects, but our conclusion from this study is that normalization is more effective than the use of weights in reducing the effects of distant neighbors and it also does not require additional parameters.

Comparison of all methods: Figure 4 shows the average orientation error over all 3D models for PlaneSVD, PlanePCA, VectorSVD, QuadSVD, NWR and N. We chose N as the top performing among the modifications we evaluated above and NWR due to its more stable behavior with respect to variations in k.

The lowest error is achieved by N on the noise-free data for $k = 10$, while NWR shows greater stability as k increases. (There is no benefit from considering more neighbors since the data is noise-free.) When noise with $\sigma_n = 0.3\%$ is added, N still achieves the smallest error (8.04°) for $k = 20$. PlanePCA and PlaneSVD are virtually indistinguishable from N for all values of k. Increasing σ_n to 0.6% does not alter the results significantly. PlanePCA achieves the smallest orientation error at $k = 30$ by a very small margin over PlaneSVD and N.

VectorSVD, or equivalently R, is clearly worse than the other methods. QuadSVD does not work well based on a minimal sample of 10 neighbors and keeps improving as k grows without coming particularly close to the other methods. NWR appears to be more sensitive to noise then the other methods.

Figure 5 contains visualizations of various models with different levels of additive noise generated by the *base* method. In general, discerning the per-point differences among different methods is essentially impossible, so we do not provide visualizations of results from other methods. The vertices in these models have been color-coded according

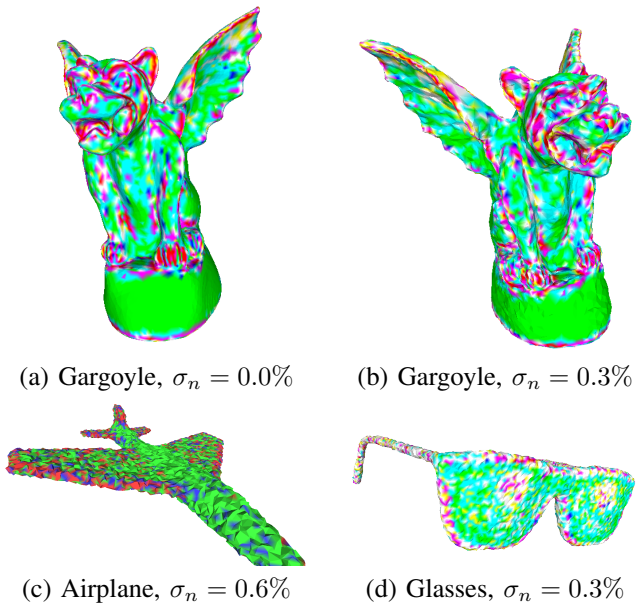


(a) Noise-free inputs

(b) Additive noise $\sigma_n = 0.3\%$

(c) Additive noise $\sigma_n = 0.6\%$

Fig. 4. Average orientation error per point in degrees over all 3D models as a function of k for PlaneSVD, PlanePCA, VectorSVD, QuadSVD, NWR and N. See text for an analysis of the results. PlaneSVD and PlanePCA overlap almost perfectly and cannot be distinguished in the plots.



(a) Gargoyle, $\sigma_n = 0.0\%$ (b) Gargoyle, $\sigma_n = 0.3\%$

(c) Airplane, $\sigma_n = 0.6\%$ (d) Glasses, $\sigma_n = 0.3\%$

Fig. 5. Screenshots of meshes color-coded according to surface normal orientation error. All models were generated by the *base* method. The color coding is with respect to the mean error for the specific mesh. Vertices within 25% of the mean are colored blue, vertices with error larger than 125% of the mean are colored red and vertices with error less than 75% of the mean are colored green. The color of the interior points in each triangle is the result of barycentric blending of the vertex colors producing shades of purple or cyan. As before, σ_n is given as a fraction of the diagonal of the bounding box of the mesh.

to the errors of the estimated normals. Figure 6 shows the average orientation error for $k = 20$ as a function of σ_n. N and PlanePCA appear to be more robust to additive noise, while QuadSVD and NWR show a steep increase in error.

V. CONCLUSIONS

We presented an evaluation of surface normal estimation methods applicable to point clouds that is extensive both in terms of the number of methods being evaluated and also in terms of the size of the validation set. The use of 40 3D meshes of complex geometry poses challenges to these algorithms and exposes potential weaknesses. These challenges are due to the presence of details (large curvature) on the surfaces, and due to the non-uniform density of the points, since density is proportional to the local level of detail. Moreover, we perturbed the vertices by adding zero-mean Gaussian noise to them.

Our results shows that the use of the reference point, the point for which the normal is estimated, as the anchor through which the plane must pass is detrimental for accuracy. This is due to the non-isotropy of the neighborhoods and, in some experiments, the presence of strong additive noise that make the reference point unsuitable for this role. Using the neighborhood mean has proven to be more effective. As a result, VectorSVD is inferior to the other methods presented in [10].

A second clear conclusion from our work is that all vectors used to construct the matrices whose spectral analysis produces the sought-after normals should be normalized.

While the use of weights did not seem to be beneficial in our experimental setup (non-isotropic neighborhoods and additive noise), it has shown to be very effective when the data are corrupted by outliers [21], [22], [23]. We plan to perform further analysis on data corrupted by both additive and outlier noise in our future work.

It is well known that PlaneSVD may suffer from numerical instability. This behavior is not observed on our data because the point clouds are centered and scaled so that the diagonal of their bounding box is equal to one, but practitioners should be aware of this drawback.

Our experiments do not support fitting quadratic surfaces

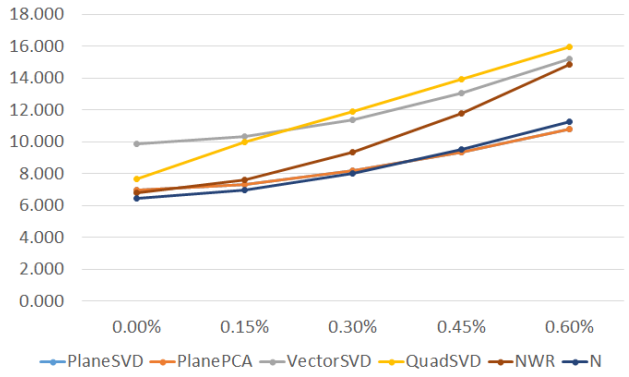


Fig. 6. Average orientation error per point in degrees over all 3D models as a function of σ_n for PlaneSVD, PlanePCA, VectorSVD, QuadSVD, NWR and N. The number of neighbors is fixed at 20 for all noise levels. PlaneSVD and PlanePCA overlap almost perfectly and cannot be distinguished in the plot.

to the data. QuadSVD does not perform particularly well and comes at a much higher computational cost due to the need to perform SVD on a 10×10 matrix for each point, in contrast to most of the other methods that operate on 3×3 scatter matrices.

Our final recommendation is the use of the optimal total least squares solution, termed PlanePCA in [10], modified to use normalized vectors. Normalization, in general, significantly improves the numerical stability and accuracy of TLS optimization. See, for example, the work of Hartley [24].

References

[1] D. Anguelov, B. Taskarf, V. Chatalbashev, D. Koller, D. Gupta, G. Heitz, and A. Ng, "Discriminative learning of markov random fields for segmentation of 3d scan data," in *IEEE. Conf. on Computer Vision and Pattern Recognition*, vol. 2, 2005, pp. 169–176.

[2] A. Golovinskiy, V. Kim, and T. Funkhouser, "Shape-based recognition of 3d point clouds in urban environments," in *Int. Conf. on Computer Vision*, 2009.

[3] A. E. Johnson and M. Hebert, "Using spin images for efficient object recognition in cluttered 3d scenes," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 21, no. 5, pp. 433–449, 1999.

[4] A. Frome, D. Huber, R. Kolluri, T. Bulow, and J. Malik, "Recognizing objects in range data using regional point descriptors," in *European Conf. on Computer Vision*, 2004, pp. Vol III: 224–237.

[5] A. Patterson, P. Mordohai, and K. Daniilidis, "Object detection from large-scale 3D datasets using bottom-up and top-down descriptors," in *European Conf. on Computer Vision*, 2008, pp. 553–566.

[6] R. B. Rusu, N. Blodow, and M. Beetz, "Fast point feature histograms (fpfh) for 3d registration," in *IEEE International Conference on Robotics and Automation (ICRA)*. IEEE, 2009, pp. 3212–3217.

[7] F. Tombari, S. Salti, and L. Di Stefano, "Unique signatures of histograms for local surface description," in *European Conf. on Computer Vision*. Springer, 2010, pp. 356–369.

[8] H. Hoppe, T. DeRose, T. Duchamp, J. McDonald, and W. Stuetzle, "Surface reconstruction from unorganized points," *ACM SIGGRAPH*, pp. 71–78, 1992.

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

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

[11] H. Dutagaci, C. Cheung, and A. Godil, "Evaluation of 3D interest point detection techniques via human-generated ground truth," *The Visual Computer*, vol. 28, pp. 901–917, 2012.

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

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

[14] M. Gopi, S. Krishnan, and C. T. Silva, "Surface reconstruction based on lower dimensional localized delaunay triangulation," in *Computer Graphics Forum*, vol. 19, no. 3, 2000, pp. 467–478.

[15] K. Kanatani, *Statistical Optimization for Geometric Computation: Theory and Practice*. Elsevier Science, 1996.

[16] W. Sun, C. Bradley, Y. Zhang, and H. T. Loh, "Cloud data modelling employing a unified, non-redundant triangular mesh," *Computer-Aided Design*, vol. 33, no. 2, pp. 183–193, 2001.

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

[18] G. Medioni, M. S. Lee, and C. K. Tang, *A Computational Framework for Segmentation and Grouping*. Elsevier, New York, NY, 2000.

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

[20] G. Taubin, "Estimating the tensor of curvature of a surface from a polyhedral approximation," in *Int. Conf. on Computer Vision*, 1995, pp. 902–907.

[21] G. Guy and G. Medioni, "Inference of surfaces, 3d curves, and junctions from sparse, noisy, 3d data," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 19, no. 11, pp. 1265–1277, 1997.

[22] W. Tong, C. Tang, P. Mordohai, and G. Medioni, "First order augmentation to tensor voting for boundary inference and multiscale analysis in 3d," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 26, no. 5, pp. 594–611, May 2004.

[23] M. Liu, F. Pomerleau, F. Colas, and R. Siegwart, "Normal estimation for pointcloud using gpu based sparse tensor voting," in *IEEE International Conference on Robotics and Biomimetics (ROBIO)*, 2012, pp. 91–96.

[24] R. I. Hartley, "In defense of the eight-point algorithm," *IEEE Trans. on Pattern Analysis and Machine Intelligence*, vol. 19, no. 6, pp. 580–593, 1997.