DOI: 10.1111/cgf.12982
Eurographics Symposium on Geometry Processing 2016
Maks Ovsjanikov and Daniele Panozzo
(Guest Editors)

Volume 35 (2016), Number 5

Mesh Statistics for Robust Curvature Estimation

L. Váša[1,2], P. Vaněček[2], M. Prantl[2], V. Skorkovská[2], P. Martínek[2], I. Kolingerová[1,2]

[1]NTIS – New Technologies for the Information Society, Faculty of Applied Sciences, University of West Bohemia, Plzeň, Czech Republic
[2]Department of Computer Science and Engineering, Faculty of Applied Sciences, University of West Bohemia, Plzeň, Czech Republic

Abstract

While it is usually not difficult to compute principal curvatures of a smooth surface of sufficient differentiability, it is a rather difficult task when only a polygonal approximation of the surface is available, because of the inherent ambiguity of such representation. A number of different approaches has been proposed in the past that tackle this problem using various techniques. Most papers tend to focus on a particular method, while an comprehensive comparison of the different approaches is usually missing.

We present results of a large experiment, involving both common and recently proposed curvature estimation techniques, applied to triangle meshes of varying properties. It turns out that none of the approaches provides reliable results under all circumstances. Motivated by this observation, we investigate mesh statistics, which can be computed from vertex positions and mesh connectivity information only, and which can help in deciding which estimator will work best for a particular case. Finally, we propose a meta-estimator, which makes a choice between existing algorithms based on the value of the mesh statistics, and we demonstrate that such meta-estimator, despite its simplicity, provides considerably more robust results than any existing approach.

Categories and Subject Descriptors (according to ACM CCS): I.3.5 [Computer Graphics]: Computational Geometry and Object Modeling—Geometric algorithms, languages, and systems; curve, surface, solid, and object representations.

1. Introduction

In this paper, we deal with the problem of estimating principal curvatures at vertices of a general triangle mesh. Curvature is a differential property of embedded manifolds. For planar curves, the curvature is a scalar value at every point. There are many equivalent definitions, such as the signed rate of change of unit tangent, the signed rate of change of unit normal, or the signed inverse radius of osculating circle. At any point of a smooth surface, one can obtain a planar curve by intersecting the surface with a plane containing the point and the surface normal, and the curvature of such curve is known as normal curvature. By varying the plane, one obtains two distinct orthogonal orientations where the normal curvature is minimal or maximal, respectively. Curvature of the corresponding intersection curves is known as the first and the second principal curvature κ_1 and κ_2, respectively.

Computing the curvature allows better understanding of the shape. Local properties, such as convexity/concavity, unfoldability and others can be derived from the principal curvatures. Many shape processing algorithms build on the notion of surface curvatures. Typically, shape matching, surface registration, shape classification, surface parameterization and many others build heavily on the availability of surface curvatures.

While unambiguously defined and usually easy to compute in the smooth case, curvature is much more difficult to evaluate when only a polygonal approximation of the surface is available. Directly applied, curvature is zero for inner areas of mesh faces and undefined at vertices and edges. One must therefore interpret the problem differently. In the usual interpretation, the task is to estimate the curvature of the surface that is approximated by the polygonal mesh. This is, however, a difficult task, since generally it is not possible to infer the exact description of such surface based on the knowledge of the vertex positions only, because many different smooth surfaces may correspond to a single discrete representation in general.

Because of the potential usefulness of the curvature and the inherent difficulty of its estimation from a polygonal representation, many approaches emerged in the past decades, using different concepts and approaches to tackle the problem. Typically, algorithms attempt to locally fit some smooth surface of known curvature to the polygonal mesh, or relate some local curvature integrals of the polygonal surface to the curvature of the approximated smooth surface. We review the possible approaches in the Related work section.

Unsurprisingly, the different approaches give different results, depending both on the algorithm, as well as on the nature of the input data. Based on the algorithm design, some curvature estimators work particularly well for some classes of surfaces, such as

© 2016 The Author(s)
Computer Graphics Forum © 2016 The Eurographics Association and John Wiley & Sons Ltd. Published by John Wiley & Sons Ltd.

delivered by EUROGRAPHICS DIGITAL LIBRARY
www.eg.org diglib.eg.org

spherical surfaces, regularly sampled surfaces or surfaces equipped with exact normals. Additionally, some algorithms use information from a wider local neighborhood in order to make the estimation more robust to noise, while sacrificing some of the precision by smoothing the result - an effect that is not always appropriately discussed. Despite the optimistic results usually presented in research papers, currently there is no single curvature estimation algorithm that works best (or even very well) in all scenarios.

For many real-life scenarios, the current gamut of curvature estimators is sufficient and an appropriate algorithm may be chosen for most practical situations. On the other hand, in general mesh processing tools, such as Meshlab (`http://meshlab.sourceforge.net`), OpenFlipper (`http://www.openflipper.org`) or Graphite (`http://alice.loria.fr/software/graphite`), there is usually no information available on the origin or purpose of the particular polygonal mesh at hand that would allow choosing an appropriate curvature estimator. This leads to serious problems, when the estimated curvature is of little use, either by being too noisy, too smoothed or simply too imprecise for the particular purpose.

In this paper, we present results of a large experiment, where a variety of curvature estimators were used on triangle meshes of different properties, such as density, regularity or amount of present noise. Having no winning estimation strategy, we investigate a set of mesh statistics which could help with choosing an appropriate estimator for a dataset of unknown origin. This leads to a concept of a meta-estimator, which first analyzes the data and chooses a proper estimation strategy based on this analysis. We build one such meta-estimator, and show that on average it easily outperforms any of the existing estimators.

The rest of the paper is organized as follows: in the following section we review related work, focusing on the most prominent curvature estimation techniques. In Section 3 we describe the structure (data sources, distortion sources, curvature estimators) of our accuracy experiment. In Section 4 we discuss the results of the experiment. Section 5 introduces different mesh statistics, which will be used in Section 6 to design a curvature meta-estimator. Finally, in Section 7 we discuss some implementation details and Section 8 concludes the paper.

2. Related Work

Our work is closely related to a variety of curvature estimation algorithms proposed in the literature. It is generally possible to distinguish two main approaches: first attempts to somehow estimate the smooth surface that is approximated by the mesh, typically using some sort of regression, and then evaluates the curvature of the fitted surface at some point. The other class of approaches directly evaluates the curvature, or curvature related quantities, of the triangle mesh at hand. In order to obtain reasonable results, these quantities are usually integrated over some domain. Pointwise estimate of curvature is then established as a normalized integral over some small area. Note that while the latter class of approaches actually estimates fundamentally different quantities, we will simply investigate its performance for the same purpose as the first class.

One of the most popular approaches is approximating the local surface and/or its normals using either circles or parabolas, as discussed in [GI04]. Such approach produces an osculating jet (see [CP03]), usually described by a 2×2 tensor, where the eigenvalues correspond to the curvature of the jet at origin. Note that all the variants of this algorithm rely on the availability and accuracy of vertex normals. For a majority of practical purposes, this might lead to problems, when normals have to be estimated and thus become inherently inaccurate [MT04].

A similar approach has been proposed by [ZGYL11] in the context of GPU estimation of curvature. Each triangle of the mesh is first replaced by a Bézier patch (a variant of the so-called PN patch [VPBM01] has been used), for which the curvature is easy to evaluate. Curvature of a vertex is then computed as a weighted sum of curvatures at barycentres of incident triangles, using weights as in vertex normal estimation by Max [Max99].

One drawback of PN triangles is that they are only C^0 continuous across edges. Constructing plain cubic Bézier patches with a higher order of continuity is, however, not always possible. Motivated by this problem, Fünfzig et al. [FMHF08] proposed the PNG1 patch, which is constructed as a blend of three cubic Bézier patches for each triangle, each providing continuity over one edge of the triangle. Curvature properties of such patches have been investigated in [BFRA12], and such approach can be used to obtain per vertex curvature estimates in the same manner as in [ZGYL11].

Another very popular approach to curvature estimation [MDSB02] is based on the cotan discretization of the Laplace-Beltrami operator [Mac49,PP93,DMSB99]. Evaluating the discrete Laplacian on a triangle mesh yields the mean curvature normal vector, from which the magnitude of the mean curvature can be computed easily. Together with a consistent set of normal estimates and an angle defect based [BCM03] estimate of the Gaussian curvature [Xu06], one may reconstruct the principal curvatures.

An approach closely related to the fitting based algorithms has been proposed by Rusinkiewicz [Rus04]. Here, an estimate of the second fundamental form is built for each triangle. These estimates are then rotated to the coordinate system of each vertex and used to obtain a per-vertex estimate as a weighted sum of the per-triangle forms. Principal curvatures are then extracted as eigenvalues of the per-vertex tensors.

An approach based on computing the integrated shape operator (with swapped eigenvalues) over a small area of the mesh has been proposed by Cohen-Steiner and Morvan [CSM03]. Their approach builds on the fact that such operator is non-zero only on edges, and even there its integral can be computed from the dihedral angles. Curvatures are obtained by eigenvalue decomposition of the estimated operator.

Principal curvatures can be also recovered using another formulation of the shape operator proposed by Grinspun et al. [GGRZ06]. Their approach uses mid-edge normals in order to obtain an discretization of the shape operator.

Another shape operator based method has been proposed by Hildebrandt and Polthier [HP11]. In their approach, two generalized versions of the shape operator $\hat{S}$ and $\bar{S}$ are proposed and their local estimates are derived using a local function. Both generalized operators can in turn be converted into the standard shape operator S, from which the curvatures can be extracted.

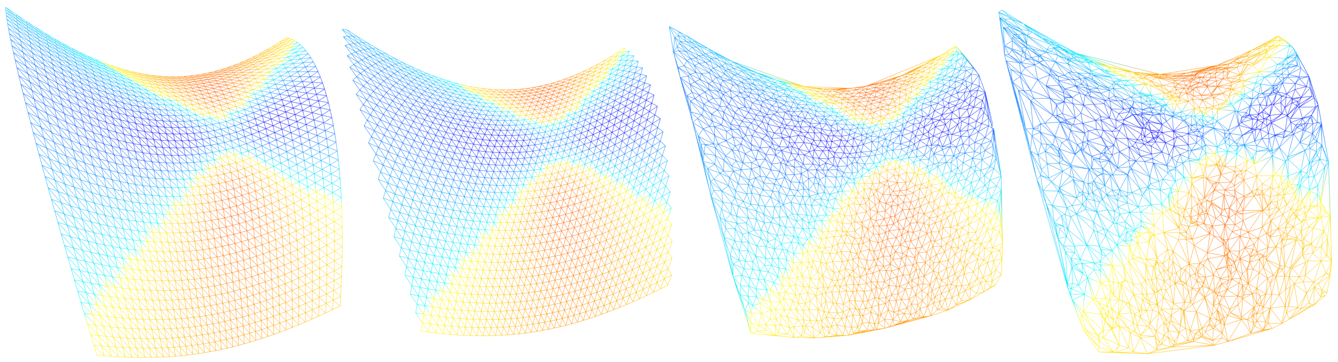

Figure 1: *Used sampling of explicit functions. From left to right: rectangular, equilateral, randomized, random. The color mapping captures mean curvature.*

Pottmann et al. [PWY*07] have derived relations between some integral invariants and mean and Gaussian curvature. The invariants are estimated using a sampled distance function together with a tree-based data structure or Fast Fourier Transform [PWHY09].

Many of the approaches use a certain distance d, up to which the surface influences the pointwise curvature estimation. Kalogerakis et al. [KSNS07] have noted that the choice of d strongly influences the estimation accuracy, especially in the presence of noise. Therefore they propose an adaptive reweighting of the vertices in the neighborhood in order to suppress the influence of outliers. This approach is in its principle most closely related to our proposed meta-algorithm. Our approach, however, takes the idea one step further, adjusting not only the weights of the vertices, but also the method itself.

A common trait of many previous works proposing new curvature estimation algorithms is rather weak testing, focusing mainly on stressing out the usefulness of a particular method rather than providing a comprehensive performance analysis. Yet as one can see, there are many different approaches to curvature estimation, and one may expect that under varying circumstances, different algorithms will perform very differently. In this work, rather than discussing **why** a particular algorithm outperforms others, we focus on a more practical question of **when** a particular algorithm works well. We will therefore not discuss whether or not it is appropriate to use each algorithm in a particular scenario, instead we simply assume that it can be used and evaluate the performance.

Even though similar studies have been done previously [MSR07, KLN08, MFM10, VKB16], our experiment involves a much wider gamut of possible data types, as well as approaches not known at the time of the previous works.

3. Experiment Design

Our aim is to test the accuracy of a variety of curvature estimation algorithms on a wide scale of different types of triangle meshes generated from smooth surfaces, for which the exact curvature is known. We hope to provide the readers with quantitative data that will help them to choose an appropriate estimation algorithm for a particular application. At the same time, we want to map the "landscape" of triangle mesh types in order to be able to comprehend the appropriateness of a particular estimator based on the properties of the input mesh only.

In order to achieve our goal, we have created a software framework which allows performing extensive testing easily. The framework is modular, providing the possibility to add data sources, data distortion sources, curvature estimators and accuracy evaluation routines easily. All available modules can be then combined to form an experiment. Our benchmarking software is freely available and because of its trivial extensibility, it is suitable for testing future curvature estimation approaches. Also, any of the experiment design choices made by us can be easily adjusted in order to match the needs of a particular application.

One can intuitively identify several properties of triangle meshes, which will have an influence on the accuracy of curvature estimation. In particular, the sampling density is a determining factor [HPW05]. Apart from it, sampling regularity also plays an important role. Many of the algorithms use vertex normals for the curvature estimation, however, in most practical cases they are not known and have to be estimated. The accuracy of the estimation then in turn also influences the curvature estimation result. Finally, presence of noise in the data is a key factor. In our experiment, we have tried to generate datasets covering the full spectrum of possibilities, yet each dataset (triangle mesh) is globally homogeneous in the sense of the above listed properties.

We have used three types of data sources: explicit functions in the form $z = f(x,y)$, implicit functions in the form $f(x,y,z) = 0$ and NURBS surfaces. Each type of smooth surface has been converted into an approximating triangle mesh using techniques specified below, and for each mesh vertex, exact principal curvatures at the corresponding smooth surface point were computed and used as reference for evaluating the curvature estimators.

3.1. Explicit Functions

Converting an explicit function to a triangle mesh is straightforward when a sampling and a triangulation of the XY plane is given. In our experiments, we have used four types of sampling:

- regular *rectangular*,

- regular *equilateral* (vertices form equilateral triangles),
- equilateral with noise (vertices of equilateral distribution are shifted in the XY plane by a random vector up to a user-specified length), denoted *randomized* and
- *random*.

The first two types of sampling imply a regular triangulation (rectangles with diagonals or equilateral triangles), while for the later two the 2D Delaunay triangulation [Kv02] is used. Fig. 1 shows an example of the used types of sampling. Finally, the z coordinate is computed for each vertex using the user specified function. Our system then uses symbolic differentiation to compute the exact principal curvatures at each mesh vertex.

In our experiment, we have used the function $z = Ax^2 + By^2 + Cxy + Dx + Ey$ with parameters A, B, C, D and E chosen randomly in the interval $\langle -1, 1\rangle$. We have sampled in the interval $x \in \langle -2, 2\rangle, y \in \langle -2, 2\rangle$ using varying spacing to achieve three levels of sampling density.

3.2. Implicit Functions

Another source of triangle meshes with known curvature is isosurface extraction from implicit surfaces. The character of the output mesh then depends on the particular surface extraction method. The basic Marching cubes [LC87] algorithm is notoriously known for the poor quality of the resulting triangulation, and effort has been invested into finding a better alternative. In our experiments we have used both Marching cubes and Dual contouring [JLSW02], which provides significantly different results (see Fig. 2).

Note that in the typical extraction task, the implicit function is first sampled using a regular grid, and the vertices of the final mesh are then computed from these samples. As a result, they do not necessarily lie on the original smooth surface $f(x,y,z) = 0$. Therefore we use two versions of the isosurface extraction algorithm:

- *inexact*, which works in the usual way. The desired curvature is computed from the implicit function at the off-surface point, i.e. it is in fact **not** the curvature of $f(x,y,z) = 0$, and
- *exact*, where the extraction algorithm has direct access to the underlying implicit function, and is therefore able to place the vertices exactly onto the isosurface. In the case of marching cubes, this translates to first identifying the intersected edges using the regular grid, and then finding the exact intersection point using interval subdivision instead of interpolation.

Our benchmarking system allows using any implicit function of sufficient differentiability. Computing the principal curvatures of the isosurface of an implicit function is straightforward, especially since there is a handy summary of the formulas available [Gol05].

In the experiment, we have used the zero isosurface of the function $A\sin(Bx) + C\sin(Dy) + sin(Ez)$ with the parameters A, B, C, D and E chosen randomly in the interval $\langle 1, 5\rangle$. The isosurface has been extracted in the interval $x \in \langle -2, 2\rangle, y \in \langle -2, 2\rangle, z \in \langle -2, 2\rangle$ using various resolutions of the volume grid.

3.3. NURBS Surfaces

Non-uniform rational Bézier spline (NURBS) [Far91] surfaces are ubiquitous in surface design applications. Since they are smooth parametric surfaces, converting them into triangle meshes is a simple matter of triangulating the parameter domain. In our experiments, we have used all the methods listed above for triangulation of the XY plane in the case of explicit functions.

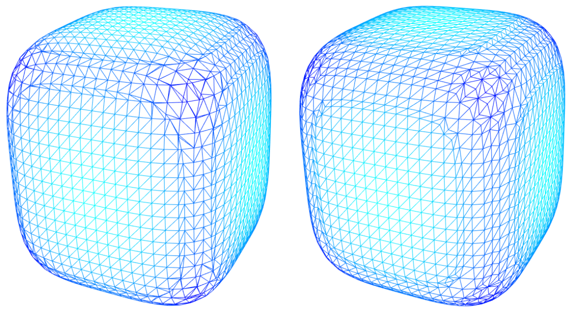
Figure 2: *Isosurface extraction methods. Left: Marching cubes; note the sliver triangles. Right: Dual contouring; note the vertices of high degree.*

In order to generate various surfaces, we use cubic basis functions, random control points in a user-specified range and random weights in the interval $\langle 0, 1\rangle$.

3.4. Mesh Distortion

It has been observed previously that the accuracy of curvature estimation depends strongly on the accuracy of the input data. Some approaches provide highly accurate results for noiseless inputs, however, such algorithms usually provide unreliable results for noisy data. Robustness to noise, on the other hand, usually leads to smoothing, which compromises the accuracy when the noise is actually low. For this reason, we test the estimators both in noiseless scenario, as well as in scenarios when additional random noise is added to the input. In the experiments, we use both uniform noise (which may appear in practice for example due to coordinate quantization during compression) and Gaussian noise (which typically appears as artifact in imprecise scanning, where a multitude of factors influence the result). We have applied noise of varying standard deviation, however, in order to align the results from meshes of different size/tessellation, we relate the standard deviation to the average edge length.

In real data, apart from noisy vertex positions, also the vertex normals are usually not exactly known to the curvature estimators, except for rare cases: for example, data exported from a CAD application might have exact normals attached (we do not discuss such possibility here). Therefore we use estimated vertex normals for all experiments involving estimators that require them. After a short comparative study, we have decided to compute the normals using the algorithm by Max [Max99], since it provides quite stable results in most situations. When working with noisy inputs, we estimate the vertex normals from the noisy vertex positions.

3.5. Tested Estimators

To the best of our knowledge, we have included a to-date widest variety of curvature estimation approaches into our experiment. We are also using some approaches not originally intended for curvature estimation. We have implemented the following estimators, which work as parameterless:

- cotan discretization of Laplace operator as an estimator of mean curvature combined with angle deficiency as estimate of Gaussian curvature, as described in [MDSB02],
- weighted average of per-triangle estimation of shape operator, as described in [Rus04],
- weighted average of PN-triangle curvatures, as described in [ZGYL11],
- weighted average of PNG-1 patch curvatures, as described in [FMHF08].

Apart from those, we have also implemented approaches that work in a local neighborhood. These approaches accept a parameter which determines the radius of the neighborhood, and we have tested various choices in the experiment. In general, we specify the radius as a multiple of the average edge length $\bar{l}$, which leads to reasonable values in most cases. In this class of algorithms, we have implemented the following:

- circle / parabola fit, as described by [GI04],
- parabola fit with adjacent normal, as described in [GI04],
- per-edge estimation of the shape operator (with swapped eigenvalues) as described by [CSM03], and
- generalized shape operators $\bar{S}$ and $\hat{S}$ as described by [HP11].

Finally, we have used a publicly available reference implementations of the following algorithms:

- curvature estimation via integral invariants computed using FFT, as described in [PWHY09, PWY*07], and
- adaptive estimation of the second fundamental form, as described in [KSNS07].

4. Experiment Evaluation

In order to quantify curvature estimation accuracy, we compute the sum of curvature estimation errors $\varepsilon = \|\kappa_1 - \kappa_1^*\| + \|\kappa_2 - \kappa_2^*\|$, where κ_i is the curvature estimation and κ_i^* is the exact curvature. For each estimator, the values ε are averaged over all vertices of each mesh $\mathcal{M}$, only excluding vertices that lie near the border in both topological and geometric sense, because some of the estimators require special handling of border vertices. This way, we obtain the error value $e_i(\mathcal{M}_j)$ of i-th estimator on j-th mesh. The errors are averaged over 10 meshes of each type (combination of source, sampling and triangulation) generated with random parameters. For illustration of the types of input data, please see the supplementary material.

Table 2 shows a summary of the results of our experiment for noiseless data and for data with Gaussian noise of standard deviation $\sigma = 0.03\bar{l}$, Table 1 details the used estimators and their parameters. Note that for the noiseless and noisy experiments, the same 10 meshes were used, with and without noise. Therefore the values in the corresponding cells are comparable. Vertex normals were estimated from the triangle mesh in both cases. Results for uniform noise and Gaussian noise of different strength can be found in the accompanying material, however, Table 2 captures the most prominent trends in the results.

For noiseless data, following trends can be identified:

- local approaches, such as Meyer or Rusinkiewicz, provide the best results, but only if the sampling is regular,
- fitting based schemes, such as the different variants discussed by Goldfeather and Interrante, provide comparably good results, even when the regularity is lower,
- the adaptive approach of Kalogerakis yields also quite good and stable results,
- integrating schemes, such as Cohen-Steiner and Morvan, Hildebrandt or Pottmann, usually provide results of considerably lower accuracy, since they tend to smooth the results.

Naturally the errors with noiseless data are much lower than with noisy data, as can be expected.

For noisy data, on the other hand, the trends are quite different:

- local approaches fail to provide any reasonable estimation of the original curvature,
- fitting approaches provide mediocre results provided that a wider neighborhood is used, otherwise they fail as well,
- the adaptive approach by Kalogerakis also provides unstable, rather mediocre results,
- best results are obtained by integrating schemes, in particular the algorithm by Hildebrandt and Polthier is quite stable when an appropriate width of the used local neighborhood is used.

Note that one may argue that the local approaches measure the curvature of the local features on the noisy surface, and it is thus unfair to measure the error with respect to the curvature of the original smooth surface. This is indeed true, **for the case of noise-free mesh**, which is the case covered above. In many practical scenarios, however, only a noisy measurement of the surface is available, yet the user wants to estimate the curvature of the original smooth one. To reiterate, we do not discuss whether a particular method has been designed for a particular purpose, we merely measure how well it works in a given scenario.

Our objective is to compare the accuracy of the different estimators. One could measure the sum of estimation errors over all the input meshes, however, such an approach leads to a bias towards meshes where the estimation error is high in general (noisy data). A better approach is to relate the inaccuracy of each predictor to some realistically achievable result for each input mesh $\mathcal{M}_i$. Using the minimum estimation error e_{min} over a group of estimators is a good way to obtain a more expressive measure $\hat{e}$ of inaccuracy of a particular estimator

$$\hat{e}_j(\mathcal{M}_i) = \frac{e_j(\mathcal{M}_i) - e_{min}(\mathcal{M}_i)}{e_{min}(\mathcal{M}_i)}, \tag{1}$$

where $e_j(\mathcal{M}_i)$ is the estimation error of the j-th estimator on $\mathcal{M}_i$. Finally, we would like to minimize the average

$$\bar{e}_j = \frac{1}{N}\sum_{i=1}^{N} \hat{e}_j(\mathcal{M}_i), \tag{2}$$

where N is the number of input meshes. In our experiments, we have used

$$e_{min}(\mathcal{M}_i) = \min_j e_j(\mathcal{M}_i), \tag{3}$$

i.e. the minimum over all tested estimators.

As expected, there is no unique winner in the terms of curvature estimation accuracy. One important observation is that those methods that work best in some cases work also very poorly on the rest of the data, providing sometimes results that are worse by multiple orders of magnitude. As a result, the average relative error of tested estimators ranges from $\bar{e} = 6.02$ for the Kalogerakis approach up to $\bar{e} = 12710.9$ for the Goldfeather and Interrante algorithm using the smallest neighborhood and fitting parabolas.

5. Mesh Statistics

As demonstrated in the previous section, the gamut of triangle meshes forms a rather diverse landscape, where different curvature estimation approaches provide results of vastly different accuracy. It is now our aim to investigate *mesh statistics* which would help in deciding which curvature estimation approach to use. A mesh statistic $\mathcal{S}(\mathcal{M})$ is a scalar number which can be determined from the triangle mesh only (vertex positions and connectivity information), i.e. without any knowledge about the original smooth surface.

Apart from being relevant to curvature estimation, such statistic should also possess other properties. In particular, one would expect a statistic to be invariant to similarity transformations, because the choice of proper curvature estimator should depend neither on the unit used to measure the vertex positions, nor on the particular spatial position and orientation of the input triangle mesh. It is also reasonable to have statistics that are independent of the total number of vertices/triangles of the mesh, as long as the character of the mesh is preserved. Therefore most statistics are computed for some local neighborhood in the data (for each vertex, for each triangle, for each edge etc.), yielding a vector $\mathbf{t} \in R^N$, where N is the number of local neighborhoods (vertices, triangles, edges etc.). The final statistic is then obtained using a pooling operator $\mathcal{P}$, i.e. $\mathcal{S}(\mathcal{M}) = \mathcal{P}(\mathbf{t})$. We have tested minimum, maximum, mean, median, standard deviation and variance as a pooling operator.

We have experimented with a number of statistics which intuitively may capture the key characteristics of a triangle mesh. Following statistics are computed in our system:

- edge lengths, normalized by average edge length
- inner angles of triangles
- unsigned dihedral angles ($cos^{-1}(\mathbf{n_i} \cdot \mathbf{n_j})$, where $\mathbf{n_i}$ and $\mathbf{n_j}$ are unit triangle normals of two adjacent triangles) for each inner edge
- signed dihedral angles for each inner edge
- solid angle adjacent to each vertex
- vertex degrees
- ratio of the circumcircle/inscribed circle radius

Additionally, we have used statistics based on a discretization of the Laplace-Beltrami operator. We have used the uniform discretization, which when applied to coordinate function assigns a vector $\mathbf{d_i}$ to each vertex, such that

$$\mathbf{d}_i^u = \frac{1}{\|N(i)\|} \sum_{j \in N(i)} (\mathbf{v}_j - \mathbf{v}_i), \tag{4}$$

where $N(i)$ is the set of vertices in the one-ring neighborhood of the i-th vertex, and $\mathbf{v}_i$ are the 3D coordinates of i-th vertex. The resulting vectors capture the offset (both tangential and normal) of each vertex from the average of its neighbors. Additionally, we have

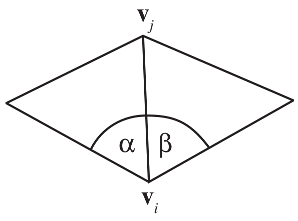

Figure 3: *Angles for the mean value Laplacian.*

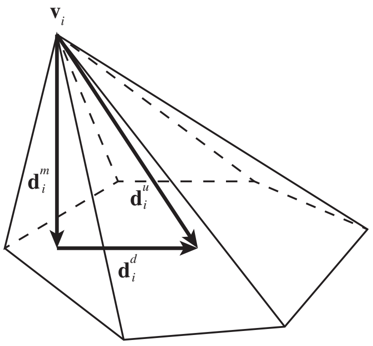

Figure 4: *Schematic of a relation between unit Laplacian vector $\mathbf{d}_i^u$, mean value Laplacian vector $\mathbf{d}_i^m$ and Laplacian vector difference $\mathbf{d}_i^d$.*

used the mean value discretization of the Laplace operator [Flo03], computed as

$$\mathbf{d}_i^m = \sum_{j \in N(i)} \frac{w_{ij}(\mathbf{v}_j - \mathbf{v}_i)}{\sum_j w_{ij}}, \tag{5}$$

where

$$w_{ij} = \frac{tan(\alpha/2) + tan(\beta/2)}{\|\mathbf{v}_i - \mathbf{v}_j\|}, \tag{6}$$

and where α and β are the adjacent angles as depicted in Fig. 3 (for more details, see [Flo03]). The resulting vectors capture the orthogonal offset of each vertex from the plane of its neighbors. Finally, one can extract the tangential offset (i.e. a measure of sampling regularity) of each vertex by subtracting the two Laplacian vectors, obtaining the difference $\mathbf{d}_i^d = \mathbf{d}_i^u - \mathbf{d}_i^m$. For illustration see Fig. 4.

For the Laplacian-based mesh statistics, we use the normalized vector lengths $\|\mathbf{d_i}\|/\bar{l}$ of each vector. The per-vertex vector lengths are then pooled using one of the pooling operators listed above.

Apart from that, we have also used the standard cotan discretization of the Laplace operator [DMSB99], which estimates for each vertex the mean curvature normal vector. Having a matrix representation of the operator L and a vector of all vertex positions $\mathbf{v}$, the Laplace vectors are $\mathbf{d}^c = L\mathbf{v}$. The smoothness of the surface can then be estimated by evaluating the smoothness of $\mathbf{d}^c$. Therefore we have applied the discrete Laplace operator to $\mathbf{d}^c$ again in order to determine its smoothness, obtaining $\mathbf{s} = L(L\mathbf{v}) = L^2\mathbf{v}$. Note that $\mathbf{d}^c$ scales inversely with the mesh, and thus the resulting vectors $\mathbf{s}$ scale with the inverse square of the mesh scaling factor. In order to obtain a scale independent statistic, we multiply the resulting vectors by the squared average edge length:

$$\mathbf{s}^* = \bar{l}^2 L^2 \mathbf{v}. \tag{7}$$

The vector $\mathbf{s}^*$ consists of three component vectors associated to each vertex of the mesh. The component vector lengths are then pooled again using one of the pooling operators.

6. Curvature Meta-Estimator

We now want to construct a meta-estimator, which for each input selects one of the available estimators based on the available statistics. There are different possibilities of how the relation between statistics and estimator accuracy can be used. We wish to have a meta-estimator that is as simple as possible, consisting of only few estimators and using only one or two mesh statistics. In such setting, the task can be interpreted as a classification problem: we select a pair of estimators, and for each mesh, we measure which one of the pair works better, yielding two classes of meshes. The task of the meta-estimator is then to replicate the decision, using only the statistic values.

For example, the estimator no. 7 (Goldfeather & Interrante, see Tab.1) works well for noiseless meshes, while the estimator no. 23 (Hildebrandt & Polthier) achieves good accuracy with noisy data. The $\bar{e}$ of estimator no. 7 is 15.19 and $\bar{e}$ of estimator 23 is 37.99, i.e. neither of the estimators provides reliable results over the whole dataset. We use the smoothness measure $\mathbf{s}^*$ (Eq. (7)) to select one of the two estimators for each mesh. Median is used as pooling operator to obtain a single value for each mesh. A simple thresholding has been used to make the decision, using a threshold $t = 0.609$ learned from a subset (random 35%) of the input meshes. The procedure yields a meta-estimator with $\bar{e} = 1.18$, i.e. reduced by more than an order of magnitude with respect to the better of the constituent estimators, and also considerably better than the result of the best single estimator ($\bar{e} = 6.02$ of Kalogerakis).

Using exhaustive search, we have found no better combination of two estimators, mesh statistic and pooling operator. Fig. 5 shows the accuracy of other mesh statistics in the role of classifier between the two selected estimators. Naturally, better results could be obtained by combining more estimators into the meta-estimator.

The simplest two-level meta-estimator works as follows: it makes a decision based on a threshold of mesh statistic $\mathcal{S}_1$, either to use estimator E_1, or to do another decision, based on a potentially different statistic $\mathcal{S}_2$, between two different estimators E_2 and E_3 (see the structure in Fig. 6). When constructing the meta-estimator, we first establish the threshold of $\mathcal{S}_2$ by learning from a subset of data. Subsequently, the resulting partial meta-estimator is used to determine the threshold for the statistic $\mathcal{S}_1$.

We have again used exhaustive search of all combinations of two mesh statistics and three estimators. The resulting meta-estimator consists of the median of lengths of $\mathbf{s}^*$ in the role of both $\mathcal{S}_1$ and $\mathcal{S}_2$ (with different threshold), and of estimators no. 27 (Meyer) as E_1, 24 (Hildebrandt and Polthier) as E_2 and 26 (Kalogerakis) as E_3. The resulting meta-estimator achieves $\bar{e} = 0.86$.

A three-component meta-estimator roughly represents the limit of what can be reasonably tested using exhaustive searching. The next step, i.e. doing a full 2-level decision tree, has 3 unknown statistics and 4 unknown estimators, and testing all possibilities would be only possible using massive computational power, or by removing some less promising options beforehand. Creating a more sophisticated classifier, where all the statistics will be used as inputs for example for a neural network, is planned as a future work.

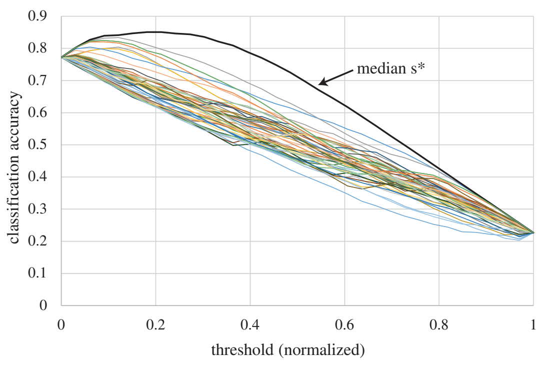

Figure 5: *Accuracy of all tested statistics in selecting the proper estimator (with lower error) out of estimators 23 and 7.*

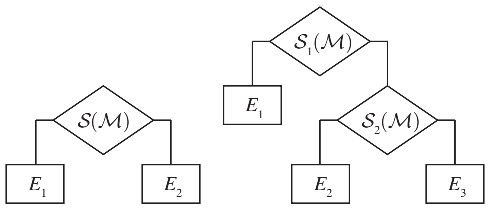
$\mathcal{S}(\mathcal{M})$
E_1
E_2
$\mathcal{S}_1(\mathcal{M})$
E_1
$\mathcal{S}_2(\mathcal{M})$
E_2
E_3

Figure 6: *Structure of the proposed meta-estimators: single level decision(left) and two level decision (right).*

Even if there is some information about the input data available, building a meta-estimator is still beneficial. In particular, when the user knows that the data contain very little noise, or that it is even completely noise-free, there are still considerable differences in the accuracy among the estimators, depending on different properties of the input.

We have used the same procedure as described above, only this time using the noise-free meshes for the experiments (inexact implicit functions were also excluded from the results). For the simplest case of a single choice meta-estimator, the best results were obtained using the Laplacian vector difference vectors $\mathbf{d}^d$ for building the statistic. When mean value is used as a pooling operator, yielding a statistic that decides between estimator 8 (Goldfeather and Interrante, fitting circles, range $6\bar{l}$, $\bar{e} = 5.83$ for noiseless data) and 27 (Meyer, $\bar{e} = 14.59$ for noiseless data), then a meta-estimator with $\bar{e} = 0.45$ is obtained. The statistic captures the regularity of sampling, and decides between an estimator that works well for regular meshes (estimator 27) and an estimator that works well for irregular ones (estimator 8).

Again, a two-choice meta estimator can be built as well. In our experiment, the best choice found by an exhaustive search is making first a choice based on the $\mathbf{d}^d$, again using the mean for pooling. Based on the first choice, either estimator no. 27 is chosen, or another choice is made, based on the standard deviation of mean value Laplacian vector lengths. This statistic captures the sampling density (for densely sampled surfaces, the standard deviation will be low), and it is used to make a choice between estimators 7 and 9 (different versions of Goldfeather and Interrante). The resulting two-choice meta-estimator has $\bar{e} = 0.31$.

7. Implementation Notes

An implementation of our benchmarking software is available at our website `http://graphics.zcu.cz/curvature.html`. It not only allows replicating the results presented here, but also adjusting the experiments to suit the needs of a particular application. In our implementation, an experiment is described by an XML file which specifies the used data sources, distortion procedures, estimation algorithms and evaluation algorithms that should be used. Each module is configurable, and it is therefore easy to design different experiments (using different functions in particular) by simply changing the XML file.

Additionally, thanks to the modular structure of the implementation, it is easy to extend the system by a new data source, distorter, estimator or evaluation algorithm, only by implementing a simple interface. More details can be found in the supplied Programmers guide and Class reference.

We hope that by publishing the benchmarking software, we will make it easier to build more comprehensive curvature estimation experiments in the future. Naturally, we welcome any additions to the software from the community.

8. Conclusion

We have shown results of a largest curvature estimation accuracy experiment to date, involving recent estimators and a large variety of input data types. The results demonstrate conclusively that no single estimator is suitable for all possible input data.

Based on simple mesh statistics, however, it is possible to choose an estimator suited for the properties of a particular input. We have demonstrated that even a very simple meta-estimator, which only chooses between two and three estimators respectively, improves the average imprecision considerably. Our results show promise for future possible improvements, if more complex classifiers, such as neural networks, are used to choose the curvature estimator.

We use the difference $\mathbf{d}^d$ of mean value Laplacian vector and unit Laplacian vector as a measure of sampling irregularity, which is independent of sampling density. It turns out that this measure works well for our purposes, however, we believe that it could be used for other applications in mesh processing as well.

In our work, we assume that the properties of the meshes are uniform for all vertices, which in practice may not always be the case. However, it should be straightforward to extend the meta-estimator so that it uses a local neighborhood to evaluate the mesh statistic and to choose the appropriate estimator on a per-vertex rather than per-mesh basis.

In the future, we would like to expand the experiment by using real world CAD NURBS models, additional sampling schemes and more estimation approaches, such as [CP03]. Another interesting possibility would be to perform a similar experiment and meta-estimator construction for principal curvature directions.

Acknowledgments

The authors would like to thank Yongliang Yang for providing a reference implementation of [PWHY09] and Nikola Kottová, Katka Kubásková and Tomáš Zítka for the help with the implementation. This work was supported by the Czech Ministry of Education, Youth and Sports - the project LO1506 and University spec. research - 1311; by the UWB grant SGS-2016-013 Advanced Graphical and Computing Systems; and by the 1st Internal grant scheme of DCSE+P2, 2015.

References

[BCM03] BORRELLI V., CAZALS F., MORVAN J.-M.: On the angular defect of triangulations and the pointwise approximation of curvatures. *Comput. Aided Geom. Des. 20*, 6 (Sept. 2003), 319–341. 2

[BFRA12] BOSCHIROLI M., FÜNFZIG C., ROMANI L., ALBRECHT G.: G1 rational blend interpolatory schemes: A comparative study. *Graph. Models 74*, 1 (Jan. 2012), 29–49. 2

[CP03] CAZALS F., POUGET M.: Estimating differential quantities using polynomial fitting of osculating jets. In *Proceedings of the 2003 Eurographics/ACM SIGGRAPH Symposium on Geometry Processing* (Aire-la-Ville, Switzerland, Switzerland, 2003), SGP '03, Eurographics Association, pp. 177–187. 2, 8

[CSM03] COHEN-STEINER D., MORVAN J.-M.: Restricted delaunay triangulations and normal cycle. In *Proceedings of the Symposium on Computational Geometry* (2003), SCG '03, ACM, pp. 312–321. 2, 5, 10

[DMSB99] DESBRUN M., MEYER M., SCHRÖDER P., BARR A. H.: Implicit fairing of irregular meshes using diffusion and curvature flow. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1999), SIGGRAPH '99, ACM Press/Addison-Wesley Publishing Co., pp. 317–324. 2, 6

[Far91] FARIN G. E.: *NURBS for Curve and Surface Design*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA, 1991. 4

[Flo03] FLOATER M. S.: Mean value coordinates. *Comput. Aided Geom. Des. 20*, 1 (Mar. 2003), 19–27. 6

[FMHF08] FÜNFZIG C., MÜLLER K., HANSFORD D., FARIN G.: Png1 triangles for tangent plane continuous surfaces on the gpu. In *Proceedings of Graphics Interface 2008* (Toronto, Ont., Canada, Canada, 2008), GI '08, Canadian Information Processing Society, pp. 219–226. 2, 5, 10

[GGRZ06] GRINSPUN E., GINGOLD Y., REISMAN J., ZORIN D.: Computing discrete shape operators on general meshes. *Computer Graphics Forum (Eurographics) 25*, 3 (2006), 547–556. 2

[GI04] GOLDFEATHER J., INTERRANTE V.: A novel cubic-order algorithm for approximating principal direction vectors. *ACM Trans. Graph. 23*, 1 (Jan. 2004), 45–63. 2, 5, 10

[Gol05] GOLDMAN R.: Curvature formulas for implicit curves and surfaces. *Comput. Aided Geom. Des. 22*, 7 (Oct. 2005), 632–658. 4

[HP11] HILDEBRANDT K., POLTHIER K.: Generalized shape operators on polyhedral surfaces. *Comput. Aided Geom. Des. 28*, 5 (June 2011), 321–343. 2, 5, 10

[HPW05] HILDEBRANDT K., POLTHIER K., WARDETZKY M.: On the convergence of metric and geometric properties of polyhedral surfaces. *GEOMETRIAE DEDICATA 123* (2005), 89–112. 3

data	sampling	triangulation	1	2	3	4	5	6	7	8	9	10	11	12	13	14	15	16	17	18	19	20	21	22	23	24	25	26	27	28	29	30	31	32	33	34
explicit	coarse	rectangular	0.078	0.22	0.361	0.473	0.019	0.004	0.013	0.025	0.039	0.008	0.036	0.075	0.115	0.012	0.041	0.089	0.149	0.85	0.84	0.848	0.855	0.056	0.131	0.238	0.347	0.022	0.003	0.794	0.083	0.128	0.372	0.812	0.009	0.009
		equilateral	0.06	0.175	0.304	0.415	0.021	0.004	0.012	0.024	0.037	0.006	0.025	0.053	0.081	0.01	0.037	0.081	0.136	0.897	0.889	0.884	0.884	0.048	0.102	0.194	0.296	0.018	0.003	0.891	0.087	0.111	0.341	0.812	0.007	0.008
		randomized	0.062	0.209	0.407	0.536	0.11	0.023	0.016	0.024	0.038	0.035	0.039	0.059	0.089	0.019	0.035	0.071	0.117	0.85	0.84	0.837	0.837	0.056	0.108	0.206	0.309	0.028	0.113	0.891	0.087	0.105	0.323	0.728	0.041	0.046
		random	0.083	0.236	0.416	0.524	0.514	0.066	0.034	0.04	0.057	0.071	0.078	0.098	0.13	0.044	0.048	0.1	0.172	0.831	0.805	0.812	0.819	0.081	0.127	0.229	0.332	0.052	0.284	0.795	0.082	0.118	0.349	0.738	0.085	0.217
	medium	rectangular	0.039	0.015	0.026	0.044	0.018	2E-04	7E-04	0.001	0.003	4E-04	0.002	0.005	0.009	0.002	0.002	0.005	0.008	1.143	1.086	1.075	1.07	0.075	0.018	0.025	0.054	0.001	2E-04	1.985	0.175	0.047	0.174	0.683	5E-04	5E-04
		equilateral	0.047	0.015	0.021	0.035	0.023	2E-04	7E-04	0.001	0.003	3E-04	0.002	0.003	0.006	0.001	0.002	0.004	0.008	1.179	1.13	1.114	1.106	0.069	0.018	0.022	0.05	0.001	2E-04	2.204	0.196	0.046	0.164	0.693	5E-04	5E-04
		randomized	0.038	0.015	0.022	0.037	0.132	0.022	0.007	0.004	0.004	0.027	0.012	0.01	0.011	0.017	0.008	0.007	0.009	1.196	1.133	1.115	1.107	0.112	0.023	0.023	0.052	0.007	0.138	2.18	0.192	0.045	0.164	0.691	0.043	0.044
		random	0.06	0.062	0.22	0.382	0.528	0.073	0.022	0.013	0.012	0.055	0.03	0.029	0.035	0.05	0.015	0.011	0.014	1.176	1.091	1.068	1.057	0.167	0.042	0.04	0.067	0.032	0.347	1.993	0.179	0.088	0.238	0.659	0.091	0.206
	fine	rectangular	0.076	0.022	0.012	0.008	1.644	4E-06	2E-05	4E-05	1E-04	1E-05	5E-05	1E-04	3E-04	2E-04	1E-04	2E-04	9E-04	1.899	1.824	1.803	1.799	0.121	0.033	0.024	0.036	2E-04	8E-06	7.798	0.685	0.084	0.119	1.949	2E-05	1.644
		equilateral	0.084	0.023	0.014	0.009	1.654	5E-06	2E-05	4E-05	1E-04	7E-06	4E-05	8E-05	2E-04	2E-04	7E-05	3E-04	0.001	1.93	1.866	1.843	1.838	0.104	0.031	0.022	0.035	2E-04	7E-06	8.499	0.747	0.09	0.12	2.07	2E-05	1.654
		randomized	0.061	0.019	0.01	0.007	1.651	0.022	0.007	0.003	0.002	0.029	0.009	0.005	0.003	0.019	0.008	0.005	0.005	1.962	1.87	1.848	1.839	0.377	0.051	0.027	0.038	0.007	0.162	8.524	0.753	0.09	0.118	2.055	0.048	1.65
		random	0.102	0.16	0.642	1.118	1.525	0.089	0.024	0.011	0.007	0.064	0.025	0.015	0.012	0.064	0.019	0.011	0.009	1.979	1.845	1.809	1.796	0.479	0.117	0.057	0.064	0.039	0.43	7.836	0.73	0.277	0.399	1.889	0.113	1.425
implicit exact	coarse	MC	0.46	0.985	1.418	1.681	49.54	0.259	0.34	0.457	0.649	0.516	0.653	0.78	1.046	0.353	3.212	4.998	5.706	1.916	1.873	1.871	1.903	0.346	0.734	1.072	1.407	0.39	0.739	0.872	0.588	1.293	1.9	1.966	0.454	2.361
	medium		0.173	0.258	0.456	0.661	4.848	0.132	0.111	0.156	0.213	0.218	0.257	0.341	0.43	0.105	0.145	0.884	1.263	2.042	1.929	1.941	1.956	0.219	0.176	0.327	0.505	0.107	0.856	1.258	0.432	1	1.852	2.06	0.23	0.83
	fine		0.114	0.039	0.04	0.061	1.215	0.254	0.078	0.044	0.037	0.293	0.124	0.088	0.084	0.228	0.067	0.048	0.047	1.669	1.54	1.513	1.523	0.294	0.058	0.036	0.073	0.05	1.245	3.621	0.338	0.262	0.747	2.181	0.451	0.637
	coarse	DC	0.322	0.655	1.004	1.193	0.732	0.154	0.229	0.355	0.521	0.27	0.433	0.625	0.824	0.227	0.97	3.15	1.862	1.572	1.513	1.509	1.529	0.254	0.479	0.738	1.016	0.775	0.462	1.318	0.275	0.575	1.23	1.528	0.237	0.315
	medium		0.117	0.193	0.321	0.456	0.47	0.072	0.068	0.102	0.14	0.097	0.134	0.2	0.269	0.059	0.182	1.182	0.829	1.805	1.72	1.708	1.707	0.16	0.121	0.212	0.334	0.171	0.381	2.014	0.257	0.406	0.961	1.398	0.106	0.135
	fine		0.136	0.093	0.102	0.137	1.06	0.157	0.059	0.04	0.042	0.116	0.051	0.056	0.079	0.091	0.034	0.036	0.054	4.031	3.885	3.846	3.815	0.336	0.063	0.058	0.129	0.227	0.786	4.298	0.418	0.416	1.136	3.04	0.144	0.256
implicit inexact	coarse	MC	0.466	1.004	1.448	1.714	63.88	1.1	0.563	0.594	0.761	0.709	0.695	0.836	1.1	0.737	3.06	5.026	5.264	1.927	1.897	1.9	1.935	0.33	0.749	1.102	1.446	0.522	1.237	0.891	0.595	1.319	1.65	1.999	0.658	4.837
	medium		0.185	0.257	0.457	0.663	8.483	0.603	0.239	0.193	0.226	0.489	0.295	0.351	0.435	0.324	0.188	0.457	4.406	2.048	1.929	1.942	1.958	0.22	0.172	0.326	0.505	0.206	1.226	1.259	0.432	1.003	1.548	2.065	0.534	3.051
	fine		0.123	0.042	0.04	0.061	1.807	0.599	0.209	0.11	0.073	0.385	0.134	0.083	0.077	0.265	0.075	0.047	0.043	1.674	1.54	1.513	1.522	0.297	0.057	0.036	0.072	0.121	1.048	3.621	0.338	0.263	0.577	2.183	0.436	0.671
	coarse	DC	0.371	0.674	0.92	1.08	1.647	0.286	0.274	0.363	0.504	0.314	0.416	0.589	0.769	0.383	1.29	2.085	1.313	1.595	1.521	1.504	1.512	0.315	0.509	0.713	0.964	0.83	0.792	1.384	0.356	0.694	0.842	1.485	0.323	0.428
	medium		0.127	0.205	0.329	0.462	1.143	0.167	0.097	0.113	0.147	0.138	0.14	0.201	0.269	0.103	0.147	0.617	8.703	1.811	1.722	1.708	1.707	0.167	0.127	0.216	0.336	0.177	0.657	2.019	0.262	0.407	0.858	1.398	0.148	0.206
	fine		0.141	0.115	0.123	0.155	1.532	0.246	0.09	0.054	0.048	0.166	0.061	0.059	0.08	0.128	0.042	0.04	0.056	4.032	3.884	3.846	3.815	0.339	0.063	0.059	0.13	0.229	1.163	4.307	0.42	0.417	1.042	3.039	0.194	0.314
nurbs	coarse	rectangular	0.056	0.082	0.151	0.217	0.027	0.003	0.009	0.018	0.03	0.011	0.044	0.09	0.146	0.008	0.023	0.047	0.081	1.261	1.212	1.206	1.198	0.105	0.058	0.135	0.277	0.01	0.002	4.614	0.42	0.094	0.263	0.799	0.004	0.004
		equilateral	0.047	0.064	0.12	0.175	0.024	0.002	0.006	0.012	0.02	0.006	0.025	0.051	0.086	0.006	0.017	0.036	0.061	1.233	1.18	1.174	1.171	0.102	0.049	0.115	0.251	0.008	0.001	5.001	0.448	0.087	0.229	0.849	0.003	0.003
		randomized	0.048	0.068	0.123	0.178	0.174	0.041	0.019	0.018	0.023	0.049	0.042	0.059	0.088	0.025	0.021	0.037	0.061	1.229	1.166	1.158	1.156	0.152	0.054	0.116	0.254	0.024	0.174	4.993	0.451	0.086	0.224	0.83	0.063	0.072
		random	0.159	0.396	0.814	1.049	0.809	0.11	0.049	0.049	0.067	0.099	0.105	0.148	0.227	0.067	0.038	0.072	0.124	1.448	1.329	1.29	1.265	0.25	0.133	0.243	0.411	0.062	0.545	4.673	0.43	0.13	0.363	0.846	0.131	0.324
	medium	rectangular	0.054	0.041	0.074	0.116	0.027	0.001	0.003	0.006	0.011	0.004	0.015	0.032	0.056	0.004	0.008	0.016	0.027	1.559	1.488	1.464	1.438	0.114	0.04	0.085	0.192	0.004	8E-04	4.583	0.417	0.154	0.395	1.043	0.001	0.002
		equilateral	0.046	0.033	0.059	0.095	0.026	7E-04	0.002	0.004	0.007	0.002	0.009	0.019	0.034	0.003	0.006	0.012	0.02	1.524	1.454	1.43	1.408	0.107	0.036	0.074	0.177	0.003	4E-04	4.951	0.452	0.141	0.362	1.08	1E-03	0.001
		randomized	0.045	0.035	0.062	0.098	0.219	0.046	0.017	0.012	0.012	0.044	0.026	0.028	0.039	0.025	0.013	0.014	0.021	1.531	1.448	1.421	1.399	0.211	0.044	0.078	0.18	0.016	0.205	4.965	0.455	0.139	0.358	1.065	0.066	0.089
		random	0.127	0.329	0.73	1.003	0.671	0.111	0.04	0.03	0.034	0.086	0.066	0.081	0.119	0.07	0.025	0.027	0.048	1.673	1.552	1.503	1.467	0.334	0.101	0.155	0.27	0.047	0.611	4.642	0.434	0.203	0.471	1.037	0.131	0.334
	fine	rectangular	0.056	0.021	0.024	0.036	1.408	2E-04	6E-04	0.001	0.002	7E-04	0.003	0.006	0.011	0.002	0.002	0.003	0.006	1.766	1.692	1.67	1.655	0.121	0.033	0.048	0.105	0.001	1E-04	4.578	0.427	0.237	0.52	1.219	3E-04	1.402
		equilateral	0.048	0.018	0.019	0.029	1.5	1E-04	4E-04	8E-04	0.002	4E-04	0.002	0.003	0.006	0.001	0.001	0.002	0.005	1.729	1.656	1.636	1.621	0.115	0.031	0.044	0.099	0.001	8E-05	4.907	0.452	0.221	0.483	1.246	2E-04	1.495
		randomized	0.045	0.019	0.021	0.03	1.472	0.049	0.015	0.008	0.006	0.043	0.017	0.012	0.012	0.027	0.011	0.008	0.008	1.742	1.658	1.634	1.617	0.352	0.05	0.048	0.102	0.01	0.229	4.872	0.446	0.218	0.48	1.238	0.07	1.394
		random	0.137	0.361	0.823	1.166	1.436	0.118	0.036	0.02	0.017	0.08	0.041	0.039	0.05	0.075	0.023	0.015	0.016	1.87	1.747	1.709	1.686	0.53	0.127	0.113	0.162	0.041	0.677	4.618	0.478	0.293	0.573	1.212	0.137	1.192
explicit	coarse	rectangular	0.125	0.23	0.367	0.477	3.779	0.715	0.299	0.173	0.122	0.437	0.135	0.106	0.128	0.392	0.132	0.119	0.163	0.864	0.84	0.847	0.855	0.081	0.134	0.24	0.35	0.062	1.601	0.997	0.154	0.135	0.372	0.812	0.317	0.613
		equilateral	0.115	0.187	0.31	0.419	3.517	0.647	0.27	0.155	0.109	0.385	0.116	0.083	0.094	0.357	0.116	0.105	0.146	0.92	0.892	0.883	0.883	0.085	0.106	0.196	0.299	0.057	1.405	1.086	0.16	0.119	0.342	0.813	0.285	0.582
		randomized	0.116	0.224	0.423	0.559	4.536	0.743	0.298	0.169	0.118	0.481	0.157	0.106	0.112	0.395	0.126	0.102	0.141	0.862	0.839	0.836	0.838	0.083	0.111	0.208	0.311	0.085	1.774	1.118	0.167	0.114	0.324	0.731	0.41	0.821
		random	0.216	0.313	0.46	0.55	66.92	2.218	0.808	0.454	0.301	2.423	0.695	0.368	0.285	13.47	2.097	2.916	0.526	0.837	0.806	0.818	0.824	0.114	0.148	0.248	0.349	0.454	5.077	1.066	0.163	0.126	0.351	0.74	1.397	5.401
	medium	rectangular	0.426	0.148	0.092	0.081	16.63	2.906	1.203	0.676	0.441	1.757	0.489	0.231	0.14	1.57	0.47	0.239	0.151	1.373	1.114	1.081	1.07	0.424	0.066	0.038	0.06	0.175	6.986	2.007	0.2	0.056	0.178	0.685	1.292	3.806
		equilateral	0.474	0.165	0.096	0.078	15.97	2.666	1.098	0.615	0.4	1.544	0.428	0.202	0.122	1.43	0.418	0.208	0.131	1.475	1.171	1.124	1.11	0.51	0.08	0.04	0.057	0.181	6.334	2.231	0.221	0.056	0.165	0.693	1.14	3.827
		randomized	0.474	0.164	0.097	0.081	19.84	3.012	1.206	0.669	0.432	1.928	0.581	0.28	0.169	1.547	0.446	0.233	0.153	1.463	1.166	1.124	1.11	0.505	0.078	0.04	0.059	0.311	7.649	2.212	0.22	0.055	0.165	0.689	1.655	4.69
		random	0.592	0.351	0.447	0.567	61.64	8.57	3.177	1.842	1.269	12.33	4.55	2.269	1.349	60.87	9.422	12.08	5.319	1.382	1.112	1.079	1.069	0.495	0.091	0.063	0.083	1.889	21.24	2.036	0.211	0.097	0.235	0.666	6.044	16.6
	fine	rectangular	3.224	1.092	0.584	0.382	1.647	19.58	7.931	4.448	2.896	11.9	3.261	1.524	0.89	10.69	3.122	1.581	1.003	4.381	1.941	1.884	1.842	3.423	0.516	0.185	0.11	1.177	49.6	7.834	0.703	0.1	0.123	1.938	9.206	1.645
		equilateral	3.561	1.216	0.651	0.427	1.654	17.41	6.968	3.88	2.517	9.99	2.714	1.265	0.741	9.244	2.648	1.314	0.823	5.132	2.007	1.919	1.901	4.197	0.625	0.223	0.126	1.22	43.21	8.513	0.758	0.106	0.125	2.066	7.798	1.654
		randomized	3.618	1.206	0.654	0.431	2.008	19.99	7.763	4.265	2.745	12.65	3.692	1.759	1.037	10.22	2.848	1.476	0.97	4.938	1.987	1.917	1.899	4.042	0.605	0.215	0.123	2.085	51.82	8.531	0.765	0.106	0.119	2.055	11.72	1.819
		random	4.387	2.329	2.392	2.675	10.2	54.32	19.9	11.41	8	84.7	45.68	30.93	21.78	384.2	193	81.83	261.8	4.577	1.964	1.895	1.876	3.922	0.605	0.233	0.148	12.98	137.6	7.854	0.764	0.294	0.39	1.903	40.63	6.557
implicit exact	coarse	MC	0.599	1.044	1.443	1.693	364.8	2.041	1.005	0.849	0.929	5.143	1.675	1.226	1.323	26.77	8.416	6.086	9.234	1.933	1.885	1.885	1.916	0.446	0.789	1.126	1.454	0.866	4.634	0.97	0.605	1.298	1.9	1.967	1.7	8.377
	medium		0.696	0.565	0.68	0.823	168	6.355	2.892	1.935	1.494	25.76	8.883	4.467	2.792	141.4	50.29	20.15	12.88	2.158	1.951	1.956	1.973	0.517	0.272	0.401	0.571	1.761	14.4	1.286	0.445	1.005	1.856	2.062	5.132	21.88
	fine		2.552	1.482	1.218	1.094	66.37	26.47	11.76	7.889	6.148	136.1	71.27	47.27	33.53	774.6	240.8	210.4	120.7	2.651	1.654	1.57	1.567	1.668	0.311	0.159	0.158	7.213	62.32	3.643	0.363	0.279	0.754	2.184	22.64	23.96
	coarse	DC	0.373	0.674	1.012	1.196	5.134	0.677	0.373	0.406	0.544	0.531	0.472	0.637	0.831	1.026	1.223	2.365	2.146	1.591	1.511	1.509	1.528	0.303	0.484	0.742	1.019	0.8	1.925	1.583	0.358	0.581	1.232	1.527	0.471	0.95
	medium		0.396	0.314	0.396	0.508	14.3	1.889	0.729	0.425	0.32	1.181	0.402	0.3	0.317	1.013	0.36	0.592	0.809	1.952	1.73	1.707	1.704	0.409	0.151	0.219	0.337	0.335	5.685	2.069	0.295	0.412	0.962	1.397	1.097	3.147
	fine		1.453	0.936	0.723	0.643	50.3	7.442	2.735	1.474	0.954	4.475	1.245	0.603	0.4	3.757	1.037	0.531	0.395	4.403	3.888	3.839	3.801	1.517	0.242	0.114	0.152	0.908	24.26	4.359	0.433	0.433	1.17	3.04	4.327	13.01
implicit inexact	coarse	MC	0.606	1.063	1.47	1.725	331	2.423	1.117	0.912	0.985	5.109	1.628	1.235	1.351	25.82	8.246	7.331	4.83	1.945	1.907	1.913	1.946	0.437	0.807	1.158	1.488	0.93	4.892	0.986	0.613	1.324	1.651	1.999	1.768	9.119
	medium		0.696	0.562	0.676	0.819	168.3	6.598	3.033	2.025	1.549	25.16	8.731	4.402	2.662	164.7	612.8	17	17.18	2.165	1.95	1.958	1.975	0.519	0.27	0.401	0.573	1.784	14.51	1.281	0.446	1.009	1.55	2.066	5.167	21.99
	fine		2.542	1.506	1.252	1.135	66.23	26.6	11.91	8.006	6.239	136.5	72.81	48.65	33.62	551.1	239.7	126.6	107.3	2.653	1.657	1.569	1.567	1.666	0.312	0.16	0.159	7.139	62.33	3.624	0.36	0.277	0.585	2.176	22.6	23.88
	coarse	DC	0.42	0.692	0.929	1.085	5.573	0.748	0.415	0.418	0.53	0.564	0.461	0.603	0.777	0.704	1.481	1.428	1.673	1.611	1.523	1.505	1.513	0.363	0.514	0.716	0.967	0.849	2.116	1.63	0.43	0.7	0.843	1.484	0.54	1.078
	medium		0.402	0.323	0.401	0.513	14.47	1.909	0.737	0.432	0.326	1.193	0.407	0.303	0.318	1	0.365	0.699	0.938	1.953	1.732	1.707	1.703	0.417	0.156	0.222	0.339	0.339	5.779	2.086	0.301	0.414	0.858	1.396	1.122	3.199
	fine		1.444	0.927	0.719	0.638	49.72	7.363	2.71	1.462	0.946	4.449	1.233	0.599	0.395	3.736	0.994	0.527	0.389	4.407	3.896	3.839	3.801	1.521	0.241	0.115	0.154	0.898	24.01	4.314	0.429	0.437	1.039	3.029	4.312	12.83
nurbs	coarse	rectangular	0.422	0.174	0.174	0.223	15.77	2.823	1.178	0.662	0.434	1.696	0.501	0.275	0.234	1.517	0.467	0.249	0.189	1.513	1.233	1.206	1.195	0.442	0.096	0.142	0.28	0.173	6.73	5.457	0.755	0.167	0.263	0.793	1.254	3.452
		equilateral	0.451	0.174	0.154	0.186	16.36	2.922	1.203	0.676	0.442	1.74	0.504	0.264	0.198	1.645	0.484	0.25	0.179	1.542	1.2	1.176	1.172	0.488	0.093	0.123	0.253	0.177	6.931	5.813	0.778	0.161	0.23	0.848	1.296	3.689
		randomized	0.463	0.177	0.166	0.2	20.56	3.318	1.335	0.743	0.484	2.159	0.662	0.35	0.254	1.823	0.529	0.291	0.215	1.519	1.175	1.162	1.159	0.497	0.091	0.123	0.255	0.314	8.405	5.863	0.801	0.166	0.23	0.836	1.886	4.699
		random	0.791	0.72	1.041	1.224	72.51	10.12	3.845	2.219	1.577	15.26	5.886	3.12	2.19	28.14	87.65	252.6	19.27	1.64	1.344	1.302	1.296	0.515	0.182	0.272	0.429	2.143	26.01	6.012	0.848	0.218	0.368	0.832	7.371	19.11
	medium	rectangular	0.841	0.297	0.191	0.174	35.64	6.481	2.662	1.496	0.973	3.819	1.072	0.525	0.338	3.421	1.017	0.521	0.335	2.064	1.533	1.47	1.436	0.84	0.14	0.111	0.201	0.352	15.46	4.685	0.493	0.175	0.396	1.048	2.858	8.489
		equilateral	0.932	0.323	0.196	0.164	36.47	6.536	2.661	1.495	0.973	3.871	1.082	0.524	0.33	3.667	1.058	0.525	0.332	2.16	1.52	1.443	1.41	0.969	0.157	0.108	0.188	0.379	15.7	5.029	0.52	0.162	0.362	1.072	2.908	8.95
		randomized	0.932	0.323	0.201	0.172	44.51	7.482	2.958	1.64	1.06	4.786	1.431	0.699	0.44	4.002	1.12	0.582	0.391	2.12	1.505	1.432	1.401	0.951	0.149	0.107	0.191	0.666	19.01	5.041	0.525	0.167	0.363	1.067	4.35	10.95
		random	1.485	1.188	1.464	1.61	81.4	20.77	7.675	4.363	3.058	27.64	11.49	7.252	4.779	109.2	26.57	27.43	10.11	2.135	1.593	1.524	1.5	0.97	0.21	0.203	0.301	4.295	51.8	4.835	0.536	0.252	0.494	1.041	14.84	27.21
	fine	rectangular	2.164	0.731	0.403	0.273	24.51	17.66	7.224	4.037	2.616	10.31	2.842	1.339	0.792	9.231	2.714	1.375	0.873	3.322	1.858	1.711	1.667	2.146	0.328	0.144	0.146	0.935	41.66	4.608	0.44	0.243	0.519	1.215	7.953	7.698
		equilateral	2.373	0.799	0.439	0.295	19.34	17.57	7.055	3.934	2.551	10.28	2.826	1.329	0.783	9.741	2.76	1.353	0.833	3.583	1.865	1.686	1.639	2.431	0.37	0.155	0.148	0.964	42.16	4.914	0.46	0.229	0.489	1.25	7.936	6.342
		randomized	2.403	0.799	0.441	0.3	22.29	20.02	7.825	4.305	2.771	12.68	3.744	1.794	1.062	10.74	2.949	1.497	0.963	3.495	1.849	1.679	1.633	2.427	0.359	0.152	0.148	1.775	50.69	4.9	0.459	0.226	0.48	1.235	11.92	8.369
		random	3.868	2.562	2.607	2.68	35.75	55.38	21.23	12.46	8.91	80.12	43.02	30.4	21.81	331.5	133	86.84	48.13	3.414	1.909	1.768	1.738	2.493	0.415	0.233	0.231	12.24	132.9	4.602	0.497	0.299	0.561	1.207	38.89	17.69

Table 2: *Results summary. For column legend see Table 1.*

[JLSW02] JU T., LOSASSO F., SCHAEFER S., WARREN J.: Dual contouring of hermite data. *ACM Trans. Graph. 21*, 3 (July 2002), 339–346. 4

[KLN08] KERAUTRET B., LACHAUD J.-O., NAEGEL B.: Comparison of Discrete Curvature Estimators and Application to Corner Detection. In *4th International Symposium on Advances in Visual Computing - ISVC 2008* (Las Vegas, United States, 2008), et al. G. B., (Ed.), vol. 5358, Springer, pp. 710–719. 3

[KSNS07] KALOGERAKIS E., SIMARI P., NOWROUZEZAHRAI D., SINGH K.: Robust statistical estimation of curvature on discretized surfaces. In *Proceedings of the Fifth Eurographics Symposium on Geometry Processing* (2007), SGP '07, pp. 13–22. 3, 5, 10

[Kv02] KOLINGEROVÁ I., ŽALIK B.: Improvements to randomized incremental delaunay insertion. *Computers & Graphics 26*, 3 (2002), 477–490. 4

[LC87] LORENSEN W. E., CLINE H. E.: Marching cubes: A high resolution 3d surface construction algorithm. *SIGGRAPH Comput. Graph. 21*, 4 (Aug. 1987), 163–169. 4

[Mac49] MACNEAL R. H.: *The solution of partial differential equations by means of electrical networks*. PhD thesis, California Institute of Technology, 1949. 2

[Max99] MAX N.: Weights for computing vertex normals from facet normals. *J. Graph. Tools 4*, 2 (Mar. 1999), 1–6. 2, 4

[MDSB02] MEYER M., DESBRUN M., SCHRÖDER P., BARR A.: Discrete Differential Geometry Operators for Triangulated 2-Manifolds. In *International Workshop on Visualization and Mathematics* (2002). 2, 5, 10

[MFM10] MESMOUDI M. M., FLORIANI L. D., MAGILLO P.: Discrete curvature estimation methods for triangulated surfaces. In *WADGMM* (2010), Köthe U., Montanvert A., Soille P., (Eds.), vol. 7346 of *Lecture Notes in Computer Science*, Springer, pp. 28–42. 3

[MSR07] MAGID E., SOLDEA O., RIVLIN E.: A comparison of gaussian and mean curvature estimation methods on triangular meshes of range image data. *Comput. Vis. Image Underst. 107*, 3 (Sept. 2007), 139–159. 3

[MT04] MORVAN J.-M., THIBERT B.: Approximation of the normal vector field and the area of a smooth surface. *Discrete & Computational Geometry 32*, 3 (2004), 383–400. 2

[PP93] PINKALL U., POLTHIER K.: Computing discrete minimal surfaces and their conjugates. *Experimental Mathematics 2* (1993), 15–36. 2

[PWHY09] POTTMANN H., WALLNER J., HUANG Q.-X., YANG Y.-L.: Integral invariants for robust geometry processing. *Comput. Aided Geom. Des. 26*, 1 (Jan. 2009), 37–60. 3, 5, 8, 10

[PWY*07] POTTMANN H., WALLNER J., YANG Y.-L., LAI Y.-K., HU S.-M.: Principal curvatures from the integral invariant viewpoint. *Computer Aided Geometric Design 24*, 8-9 (2007), 428–442. 3, 5

[Rus04] RUSINKIEWICZ S.: Estimating curvatures and their derivatives on triangle meshes. In *Symposium on 3D Data Processing, Visualization, and Transmission* (2004). 2, 5, 10

[VKB16] VÁŠA L., KÜHNERT T., BRUNNETT G.: Multivariate analysis of curvature estimators. *Computer Aided Design and Applications, to appear* (2016). 3

[VPBM01] VLACHOS A., PETERS J., BOYD C., MITCHELL J. L.: Curved pn triangles. In *Proceedings of the 2001 Symposium on Interactive 3D Graphics* (New York, NY, USA, 2001), I3D '01, ACM, pp. 159–166. 2

[Xu06] XU G.: Convergence analysis of a discretization scheme for gaussian curvature over triangular surfaces. *Comput. Aided Geom. Des. 23*, 2 (Feb. 2006), 193–207. 2

[ZGYL11] ZHIHONG M., GUO C., YANZHAO M., LEE K.: Curvature estimation for meshes based on vertex normal triangles. *Comput. Aided Des. 43*, 12 (Dec. 2011), 1561–1566. 2, 5, 10

id.	estimator	$\bar{e}$
1	Cohen-Steiner & Morvan [CSM03], range $2\bar{l}$	94.2
2	Cohen-Steiner & Morvan [CSM03], range $4\bar{l}$	29.0
3	Cohen-Steiner & Morvan [CSM03], range $6\bar{l}$	20.8
4	Cohen-Steiner & Morvan [CSM03], range $8\bar{l}$	19.4
5	Fünfzig et al. [FMHF08]	2001.5
6	Goldfeather & Interrante [GI04], fitting circle, range $2\bar{l}$	39.4
7	Goldfeather & Interrante [GI04], fitting circle, range $4\bar{l}$	15.2
8	Goldfeather & Interrante [GI04], fitting circle, range $6\bar{l}$	8.7
9	Goldfeather & Interrante [GI04], fitting circle, range $8\bar{l}$	6.2
10	Goldfeather & Interrante [GI04], fitting parabola, range $2\bar{l}$	107.9
11	Goldfeather & Interrante [GI04], fitting parabola, range $4\bar{l}$	39.2
12	Goldfeather & Interrante [GI04], fitting parabola, range $6\bar{l}$	22.3
13	Goldfeather & Interrante [GI04], fitting parabola, range $8\bar{l}$	16.0
14	Goldfeather & Interrante [GI04], fitting parabola with normals, range $2\bar{l}$	12710.9
15	Goldfeather & Interrante [GI04], fitting parabola with normals, range $4\bar{l}$	1072.8
16	Goldfeather & Interrante [GI04], fitting parabola with normals, range $6\bar{l}$	386.3
17	Goldfeather & Interrante [GI04], fitting parabola with normals, range $8\bar{l}$	76.1
18	Hildebrandt & Polthier [HP11], $\hat{S}$ estimation, range $2\bar{l}$	2214.0
19	Hildebrandt & Polthier [HP11], $\hat{S}$ estimation, range $4\bar{l}$	2130.2
20	Hildebrandt & Polthier [HP11], $\hat{S}$ estimation, range $6\bar{l}$	2104.3
21	Hildebrandt & Polthier [HP11], $\hat{S}$ estimation, range $8\bar{l}$	2098.0
22	Hildebrandt & Polthier [HP11], $\bar{S}$ estimation, range $2\bar{l}$	136.3
23	Hildebrandt & Polthier [HP11], $\bar{S}$ estimation, range $4\bar{l}$	38.0
24	Hildebrandt & Polthier [HP11], $\bar{S}$ estimation, range $6\bar{l}$	30.6
25	Hildebrandt & Polthier [HP11], $\bar{S}$ estimation, range $8\bar{l}$	50.2
26	Kalogerakis et al. [KSNS07]	6.0
27	Meyer [MDSB02]	100.4
28	Pottmann et al. [PWHY09], ScaleRatio 0.02	9116.8
29	Pottmann et al. [PWHY09], ScaleRatio 0.045	802.6
30	Pottmann et al. [PWHY09], ScaleRatio 0.1	109.1
31	Pottmann et al. [PWHY09], ScaleRatio 0.225	168.6
32	Pottmann et al. [PWHY09], ScaleRatio 0.5	2260.6
33	Rusinkiewicz [Rus04]	25.7
34	Zhihong et al. [ZGYL11]	1871.0

Table 1: *Columns notation for Table 2. $\bar{l}$ stands for average edge length.*