A ROBUST STATISTICAL APPROACH FOR CURVATURE ESTIMATION IN DISCRETIZED SURFACES

by

Evangelos Kalogerakis

A thesis submitted in conformity with the requirements
for the degree of Master of Science
Graduate Department of Computer Science
University of Toronto

Abstract

A robust statistical approach for curvature estimation in discretized surfaces

Evangelos Kalogerakis

Master of Science

Graduate Department of Computer Science

University of Toronto

2007

The curvature of discretized surfaces is playing a crucial role in numerous computer graphics and vision applications as it is directly related to the problem of shape understanding. Curvature is typically computed at mesh vertices, on their associated ring neighborhoods or arbitrary user-defined regions. However, such approaches are not well suited to noisy, non-uniformly sampled and tesselated surfaces, as they can become unstable, in the presence of noise, mesh irregularities and structured outliers. In this thesis, a robust statistical approach, based on M-estimation, is presented, that is highly tolerant of noisy configurations on such discretized surfaces, holding the desirable properties of accuracy, stability and consistency in the curvature computation. The main novelty of the approach is the rejection of noise and outliers, by appropriately sampling and weighting normal variations in varying regions around each point of interest, that the algorithm automatically converges to, with minimum user intervention.

Dedication

Στην γιαγιά μου Αρχοντούλα

Acknowledgements

First of all, I thank the one responsible for having me sentenced into this life form.

Then, I thank my supervisor Professor Karan Singh and the DGP PhD student Patricio Simari for their valuable help and support during my thesis work. I also thank the second reader of my thesis, Professor Eugene Fiume.

Moreover, I would like to thank Professor Eitan Grinspun and Professor Christina Christara for their help and suggestions.

Contents

Chapter 1

Introduction

The study of differential properties of curves and surfaces has been an intriguing area of research in mathematics over the centuries. Throughout the history of mathematics, the analysis of curvature of curves and surfaces has played an important role in the development of the fields of analytic and differential geometry and manifold theory, as curvature is a fundamental descriptor for their understanding. Curvature was extensively studied by ancient Greek mathematicians (Aristotle, Apollonius of Perga), physicists, astronomers and analytic and differential geometers of the common era (Johannes Kepler, Sir Isaac Newton, Gottfried Wilhelm Leibniz, James Bernoulli, Pierre de Fermat, Rene Descartes, Christiaan Huygens, Leonhard Euler, Augustin-Louis Cauchy etc and especially Karl Friedrich Gauss who brought differential geometry to a whole new level) and finally modern physicists and mathematicians (Friedrich Bernhard Riemann, Elwin Bruno Christoffel, Gregorio Ricci, Albert Einstein whose general relativity field equations describe curvature of spacetime in universe due to the existence of matter and energy, etc).

In general, curvature is an important concept in mathematics as in very simple words, it is a measure of ”bending”. In the case of curves, curvature corresponds to how much the curve

”bends” at each point, while, in the case of surfaces, this is translated to the local ”bending” of all curves passing though a point in the surface. The main curvatures in surfaces that emerged from this scrutiny are the mean curvature, Gaussian curvature, principal curvatures and the Weingarten map, where each ”type” of curvature is related to different intrinsic or extrinsic properties of the surfaces.

Beyond the importance of study of curvature in the area of mathematics, physics, astronomy and engineering, there are numerous and very important applications, involving the estimation and exploitation of curvature, in computer science and especially, in computer vision and computer graphics. In section 1.1 of this chapter, we will present and categorize briefly the most important applications of curvature in computer graphics and vision, in order to show where and why curvature is used in these areas. Then, in section 1.2, we will refer to the basic motivation that led us to the development of our new approach to robustly estimate curvature in discretized surfaces. Finally, in the last section, we summarize the basic topics of this thesis.

1.1 Why do we need curvature in computer graphics?

The estimation of curvature of curves and surfaces plays a crucial role in numerous computer graphics and computer vision applications. Here, we will mention the most significant ones and we briefly describe how the curvature is used in these applications.

- *surface segmentation*: surfaces representing objects are usually segmented into simpler meaningful components by examining the zero-crossings and extremal values of the surface curvature measures. In computer vision, segmentation is an important step towards object recognition as the simple meaningful components can be analyzed in terms of their geometric features and their connectivity in order to build descriptors that will be used for a partial or full matching between different objects. Segmentation is usually

applied in 3D range data, which is the main input for vision applications [13, 73]. In computer graphics, 3D surface meshes are partitioned into simpler components benefiting many other applications like geometric modeling, metamorphosis, compression, simplification, 3D shape retrieval, texture mapping and skeleton extraction. Watershed segmentation is a common method to deconstruct CAD and other synthetic models, made from simpler solid primitives. In watershed segmentation, curvature (mean, gaussian, or some combination of both) is used as a height function over the surface and starting by its local minima, the algorithm incrementally "floods" the regions around the minima until they are connected to their neighbors [45, 55]. Alternatively, other algorithms, such as *k*-means clustering can be used to decompose a 3D-mesh into homogeneous curvature surface patches with clean boundaries [37]. Other segmentation algorithms use curvature to simply rectify boundaries between regions that were roughly segmented based on some other metric (convexity, geodesic distance etc) [39, 35]. Another interesting area of segmentation in computer graphics is the partitioning of surfaces into nearly developable patches. A developable surface is a surface which can be unfolded into a plane without stretching or tearing. Thus, this type of segmentation is used in sheet-metal and plate-metal based industries. In this case, the segmentation algorithms are based on a metric involving gaussian curvature (as developable surfaces have gaussian curvature zero everywhere) [76].

- *mesh simplification*: discrete surface curvature is directly related to a quadric error metric, defined over the edges of the manifold surface, that can be minimized, by greedily removing low cost edges from the initial surface mesh. The cost assigned to every edge, based on this quadric error metric, reflects the geometric error introduced into the model as a result of contracting that edge. The minimization of the quadric metric between the initial mesh and a reduced mesh, yield simplified triangulations in the final mesh with optimal aspect ratio [27].

- *surface de-noising, smoothing*: in the surface fairing literature, there are techniques that try to minimize energy functionals that are related to the curvature of surface (e.g. total curvature) [74]. However, total curvature (sum of squared principal curvatures) depend non-linearly on the surface, thus, this minimization in non-linear. Desbrun *et al.* suggest an iterative technique by solving a linear system at each step involving the matrix of mean curvature normals [12]. Another smoothing approach is to define a weighted mean curvature flow penalizing vertices with large ratio between their principal curvatures [49].

- *symmetry detection*: symmetry, which is a complexity-reducing concept, can significantly benefit mesh compression, repair, skeletal extraction and advanced mesh editing. As curvature is invariant under Euclidean transformations, it can be used to pair up compatible points that could be mapped to each other under a candidate symmetry transform [51]. Moreover, the 3D symmetry-curvature theorem proves that a symmetry axis in a 3D shape starts from curvature extrema along the lines of curvature [40].

- *deformations*: the amount of deformation of an elastically deformable surface away from its natural state depends on the net instantaneous potential energy of the elastic deformation of the body. This strain energy is expressed as a functional in a differential equation that describes the state of the deformable model. The strain energy for elastic bodies can be expressed as a norm of the difference between the first and second fundamental forms of the deformed body and the fundamental forms of the natural undeformed body [68]. In the case of inextensible shells (infinitely thin flexible objects), we are mostly interested in isometric deformations, meaning that there are possibly significant deformations associated with the bending energy, but unnoticeable deformation associated with strain energies. That bending energy is expressed as the integral of the squared difference of mean curvature between the deformed and natural state of the shell over the surface [22].

- *shape analysis and surface recognition*: the analysis of geometric features of a surface

based on curvature is very important for partial matching of shapes, object recognition and 3D search engines. Maekawa *et al.* suggest that the umbilical points, which are the singular points of the orthogonal net of lines of curvature (where the two principal curvatures are also the same), can act like fingerprints for shape recognition. The curvature derivatives can be used to detect view and scale independent ridges and valleys, which are geometrically and perceptually salient surface features that can also be used for shape recognition or visualization [53, 38, 34]. Another surface analysis method, used to identify local similarities between objects for partial matching, accumulates curvature information around each vertex in order to build a curvature map, which can be used to create a unique signature of every vertex. The curvature map samples either mean or gaussian curvature using geodesic fans starting from a vertex inside an increasing set of N-ring neighborhoods around it [18].

- *assembling 3D puzzles*: another interesting problem in computer graphics problems is the assembly a 3D solid object from its potential fragments. The analysis of the geometry of the fractured surfaces is necessary to find a globally consistent reconstruction of the original object. More specifically, curvature can be used to determine the surface sharpness, used to segment the fragments, and the surface roughness, used to distinguish the fragmented (internal) surfaces from the external surface of the object [31]. Then, curvature can be used as a descriptor to build potential matchings between the different candidate fragmented surfaces.

- *anisotropic remeshing*: lines of minimum and maximum (principal) curvatures can be used to drive a remeshing procedure which is adapted to the natural anisotropy of a surface. The lines of curvatures determine the appropriate edges for the re-meshed version of the initial surface in anisotropic regions. Another nice property of the lines of curvature is that they mimic the lines that artists themselves would follow while designing 3D models from scratch [2, 46].

- *non-photorealistic rendering*: principal curvature directions are also used in the field on non-photorealistic rendering. For example, Hertzmann and Zorin [28] exploit lines of curvatures to define hatching directions [28] so that hatch marks convey surface shape like artists do. Curvature is also important to generate suggestive contours in NPR applications [11]. Suggestive contours are lines drawn on clearly visible parts of a surface, where a true contour would first appear with a minimal change in viewpoint. More specifically, this type of contours rely on the notion of radial curvature. Radial curvature is the normal curvature of the surface in the tangential direction defined by the projection of the camera view vector to the tangent plane of a point. Suggestive contours are generated along points that have their radial curvature zero and its directional derivative in that tangential direction is positive.

- *texture synthesis*: the principal curvature directions define a "natural" flow over the surface of an object and therefore, can be used to guide the placement of the lines of a stroke texture that aims at enhancing a viewer's perception of the 3D shape, similarly to the NPR applications [33]. An oriented texture pattern can be applied to the surface of an object such that it will be everywhere aligned with the principal directions of curvature [20].

1.2 Basic motivation and goals for our new curvature estimation approach

Although the computation of curvature in discretized surfaces is very useful in many computer graphics applications, as shown in the previous section, in general, there is no consensus on the most appropriate way to estimate normal vectors and curvatures on them as pointed out by Meyer et al. [49]. One reason for this is that the surface represented by a discrete mesh or a

point cloud is not unique. Furthermore, scanned surfaces often contain signal noise that gets amplified on normal and curvature calculations but on the other hand, many surface de-noising algorithms (e.g. [12]) depend upon robust normal and curvature estimation. Many existing approaches, that try to address the problem of curvature computation in discrete approximations of surfaces, exhibit accuracy, stable behavior and convergence in noiseless, perfectly tesselated and uniformly sampled surfaces, but they can easily become unstable even in cases of light noise. In addition, the problem of curvature computation becomes harder because of the existence of irregularities [1] and non-uniform tessellations in polygon meshes (which are also very common especially in synthetic, artistic and CAD driven meshes) or non-uniform sampling in point clouds. The underlying surfaces represented by such discretized surfaces, may be piecewise smooth themselves, meaning that they can possess intrinsic discontinuities in their derivatives (e.g. in their feature edges). The above issues trigger the problem of which is the most proper region where a curvature operator should work. The algorithms that are based on the one-ring neighborhood or the incident edges of a vertex for estimating curvature may produce the most accurate and safest results in noise-free cases, but they can become very sensitive to noise [17]. On the other hand, when the area of a curvature operator increases too large e.g. beyond one-ring neighborhoods, its accuracy and reliability may be decreased significantly [59], resulting in oversmoothing the curvature field. The choice of the proper region that minimizes the corresponding error in the estimation of curvature seems still an open issue [10].

In order to face the degeneracies, inconsistencies, low accuracy and discontinuities in the curvature field of noisy and irregular meshes, some existing approaches suggest the a-posteriori smoothing of the field. As mentioned in [19], this can result in many of the principal curvature vectors and values falling out of alignment with the true vectors and values. Moreover, if

[1] the term mesh irregularities refers to cases where the vertices of the mesh have varying valence, the triangles have varying aspect rations with usually small acute or large obtuse angles, the edges have varying length, see figure 6.9

the underlying surface has feature (sharp) edges, non-judicious a-posteriori smoothing would spoil any intrinsic discontinuities of the surface derivatives on them and propagate the extreme values of curvatures to neighbor vertices. Alternatively, in order to deal with the cases where significant noise exists, a priori smoothing on the mesh is usually suggested. However, a priori smoothing can depreciate surface detail if not applied judiciously [17, 25]. The issues caused by mesh irregularities could be faced with a regular remeshing algorithm (e.g. [23]), but the computational burden added to the curvature estimation and any existing angle and area distortions in the surface parametrization would unnecessarily complicate the implementation and could decrease the accuracy of the curvature operator.

Our goal is to develop an algorithm for estimating principal curvature values and directions in discrete surfaces (represented as either polygon meshes or point clouds), that is highly tolerant of noisy and irregular configurations on them. Firstly, we present a new fast, stable approach, which computes the curvature tensor at each point of interest, by properly constraining its estimation in a fixed small region around it (section 3). Then, we extend this approach, by developing a robust M-estimation algorithm (section 4), that samples normal variations in a region of the mesh and weights these samples appropriately, so that the the effect of noise and structured outliers are minimized without over-smoothing the curvature field within a maximum likelihood framework. In noise-free cases, our algorithm automatically tends to take into account an area closer to every point of interest, preserving accuracy, while in noisy cases, our algorithm tends to rely on a bigger area to compensate for noise. To the best of our knowledge, this is the first attempt a robust statistical technique is used for estimation of differential properties on discrete surfaces.

1.3 Thesis overview and structure of this document

This rest of this thesis document is organized as follows:

- *chapter 2*: In the next chapter, we will review the most essential concepts and terminology from the 2-manifold and discrete differential geometry theory which are necessary for the understanding of our method.

- *chapter 3*: here, we present the related work which has been done in the field of curvature computation. We note that there is an enormous variety of curvature-related discretizations, which have been developed in the fields of applied mathematics, engineering, computer science (especially computer vision and computer graphics). We review the most well known and important approaches. We also mention an interesting extension that we built upon an existing computer vision approach and that we developed prior to the main approach presented in this thesis.

- *chapter 4*: in this chapter, we present the details and implementation of our fast fixed-region approach for curvature estimation.

- *chapter 5*: we proceed with the description of our main contribution, which is the development of our robust statistics approach, by extending the framework of the method presented in the chapter 4.

- *chapter 6*: in this chapter, we show comparison results in a variety of noisy and irregular meshes to exhibit the robustness and consistency of our above approaches.

- *chapter 7*: finally, we conclude and present some interesting future work that involves our robust statistics method for curvature computation.

Chapter 2

Basic theory about curvature of manifolds embedded in R^3 space

In this chapter, we will review some important aspects of differential geometry that will be necessary for the understanding of our approach. In section 2.1, we will present the basic concepts of the theory of 2-manifolds in the continuous case while in next section 2.2, we will focus on discrete differential geometry issues which are also the main interest of this thesis. Proofs are skipped; a detailed analysis about the definition and properties of curvature in surfaces and proofs can be found in classical differential geometry books [54, 64].

2.1 Review of differential geometry in the continuous case

A smooth two dimensional surface in R^3 space is a subset $X \subseteq R^3$ such that each point on the surface has a neighborhood $U \subset X$ and there is a mapping from an open 2D set V to the 3D coordinate space such that:

a) $\vec{r} : V \to U$ is an homeomorphism

b) $\vec{r}(u,v) = [x(u,v)\ y(u,v)\ z(u,v)]^T$ has derivatives of all orders

c) at each point on the surface $\vec{r}_u = \partial\vec{r}/\partial u$ and $\vec{r}_v = \partial\vec{r}/\partial v$ are linearly independent.

The tangent plane of a surface at any point p is the vector field spanned by $\vec{r}_u(x)$ and $\vec{r}_v(x)$ while the unit normal vectors (inward and outward) are defined as cross products: $\pm\frac{\vec{r}_u \wedge \vec{r}_v}{|\vec{r}_u \wedge \vec{r}_v|}$.

2.1.1 The first and second fundamental form of a surface

Two very important geometric structures completely characterize the shape of a surface: the first fundamental form I and the second fundamental form II. Practically, the first fundamental form describes the metrical properties of the surface (length, area, and angles between two curves on the surface). A smooth curve lying on the surface is a mapping $t \to (u(t), v(t))$ with well defined derivatives of all orders such that $\vec{\gamma}(t) = \vec{r}(u(t), v(t))$ is a parametric curve in 3D space. The arc length of such curve can be found as:

$$\int_a^b \left|\vec{\gamma}'(t)\right| = \int_a^b \sqrt{\vec{\gamma}'(t)\cdot\vec{\gamma}'(t)} = \int_a^b \sqrt{E[u'(t)]^2 + 2Fu'(t)v'(t) + G[v'(t)]^2}$$

or in a more compact form

$$d\vec{r}\cdot d\vec{r} = Edu^2 + 2Fdudv + Gdv^2 \tag{2.1}$$

where $E = \vec{r}_u \cdot \vec{r}_u \quad F = \vec{r}_u \cdot \vec{r}_v \quad G = \vec{r}_v \cdot \vec{r}_v$. The area of a domain U in the surface can also be expressed in terms of the first fundamental form $\int_U |\vec{r}_u \wedge \vec{r}_v|\, dudv = \int_U \sqrt{EG - F^2}\, dudv$. The above expression 2.1 is the first fundamental form and is also usually represented as the symmetric matrix

$$I = \begin{bmatrix} E & F \\ F & G \end{bmatrix}$$

More intuitively, the first fundamental form describes the intrinsic geometry of a surface "the experience of an insect crawling around it". On the other hand, the second fundamental form

"relates to the way the surface sits in R^3". It basically describes the surface curvature and is defined as the dot product of infinite displacement $d\vec{r}$ of any smooth curve on the surface and infinite variation $d\vec{n}$ of the unit normal vector along this curve:

$$-d\vec{r}\cdot d\vec{n} = Ldu^2 + 2Mdudv + Ndv^2 \tag{2.2}$$

where $L = \vec{n}\cdot\vec{r}_{uu}$ $M = \vec{n}\cdot\vec{r}_{uv}$ $N = \vec{n}\cdot\vec{r}_{vv}$ ($\vec{r}_{uu}$, $\vec{r}_{uv}$, $\vec{r}_{vv}$ are the second partial derivatives).

2.1.2 The shape operator and normal curvature

The second fundamental form of the surface can alternatively be expressed in terms of its shape operator. Let $\vec{u}$ be a tangent vector on the surface at a point p, then the shape operator at p is defined as $S(\vec{u}) = \nabla_u\vec{N}$ where $\vec{N}$ is a unit normal vector field on a neighborhood of p in the surface. The notation $\nabla_u\vec{N}$ refers to the covariant derivative of the vector field in the direction $\vec{u}$. The covariant derivatives are a generalization of directional derivatives of real valued functions in the case of vector fields in Euclidean space and they basically measure the rate of change of a vector field on each given direction. In the case of surfaces, the normal vector field $\vec{N}$ is defined only at points of the surface and it makes sense as long as the given direction is tangential to the surface.

Given a pair of tangent vectors $\vec{u}$, $\vec{v}$, the second fundamental form is expressed in terms of the shape operator as: $II = S(\vec{u})\cdot\vec{v}$ for all pairs of tangent vector to an oriented surface. Alternatively, in the form of a symmetric $2x2$ matrix, we can write:

$$\begin{bmatrix} \nabla_{\vec{u}}\vec{N}\cdot\vec{u} & \nabla_{\vec{v}}\vec{N}\cdot\vec{u} \\ \nabla_{\vec{u}}\vec{N}\cdot\vec{v} & \nabla_{\vec{v}}\vec{N}\cdot\vec{v} \end{bmatrix} \tag{2.3}$$

Assuming again now that the surface is oriented by the choice of the unit normal vector field $\vec{N}$, we can define the normal curvature of the surface at any point p: if $\vec{u}$ is a tangential unit

vector at p, then the real-valued number $k(\vec{u}) = k(-\vec{u}) = S(\vec{u}) \cdot \vec{u}$ is the normal curvature of the surface in the direction of $\vec{u}$ at p. If $\vec{n}$ is the surface normal at p, the normal curvature vector is expressed as $\vec{k} = k\vec{n}$ (the normal curvature is the scalar magnitude of $\vec{k}$). Alternatively, the normal curvature can be expressed as a quadratic form in terms of the second fundamental tensor: given a tangent vector $\vec{t} = t_1\vec{u} + t_2\vec{v}$, the normal cuvature at this direction $\vec{t}$ satisfies the identity:

$$k(\vec{t}) = \begin{bmatrix} t_1 \\ t_2 \end{bmatrix}^T \cdot II \cdot \begin{bmatrix} t_1 \\ t_2 \end{bmatrix} \tag{2.4}$$

If we can think again of a smooth curve on the surface that passes through p and has curvature vector $\vec{\kappa}$ (which can be obtained by differentiating its unit tangent vector with respect to arc length), we can represent its curvature as a sum of a normal and a tangential component. The normal component is the normal curvature $\vec{k}$ vector and the tangential component is the geodesic curvature vector. In the special case, where the curve is the intersection of a plane determined by $\vec{u}$ and $\vec{n}$ at p with the surface, the curvature of the curve is equal to the normal curvature at p. These special curves on the surface are called normal sections. This allows us to understand the curvature of a surface in any tangential direction at p by taking the corresponding normal sections and compute their associated curvature at p.

2.1.3 Principal curvatures, mean and gaussian curvature

Given a point p on a surface in R^3, the maximum and the minimum values of normal curvature of the surface at p are called principal curvatures and are denoted as k_1 and k_2 respectively. The associated directions in which the normal curvature at p is maximized and minimized are the principal directions (or axes) $\vec{e}_1$, $\vec{e}_2$ of M at p. The net of lines, that have as tangents the principal directions at all of their points are called lines of curvature. An interesting case is

when the two principal curvatures are identical at p, thus, the normal curvature is constant at this point, or equivalently, the surface bends the same amount in all directions there. In this case, the point p is called umbilic and the lines of curvature there have singular properties. Characteristic examples, here, are the sphere and the plane where all their points are umbilic.

An important formula involving the principal curvature values in surfaces is Euler's formula. If $u = cos\theta \cdot \vec{e}_1 + sin\theta \cdot \vec{e}_2$, it can be proven that the normal curvature of the surface at p in the given $\vec{u}$ direction is:

$$k(\vec{u}) = k_1 \cos^2\theta + k_2 \sin^2\theta \tag{2.5}$$

Another important fact for the principal curvatures is that they are the eigenvalues of the second fundamental form matrix and the eigenvectors of II give the corresponding principal directions. The determinant of II, which is the product of the principal curvature values, is the Gaussian curvature G while half of its trace (or equivalently, the mean of the principal curvature values) is the mean curvature H:

$$K = k_1 k_2 = det(II) \text{ and } H = 0.5(k_1 + k_2) = 0.5 trace(II)$$

In addition, it is straightforward that:

$$k_1, k_2 = H \pm \sqrt{H^2 - K}$$

The term $\sqrt{H^2 - K}$ is also called total curvature of the surface at p. Note that in an umbilic point $K = H^2$ and in a flat point $H = K = 0$. Equivalently, the mean curvature H can also be formulated as the average of the normal curvatures:

$$H = \frac{1}{2\pi} \int_0^{2\pi} k(\theta) d\theta \tag{2.6}$$

where θ is an angle respect to an initial tangent direction at the point of interest (thus, we integrate over all the possible tangential directions around a point).

A significant fact about the Gaussian curvature is that it is independent of the choice of the unit normal vector $\vec{n}$. If it is changed to $-\vec{n}$, the Gaussian curvature will be unaffected. In fact, Gaussian curvature is an intrinsic property of the surface (meaning it does not depend on the particular embedding of the surface); intuitively, this means that ants living on the surface could determine the Gaussian curvature (it was Gauss who showed this in 1828). The sign of Gaussian curvature characterizes a point in the surface; if it is positive, the point is called an elliptic point (e.g. any patch on an ellipsoid is an elliptic region). If it is negative, the point is characterized as hyperbolic (e.g. any point on a hyperbolic paraboloid is hyperbolic). If it is zero, the point is parabolic. Surfaces that have their Gaussian curvature zero are called developable surfaces (thus, all of their points are parabolic). Intuitively, Gaussian curvature measures "how much distortion" is needed to flatten a patch onto the plane. For example, a piece of paper representing a developable surface, like a cone or a cylinder, can be unfolded to the plane without distortion.

2.1.4 Gauss-Bonnet theorem

Another important fact about Gaussian curvature is that it is directly related to the the ultimate topological characteristics of the manifold. This is shown by the Gauss-Bonnet theorem. Firstly, Gauss-Bonnet theorem connects the total Gaussian curvature of a compact 2D manifold M having boundary ∂M, with the geodesic curvature k_g at points of the boundary and the Euler characteristic of the manifold $\chi(M)$, which is practically related to the number of "holes" in the manifold; it is equal to $2-2g$, where g is the genus (roughly speaking, the number of holes) of the manifold:

$$\iint_M K\, dA + \int_{\partial M} k_g\, ds = 2\pi\chi(M)$$

where dA is the element of area of the surface, and ds is the line element, along the boundary of M. If the boundary ∂M is piecewise smooth, then the integral $\int_{\partial M} k_g\, ds$ is interpreted as the sum of the corresponding integrals along the smooth portions of the boundary, plus the sum

of the angles by which the smooth portions turn at the corners of the boundary (called jump angles α_i):

$$\iint_M K dA = 2\pi - \sum \alpha_i - \int_{\partial M} k_g ds \tag{2.7}$$

Another common formulation of the Gauss-Bonnet formula is that for any compact, boundary-less 2D manifold, the integral of the Gaussian curvature over the entire manifold is only related to the Euler characteristic of the manifold or practically its number of holes:

$$\iint_M K \, dA = 2\pi\chi(M)$$

Of course, this may be surprising because if you distort the surface and change its curvature at any location, without changing its topology, the same total gaussian curvature is maintained!

2.1.5 Euler-Lagrange equation for surface area minimization

A special type of surfaces are minimal surfaces that have their mean curvature zero everywhere, e.g. a helicoid, a catenoid or a plane are such examples. In general, finding a minimal surface of a boundary with specified constraints is a problem in the calculus of variations (sometimes also known as Plateau's problem). Lagrange was the first to notice that H = 0 is the formulation of the Euler-Lagrange differential equation for surface area minimization. This remark provided a relation between area minimization and mean curvature flow on a surface:

$$\iint_{A_M} 2H\vec{n} \, dA = \nabla A$$

or

$$2H\vec{n} = \lim_{diam(A)\to 0} \frac{\nabla A}{A} \tag{2.8}$$

where A_M is an infinite small region around a point on the surface, A is its area, diam(A) is its diameter, and ∇A is the gradient with respect to the (x, y, z) coordinates of the point. Note that the expression $2H(p)\vec{n}(p)$ at a point p, used in the above equations, is also known as the Laplace-Beltrami operator for surfaces, that maps every point p on the surface to a vector. The Laplace-Beltrami operator is the generalization of the Laplacian operator of functions defined on surfaces.

2.2 Discrete differential geometry

The need to reliably estimate the differential operators in discrete approximations of curves and surfaces gave birth to the field of discrete differential geometry. From a computational standpoint, these discrete approximations are necessary for algorithms and data structures in computer graphics. But on the other hand, they must share common properties with the continuous objects and this is not straightforward as for example, a continuous surface represented by a discrete polygon mesh or a point cloud cannot uniquely be defined. In this section, we will see how the basic theorems and concepts of the previous section are applied on the discrete case.

2.2.1 Gauss-Bonnet theorem in the discrete case

It is straightforward to apply the Gauss-Bonnet theorem in discrete surfaces. Consider a finite surface patch, e.g. a one-ring neighborhood around a vertex in a polygon mesh (a one-ring neighborhood consists of all the triangles that are incident to the center vertex having all its neighbor vertices). Then the Gauss-Bonnet theorem simply states the following:

$$\iint_{A_M} K \, dA = 2\pi - \sum_{j=1}^{\#f} \theta_j$$

or

$$K = \frac{2\pi - \sum_{j=1}^{\#f} \theta_j}{A} \tag{2.9}$$

where θ_j is the angle of the face j at the vertex of interest and #f is the number of incident faces incident to this vertex. A is the area of the region A_M where we take the above integral. Of course, one can simply answer that this region is the one-ring neighborhood in a polygon mesh, thus, A is the sum of the face areas in the one-ring neighborhood. In subsection 2.2.3, we show that this does not always guarantee a correct discrete approximation.

2.2.2 Discrete Mean Curvature

In the previous section, we had defined mean curvature as the average of the normal curvatures (see eq.2.6). In the discrete case, this equation can easily be discretized as:

$$H = \frac{1}{n} \sum_{i=0}^{n} k_i \tag{2.10}$$

where k_i is the normal curvature in a sampling tangential direction and the sum is performed in all n sampling directions e.g. the directions given by the projections of the edges incident to the vertex of interest in a polygon mesh. Usually, mean curvature is expressed as a weighted average of the normal curvatures to achieve better approximations. The choice of weightings, the computation of normal curvature or other possible approximations of the integral of the eq.2.6 gave birth to a considerable body of literature that will be presented in chapter 3.

Alternatively, mean curvature could be estimated by discretizing the Euler-Lagrange eq. 2.8. But, firstly, it is necessary to express the gradient of the area in a discrete way. In a polygon mesh discretization, which is usually represented as a set of faces, and assuming a local parameterization in a conformal space, this can be expressed as a function of the node values and the

angles of the triangulation, thus, the integral is the Laplace-Beltrami operator [49]:

$$\iint_{A_M} 2H\vec{n}\, dA = \frac{\sum_{j\in N(i)} (\cot a_{ij} + \cot b_{ij})(x_i - x_j)}{A}$$

thus, the mean curvature can be found as:

$$H = \frac{1}{2}\frac{\sum_{j\in N(i)} (\cot a_{ij} + \cot b_{ij})(x_i - x_j)}{A} \tag{2.11}$$

where a_{ij} and b_{ij} are the two angles opposite to the edge in the two triangles sharing the same edge given be the vertices (x_i, x_j) and $N(i)$ is the set of 1-ring neighbor vertices of the vertex of interest i.

Note, again, that the choice of "best" area A does not necessarily depends on the sum of face areas as we will show in the next subsection. Note that the choice of cotangents to express the integral Laplace-Beltrami operator is not a unique choice and other alternative expressions have been proposed in the literature that will be discussed in the chapter 3.

2.2.3 Voronoi regions

Although the above expression for the mean and gaussian curvature are independent of the underlying finite volume discretization, the minimization of the error, computed as the difference between the real curvature values in the continuous case and the ones in the discrete case, is achieved by appropriately choosing a well-defined region, which contains the closest meaningful area to each sample. Desbrun et. al [12] suggests the choice of Voronoi regions in order to provide tight error bounds for the discrete operators in polygon meshes. More specifically, they show that the error E, introduced by the discrete approximation of the mean curvature normal from the expression 2.8, is bounded by the value of $C^2_{\max}\sum_i \iint_{A_M} ||x - x_i||^2 dA$ where $C^2_{\max}$ is the Lipschitz constant of the Laplace-Beltrami operator over the considered smooth surface patch

A_M. The Voronoi region clearly minimizes $||x - x_i||^2$ as it contains the closest samples to the point of interest x_i.

In a polygon mesh, represented as a set of triangles, the Voronoi region should be defined for every triangle. In the case that the triangle is acute, the Voronoi region lies inside the triangle and respect to the vertex P of interest, can be found by drawing the bisectors of the incident edges PR and PQ on P (which meet at the circumcenter) and estimating the area between the circumcenter, the midpoints of the two edges and P (see 2.1):

$$VA = \frac{1}{8}(|PR|^2 cot(\angle Q) + |PQ|^2 cot(\angle R))$$

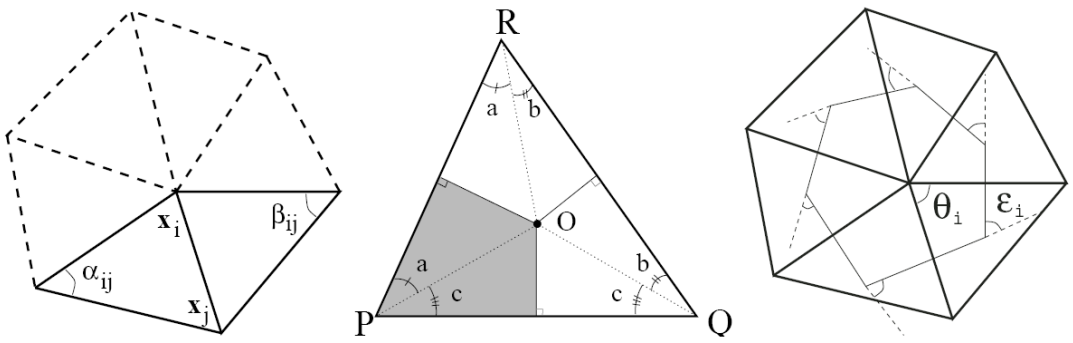

Figure 2.1: **Left:** one-ring neighbors and angles opposite to an edge, **Center:** Voronoi region on a non-obtuse triangle, **Right:** External angles of a Voronoi region [49]

Thus, in the equations for discrete mean and gaussian curvature 2.10 and 2.9, in the position of the area term A, the Voronoi Area (VA) is used instead. However, a problem arises in non-acute triangles. The Voronoi area lies beyond the triangles and the cotangent terms in the above equations become negative. A solution for this problem is proposed by Meyer at al. [49], where the notion of mixed area is used to compensate for the closest region to the vertex of interest for each triangle. If the triangle is acute, the mixed area is the voronoi region defined above. In the case that the triangle is obtuse, if the angle of the triangle at the vertex of interest x_i is obtuse, the closest region is approximated by half of the face area and if this angle is acute, the mixed area is defined to be a quarter of the face area. Even for this definition, the derivation

for the integral of the mean curvature normal is still valid as the boundaries of the area remain inside the one-ring neighborhood of each vertex and go through the midpoint of each edge.

Together with the definition of Voronoi cells, we have already introduced the basic concepts in discrete differential geometry and actually, some first valid discrete differential operators, on which the related work, that will be presented in the next chapter, is based and extended through the literature in computer graphics and vision. A more thorough analysis of these proposed extensions and new methods to approximate curvature in discrete surfaces will be given in the next chapter.

Chapter 3

Related work for the estimation of curvature in discretized surfaces

A considerable number of papers exist in the literature of computer graphics, vision and engineering concerning the computation of differential operators in discrete surfaces, especially polygon meshes. A survey of most existing approaches is also given by Gatzke and Grimm [17]. Similarly to Grinspun *et al.* [21], we will try here to present a categorization of the existing methods based on the local area where they operate on, in order to estimate curvature (section 3.1). To the best of our knowledge, all methods in the literature operate either on a fixed or a predefined standard local region around each point of interest; this is a first important difference with our robust statistics approach which adapts to noise and mesh irregularity and weights the samples of a surface area anisotropically in order to provide a reliable outlier-free curvature estimation automatically within a maximum likelihood framework.

In section 3.2, we will show an interesting extension that we built over the methods presented in [25] and [63], but, still, it lacked the desired properties of stability and accuracy, that we pursued in this thesis and were much better achieved by the methods that will be presented in

chapter 4 and 5.

3.1 Related Work

We present the related work on discrete differential operators which is categorized into edge and hinge based methods (section 3.1.1), primitive fitting approaches (section 3.1.2), vertex ring methods (section 3.1.3), per-triangle curvature estimation (section 3.1.4) and methods operating inside a ball of a given radius (section 3.1.5).

3.1.1 Edge and hinge-based methods

Perhaps, the fastest methods for discrete operators in the literature are the ones that are based on edge (two-points) and hinge (four-points) stencils. These methods are basically used in cloth simulation and are mostly interested in deriving an expression for the bending energy which is linked to the difference between the shape operator evaluated on the deformed and undeformed surfaces [3, 22, 6]. Of course, real-time cloth simulation needs these very fast and simple methods for the discretization of bending energy, but on the other hand, they cannot capture the curvature tensor (i.e. the principal curvature directions and values) or even the mean and gaussian curvature, only a functional corresponding to the bending energy e.g. the simplest functional is the integral of the squared mean curvature (often known as Willmore energy) over the surface. Moreover, these methods are highly dependent on the mesh structure, thus are sensitive to mesh irregularities and noise that affects the angles between the face normals that these methods are usually using in their estimations.

In our case, we are interested in estimating the whole curvature characteristics at every point of interest robustly to noise, non-uniform sampling and irregular tessellations as much as possible.

That's why we haven't considered these methods in our experiments. We also note that real-time applications of our curvature operator are not our main focus of interest in this thesis, although our first fixed region method (chapter 4) is fast enough and could potentially be used in such applications in the future.

3.1.2 Primitive fitting approaches

The main idea behind these methods is to locally approximate the polygon mesh with a polynomial function chosen to model the local surface shape. Firstly, these approaches approximate the normal by the local surface fitting, otherwise the normal at each point is independently computed as a weighted average of the nearby face normals or the normal of the best fit plane in a close region around it. The fitting methods are merely based on the fact that a smooth surface can be approximated well by an n-order surface in a neighborhood of a point p. By parameterizing the surface in terms of the distance of a point to the tangent plane at p, meaning that we can express this distance as a height function h(s, t) above the tangent plane, we can apply the Taylors formula for two-variable functions approximating h(s, t) up to order n around the point p.

Hammann [24] fits a quadratic function locally to the surface based on the above planar projection of the points of each one-ring neighborhood. The curvature tensor is then determined analytically from the derived polynomial surface. Meek and Wilton [48] proposes a similar quadratic fitting. In addition, quadratic patch fitting is used in computer vision approaches in the same framework with the difference of using a windows of size NxN pixels (samples) from a range image [1]. Stokelt and Wu [66] and Gatzke and Grimm [17] suggest building a natural local parameterization of the surface to fit a patch instead of using the planar projection with the height field formulation.

Chen and Schmitt [9] locally fit circular curves, given the center point and two of its neighbors.

If the angle between the normal of the circle at the center point and its estimated normal is ϕ, then the normal curvature in the direction $\vec{t}$, tangent to the circle, is given by the formula: $k(\vec{t}) = k_c \cos(\phi)$ where k_c is the curvature of the circle. Taking samples of the normal curvature in different directions, Euler's formula can be fit in order to find the principal curvatures and directions. Hameiri and Shimshoni [25] , instead, suggest to fit quadratic curves on the two immediate neighbor points of the center point or all the neighbor points inside a given radius across the normal sections to improve stability. Similarly, Simari *et al.* [63] suggests fitting polynomial curves across normal sections to estimate curvature in regular sampling directions. The above curve fitting methods seem to be more sensitive to noise than the patch fitting methods and they are also based on explicit user parameters, like the size of their associated region or the degree of the polynomial curve to fit.

Cazals and Pouget [8] show that in the case of a general smooth surface, the fitting of an osculating n-jet, which simply is the Taylor expansion of the height function truncated at order n, can cause convergence of the estimated curvatures to the true ones depending upon the degree of the jet used and as the samples get denser. The choice of the neighborhood of approximation depends on the number of points that are necessary for a stable unique fitting and is taken based either on an N-ring neighborhood in a mesh or the closest neighbor points in a point cloud. Note however, that in noisy configurations, higher-order terms causes overfitting and curvature estimation is severely damaged. Also, note that these patch fitting methods become unstable near degenerate configurations when normal information is not used (see figure 3.2).

Goldfeather and Interrante [19] expanded the quadratic patch fitting method, by using cubic fits from the points and their normals in one-ring neighborhood or two-ring neighborhoods, so that the degenerate cases can be avoided and accuracy is improved.

However, in the end, all the fitting methods arbitrarily use a notion of topological locality in their approximation based on the use of a one-ring, two-ring or N-ring neighborhood of a

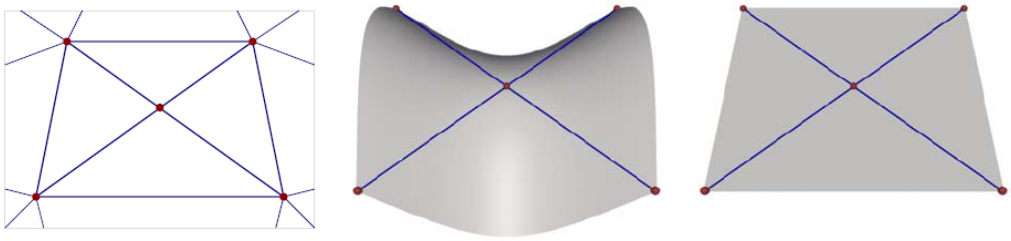

Figure 3.1: **Left:** degenerate configuration for fitting primitive methods [59]. Including any number of vertices lying on two intersecting lines in the least-square fit causes unstable estimates. **Center, right:** continuous surfaces with different curvatures consistent with the given vertices

vertex in a mesh or the closest N neighbors to each point of interest; in the presence of noise or non-uniform tessellations, it is not clear how this neighborhood should be chosen to maximize stability and accuracy. In addition, the accuracy in their curvature estimation always depends on the model that is being fit. Outliers and noisy normals may seriously affect the estimation whether higher-order approximations results in overfitting due to noise.

In our experiments, we have chosen Goldfeather and Interrante's method [19], as the best representative patch fitting method, applied on 2-ring neighborhoods as usually suggested in the literature [17, 42].

3.1.3 Discrete vertex-ring methods

One of the main advantages for discrete methods on associated vertex rings (usually one-rings) around the vertices in a polygon mesh or in a small window around a point of interest in voxel grids or range images, is the low computational cost and simplicity. One of the most well-known methods in this category is Taubin's algorithm [67]. The algorithm introduced a 3D curvature tensor that has the principal curvature directions as two of the three eigenvectors and the principal curvatures as linear combinations of two of the three eigenvalues (the third

eigenvalue is always zero by construction). Consider the normal curvature function, expressed in terms of the second fundamental tensor, as shown in equation 2.4. Let $\vec{u}$, $\vec{v}$ the principal directions at a point; if we add the normal to the basis $\vec{u}$, $\vec{v}$ we have chosen, to obtain a 3D orthonormal basis $\vec{n}$, $\vec{u}$, $\vec{v}$ and choose an arbitrary 3D direction $\vec{t} = t_0\vec{n} + t_1\vec{u} + t_2\vec{v}$, the normal curvature in this direction is:

$$k(\vec{t}) = \begin{bmatrix} t_0 \\ t_1 \\ t_2 \end{bmatrix}^T \begin{bmatrix} 0 & 0 & 0 \\ 0 & k_1 & 0 \\ 0 & 0 & k_2 \end{bmatrix} \begin{bmatrix} t_0 \\ t_1 \\ t_2 \end{bmatrix}$$

If we have another system of orthonormal directions and an arbitraty 3D direction $\vec{t}$ expressed as $\vec{t} = u_0\vec{U}_0 + u_1\vec{U}_1 + u_2\vec{U}_2$, the normal curvature at $\vec{t}$ is: $k(\vec{t}) = u^T M u$ where $u = \{u_0, u_1, u_2\}$. M is the 3x3 symmetric matrix, called Taubin's tensor. This tensor is found by the integral:

$$M = \frac{1}{2\pi} \int_{-\pi}^{\pi} k(\vec{t})\vec{t} \cdot \vec{t}^T$$

which can be discretized as a summation in a one-ring neighborhood for the incident edges, similarly to 2.10, as:

$$\tilde{M} = \sum_{j \in N(i)} w_{ij} k_{ij} \vec{e} \cdot \vec{e}^T$$

where $\vec{e}$ is the unit length normalized projection of each incident edge $\vec{u}_j - \vec{u}_i$ to the vertex (with position vector $\vec{u}_i$) of interest to its tangent plane, w_{ij} is a weight proportional to the sum of the face areas of the triangles sharing the edge and k_{ij} is the normal curvature of the edge, that is found by the approximation:

$$k_{ij} = \frac{2\vec{n}(\vec{u}_j - \vec{u}_i)}{||(\vec{u}_j - \vec{u}_i)||^2} \tag{3.1}$$

The above normal curvature approximation assumes a fitting of a quadratic curve on the points $\vec{u}_j, \vec{u}_i$, as in [25].

The best choice of weights for the curvature tensor approximation is an open question in the literature. Hameiri and Shimsoni [25] propose modifications of Taubins method by getting

more neighbor points, primarily for range data, and suggest a weighting based on the inverse of their geometric distance from the center points. Meyer *et al.* [49] suggest a weighting based on the sum of the cotangent angles opposite to each edge. However, the convergence of the discrete mean curvature to the Laplace-Beltrami operator, from the eq. 2.11, is met only under certain regularity conditions at the vertices [75]. Langer *et al.* [36] also show that the Meyer's weighting does not give accurate results for differing edge lengths and angles and suggest weights based on the sum of the tangents of the angles touching each edge on the vertex of interest. But in special cases, e.g. if the angle is close to $\pi/2$, the weighting fails to compute the whole curvature tensor. Apart from the weighting issues, these techniques based on the above normal curvature definition have the same weakness with the fitting methods, as again it is assumed that parabolas are fitted along the edges. As Rusinkiewicz [59] states they are unstable when the neighbors of a vertex are close to a pair of intersecting lines again like in figure 3.2.

Meyer *et al.* [49] suggests the approximation of discrete mean and gaussian curvature based on the Euler-Lagrange equation and Gauss-Bonnet theorem 2.11 and 2.7 based on the definition of Voronoi areas together with the notion of mixed areas (see section 2.2.3). Stokely and Wu [66] approximates the Gaussian curvature similarly, but they consider a closed path around each point of interest and use the whole area enclosed in this path. Meyer *et al.* also suggest the solution of a least squares system to find the curvature tensor based on the quadratic form $e^T IIe = k(\vec{e})$ where $\vec{e}$ is the normalized projection of the one-ring neighborhood edges and $k(\vec{e})$ is the normal curvature from 3.1. This system is used to find the principal directions only (taken from the eigenvectors of II) as they state that it does not give accurate results for the principal, mean and gaussian curvatures. Our fixed region method, which is also based on one-ring neighborhood in polygon meshes, also uses a linear least squares fitting to constrain the curvature tensors inside this local region. But in our least squares formulation, we do not only consider the tangential components of each edge but we solve the system by considering all the components of the edge vectors in a 3D coordinate orthonormal basis. Moreover, we do

not use parabola fitting to estimate normal curvature and we use a different weighting scheme that causes accurate approximations of the curvature tensor (concerning both eigenvectors and eigenvalues).

Steiner and Morvan [10] also gives discrete definitions for the mean and gaussian curvature and the curvature tensor based on the normal cycle theory. The discrete gaussian curvature is again based on the angle-deficit method; it is defined as the sum of the angle defects of the mesh at a point, that is 2π minus the sum of angles between consecutive edges incident on the point. The discrete mean curvature is defined along edges and is the sum of the signed angles between the normals of the oriented triangles of the mesh incident on them. In a one-ring neighborhood, the curvature tensor at a point is defined as:

$$II = \frac{1}{A} \sum_{edges\ e} \beta(e)\,|e|\,\vec{e}\vec{e}^{T}$$

where the summation is done over all edges e of the one-ring neighborhood, A is the surface area in the one-ring neighborhood, $\beta(e)$ is the signed angle between the normals to the two oriented triangles incident to edge e (positive if convex, negative if concave), $\vec{e}$ is a unit vector in the direction of the edge. A similar definition of the curvature tensor is also proposed by Hildebrandt and Polthier [29]. Steiner and Morvan prove that if the points on the mesh are connected by a restricted Delaunay triangulation, under a reasonable sampling condition, the estimated curvature tensors converge to the true ones of the smooth surface. This "tensor averaging" method exhibits stability, but can produce large errors for certain vertex arrangements (see figure 3.2). In addition, if the curvature tensor averaging is performed inside a one-ring neighborhood, there is high sensitivity to noise. In order to smooth out the noise, the authors suggest taking a larger region, usually a ball as we will show in section 3.1.5. However, it is unclear how to choose the neighborhood to compensate for noise.

Finally, the spherical image method, proposed by Meek and Walton [48], is another discrete one-ring method to compute Gaussian curvature only. If we traverse a closed path around a point of interest on the surface and take the "tails" of the normals of the points along the path,

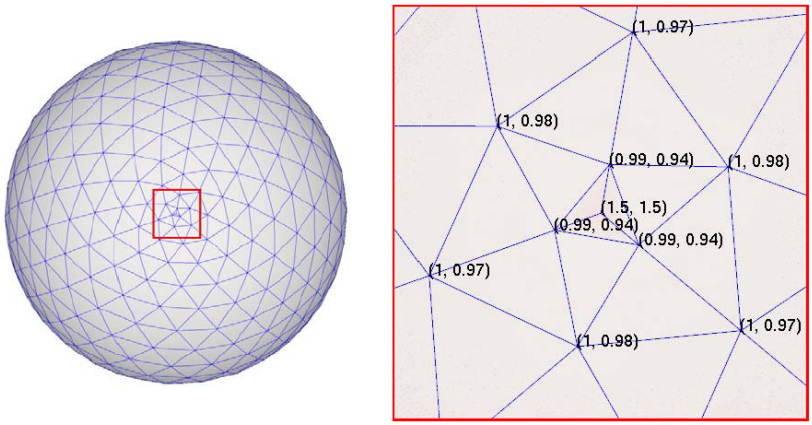

Figure 3.2: the tensor averaging technique produces large errors under certain vertex arrangements. In the case of sphere, the principal curvatures should be one at every point [59].

placed at the origin point, the heads of these normals trace out a closed curve on a unit sphere. This is the spherical image and the gaussian curvature can be approximated by the limit of the area of the spherical image of the path divided by the area of the path as the path shrinks around the point.

In our experiments, we include Meyer's method [49] based on Voronoi and mixed areas and Steiner's and Morvan's approach as good representatives from this discrete vertex-ring category.

3.1.4 Per triangle curvature estimation

Another category of methods suggests the estimation of curvature tensor per each finite element, i.e. a triangle in a mesh. The operator estimates the tensor by considering all the three edges of each triangle instead of the incident edges to a vertex or the edges of one-ring neighborhoods. These methods do not need projections to the tangent plane of each vertex, however, some calculations are needed to re-project the tensors from the faces to the vertices, if the curvature tensors on vertices are needed.

Rusinkiewicz [59] suggests the constraining of the curvature tensor in each triangle. In each triangle, we can define a $(\vec{u},\vec{v})$ orthonormal basis in the tangent frame which obviously is the plane of the triangle. Then we can solve for the second fundamental tensor, expressed in terms of the covariant derivatives of the normal curvature at the $(\vec{u},\vec{v})$ directions (see eq.2.3, by linear least squares fitting the triangle edge vectors $\vec{e}$ and their associated normal differences $\Delta\vec{n}$ in this 2D coordinate system:

$$\begin{bmatrix} \nabla_{\vec{u}}\vec{N}\cdot\vec{u} & \nabla_{\vec{v}}\vec{N}\cdot\vec{u} \\ \nabla_{\vec{u}}\vec{N}\cdot\vec{v} & \nabla_{\vec{v}}\vec{N}\cdot\vec{v} \end{bmatrix} \cdot \begin{bmatrix} \vec{e}\cdot\vec{u} \\ \vec{e}\cdot\vec{v} \end{bmatrix} = \begin{bmatrix} \Delta\vec{n}\cdot\vec{u} \\ \Delta\vec{n}\cdot\vec{v} \end{bmatrix}$$

Thus, every edge in the triangle provides a set of linear constraints on the elements of the second fundamental tensor that can be determined using linear least squares. After determining the tensor per each triangle, the tensor per vertex can be obtained by rotating the adjacent face tensors such that their chosen coordinate system aligns with the coordinate system of each vertex (given by its tangent plane and normal) and by just taking a weighted average of them, similarly to the classical vertex normal estimation, where we average the face normals. The weight for each face curvature tensor is its associated Voronoi area for each corresponding vertex. A similar per triangle curvature tensor estimation is also shown by Theisel et al. [69].

In general, the per triangle curvature operators, exhibit stability and robustness, but significant noise may still affect them as they are very localized. Our fixed region approach can be seen as an extension of Rusinkiewicz approach [59] as we follow the same formulation of the least squares fit of edges and their associated normal differences. However, we use different weights for each edge and we are able to constrain the fitting not inside a triangle but inside the whole one-ring neighborhood or even bigger areas, that are determined appropriately, so that the extended set of introduced linear constraints increases both the robustness and accuracy in the curvature tensor estimation.

3.1.5 Curvature estimation inside balls around points

In order to increase the robustness to noise, some vertex-ring methods are extended to operate beyond the one-ring neighborhoods, inside a ball of some specified radius so that the results of noise and mesh irregularities are compensated. For example, Steiner and Morvan [10] define the curvature tensor over an arbitrary region B:

$$II = \frac{1}{|B|} \sum_{edges\, e} \beta(e)\, |e \cap B| \vec{e} \otimes \vec{e}^T$$

where $|e \cap B|$ is the length of the edge inside the region B, which is always between 0 and $|e|$). In the anisotropic polygon remeshing application [2], Alliez *et al.* averages the tensor at every vertex location, for a neighborhood B that approximates a geodesic disk around each vertex. This approximation is done by simply computing the disk around each vertex of interest that is within a sphere centered at each vertex. The sphere radius is a user parameter and by default is chosen to be $1/100^{th}$ of the bounding box diagonal of the mesh.

Pottman *et al.* [44, 56] suggest performing a Principal Component Analysis (PCA) inside a ball of a given radius centered at each vertex. Given the vertices inside this sphere centered at a point of interest, the PCA returns three eigenvectors and associated eigenvalues. As the ball radius tends to zero, two of the eigenvectors converge to the principal directions and the third one with the normal at each point. Moreover, the principal curvature values can be determined by the eigenvalues. The approach seems to produce more robust principal directions against noise and minor perturbations than the normal cycles method [10].

Tong and Tang [71] proposed an alternative tensor voting technique to robustly estimate curvature tensors, which still relies on fixed preprocessing passes to reject outliers that are decoupled from curvature estimation. Maximum likelihood estimation of curvature tensors is not guaranteed. The optimal radius of their operator is globally chosen for the input surface by exhaustively searching an input range of values, preferring the one that results in the highest peaks among all the acquired curvature histograms.

Still, it is again unclear how to choose the radius of the ball in both approaches to get the best stable and robust results. Especially, in the case that the mesh has varying sampling density, a ball of constant global radius may produce inconsistent results in different regions of the discrete surface with varying detail and different sampling density, destroying the fine features of the curvature field. Finally, the above methods can fail if the ball contains samples that are totally irrelevant to the curvature tensor of a point e.g. feature edges or even worse, samples beyond these edges, for example imagine a ball centered on a point on a face of a cube that includes one of its feature edges as well as points of another face). Our approach rejects such structured outliers, behaving anisotropically on the discrete surface and is adaptive to non-uniform sampling for both accurate and robust curvature estimation. Also, our formulation yields a maximum likelihood estimate of the curvature tensor under Gaussian noise on surface points and normals and to the best of our knowledge, it is the first time a statistical maximum likelihood technique is used for estimation of differential properties on discrete surfaces.

3.2 Estimating curvature in polygon meshes with anisotropic normal sections

In this section, we present an interesting extension of the Hameiri and Shimshoni [25] and Simari *et al.* [63] approaches, that proposed estimating curvature based on polynomial curve fitting on normal sections. Our extension suggests an anisotropic operator that does not explicitly incorporate a user parameter indicating the size of area where normal sections are found. The normal sections are not considered to be of equal weight and length; the operator adjusts their weight and length based on the noise and sampling density on the direction of each normal section. Furthermore, we incorporate not only surface points in our fitting but also their associated normals as Goldfeather and Interrante did [19]. As a result, this updated operator exhibits more robustness and accuracy than the previous similar methods. However, the method again

has the disadvantages of fitting methods: it behaves well only when the normal sections can locally be modeled with quadratic curves adequately. In addition, in the case of significant noise, curve fitting becomes more unstable then patch fitting which is naturally more constrained (because more data points, participate in the polynomial model fitting, as they come from all over the local region instead of distinct sampling directions). In the following subsections, we show the basic ideas of our method.

3.2.1 Finding a normal section

Consider a triangulated mesh M = {V, E}, represented by a list of n vertices $V = \{v_i : 1 \leq i \leq n\}$ and a list of m faces $F = \{f_j : 1 \leq j \leq m\}$ where $f_j = \{i_j^1, i_j^2, i_j^3\}$ holds the indices of the vertices of a triangle. We will show how we can estimate the curvature $k(\vec{u})$ in the direction $\vec{u}$ on the tangent plane of M at a vertex v by taking a normal section s of the plane Q determined by $\vec{u}$ and the given normal $\vec{n}$ at this vertex.

The equation of Q is $(\vec{n} \times \vec{u}) \cdot (\vec{X} - \vec{P}_j) = 0$ where $\vec{X} = (x, y, z)$ and $\vec{P}_j = (v_x, v_y, v_z)$ are the coordinates of v. We retrieve the edges in the one-ring neighborhood which are not incident on v and we check for intersections of the plane with these edges. Given an edge S between two vertices v_k and v_l with coordinates $\vec{P}_k$ and $\vec{P}_l$ with equation $S(t) = \vec{P}_k + t \cdot \vec{d}$ where $t = [0, 1]$ and $\vec{d} = (\vec{P}_l - \vec{P}_k)$, we solve for parameter t:

$$t = \frac{-(\vec{n} \times \vec{u}) \cdot (\vec{P}_k - \vec{P}_j)}{(\vec{n} \times \vec{u}) \cdot \vec{d}} \tag{3.2}$$

If $0 \leq t \leq 1$, there is an intersection between the plane Q and the given edge. If v is not a boundary vertex, there are two initial intersection points between Q and the mesh, located in opposite directions, in the one-ring neighborhood of v. We can quickly compute the next intersection points based on the following scheme: *a)* if we had $0 < t < 1$ for the previous intersection point located on the edge S, there are only two candidate edges that could possibly intersect

with Q next (these are the other two edges of the triangle that S belongs to). *b)* otherwise, the intersection point is a vertex and we check the edges of its one-ring neighborhood for possible intersections (except for the edge where we already found a previous intersection).

The result is that we quickly build an increasing list of intersection points where we store only their 2D local coordinates (x_i, y_i) on the plane Q sorted by x_i. The list defines a polyline, which starts from the initial point of the vertex v and is continuously expanding in two opposite directions. We always expand the polyline in the direction of the point that is closer to v, thus, the expansion is equivalently fair in both directions.

3.2.2 Stopping criterium

Of course, it is necessary to define a stopping criterium as the process we described above, cannot continue forever in the mesh. Our key observation is that the polyline should stop in an area where a quadratic curve fits best on the intersection points we have found. Thus, we introduce a criterium that is as independent as possible from the notion of N-ring neighborhood around the vertex. Our goal is to detect an area where we can safely expect that a quadratic surface will fit well. However, as the mesh can be highly irregular, this area cannot be isotropic (e.g. the quality of approximation of the underlying surface can arbitrarily vary from vertex to vertex or even from direction to direction around the same vertex due to noise or mesh irregularity). Thus, we examine how each separate quadratic curve can fit well in the corresponding normal section of each different tangential direction around each vertex.

Every time we find a new intersection point, we fit the quadratic curve $y = ax^2$. There is only one parameter to solve for and the least squares fitting solution is given by:

$$a = \frac{\sum_i x_i^2 y_i}{\sum_i x_i^4} \tag{3.3}$$

By implementing this, carefully, every time a new intersection point is computed, we just add the appropriate values to the current sums and we just perform a division to find a. We can also fit the normals at the intersection points. By linearly interpolating the normal between the vertices of the edge where the intersection point lies, we compute the angle ϕ_i between this normal and the normal at the vertex v. Least squares fitting is performed based on the following equation:

$$a = \frac{\sum_i x_i^2 y_i + \sum_i x_i \tan(\phi_i) sign(y_i)}{\sum_i x_i^4 + \sum_i x_i^2} \tag{3.4}$$

Finally, in order to be as independent as possible from the tessellation of the mesh, we choose a weighting scheme that adds more weight to the intersection points that have larger distance from their neighbor points, meaning that the triangles which "participate" more in the polyline, have bigger influence on the fitting. This weighting scheme replaces the need for resampling the polyline at regular intervals that Simari *et al.* [63] had suggested. Moreover, the further the samples are from the center vertex of interest, the less they should be trusted. Therefore, in our weighting scheme, we also incorporate Hameiri's and Shimshoni's suggestions where the weight is proportional to the inverse of the distance of the sample points from the center point:

$$a = \frac{\sum_i w_i x_i^2 y_i + \sum_i w_i x_i \tan(\phi_i) sign(y_i)}{\sum_i w_i x_i^4 + \sum_i w_i x_i^2} \tag{3.5}$$

where $w_i = ((x_i - x_{i-1}) + (x_{i+1} - x_i))/2{d_i}^2$ for $i = 1, 2, ..., K-1$ (x_0 is the center point, K is the number of intersection points in each direction) and $w_K = (x_i - x_{i-1})/{d_i}^2$ and ${d_i}^2 = {x_i}^2 + {y_i}^2$.

After fitting the quadratic curve, we compute the maximum absolute residual of the points: $r_{max} = \max_i(|y_i - a x_i^2|)$. The expansion stops when r_{max} is bigger than the maximum tolerated residual r_{tol}. By default, the tolerated residual is chosen to be the maximum residual when we firstly fitted the two initial intersection points and the center point. The meaning of this choice is that the polyline is expanding as long as the fitting (in terms of resulting residuals) is always better that the initial fitting in the one-ring neighborhood. Alternatively, the tolerated residual

can be determined with a user-interaction mechanism, suggested by Fleishman *et al.* [14], where the user marks a small region on the surface that is known to be smooth, a polynomial local surface patch is fit to that region and the tolerated residual is set to be the largest residual of the fitting. In our case, typically eight, quadratic curves are fit to normal sections, taken at regular angle intervals (from 0 to π) around the center vertex, and the tolerated residual is set to be the largest residual of all the quadratic fits to the normal sections in the selected region.

3.2.3 Estimating principal curvatures and directions

Using the above method, we estimate curvature along tangential directions $\vec{u}_j$, taken at regular angle intervals (from 0 to π) around the center vertex. We typically choose eight tangential directions (at least three tangential directions are needed, but we experimentally noticed that satisfactory accuracy is achieved for at least four and preferably eight directions. Taking even more sampling directions has very little effects on the accuracy of the method). Then, the curvature of each quadratic curve $y = ax^2$ is $2a$ at the vertex. In the end, we have pairs of values $(k(\theta_j), \theta_j)$ where θ_j is the angle between the direction $\vec{u}_j$ and the initial direction $\vec{u}_1$ and $(k(\theta_j)$ is the curvature at the associated direction. The final step of our method is to compute the principal curvatures based on these pairs of values.

Based on eq. 2.5, we perform a fitting of the following function:

$$k(\theta) = k_1 \cos^2(\theta - \omega) + k_2 \sin^2(\theta - \omega) \Rightarrow \tag{3.6}$$

$$k(\theta) = \frac{1}{2}(k_1 + k_2) + \frac{1}{2}(k_1 - k_2)\cos(2\theta - 2\omega) \Rightarrow \tag{3.7}$$

$$k(\theta) = a + b\cos(2\theta) + c\sin(2\theta) \tag{3.8}$$

where $\frac{1}{2}(k_1 + k_2) = a$, $\frac{1}{2}(k_1 - k_2) = \sqrt{b^2 + c^2}$, $\tan 2\omega = -\frac{c}{b}$. Solving for a, b and c, we find the principal curvatures k_1 and k_2, while the angle ω give us the principal directions $\vec{e}_1$ and $\vec{e}_2$.

Least squares fitting of a,b,c simply requires the solution of the following $3x3$ linear system, than can be solved in closed form:

$$\sum_i k(\theta_i) = a + b\sum_i \cos(2\theta_i) + c\sum_i \sin(2\theta_i)$$
$$\sum_i k(\theta_i)\cos(2\theta_i) = a\sum_i \cos(2\theta_i) + b\sum_i \cos^2(2\theta_i) + \tfrac{c}{2}\sum_i \sin(4\theta_i)$$
$$\sum_i k(\theta_i)\sin(2\theta_i) = a\sum_i \sin(2\theta_i) + \tfrac{b}{2}\sum_i \sin(4\theta_i) + c\sum_i \sin^2(2\theta_i)$$

We also add weights to the above system based on the error E_i of the fitting of the corresponding quadratic curve in every direction which is equal to the sum of the squared residuals of the intersection points. In every term of the above system, we multiply by a factor $W_i = 1/\hat{E}_i^2$ where $\hat{E}_i = E_i/(\max_i(E_i))$. Thus, the values of curvatures that correspond to better fittings in their associated directions are trusted more.

3.2.4 Implementation

The above method was implemented in Matlab 7. We include this methods in our experiments, that will be presented in chapter 6, and we show in the end of this chapter some of the plots of its curvature estimation error. Although the method could compete the one-ring methods in some test cases, its accuracy was disappointing in the general case, especially compared to the surface patch fitting ([19]) and per triangle curvature tensor estimation ([59]) and our methods presented in the next chapters. Despite the adaptive anisotropy of the operator, its main disadvantage, responsible for its low accuracy, was the locality error of curve fitting.

Chapter 4

Fast curvature estimation in one-ring neighborhoods

In this chapter, we present a new fast fixed region approach to estimate the curvature tensor in polygon meshes. The basic idea is that we formulate the estimation problem as a linear least squares system, which is constrained inside one-ring neighborhoods of points in polygon meshes. This first idea serves as the base to further constrain the curvature tensor insider bigger neighborhoods of points in either polygon meshes or point clouds and apply robust statistics methods to refine the fitting (as we will show in the next chapter). Our approach takes as input the vertices and their associated normals in one-ring neighborhoods of every vertex of interest, thus, the normals are not explicitly calculated in our method, but they are used as degrees of freedom.

In section 4.1, we briefly refer to direct estimation of normals in polygon meshes and point clouds. Then, in section 4.2, we give the formulation of our approach and in section 4.3, we present the solution for the curvature tensor. Finally, in section 4.4, we also extend our method to the estimation of derivatives of curvature.

4.1 Normal estimation

The estimation of the normal at each vertex of a polygon mesh or at each point in a point cloud is out of scope of this thesis. But as normals are input of our fixed region approach, their reliable calculation is important.

In polygon meshes, the normal is usually calculated as a weighted average of the incident face normals. However, several formula exist for the choice of weights of every face normal and each one has different properties. Apart from the obvious choice of face areas as weights for the face normals, Thurmer and Wuthrich [70] suggest to take the incident angles of each face at a vertex as weights based on the idea that the normal vectors should be defined very locally independently from the face areas or edge lengths. Max [47] derives a weighting scheme that allows to calculate normals exactly if the mesh approximates a sphere locally, independently from the exact tessellation. The weight for each face normal is the face area divided by the squared edge lengths that touch the center vertex. Langer *et al.* [36] suggests taking the sine of the incident triangle angle divided by the length of the triangle edge, opposing the center vertex and they show that the approximated normals at this case converge to the real normals linearly as the edge lengths tend to 0.

We decided to follow Max's weighting scheme, as Rusinkiewicz also did [59], and in all our experiments the same weighting scheme is chosen so that the comparisons between the curvature approaches are fair.

In point clouds, firstly, normals can directly be acquired as a part of the 3D scanning process. If not, there also exist approaches to calculate normals in point clouds. For example, one can best fit a plane in a neighborhood of each point (defined from the k-nearest points), and take its normal to be the normal of this plane [30, 44]. Alternatively, a patch can be fitted locally around a point and determine the normal analytically [19]. Mitra and Nguyen [50] find optimal neighborhood size to estimate normals, given the mean curvature, the local sampling density

and distribution at each point.

4.2 Least squares formulation for curvature tensor estimation

Our goal is to estimate the curvature tensors from the second fundamental form, expressed in terms of the shape operator, based on the covariant derivatives of the normal vector field:

$$\begin{bmatrix} \nabla_{\vec{u}}\vec{N}\cdot\vec{u} & \nabla_{\vec{v}}\vec{N}\cdot\vec{u} \\ \nabla_{\vec{u}}\vec{N}\cdot\vec{v} & \nabla_{\vec{v}}\vec{N}\cdot\vec{v} \end{bmatrix} \tag{4.1}$$

Rusinkiewicz [59] suggested to linear least squares fit the tensor per triangle given the samples of the triangle edge vectors $\vec{e}$ and their associated normal variations $\Delta\vec{n}$ in a 2D coordinate system defined on the tangent plane of the triangle:

$$\begin{bmatrix} \nabla_{\vec{u}}\vec{N}\cdot\vec{u} & \nabla_{\vec{v}}\vec{N}\cdot\vec{u} \\ \nabla_{\vec{u}}\vec{N}\cdot\vec{v} & \nabla_{\vec{v}}\vec{N}\cdot\vec{v} \end{bmatrix} \cdot \begin{bmatrix} \vec{e}\cdot\vec{u} \\ \vec{e}\cdot\vec{v} \end{bmatrix} = \begin{bmatrix} \Delta\vec{n}\cdot\vec{u} \\ \Delta\vec{n}\cdot\vec{v} \end{bmatrix} \tag{4.2}$$

The above system is derived from the linearity property of the covariant derivative operator. The approximation is based on the initial equation: $\nabla_{\vec{e}}\vec{N} = \Delta\vec{\mathrm{n}}$, where $\vec{e}$ is an edge vector and $\Delta\vec{\mathrm{n}}$ is the finite normal difference across this edge. Given the tangent directions $\vec{u}$ and $\vec{v}$ on the plane of the triangle, because of the linearity of the covariant derivative, we have:

$$\begin{aligned} &\nabla_{\vec{e}}\vec{N} = \Delta\vec{\mathrm{n}} \Rightarrow \\ &\nabla_{(\vec{e}\cdot\vec{u})\vec{u}+(\vec{e}\cdot\vec{v})\vec{v}}\vec{N} = \Delta\vec{\mathrm{n}} \Rightarrow \\ &(\vec{e}\cdot\vec{u})\nabla_{\vec{u}}\vec{N} + (\vec{e}\cdot\vec{v})\nabla_{\vec{v}}\vec{N} = \Delta\vec{\mathrm{n}} \end{aligned} \tag{4.3}$$

In our approach, we would like to constrain the fitting not only per triangle but per one-ring neighborhoods around each vertex, thus, the curvature tensor is directly computed from the fitting in the one-ring neighborhood, instead of rotating and averaging the curvature tensors per incident triangle for each vertex. Our idea was based on the remark that constraining the curvature tensor fitting from all the samples (edges and normal differences) inside the one-ring neighborhood, instead of partially constraining it per triangle and then averaging, would yield more robust and stable approximation of the tensor. Notice also the fact that the least squares method solves for minimized squared error, derived from maximum likelihood under Gaussian noise. This means that if the surface point samples and normals, around each point of interest, are independently random and distributed as a Gaussian distribution around the true model, implied by the underlying surface (which is a common and reasonable assumption), least squares guarantees maximum likelihood estimation of the fitted parameters, which give the curvature tensor in our case. A second important remark is that formulating the curvature tensor estimation problem, as a ”pure” least squares problem, given all the point and normal samples in a neighborhood around a point, allows us to employ all the appropriate robust statistics methods (for example, M-estimation in our case; see next chapter), to guarantee a robust estimation of curvature tensor by rapidly decreasing the influence of outliers that can probably exist in the associated neighborhoods of each point of interest, along with noise or imperfect tessellations.

However, an immediate problem that arises from the above remarks, is how the curvature tensor can be constrained inside a 3D neighborhood of each point, as the above solution of the system 4.2 is formulated based on a 2D parametrization, given the tangent vectors $\vec{u}$, $\vec{v}$, which is obviously the natural selection at each triangle (given by its tangent plane). But this is not straightforward in a one-ring or bigger neighborhood of a surface, as the edges now do not lie on the same plane and there can be no implied unique parametrization.

We overcome the above problem by expressing the equations for the covariant derivative 4.3

within a 3D orthonormal basis $(\vec{u}, \vec{v}, \vec{w})$ for each edge $\vec{e}$:

$$\begin{aligned} &\nabla_{\vec{e}}\vec{N} = \Delta\vec{\mathrm{n}} \Rightarrow \\ &\nabla_{(\vec{e}\cdot\vec{u})\vec{u}+(\vec{e}\cdot\vec{v})\vec{v}+(\vec{e}\cdot\vec{w})\vec{w}}\vec{N} = \Delta\vec{\mathrm{n}} \Rightarrow \\ &(\vec{e}\cdot\vec{u})\nabla_{\vec{u}}\vec{N} + (\vec{e}\cdot\vec{v})\nabla_{\vec{v}}\vec{N} + (\vec{e}\cdot\vec{w})\nabla_{\vec{w}}\vec{N} = \Delta\vec{\mathrm{n}} \end{aligned} \tag{4.4}$$

However, a mathematical abnormality occurs in the above formulation. When the $\vec{u}$ and $\vec{v}$ are chosen to be the tangent directions, thus $\vec{w} = \vec{n}$ where $\vec{n}$ is the normal of the point of interest, the definition of the normal vector field upon the points of the surface embedded in the R^3 space does not allow the existence of the covariant derivative of the vector field $\vec{N}$ at the direction of the normal $\vec{n}$ of each point. Note that the derivative is not equal to zero, as commonly misunderstood (the covariant derivative of a Euclidean vector field at the direction of its vector value at a point is not necessarily $\vec{0}$). It is undefined by the Levi-Civita connection on the tangent bundle of smooth Riemann manifolds, that provides a well-defined method for differentiating vector fields by defining a rule according to which, a vector in a surface point can be displaced from one point of a surface to another, which is trivial in a Cartesian coordinate system with a simple linear transformation but impossible in the tangent bundle of an arbitrary Riemann manifold. Of course, the Levi-Civita connection considers vectors in a given surface that are always bound at its points and lie in the corresponding tangent planes. However, in the case of a discrete surface, we pointed out from the very beginning, that the underlying represented surface may not be uniquely defined. Furthermore, in the case of noise, the sample points and their associated vectors are displaced from any underlying hypothesized surface.

In order to proceed with the 3D fitting of the curvature tensor and overcome this abnormality, without assuming any arbitrary parametrization or making any other assumption about the local behavior or noise of the surface, which are clearly unknown, we allow the existence of all covariant derivatives of the given vector field $\vec{N}$, now, in Euclidean space, especially, driven by the fact that the nature of a discrete surface, together with the existence of noise,

causes the sample points and their associated vectors to be displaced in an unknown way in Euclidean space and not to be necessarily bound in the tangent bundle of a uniquely defined embedded smooth manifold. More specifically, we treat the covariant derivatives in all three directions of the chosen orthonormal basis as unknowns that will be solved for, in the linear least squares formulation. That, at least, guarantees their maximum likelihood estimation under Gaussian noise. Then, by solving for the unknown covariant derivatives of the Euclidean vector field, having known values only on the given sample points, any candidate 2D manifold, at each neighborhood of sample point, inherits, in its shape operator, the estimated covariant derivatives in any candidate tangent directions $\vec{u}$ and $\vec{v}$:

$$\begin{bmatrix} \nabla_{\vec{u}}\vec{N}\cdot\vec{u} & \nabla_{\vec{v}}\vec{N}\cdot\vec{u} & \nabla_{\vec{w}}\vec{N}\cdot\vec{u} \\ \nabla_{\vec{u}}\vec{N}\cdot\vec{v} & \nabla_{\vec{v}}\vec{N}\cdot\vec{v} & \nabla_{\vec{w}}\vec{N}\cdot\vec{v} \end{bmatrix} \cdot \begin{bmatrix} \vec{e}\cdot\vec{u} \\ \vec{e}\cdot\vec{v} \\ \vec{e}\cdot\vec{w} \end{bmatrix} = \begin{bmatrix} \Delta\vec{n}\cdot\vec{u} \\ \Delta\vec{n}\cdot\vec{v} \end{bmatrix} \tag{4.5}$$

or more compactly:

$$X \cdot A = B \text{ where:}$$

$$X = \begin{bmatrix} \nabla_{\vec{u}}\vec{N}\cdot\vec{u} & \nabla_{\vec{v}}\vec{N}\cdot\vec{u} & \nabla_{\vec{w}}\vec{N}\cdot\vec{u} \\ \nabla_{\vec{u}}\vec{N}\cdot\vec{v} & \nabla_{\vec{v}}\vec{N}\cdot\vec{v} & \nabla_{\vec{w}}\vec{N}\cdot\vec{v} \end{bmatrix} = \begin{bmatrix} a & b & c \\ b & d & e \end{bmatrix}$$

$$A = \begin{bmatrix} \vec{e}\cdot\vec{u} \\ \vec{e}\cdot\vec{v} \\ \vec{e}\cdot\vec{w} \end{bmatrix}, B = \begin{bmatrix} \Delta\vec{n}\cdot\vec{u} \\ \Delta\vec{n}\cdot\vec{v} \end{bmatrix}$$

This system needs at least three edge vectors with their normal differences in order to produce a non-degenerate solution. Of course, at every vertex in a polygon mesh, there must be at least one triangle (even if it is a border vertex), thus, the system produces always a solution except if a triangle is degenerate, meaning that it has two co-incident edges (thus, face area zero). In this case, we simply ignore any such triangle.

Our goal, then, is to apply the least squares method with the use of the proper weights for each data sample i.e. for each edge with its associated normal curvature ($\vec{e}\cdot\vec{u}$, $\vec{e}\cdot\vec{v}$, $\vec{e}\cdot\vec{w}$, $\Delta\vec{n}\cdot\vec{u}$, $\Delta\vec{n}\cdot\vec{v}$), where $\vec{u}$ and $\vec{v}$ are orthogonal tangent directions and $\vec{w}=\vec{n}$ is the estimated normal at the vertex of interest. Our experiments, firstly, confirmed Rusinkiewicz's choice [59] to use the mixed Voronoi area as defined in [49]. In our case, every edge, that is shared with two triangles, is associated with a weight depending on the sum of the mixed Voronoi areas of these two triangles respect to the center vertex. This is a safe choice, which works for all different cases of triangle angles around the point of interest (acute, obtuse, or right angles). If the edge is border edge or not touching the center vertex, we use only the mixed Voronoi area of the one triangle it belongs to.

However, as Langer and Belayev [36] noticed, convergence of the discrete operators with the use of Voronoi areas as weights for the normal curvatures, is achieved when the edge lengths and angles in the one-ring neighborhood are the same. Driven by Hameiri's and Shimshoni's [25] suggestions, we also penalize normal curvatures based on the inverse of distance of points from the center vertex i.e. the edge lengths, so the discrete operator becomes more isotropic in each neighborhood. In all of our experiments, this yielded much more accurate and stable results.

Thus, the weighted least squares solution for the above system 4.5 is given by the minimization of the weighted squared error:

$$E(a,b,c,d,e)=\sum_{edges\ \vec{e}_i} W_i \left\|X\cdot A_i - B_i\right\|_2 \tag{4.6}$$

where the weight for each edge is

$$W_i=\frac{\sum_{triangles\ j\ sharing\ i} VA_j}{l_i^2}$$

where l_i is the edge length. By taking the partial derivatives of the error E respect to the five

parameters, we can find the associated linear system that solves the system, that will be shown in the next section.

4.3 Solving the least squares problem for the curvature tensor efficiently

We would like to solve the system $XA = B$, where for simplicity reasons, we use the following symbols:

$$X = \begin{bmatrix} a & b & c \\ b & d & e \end{bmatrix}, A = \begin{bmatrix} \vec{e}_i \cdot \vec{u} \\ \vec{e}_i \cdot \vec{v} \\ \vec{e}_i \cdot \vec{w} \end{bmatrix} = \begin{bmatrix} u_i \\ v_i \\ w_i \end{bmatrix}, B = \begin{bmatrix} \Delta\vec{n}_i \cdot \vec{u} \\ \Delta\vec{n}_i \cdot \vec{v} \end{bmatrix} = \begin{bmatrix} \kappa_i \\ \lambda_i \end{bmatrix}$$

Typically, the linear least squares solution is given by the normal equation: $X = (U^T U)^{-1} U^T b$, where U is a $2nx5$ matrix and b is a $2nx1$, where n is the number of samples:

$$U = \begin{bmatrix} u_1 & v_1 & w_1 & 0 & 0 \\ \dots & \dots & \dots & 0 & 0 \\ u_n & v_n & w_n & 0 & 0 \\ 0 & u_1 & 0 & v_1 & w_1 \\ 0 & \dots & 0 & \dots & \dots \\ 0 & u_n & 0 & v_n & w_n \end{bmatrix}, b = \begin{bmatrix} \kappa_1 \\ \dots \\ \kappa_n \\ l_1 \\ \dots \\ l_n \end{bmatrix}$$

Note that the system is over-determined when $n > 3$. Alternatively, the same result for X can be acquired by setting the partial derivatives of the objective function 4.6, to zero, so that we

get the following:

$$E = \sum_i W_i(au_i + bv_i + cw_i - \kappa_i)^2 + \sum_i W_i(bu_i + dv_i + ew_i - \lambda_i)^2$$

$$\frac{\partial E}{\partial a} = 2\sum_i u_i W_i(au_i + bv_i + cw_i - \kappa_i) = 0 \Rightarrow$$

$$a\sum_i u_i^2 W_i + b\sum_i u_i v_i W_i + c\sum_i u_i w_i W_i - \sum_i u_i \kappa_i W_i = 0$$

$$\frac{\partial E}{\partial \beta} = 2\sum_i v_i W_i(au_i + bv_i + cw_i - \kappa_i) + 2\sum_i u_i W_i(bu_i + dv_i + ew_i - \lambda_i) = 0 \Rightarrow$$

$$a\sum_i u_i v_i W_i + b\sum_i (v_i^2 + u_i^2) W_i + c\sum_i v_i w_i W_i + d\sum_i u_i v_i W_i + e\sum_i u_i w_i W_i - \sum_i v_i \kappa_i W_i - \sum_i u_i \lambda_i W_i = 0$$

$$\frac{\partial E}{\partial c} = 2\sum_i w_i W_i(au_i + bv_i + cw_i - \kappa_i) = 0 \Rightarrow$$

$$a\sum_i u_i w_i W_i + b\sum_i v_i w_i W_i + c\sum_i w_i^2 W_i - \sum_i w_i \kappa_i W_i = 0$$

$$\frac{\partial E}{\partial d} = 2\sum_i v_i W_i(bu_i + dv_i + ew_i - \lambda_i) = 0 \Rightarrow$$

$$b\sum_i u_i v_i W_i + d\sum_i v_i^2 W_i + e\sum_i v_i w_i W_i - \sum_i v_i \lambda_i W_i = 0$$

$$\frac{\partial E}{\partial e} = 2\sum_i w_i W_i(bu_i + dv_i + ew_i - \lambda_i) = 0 \Rightarrow$$

$$b\sum_i u_i w_i W_i + d\sum_i w_i v_i W_i + e\sum_i w_i^2 W_i - \sum_i w_i \lambda_i W_i = 0$$

Thus, first of all, we precompute the mixed Voronoi areas for each triangle respect to each of its three vertices. Then, at every point of interest, we get the edges in its one-ring neighborhood (a half-edge structure accelerates such query) and we solve the above 5x5 system. The pseudocode for all of these is the following:

for every vertex i {

for every edge j in neighborhood of i{

1. initialiaze 5x5 empty matrix A and 5x1 matrix B.
2. retrieve the mixed Voronoi area of the triangles associated with j, respect to i, and divide their average with squared edge length to get the weight W_i.
3. compute u_i, v_i, w_i, κ_i, λ_i (just dot products) and self multiply them with $\sqrt{W_i}$.
4. A[0][0] += u_i * u_i; A[0][1] += u_i * v_i; A[0][2] += u_i * w_i; B[0] += κ_i * u_i;

A[1][2] += v_i * w_i; B[1] += κ_i * v_i + λ_i * u_i

A[2][2] += w_i * w_i; B[2] += κ_i * w_i;

A[3][3] += v_i * v_i; B[3] += λ_i * v_i; B[4] += λ_i * w_i;

}

A[1][0] = A[0][1]; A[1][3] = A[0][1]; A[1][4] = A[0][2]; A[1][1] = A[0][0] + A[3][3];

A[2][0] = A[0][2]; A[2][1] = A[1][2]; A[3][1] = A[0][1]; A[3][4] = A[1][2];

A[4][1] = A[0][2]; A[4][3] = A[3][4]; A[4][4] = A[2][2];

solve for X = inv(A) * B (e.g. use Lapack)

create 2x2 matrix $Y = \begin{bmatrix} X[0] & X[1] \\ X[1] & X[3] \end{bmatrix}$

diagonalize X and get eigenvalues (principal curvatures)

and eigenvectors (that yield the principal directions vectors with a simple linear transformation)

}

The above pseudocode yields the minimum required mathematical operations to solve the system. For example, we don't need to put all the elements of A and B inside the second loop as most of them are redundant. This saves a lot of additions and multiplications per edge. The solution for X by inversing A and multiplying it by B can be executed e.g. with an efficient linear programming library or function (e.g. in our case, we use the dgesv function of Lapack [52]). The diagonalization of a 2x2 matrix to its eigenvalues and eigenvectors is trivial. In the end, we get the principal curvature values and principal directions for each vertex.

By incorporating the above code to an efficient library, like trimesh2 [58], the principal curvature values and directions can be computed for a polygon mesh, with about 1M faces, in 10 seconds approximately (Pentium 4 3GHz with 2GB memory).

4.4 Solving the least squares problem for the derivative of curvatute tensor efficiently

Like Rusinkiewicz [59], we can also extend our method to estimate the derivative of the curvature tensors. This derivative shows how the curvature is changed from point to point in the surface along a direction and is mostly used for crease detection [38] and line drawing [11] in computer graphics applications. In our case, we do not need to re-project the curvature tensor from each vertex to its faces, estimate the derivative of curvature tensor per face and then re-project again the derivative of the curvature tensor back to the vertices, as Rusinkiewicz did. We again constrain the derivative curvature tensor per one-ring neighborhood and we directly solve for the derivative in a least-square formulation for each vertex.

The derivative of the curvature tensor is a symmetric 2x2x2 tensor, having the following form:

$$C = \begin{pmatrix} \nabla_u II & \nabla_v II \end{pmatrix} = \left\{ \begin{pmatrix} a & b \\ b & c \end{pmatrix} \begin{pmatrix} b & c \\ c & d \end{pmatrix} \right\}$$

Again, we add the necessary degrees of freedom to perform the 3D fitting of the above tensor in the following linear system:

$$\left\{ \begin{pmatrix} a & b & e \\ b & c & f \end{pmatrix} \begin{pmatrix} b & c & f \\ c & d & h \end{pmatrix} \right\} \begin{bmatrix} \vec{e}_i \cdot \vec{u} \\ \vec{e}_i \cdot \vec{v} \\ \vec{e}_i \cdot \vec{w} \end{bmatrix} = \left\{ \begin{bmatrix} \Delta(\nabla_u \vec{N} \cdot \vec{u}) \\ \Delta(\nabla_u \vec{N} \cdot \vec{v}) \end{bmatrix} \begin{bmatrix} \Delta(\nabla_v \vec{N} \cdot \vec{u}) \\ \Delta(\nabla_v \vec{N} \cdot \vec{v}) \end{bmatrix} \right\}$$

With some notation simplification, we have:

$$\left\{ \begin{pmatrix} a & b & e \\ b & c & f \end{pmatrix} \begin{pmatrix} b & c & f \\ c & d & h \end{pmatrix} \right\} \begin{bmatrix} u_i \\ v_i \\ w_i \end{bmatrix} = \left\{ \begin{bmatrix} \kappa_\iota \\ \lambda_\iota \end{bmatrix} \begin{bmatrix} \lambda_\iota \\ \mu_\iota \end{bmatrix} \right\}$$

That leads to the minimization of the following objective function:

$$E(a,b,c,d,e,f,h) = \sum_i W_i(au_i + bv_i + ew_i - \kappa_i)^2 + 2\sum_i W_i(bu_i + cv_i + fw_i - \lambda_i)^2 + \\ + \sum_i W_i(cu_i + dv_i + hw_i - \mu_i)^2$$

which, obviously, by setting the partial derivatives to 0, leads to a 7x7 linear system, that can similarly be solved efficiently as in the above section. We note that we use for every edge, the same weight as above. By modifying the trimesh2 library with the above approach, the derivative of the curvature tensor for a polygon mesh, with about 1M faces, can be estimated in 35 seconds approximately (Pentium 4 3GHz with 2GB memory).

Chapter 5

Curvature estimation using robust statistics

In the previous chapter, we showed how we can treat the curvature estimation as a least squares fitting problem in a fixed neighborhood around each point of interest. Here, we exploit this framework in order to estimate curvature, based on an M-estimation algorithm, from the robust statistics literature [32, 26, 16]. M-estimators have the advantage of robustly fitting a model by minimizing an objective function of the residuals of the samples with an Iterative Reweighted Least Squares (IRLS) scheme. Starting from a reasonably large area (support area) initially, the algorithm detects and gradually rejects noise and structured outliers (e.g. samples past feature edges or irrelevant normal variations, so that irrelevant curvature values are not propagated to nearby points).

It has been perceived many times by many researchers (e.g. [59][17]) that discrete operators, working in small neighborhoods (e.g. using the incident edges on each vertex or the one-ring neighborhood, thus, including also the non-touching edges on each vertex) give the safest and most accurate results. But, unfortunately, this holds in non-noisy cases. In the case of noise,

which is very common practically, these methods become sensitive and lack in robustness. More stable results are achieved by considering two-ring neighborhoods (e.g. [19]) or even bigger user-defined neighborhoods, so that smooth results are returned at least (e.g. [63]). However, a user-defined region (e.g. given a fixed radius of a spherical or geodesic neighborhood), does not solve the problem as users may be unfamiliar with the amount of noise existing in a mesh, and more significantly, it is preferable to minimize user intervention in geometry applications. Fixed regions have also the disadvantage, that outliers can very easily infiltrate curvature estimation. Obviously, an important issue is that sharp edges (and in general, fine features), should be preserved in the curvature field. It would be also unwise to propagate extreme values of curvature in neighbor points of sharp edges, when the discrete operator is working with arbitrary fixed regions. Again, we would like to minimize user interaction, for example, we do not force users to choose every feature edge on the discrete surface manually. In contrast with the previous approaches, our algorithm automatically tends to give more weight to samples that are closer to each vertex of interest in non-noisy cases (preserving accuracy), while it behaves more smoothly when noise exists, so that both accuracy and robustness is preserved in the curvature estimation.

M-estimation has been used in various computer vision applications (see [65] for a survey), such as motion estimation [60, 5], robust local surface descriptors with planar patches [4], robust edge detection [43], 2D/3D object tracking [57], fundamental matrix estimation (that is used to find image-to-image transformations, camera motion, and scene structure) [72] etc. Recently, it has also been used in computer graphics for dominant global and local symmetry detection [62].

In general, robust statistics methods have three main properties: a) the breakdown point, which is the minimum fraction of outlying data that can cause an estimator to choose a totally different model from the real one (in least squares, the breakdown is 0), b) the influence function, which is the change of an estimate, after introducing an outlier, as a function of the distance from the

"uncorrupted" estimate (e.g. in least squares, the influence is proportional to the distance of the point from the estimate) and c) statistical efficiency, which is the minimum possible variance for an unbiased estimator (determined by a target distribution, such as Gaussian) divided by its actual variance. Robust estimators, having high breakdown point, tend to have low statistical efficiency, so many data points are required to obtain stable estimates.

M-estimators have the advantage of robustly fitting a model by minimizing an objective function with an Iterative Reweighted Least Squares (IRLS) scheme, so that they can relatively quickly find a local minima of the function, while other robust statistics methods, like Least Median Squares (LMS) cannot be reduced to an IRLS approach and need to be solved by an exhaustive search in the space of all possible estimates (thus, in LMS, randomly chosen subsets are analyzed). Because of their high statistical efficiency, M-estimators work well with small datasets and in our case, the samples for curvature estimation are necessarily limited to a local neighborhood of each point. However, the breakdown of M-estimators is 0, because of leverage points, which are points positioned far from the remainder data in the dimensions associated with the independent variables, as well as far from the fit to the uncorrupted data. GM-estimators treats this problem by decreasing appropriately the influence of points in both dependent and independent variables. In our case, we use GM-estimators, exhibiting this property of low non-zero breakdown point. On the other hand, bounded influence regions, such as LMS or LTS (Least Trimmed Squares) have breakdown point as high as 50%.

For the curvature estimation problem, we have experimented with other robust statistics methods, such as LMS or forward search with iterative refitting, like in [15], which is used in robust surface reconstruction with fine features, but we have decided to stick to M-estimation (more specifically GM-estimation) for the following reasons:

- *computational efficiency*: the IRLS framework gives a significant computational advantage in M-estimators, where the structural outliers are gradually rejected in a top-down approach [62], instead of a much slower growing region technique around each point of

interest.

- *stability*: Curvature estimation should always be localized in a neighborhood of a point, thus, the samples should be limited inside this region. The need for a high breakdown point, such as in LMS, requires having enough inlier data to satisfy it. Otherwise, there is a significant danger, that the estimate includes a relatively too small subset of data, considered as inliers, to guarantee a safe approximation of curvature. Another issue, that we noticed in our experiments with forward search, was that the estimate could easily consider many of the samples from the one-ring neighborhood of a point as outliers, tending to prefer other neighbor regions, with lower noise, but of course, the returned result did not have any correspondence with the curvature of the point of interest. On the other hand, the low breakdown point of M-estimators can be misleading; with a relatively safe fitting in the beginning, together with an influence function that quickly tends to 0, sufficient number of outliers, if exist, are rejected safely and efficiently.

- *low or no dependence on user parameters*: M-estimation does not depend heavily on user parameters, especially with an automatic robust scale estimation [32], whether LMS heavily depends on the choice of the size of the random subsets (low subset size would favor instability in the curvature estimation where high subset size would favor noise sensitivity and bigger running times) or a maximum tolerated residual in the case of forward search, where in the case of curvature estimation, it would be difficult for a user to choose it, even with an interactive method, like in [15], as they may not be able to judge the behavior of the curvature field on a surface.

Therefore, we have decided to choose M-estimation for the curvature estimation problem with robust statistics. In section 5.1, we describe how we acquire the samples and their associated weighting around each point of interest in either polygon meshes or point clouds and next, in section 5.2, we show how M-estimation is exactly employed in our problem. Finally, in section 5.3, we summarize all the necessary steps of our algorithm.

5.1 Sampling and weighting normal variations around each point of interest

In the previous chapter, where we analyzed the fixed neighborhood curvature estimation approach, the samples of the distance vectors and their associated normal differences were simply taken from the edges of each one-ring neighborhood in a polygon mesh. An important issue, now that we extend our approach beyond the one-ring neighborhoods, is how we gather the samples and their weighting meaningfully in either a polygon mesh or a point cloud around each point of interest.

5.1.1 Acquiring samples

Firstly, we remind the reader that each sample e_i is 5D, consists of the three components of each distance vector in the chosen 3D orthonormal basis, ($\vec{e}\cdot\vec{u}$, $\vec{e}\cdot\vec{v}$, $\vec{e}\cdot\vec{w}$) and the two tangential components of the normal differences ($\Delta\vec{n}\cdot\vec{u}$, $\Delta\vec{n}\cdot\vec{v}$). In order to robustly estimate curvature at a point on a discrete surface, we would like to get a detailed "holistic" view of the normal variations around it, independently from the exact mesh connectivity, which should have minimum effects on the curvature estimation i.e. curvature estimation should be independent from all possible tessellations of the surface. As a result, we acquire a dense sampling of normal variations around each point of interest inside the support area, by including the normal differences of all pairs of points inside the support region, in the linear system 4.5. This dense sampling proved to be much more stable and accurate, rather than taking into account only the associated normal variations across the existing edges of the mesh. Of course, the price to pay is that the overdetermined linear system 4.5 includes many more samples. In regular meshes with equidistant vertices, the sampling based on edges returns very similar results with the dense all-pairs sampling, however, in irregular and noisy meshes, the dense sampling of

normal variations yields more accurate and stable results.

In our tests, presented in the next chapter, we show error curves based on both the edge sampling "mode" of our algorithm and the dense sampling mode, so that their difference in accuracy becomes clear. In our implementation of our algorithm, the default mode for the execution of the method is the dense sampling mode. Edge sampling mode is only suggested when users are sure of the relatively good quality of the meshes (in terms of mesh connectivity and point sampling rate) and point clouds (in terms of the point sampling rate). The advantage of the edge sampling mode is computational, as it is e.g. 3-4 times faster than the dense sampling mode.

In order to save some time and quickly prune obvious outliers, initially, we prune any samples, whose normals have bigger than $\pi/2$ angle with the normal of the center point. In polygon meshes, especially, we also prune any samples, whose points have one of their incident face normals (per vertex face normals), creating a more than $\pi/2$ angle with the center point normal.

5.1.2 Weighting scheme for samples

The IRLS process of the M-estimation reweights the samples accordingly, in order to minimize an objective function of their residuals error. However, as Simari *et al.* [62] noticed, the M-estimation weights should be multiplied with initial weights that also capture the geometric importance of the samples, to guarantee stable behavior in the presence of non-uniform tessellations. In the fixed one-ring neighborhood, this geometric importance of the samples was captured by their associated Voronoi area and their edge length. In our case, where the samples are not taken inside the one-ring neighborhoods and are not limited to the mesh connectivity, such weighting would not make any sense. The weighting of the samples, taken by our algorithm, should be related to the fact that they are not of equal quality, otherwise the fit of the curvature tensor would be influenced by the samples of "poor quality". The statistics literature

[7], here, reminds us of the fact that weights of the samples should be scaled with a factor that depends on the variances of the residuals of the samples i.e. given the variances of the samples σ_i, their weights should be $1/{\sigma_i}^2$, so that the response variances are transformed to a constant value (note that we will not ignore the weighting of the samples as it was given in the one-ring neighborhood approach, as the initial guess that our algorithm makes, before it starts the IRLS process, is the one-ring neighboorhood estimate).

Unfortunately, these optimal weights, which are based on the inverse of the true squared variances of each data point, are never known, obviously. Thus, we have to estimate the weights that should be used. A strategy for such estimation is to find a function that relates the variance with some property values of the samples. Starting from the reasonable remark, that the normal variation samples should be less trusted, when their associated points are further from the center point of interest, we assume that a weighting that decreases with the inverse of the average squared distance of the points of the samples from the center point would be reasonable. This remark was confirmed in the plots of the variance of the residuals of the samples respect to their average distance from the center point for several test surfaces, whose curvature tensor is known and where we can compute the variances obviously. In polygon meshes, we use the geodesic distance (that can quickly be approximated with Floyd's or Dijkstra's algorithm) while in point clouds, we use the Euclidean distance between points. We assume that such weighting is reasonable in the general case and is also similar to the weighting that was suggested in Hameiri's and Shimshoni's [25] approach. In figures 5.1, we show plots for three characteristic cases of the sphere, cylinder and monkey saddle. The samples are taken inside 3-ring neighborhoods around the vertices for the purposes of these plots. Then the residuals of the samples are grouped into bins according to their corresponding average geodesic distance and we measure the variance of the residuals on each bin.

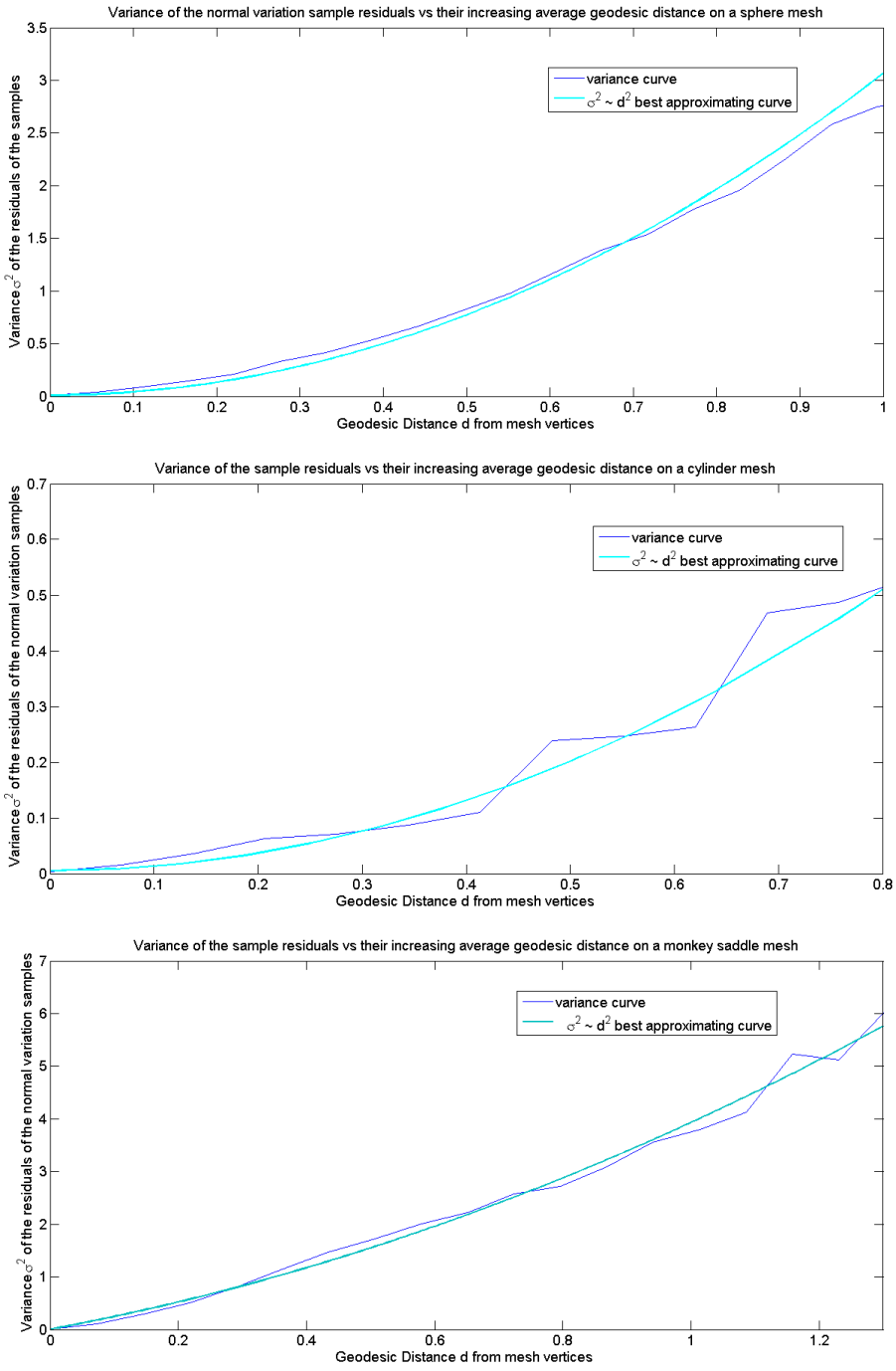

Figure 5.1: Variance σ^2 of the residuals of the normal variation samples versus their average geodesic distance from the mesh vertices on sphere, a cylinder and a monkey saddle. The best approximating quadratic curve on the variances of the samples is also drawn to justify our assumption about the initial weighting scheme we use in the M-estimation algorithm

5.1.3 Support region

The support region contains all points whose normal differences will be taken into account for the solution of the system for the curvature tensor fitting. The support area should not be too small so that it guarantees a stable fitting and not too large as we would not like to have too many outliers initially and too many samples that eventually would slow up the solution of the system. The support area includes all points inside a sphere of a radius of 3.0 multiplied by the average distance of each vertex of interest from its immediate neighbor vertices in its one-ring neighborhood in polygon meshes or the typically K=6 nearest points in a point cloud. Thus, the support area is varying in order to be adapted to the local surface sampling. Note also, in the case of polygon meshes, we always include the one-ring vertices independently from their distance, to always ensure stability. The above factor 3.0 for the support region was determined to be a good heuristic experimentally; increasing that factor had little effect on curvature estimation, due to our weighting scheme and the M-estimation process that tends to reject the increasing number of outliers, when the support region becomes larger and larger (note also that the samples outside this region would get weights almost 10 times smaller than the samples having the immediate neighbor points of each point of interest on average). However, in the case of irregular meshes, determining the support area based on average distance of their immediate neighbor (one-ring) vertices to the vertices of interest was not always the best choice. In fact, we found that determining the support region from the typically K=6 nearest points of each vertex of interest also tended to be a bit better heuristic for polygon meshes, too, in the general case. In our experiments, we used this heuristic.

5.1.4 Initial guess of the model

In order to start the M-estimation algorithm with the IRLS process, an initial guess for the curvature tensor of each point of interest should be made. The safest and most accurate guess

is the one-ring estimate in polygon meshes in the non-noisy case, as exactly described in the previous chapter. In the case of point clouds, we perform a fitting of the normal differences of the typically K=6 nearest points, where we weight them only based on their euclidean distance from the center vertex. Note that the GM-estimator, with automatic scale estimation, performs robustly even when good initial guesses do not always exist [60].

5.2 GM-estimation for curvature estimation

In the M-estimation formulation, the goal is to minimize an objective function of the residuals error:

$$\min_{x} \sum_{e_i \in E} \rho(r_{i,x}/\sigma) \tag{5.1}$$

where x is the model with the unknown parameters, containing the curvature tensor, e_i are our 5D samples, E is the sample set, $r_{i,x}$ is the squared distance of the sample e_i from the model x and σ is the given scale factor, that will automatically be estimated. Finally, ρ is a robust loss function that is monotonically non-increasing with decreasing residuals. In least squares, the loss function is: $\rho(r_{i,x}/\sigma) = \frac{r_{i,x}^2}{2\sigma^2}$. However, such quadratic loss function would be very sensitive to outliers, resulting in over-smoothing the curvature field and destroying its fine features, especially if the operator is applied to an area bigger than the associated one-ring neighborhood around each point of interest. Thus, in our case, we use the Geman-Mclure cost function (GM), which is a common choice in computer vision [16] and graphics applications [62], as it exhibits very good behavior in quickly rejecting structural outliers.

The minimization of eq. is achieved by solving:

$$\sum_{e_i \in E} \psi(r_{i,x}/\sigma) \frac{dr_{i,x}}{dx} \frac{1}{\sigma} = 0$$

where $\psi(r_{i,x}/\sigma) = \rho'(r_{i,x}/\sigma)$. A common way to solve the above equation is to introduce a process, called Iterative Reweighted Least Squares (IRLS), with a weight function, expressed for each sample, as $w(r_{i,x}/\sigma) = \frac{\psi(r_{i,x}/\sigma)}{r_{i,x}/\sigma}$.

In GM-estimation, the cost function and the weights are given as (a plot diagram is also shown in figures 5.2 and 5.3):

$$\rho(r_{i,x}/\sigma) = \frac{(r_{i,x}/\sigma)^2}{1+(r_{i,x}/\sigma)^2}, \; w(r_{i,x}/\sigma) = \frac{2}{(1+\frac{r_{i,x}^2}{\sigma^2})^2}$$

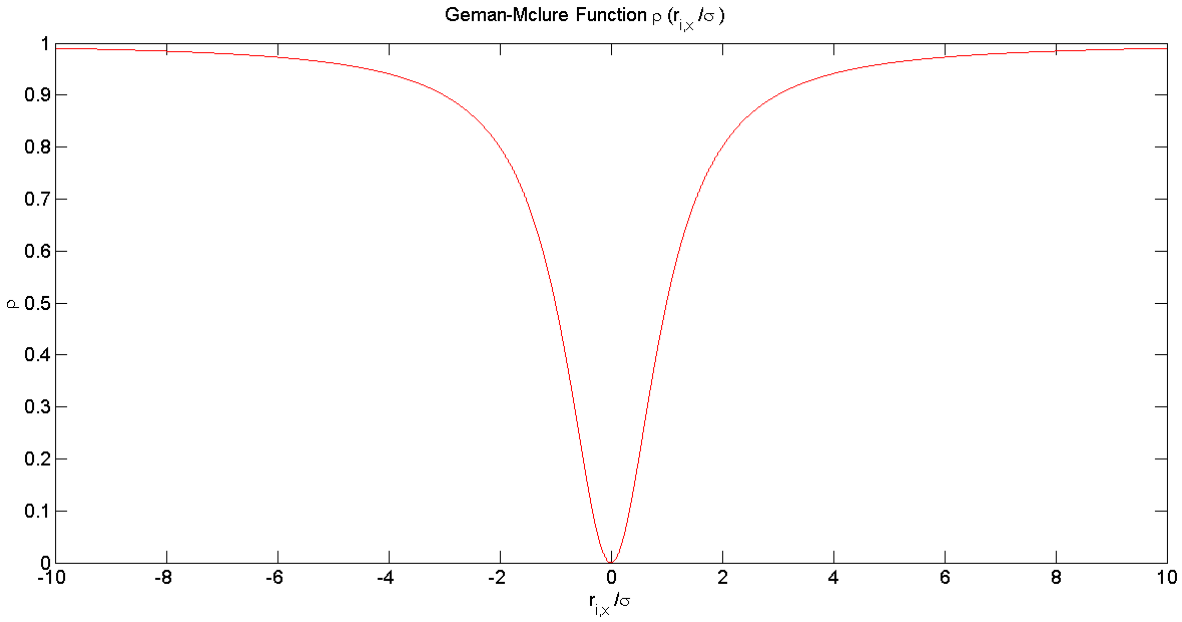

Figure 5.2: **Left:** GM cost function

The parameter σ, called scale estimation, governs the inflection point beyond, which there is sharper decrease of the weights. By assuming that the residuals are modeled with a contaminated Gaussian distribution, as severals authors also suggest [16], which is also reasonable in our case, the sigma is automatically estimated as:

$$\sigma = 1.4826 \cdot median\,|r_{i,x}|$$

following from the fact that median value of the absolute values of a large enough sample of unit variance normal-distributed 1D values is $0.6745 = 1/1.4826$. This scale estimation causes

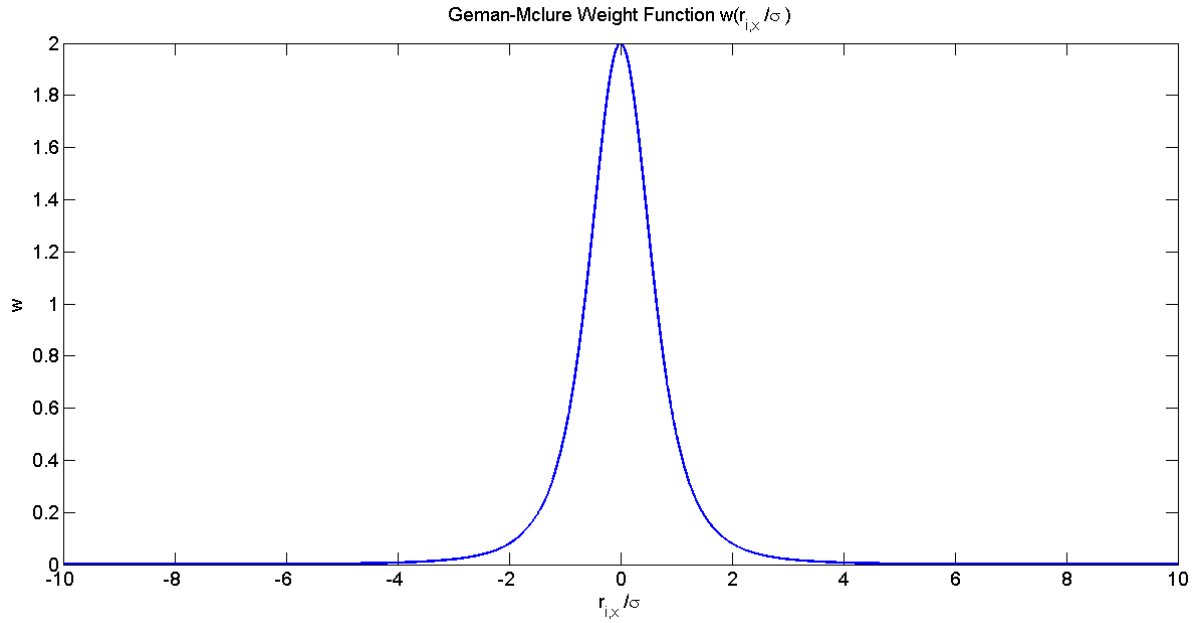

Figure 5.3: GM weight function

the estimator to tolerate almost 50% of any outliers so that it has strong resistance to outliers [60]. We not also let σ to fall below its initial value, as given by the initial fitting of the tensor from the one-ring neighborhood estimate, in order to guarantee stability (similarly to [62]). In order to control leverage, as also in [62] and in [60], we set the weights of the non one-ring neighborhood samples with residuals more than 2σ, equal to 0, so that they are considered as structured outliers and do not affect the curvature estimation at all.

5.3 The robust curvature estimation algorithm

The algorithm for the robust curvature estimation starts with gathering the normal variation samples, in which the support region of each point of interest, and initializing the model x, as described in section 5.1. We ran the IRLS process for typically 50 iterations or until convergence (the relative difference of the Frobenius norm of the matrix x with the estimated parameters of the current iteration, respect to the Frobenius norm of x of the previous iteration

is less than 1e-10), and we choose the parameter values for the iteration achieving the lowest cost function value. If there is considerable noise inside the support region around each point, then the M-estimator, because of a higher value of median of the residuals (and thus, σ), will yield a slower decreasing weight function, giving more weight to points further from the center point. Otherwise, it will tend to estimate the curvature tensor based on the closest neighbors of each point of interest. This fact is shown in figures 5.4 and 5.5.

Figure 5.4 shows how the cost function and the principal curvature estimates changes during the IRLS process for a vertex on the top of a torus mesh. We also visualize the weights, that are corresponding to its neighbor points in the support region, based on the sum of the weights given to the normal variation samples which they participate, for the first and best IRLS iteration. Then, in the next figure 5.5, we show the same characteristic figures for another point on the side of the torus with added heavy normal noise (of 20% variance of its median edge length) to its vertices.

In both cases, the figures show the fact that the M-estimation automatically converges to a smoother estimate, when noise exists, so that the accuracy of the curvature estimation is dramatically improved, compared to other methods. For example, on the examined vertex of the noisy torus case of figure 5.5, our method starts from a bad one-ring estimate of principal curvatures, which is (k_1, k_2) = $(.5787, .2266)$ and converges to (k_1, k_2) = (.9954, .2088). Meyer *et al.*'s one-ring method [49] gives (k_1, k_2) = $(2.5255, 0.3053)$, the normal cycle one-ring method [10] gives (k_1, k_2) = $(2.6454, 0.2892)$, Goldfeather's and Interrante's [19] two-ring cubic fitting method gives (k_1, k_2) = $(0.6791, 0.2279)$, Rusinkiewicz's [59] gives (k_1, k_2) = $(0.4575, 0.2001)$. Finally, Pottmann *et al.*'s [56] integral invariant method, returns (k_1, k_2) = $(0.9413, 0.1900)$, by performing a manual exhaustive search of spherical kernel radii, where the method returns the most accurate results (of course, the choice of the size of the kernel in this method is unclear if minimization of curvature tensor error is desired). The analytical principal curvature values for the same vertex are (k1,k2)=(1,0.2). Of course, these results are

indicative, extensive tests involving RMS error on all the vertices of several test meshes, under difference variances of noise and different tessellations will be given in the chapter 6.

Figure 5.6 shows how the estimator converges to a good estimation of the curvature tensor of a vertex on a locally cylindrical surface which is close to a boundary edge by rejecting any structured outliers caused by the surface samples past the feature boundary. In contrast, an estimate that was based on a two-ring or bigger neighborhood, would get a totally wrong estimate for the same point.

Finally, we note that due to the iterative local optimization nature of our robust M-estimation algorithm, curvature estimation becomes slower than our fixed ring neighborhood approach, presented in chapter 4. Indicatively, our CGAL [41] implementation needs about 2-3 seconds for a polygon mesh of 10K polygons while for a mesh of 100K polygons, it may need approximately 30-40 seconds; of course running times are varying for each surface. For 1M polygons, the running time can be as more as 5 minutes.

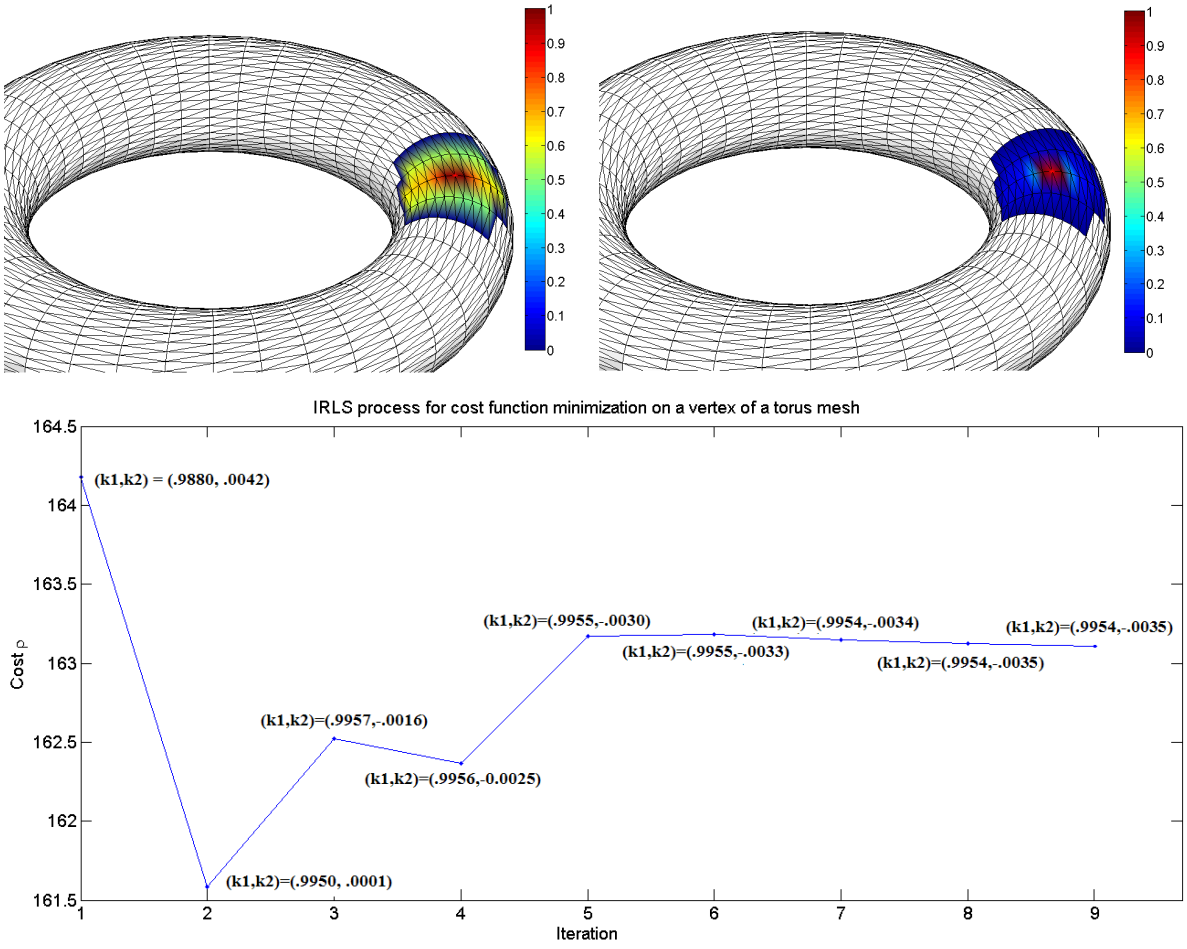

Figure 5.4: **Top Left:** Visualization of the weights corresponding to the vertices of the support region, by summing up the weights assigned to their associated normal variation samples, during the first iteration of the IRLS process. **Top Right:** Same visualization of the weights for the iteration, achieving lowest cost function value. **Bottom** Cost function minimization process during the IRLS algorithm. Note that the the optimization is non-convex and we choose the parameter values for the lowest cost function value. Convergence to another local minima of the cost function is achieved after 9 iterations. The analytical principal curvature values for the same vertex are (k1,k2)=(1,0).

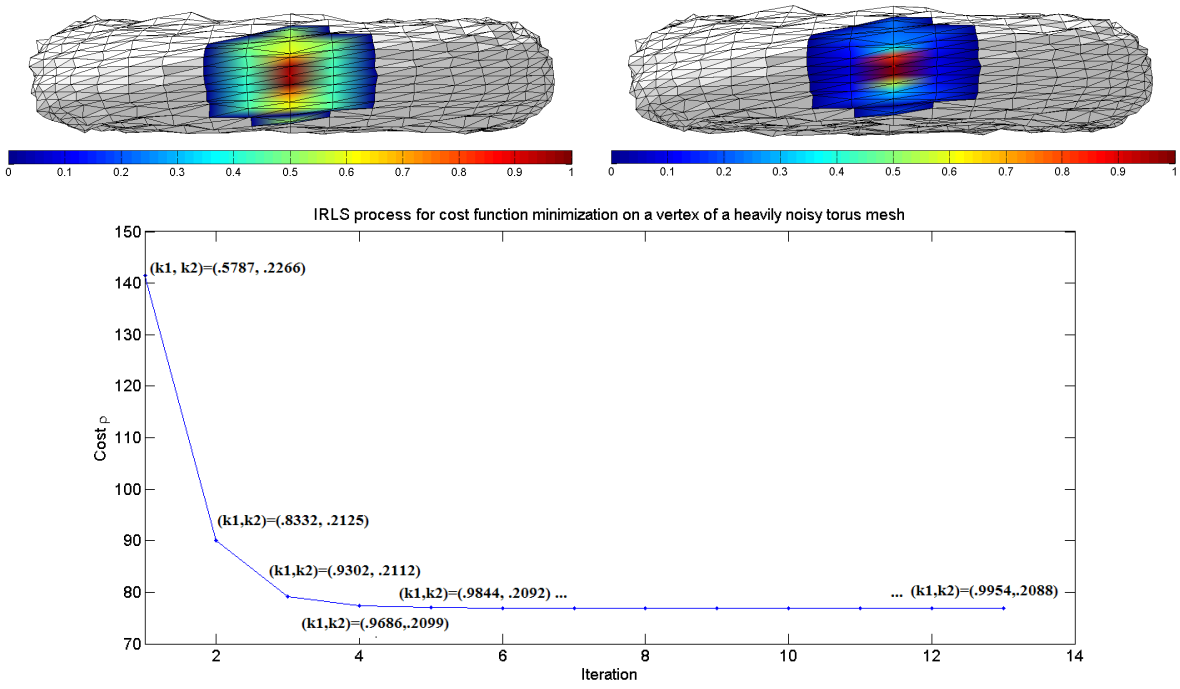

Figure 5.5: **Top:** Same visualizations of the corresponding weights for the vertices of the support region during the first and last (best) iteration of the IRLS process for the curvature estimation of another vertex on the side of a heavily noisy torus (normal noise with variance 20% of the median edge length). **Bottom:** Cost function minimization process during the IRLS algorithm. In this case, convergence to the unique minima of the cost function is achieved after 13 iterations. Note that the estimate of the final iteration is much more accurate than the initial one-ring estimate. The analytical principal curvature values for the same vertex are (k1,k2)=(1,0.2).

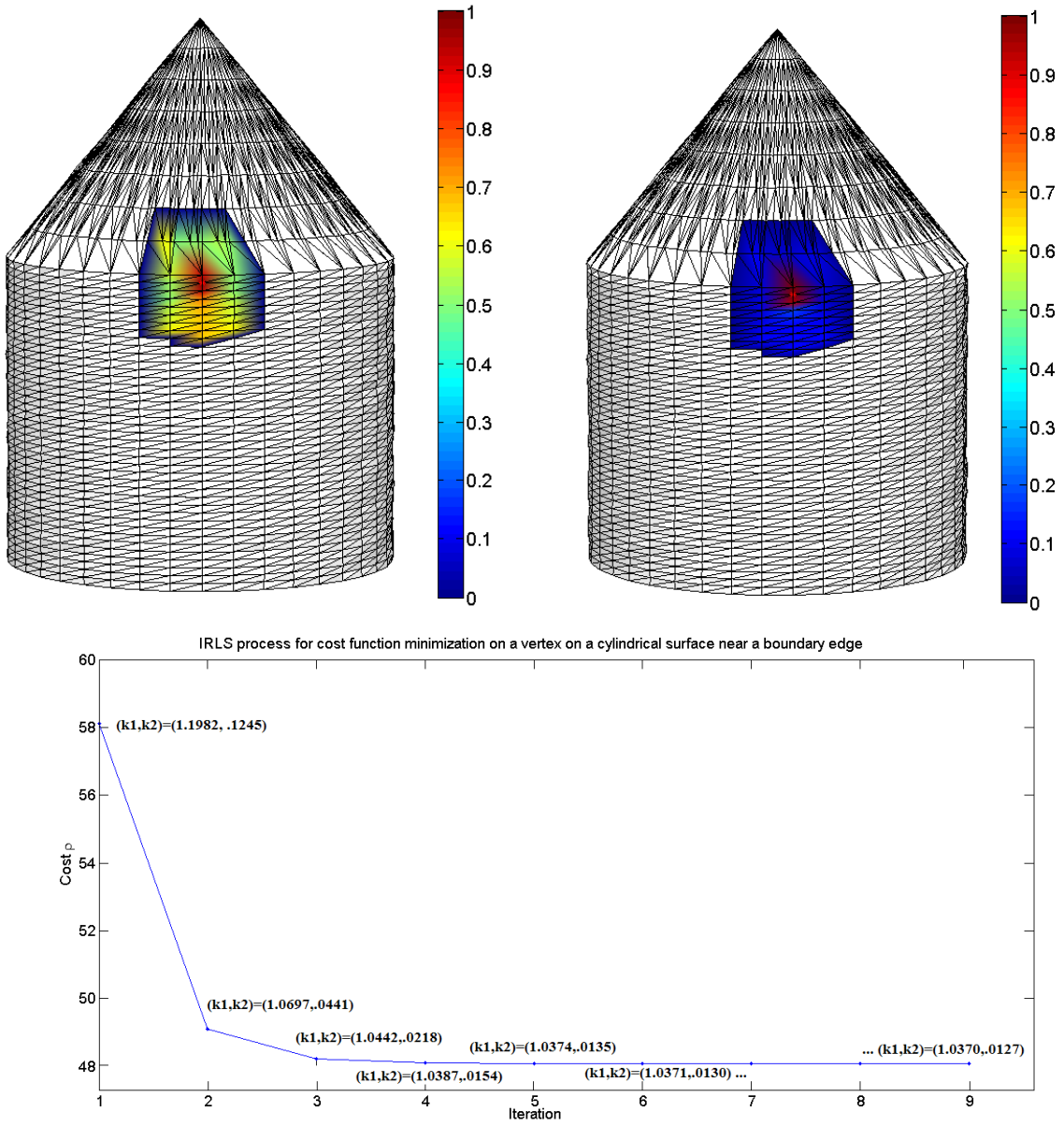

Figure 5.6: **Top:** Same visualization of the support region for the first and last (best) iteration of the IRLS process for the curvature estimation of a vertex near a boundary edge of a surface with light noise (2% normal noise). The surface consists of a cylindrical part and a cone on the top (the boundary is between them). The M-estimation algorithm rejects the structured outliers, caused by the samples past the boundary feature. **Bottom:** Cost function minimization process for the same vertex on the surface. Convergence to the unique minimum of the cost function is achieved after 9 iterations. A 2-ring method, like the cubic patch fitting method, gave (1.8181, 0.9956). The analytical principal curvature values for the same vertex are (k1,k2)=(1,0).

5.3.1 M-estimation algorithm steps

Here, we briefly summarize the necessary steps of our M-estimation algorithm:

for every vertex on the mesh (or the point cloud)

- initialize the curvature parameters from the one-ring estimate (as in chapter 5)
  or from the K nearest points estimate (for point clouds, typically K=6)
- retrieve all points inside the support region and store their normal variations and weights
- initialize σ as 1.4826 multiplied by the median of the residuals, from the initial fitting
- for every iteration (maximum 50) or until convergence
  - for every sample
    - if it does not come from inside the one-ring neighborhood or the K=6 nearest points and its current residual is more than 2σ, set its weight equal to 0
    - otherwise, compute its GM-estimation weight multiplied with its geometric weight
    - and update the cost function with the contribution of this sample
  - if the value of the cost function is lower than the lowest current one, store the current parameter values (as candidate best ones)
  - recompute sigma as maximum(1.4826 multiplied by the median of the current residuals, initial sigma)
  - re-perform the fitting (with the algorithm described in chapter 5), with the currently computed weights of the samples
- return the best curvature parameter values found for this vertex

Chapter 6

Results

In order to exhibit the robustness, the stability and the accuracy of our M-estimation approach, presented in the previous chapter, as well as the satisfactory properties of the fixed neighborhood approach, as shown in chapter 4, we have generated a series of test cases, measuring the Root-Mean-Square (RMS) error of the principal, mean and gaussian curvature in a variety of test surfaces. For these test surfaces, the errors can be measured as we know their curvature tensors analytically. We show also our results in surfaces where the quality can easily and objectively be inspected visually.

In our tests, we have included the following methods:

- The normal cycle method by Cohen-Steiner and Jean-Marie Morvan ([10]
- The discrete operator introduced by Meyer et al ([49])
- The cubic patch fitting method by Goldfeather and Interrante ([19]
- The per triangle curvature tensor estimation by Rusinkiewicz ([59])
- The ”spider” method introduced by Simari *et al.* ([63]) updated with our changes (chap-

ter 3.2) (in some indicative test cases only)

- Our per one-ring neighborhood curvature tensor estimation (chapter 4)
- Our robust M-estimation curvature tensor estimation approach (chapter 5) in both edge sampling and dense sampling mode.

In all the methods, that depend on normal estimation, we used the same Max weighting scheme [47]. All the methods, that were depending on a Voronoi area weighting scheme, used the notion of mixed voronoi areas, as exactly introduced by Meyer *et al.* [49]. The RMS error is estimated as $\sqrt{(\frac{1}{N}\left|\tilde{K}-K\right|^2}$ where $\tilde{K}$ is the experimental measurement and K is its analytical (real) value. N is the number of points where the measurement is taken. In our test cases, for fair comparisons, in the RMS error estimation, we exclude the vertices that are inside a two-ring neighbors of boundary vertices, where we have discontinuity in the curvature field. Note also that Rusinkiewicz [59] and Hamann [24] used the RMS error notion to judge the quality of their results.

In section 6.1, we have generated noisy cases by adding both normal and tangential noise to the vertices of the test meshes. In the next section 6.2, we test the methods in non-noisy, but irregular and non-uniformly tessellated surfaces, based on the test set that was firstly developed and used by Grinspun *et al.* [21]. In the next section, our test cases include irregular coarser and coarser tessellations of surfaces together with some noise added to them at the same time. Finally, in the last section of the chapter, we show visual results of some real-world meshes, based on the trimesh2 library, where the curvature tensor for each vertex on the meshes are visualized appropriately, based on Rusinkiewicz's visualization method [59]. The underlying surfaces are C^2 continuous, thus their associated curvature field should change smoothly on the surface. In all test cases, for fair comparison, the surfaces are not preprocessed or smoothed for any method.

Our six characteristic test surfaces, used in sections 6.1-6.3, are the following (the presented

principal, mean, gaussian curvature equations can be found in any classical differential books):

a) Sphere: The sphere of radius R is a surface whose all its points are umbilic, meaning that they have the same curvatures values in all directions. All the curvature values are equal to $1/R$. Of course, the mean curvature is the same while the Gaussian curvature is $1/R^2$.

b) Torus: The torus has parametric equations $x = (c + a\cos v)\cos u$, $y = (c + a\cos v)\sin u$, $z = a\sin v$ (where $u, v \in [0, 2\pi)$) where c is the radius from the center of the hole to the center of the torus tube and the radius of the tube is a. Its mean curvature is $H = -\frac{c+2a\cos v}{2a(c+a\cos v)}$ and gaussian curvature $K = \frac{\cos v}{a(c+a\cos v)}$. Its principal curvatures are $1/a$ and $-\frac{\cos v}{c+a\cos v}$

c) Cylinder: A cylinder has parametric equation $x = r\cos v, y = r\sin v, z = u$ where $v \in [0, 2\pi)$. It is a developable surface (with gaussian curvature 0), mean curvature $1/2r$, the maximum principal curvatures equal to $1/r$ and the other one equal to 0.

d) Monkey Saddle: The monkey saddle is described by the equation $z = x \cdot (x^2 - 3y^2)$ or perimetrically by $x(u,v) = u$, $y(u,v) = v$, $z(u,v) = u^3 - 3uv^2$, $u, v \in \Re$. Its gaussian curvature is $K = \frac{-36\cdot(u^2+v^2)}{[1+9(u^2+v^2)^2]^2}$ and mean curvature is $H = \frac{27u(-u^4+2u^2v^2+3v^4)}{[1+9(u^2+v^2)^2]^{3/2}}$ (note that the principal curvatures can be found as $k_1, k_2 = H \pm \sqrt{H^2 - K}$).

e) Elliptic Cone: The elliptic cone with height h, semimajor axis a, and semiminor axis b has parametric equations $x(u,v) = a\frac{h-u}{h}\cos v$, $y(u,v) = b\frac{h-u}{h}\sin v$, $z(u,v) = u$ where $u \in [0,h), v \in [0, 2\pi]$. An interesting property of the elliptic cone is that it has zero Gaussian curvature everywhere (developable surface). The mean curvature is $\frac{\sqrt{2}abh^2(h^2+a^2\cos^2 v+b^2\sin^2 v)}{(h-u)[2a^2b^2+(a^2+b^2)h^2+(b^2-a^2)h^2\cos(2v)]^{3/2}}$.

f) Helicoid: Finally, the helicoid is a minimal surface, meaning that it has zero mean curvature everywhere, and is described by the following equations: $x = u\cos v$, $y = u\sin v$, $z = cv$ where $u, v \in \Re$. The gaussian curvature is $-\frac{c^2}{(c^2+u^2)^2}$.

Note, finally, that in the case of cone, we exclude from the RMS error measurement the vertices around a 2-ring neighborhood of the apex vertex, as the mean curvature on the apex if

theoretically infinite.

6.1 Noisy test surfaces

The first test suite consists of our six test surfaces in noisy versions. We add both tangential and normal Gaussian noise with increasing variance from 0% until 5% (with .25% step) to all the vertices of the surfaces and we measure the RMS errors for all methods for all the surfaces. Figure 6.1. We repeated our tests 100 times and took the average values of RMS errors in all cases.

We present the plots of RMS errors for principal curvatures, mean and gaussian curvature respect to increasing variance of random Gaussian tangential and random noise for all the surfaces and methods in figures 6.2 - 6.8.

Notice that the Max's weighting scheme [47] to compute normals, by averaging appropriately the face normals, gives exact normals on the sphere, thus the method that are using estimated normals in their computation, like our methods, as well as Goldfeather's and Interrante's [19] and Rusinkiewicz's [59] methods are favored in the sphere and cylinder (but not in the cases of significant noise).

The results turned out to be very positive for our M-estimation method. In all surfaces, for all cases of noise, our M-estimation method (either with dense sampling or edge sampling) exhibit the best accuracy and stability with significant difference from the other approaches. The M-estimation with dense sampling, independently from the mesh connectivity gives more accurate and robust results, than using only the existing edges of the mesh (edge sampling), supporting the fact that sampling the normal variations, beyond the existing edge connectivity on the mesh yields a more accurate solution.

On the contrary, the one-ring methods became very sensitive to noise in all surfaces, while the two-ring cubic patch fitting method showed significant instability in the presence of moderate noise in the cases of the sphere and the cone. Even in the monkey saddle, which is a cubic surface, our M-estimation approach exhibited better accuracy than this method. Rusinkiewicz's method had general good behavior in these test cases. Our fixed region method also had satisfactory stability, returning results that were very close to Rusinkiewicz's method, with slightly better accuracy in the cases of torus, monkey saddle, cylinder and helicoid.

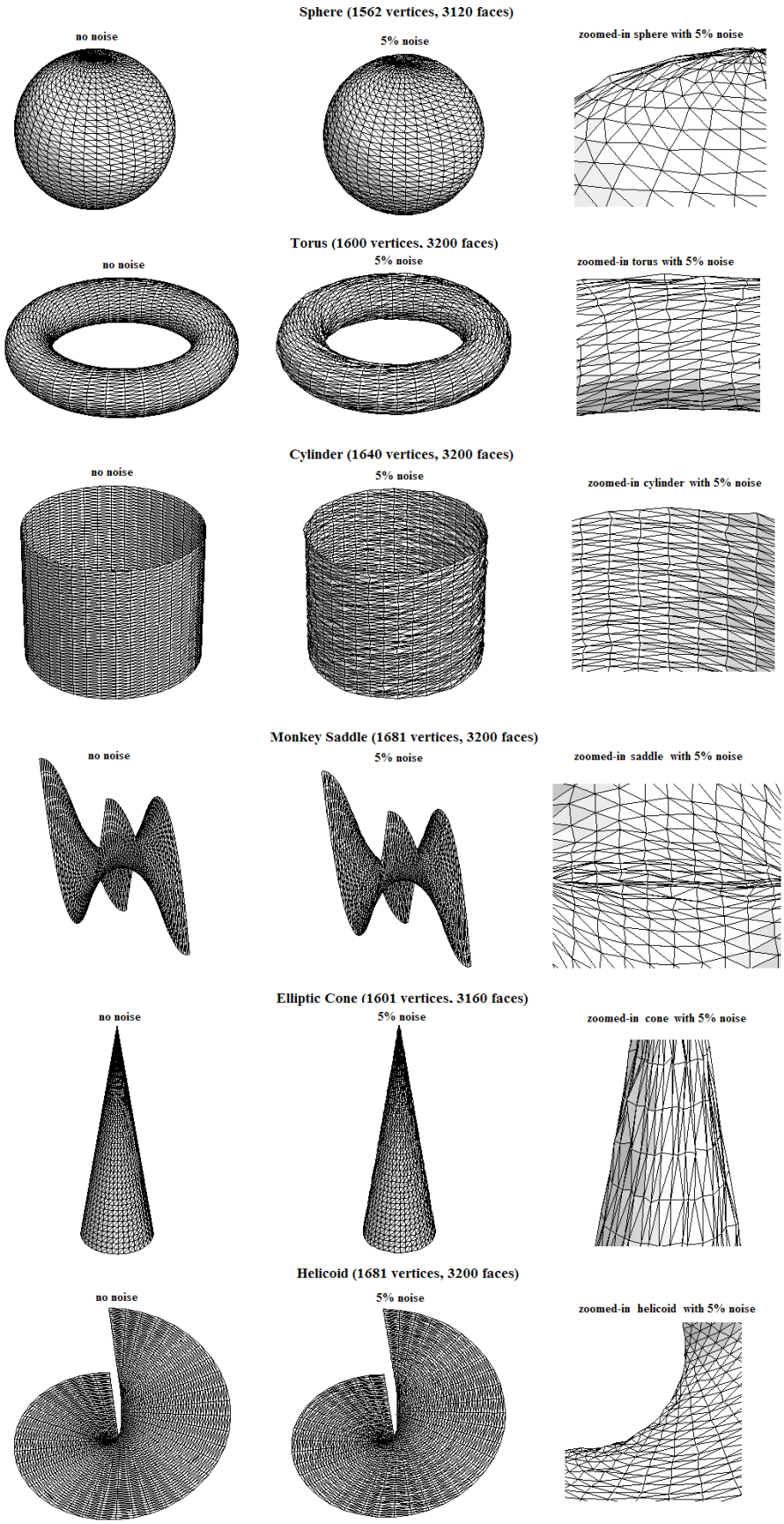

Figure 6.1: Mesh examples from our first test suite of progressively noisy surfaces

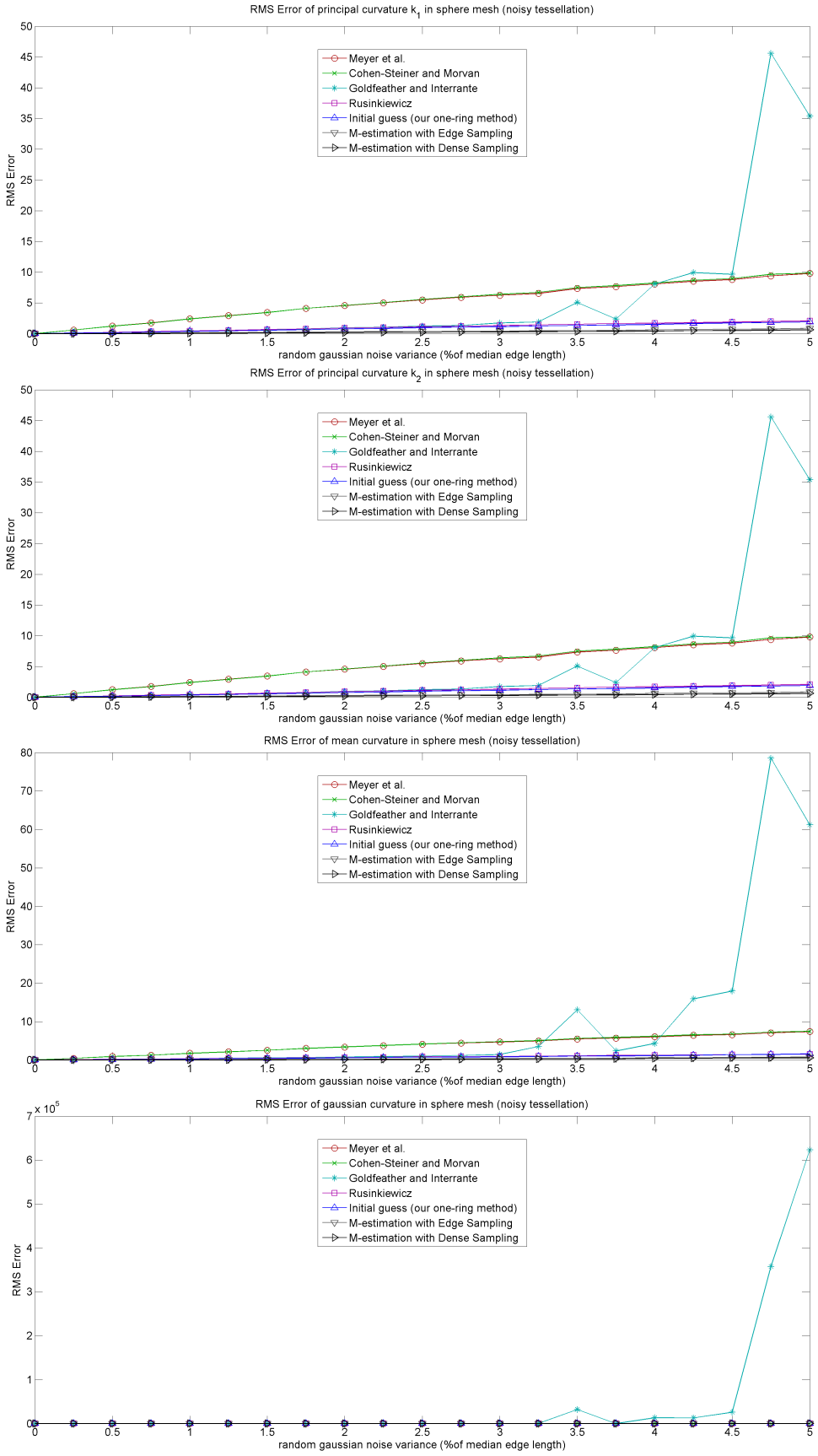

Figure 6.2: Plots of RMS Error for the noisy versions of sphere

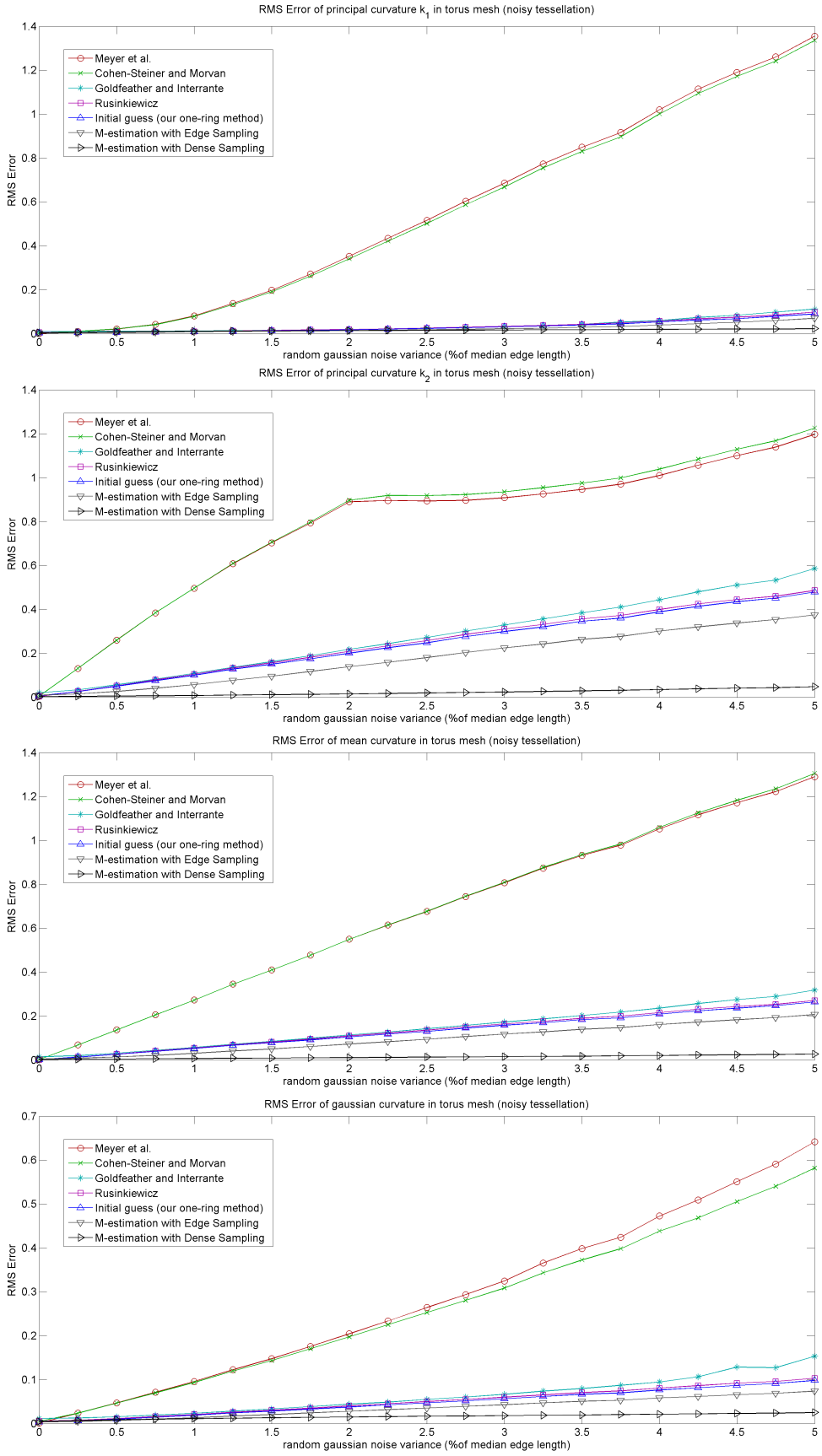

Figure 6.3: Plots of RMS Error for the noisy versions of torus

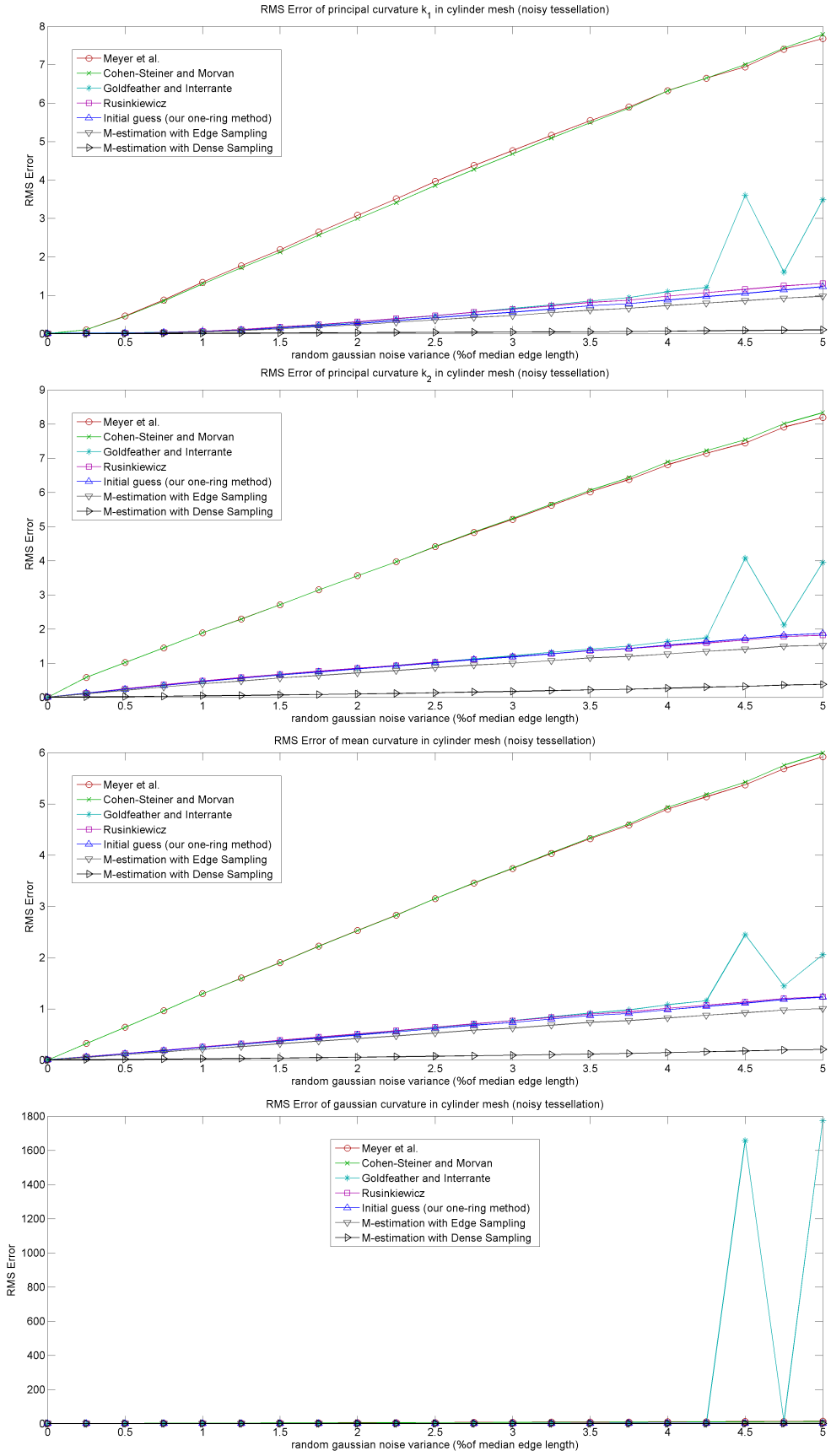

Figure 6.4: Plots of RMS Error for the noisy versions of cylinder

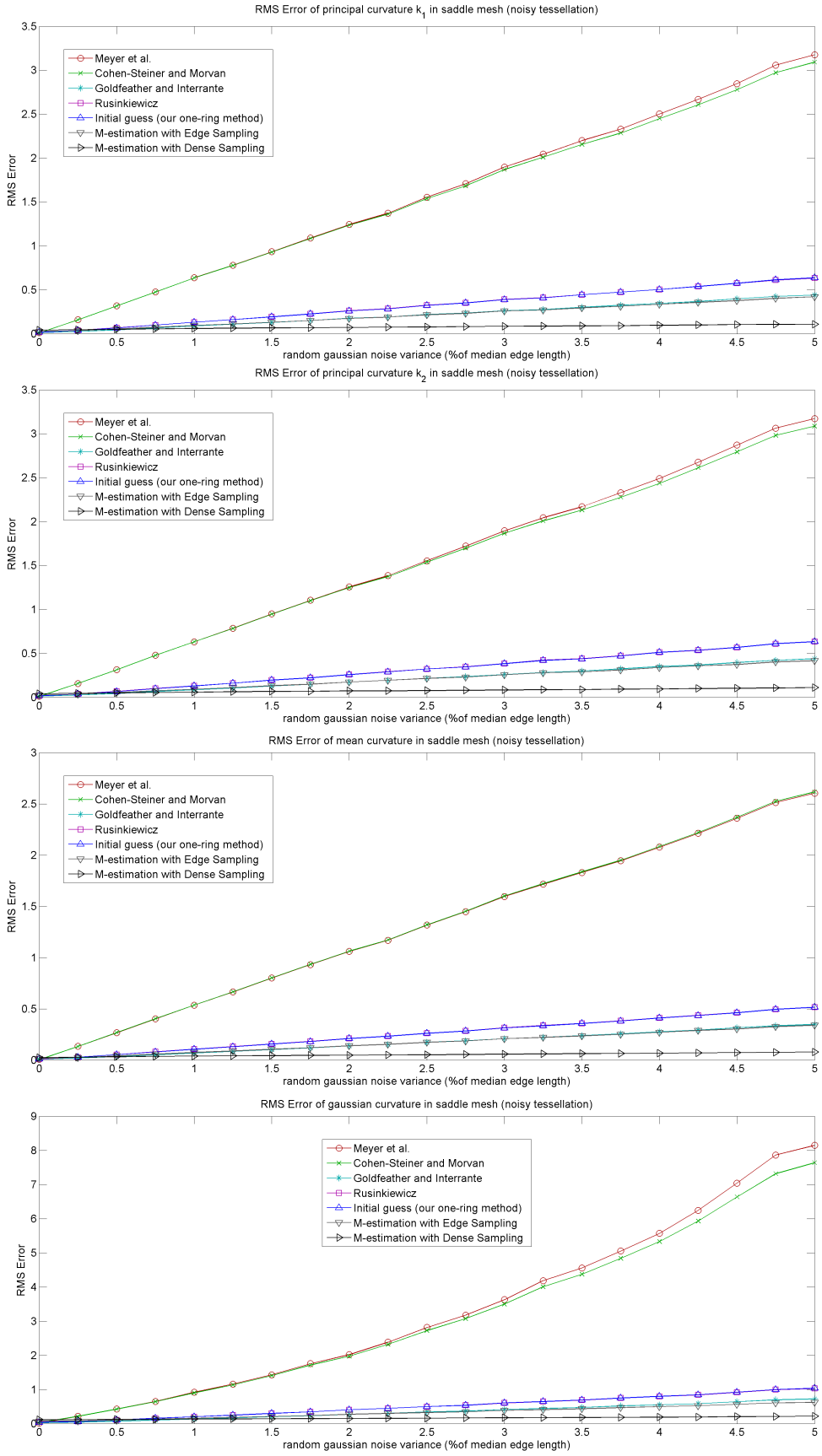

Figure 6.5: Plots of RMS Error for the noisy versions of monkey saddle

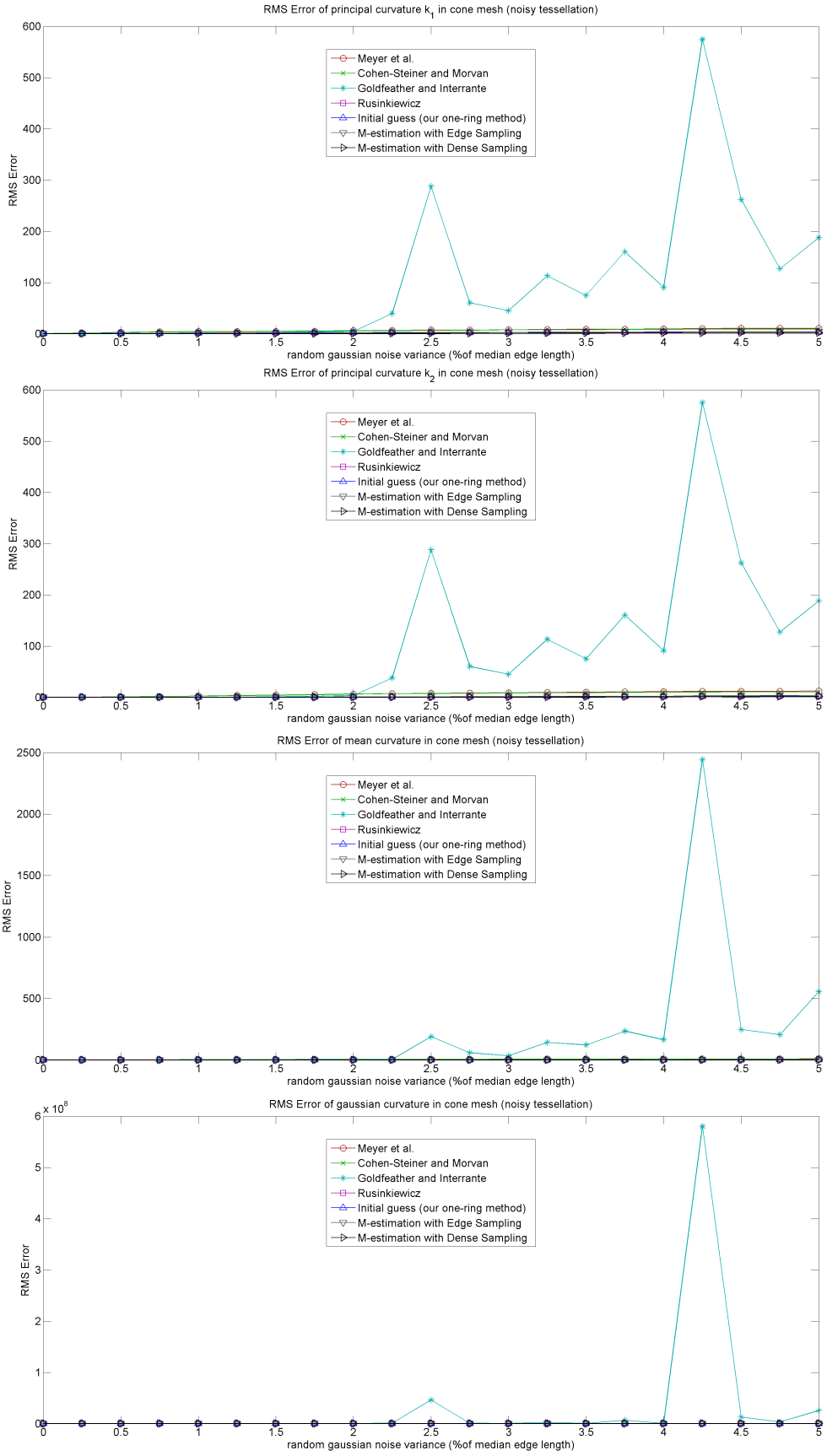

Figure 6.6: Plots of RMS Error for the noisy versions of elliptic cone

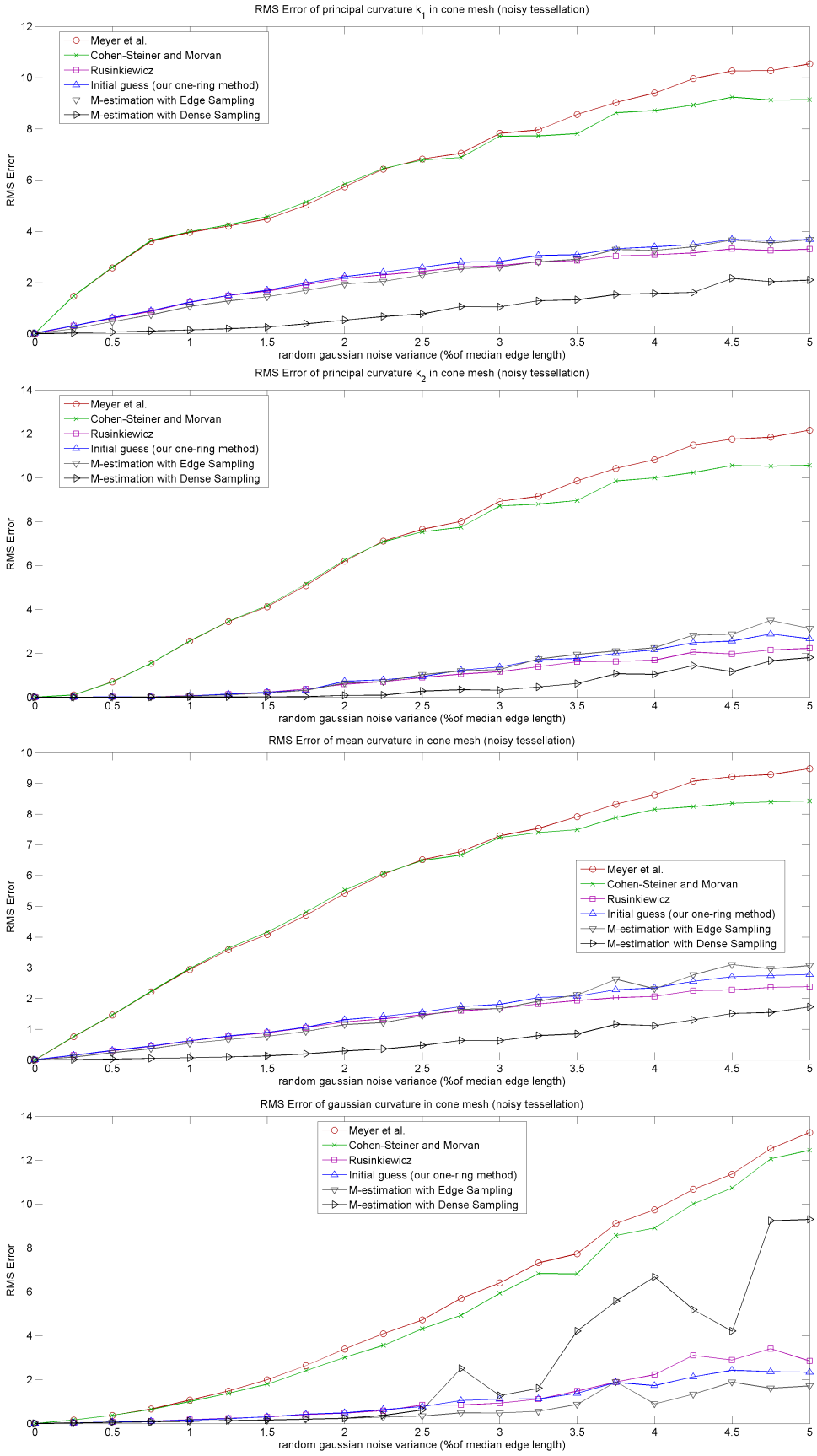

Figure 6.7: Same plots for cone without cubic patch fitting method

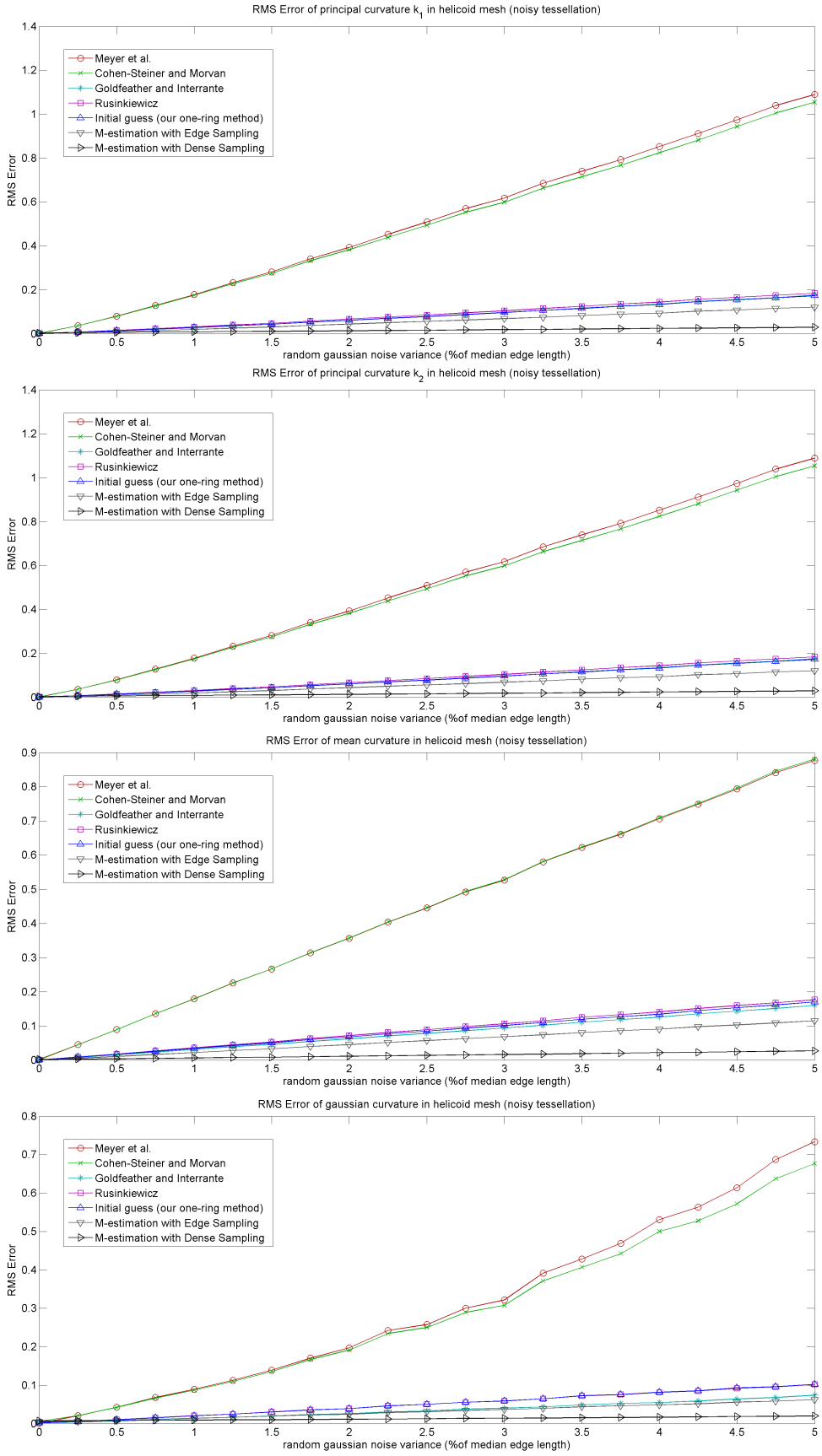

Figure 6.8: Plots of RMS Error for the noisy versions of helicoid

6.2 Irregular, non-uniformly sampled surfaces

In our second test suite, the methods are tested for irregular and non-uniformly sampled tessellations without any noise. We include four different possible tessellations of surfaces, taken from Grinspun *et al.*'s [21] test suite (see 6.9). The first "bad" tessellation includes non-uniform sampling of surfaces that also switches from regular to irregular connectivity, with different valences on the vertices. The second one has badly shaped triangles with aspect ratio 3. The third one introduces polar distortion to the sampling of the surface. Finally, the last one is a mixture of the previous bad cases of tessellations, with non-uniform sampling, highly mesh irregularity with bad triangle aspect rations, varying valences on the vertices and polar distortion. We use these mesh tessellation types in various increasing resolutions, starting from 200 faces and reaching 25000 faces. Normally, a good curvature estimation must exhibit convergence as the resolution of the mesh increases.

The reason that we present this test suite is that we would like to test our methods in cases when noise does not exist and all the vertices are exactly but irregularly lying on the underlying surface. In such cases, it could be assumed that the one-ring methods should provide best accuracy as they are working on the smallest possible neighborhood of points, thus they do not cause any smoothing of the curvature field. However, our tests did not confirm that. On the contrary, our M-estimation method, in the majority of the cases, exhibited again the best stability and accuracy, as when noise does not exist, it automatically converges to a very small neighborhood around each vertex (giving high values of weights there), although it starts from a very extended neighborhood in the support region.

The one-ring methods exhibited good convergence in the cases of the tessellation with polar distortion and triangles with bad aspect ratio. Meyer's *et al.* method showed in general slightly better accuracy than the normal cycle method and gave the best accuracy usually on the meshes of monkey saddle and cone. Goldfeather's and Interrante's cubic patch fitting method exhibited

good stability in all cases, but it had large errors usually for the coarsest versions of the meshes, especially on the sphere, cylinder and cone.

In the sphere and cylinder, our one-ring and M-estimation method as well as Rusinkiewicz's method had an error of almost zero, as these methods take advantage of the fact that the normals are exact. However, this does not hold for the cubic patch fitting method that also uses the estimated normals.

Except to the cone, our M-estimation with dense sampling had the best accuracy in the curvature estimation of meshes with non-uniform (half 3-12) and mixed tessellation, which is the hardest case. In the other two types of tessellation, it had medium accuracy, but always exhibited very good convergence behavior. The edge sampling mode of the algorithm was less accurate in general and also had some instabilities on the cone meshes. Our fixed region method had satisfactory behavior and exhibited better accuracy than Rusinkiewicz's method on the meshes of torus and helicoid but usually not on the cone and monkey saddle. In most cases, M-estimation improved the initial guess of the fixed region method; however it worsened the accuracy of the first estimate a bit, mostly on some mesh types with bad aspect ratios and polar distortion. In general, the tests again showed positive results for the M-estimation in these test cases without noise.

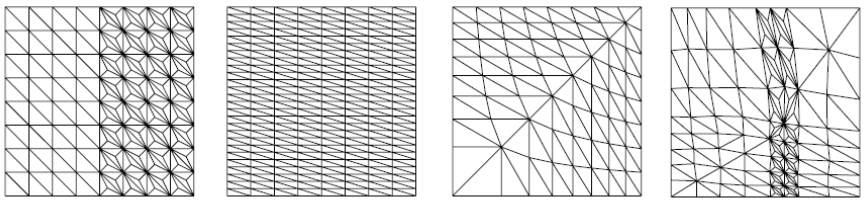

Figure 6.9: Mesh tessellation types for the second test suite (from [21])

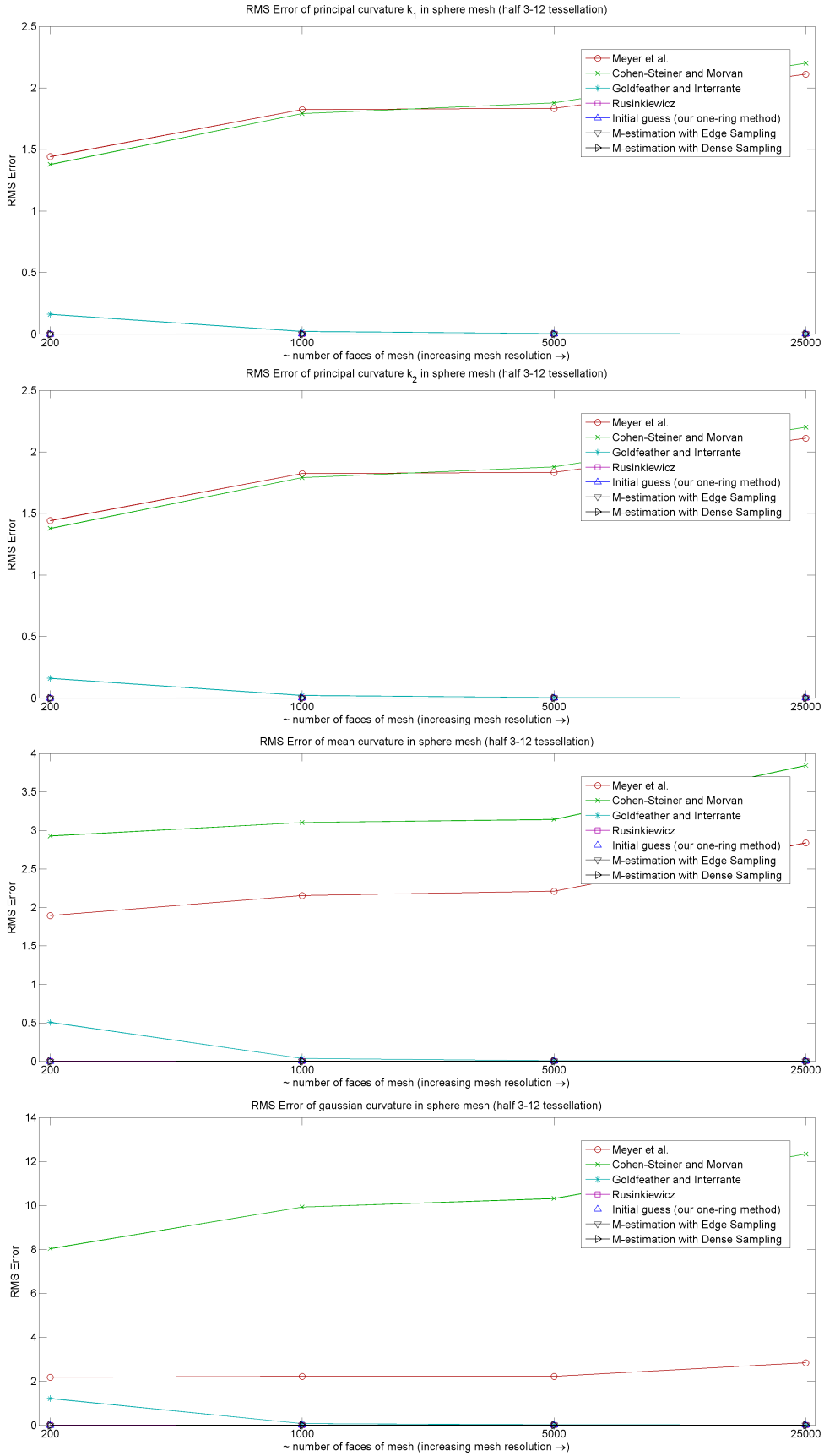

Figure 6.10: Plots of RMS Error for the half 3-12 tessellations of sphere

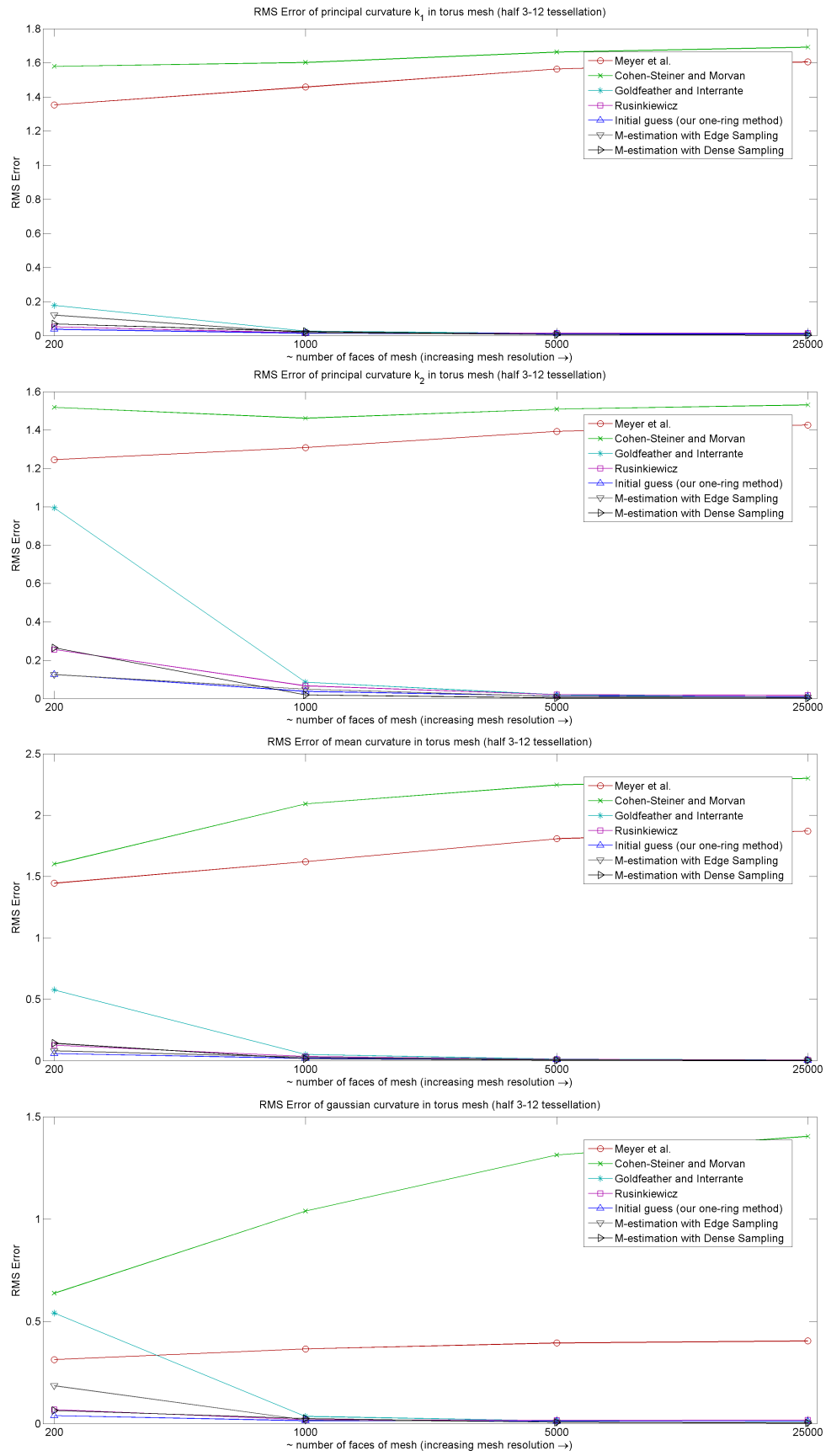

Figure 6.11: Plots of RMS Error for the half 3-12 tessellations of torus

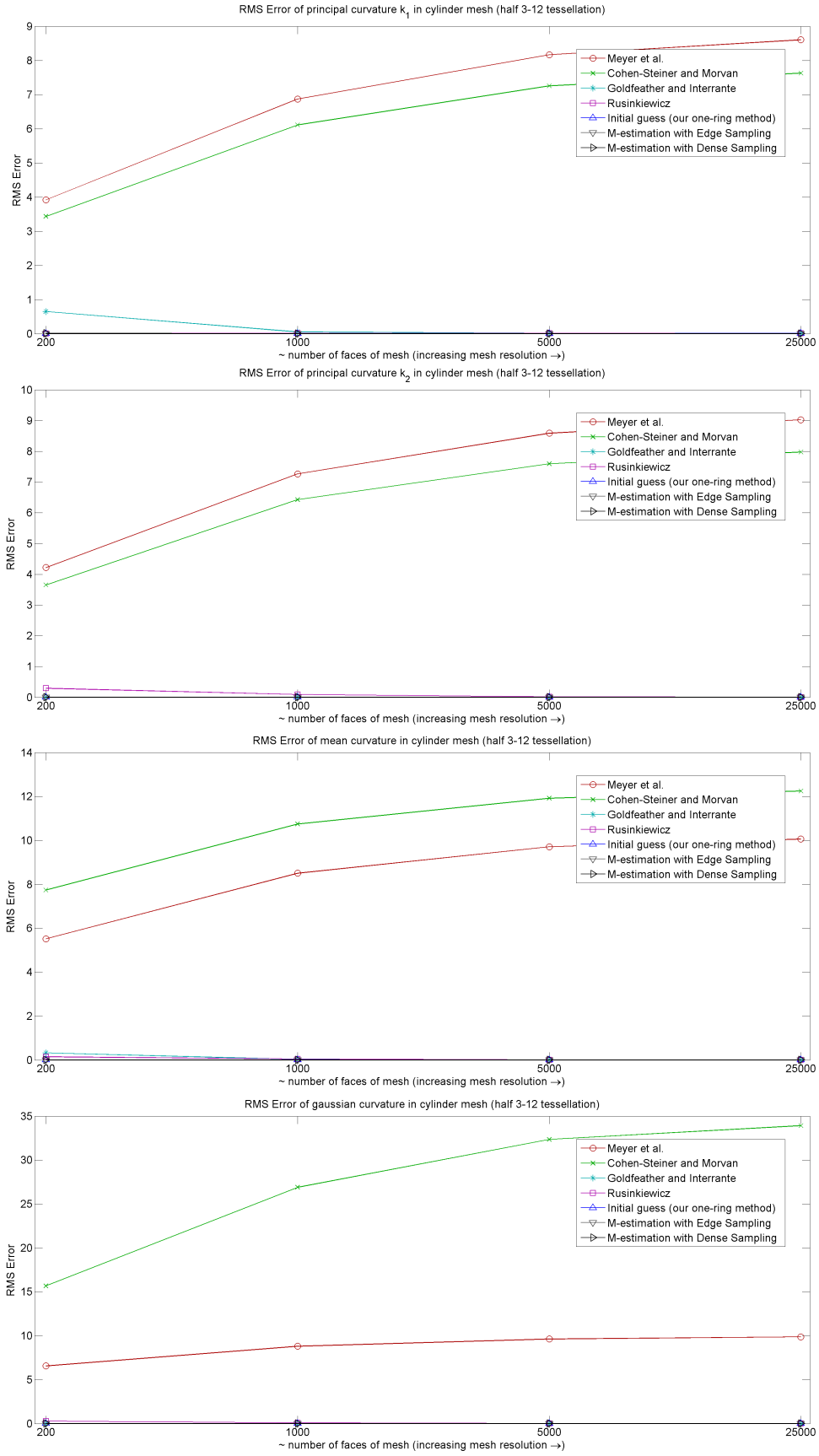

Figure 6.12: Plots of RMS Error for the half 3-12 tessellations of cylinder

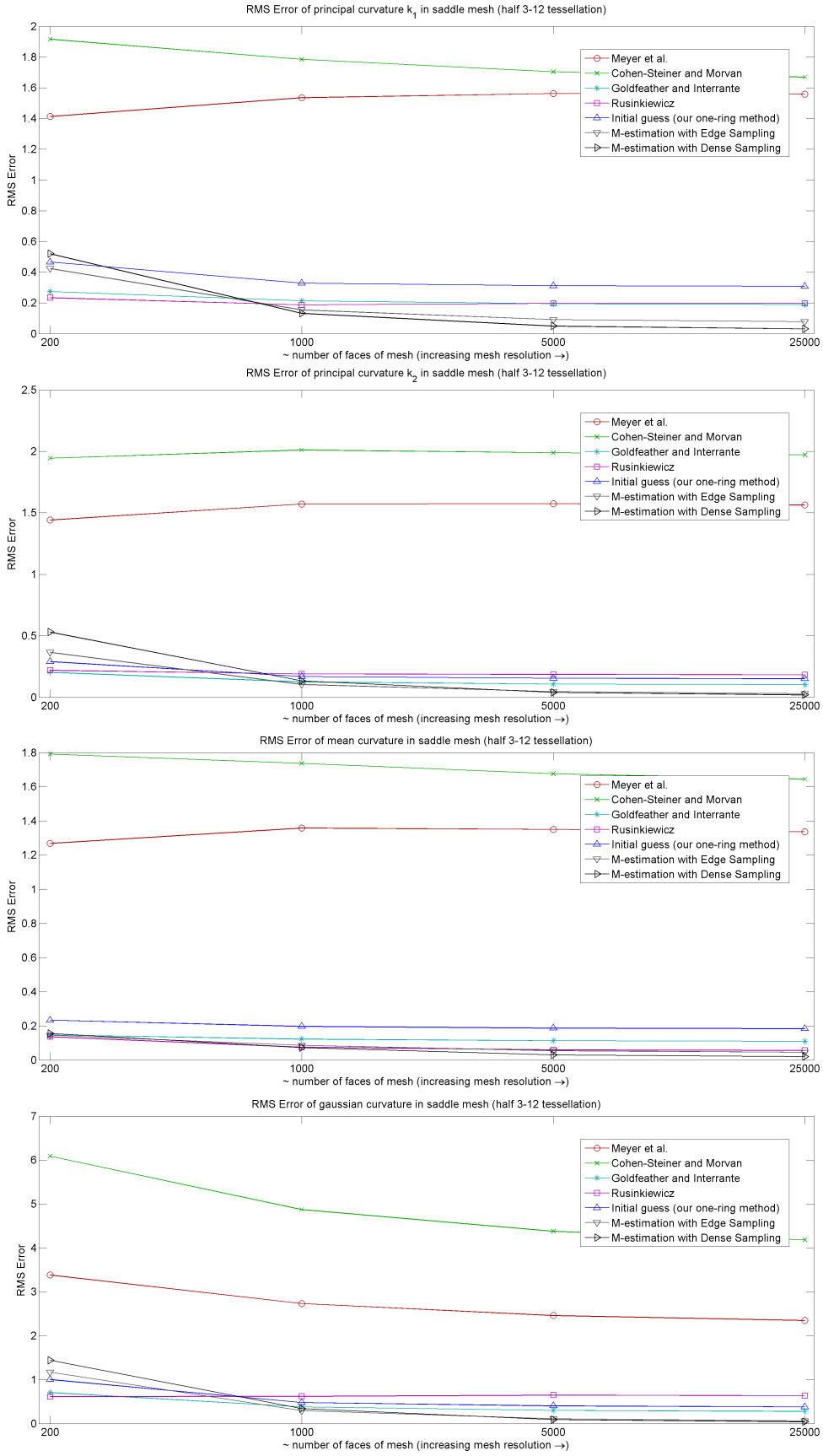

Figure 6.13: Plots of RMS Error for the half 3-12 tessellations of saddle

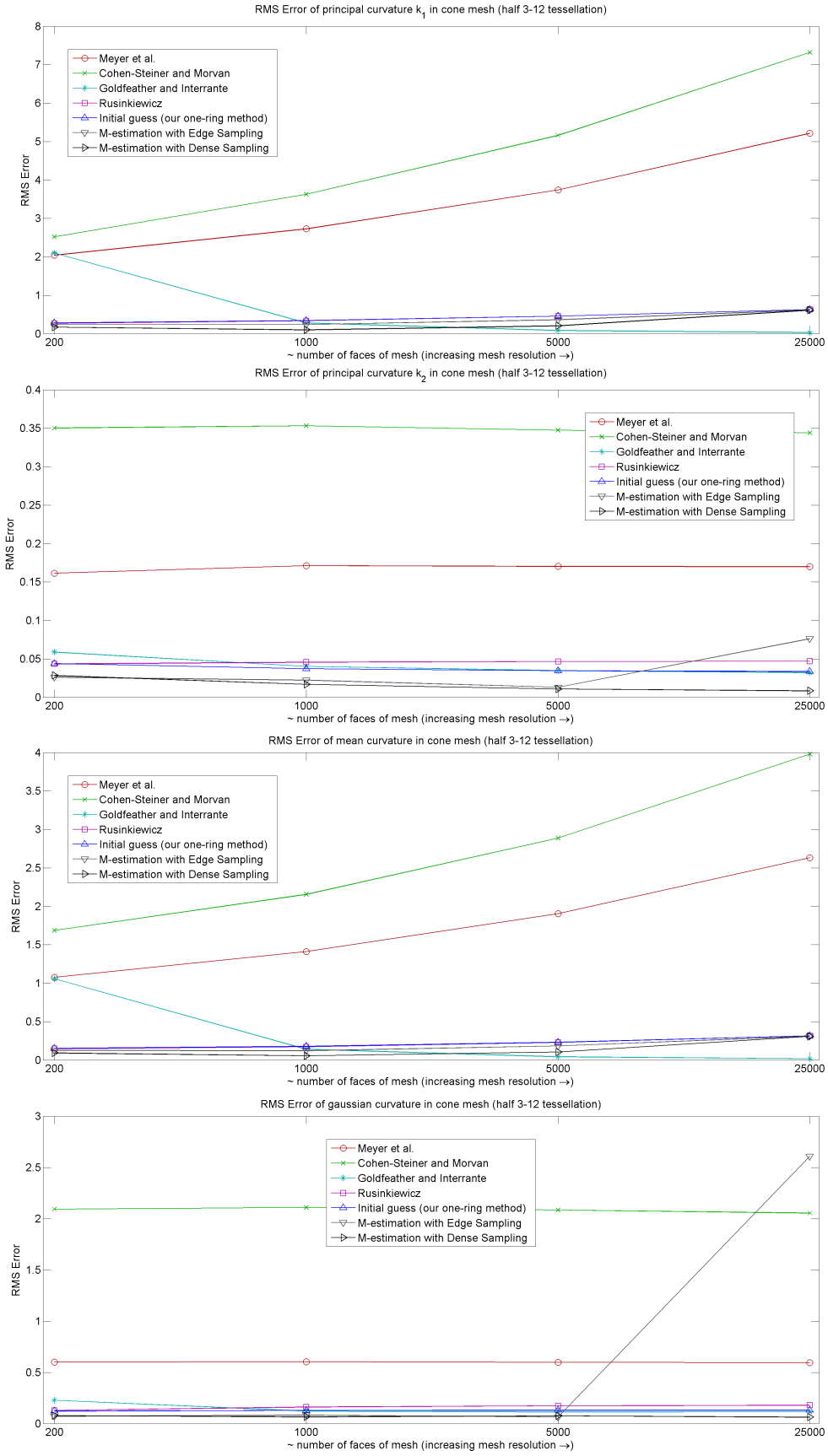

Figure 6.14: Plots of RMS Error for the half 3-12 tessellations of cone

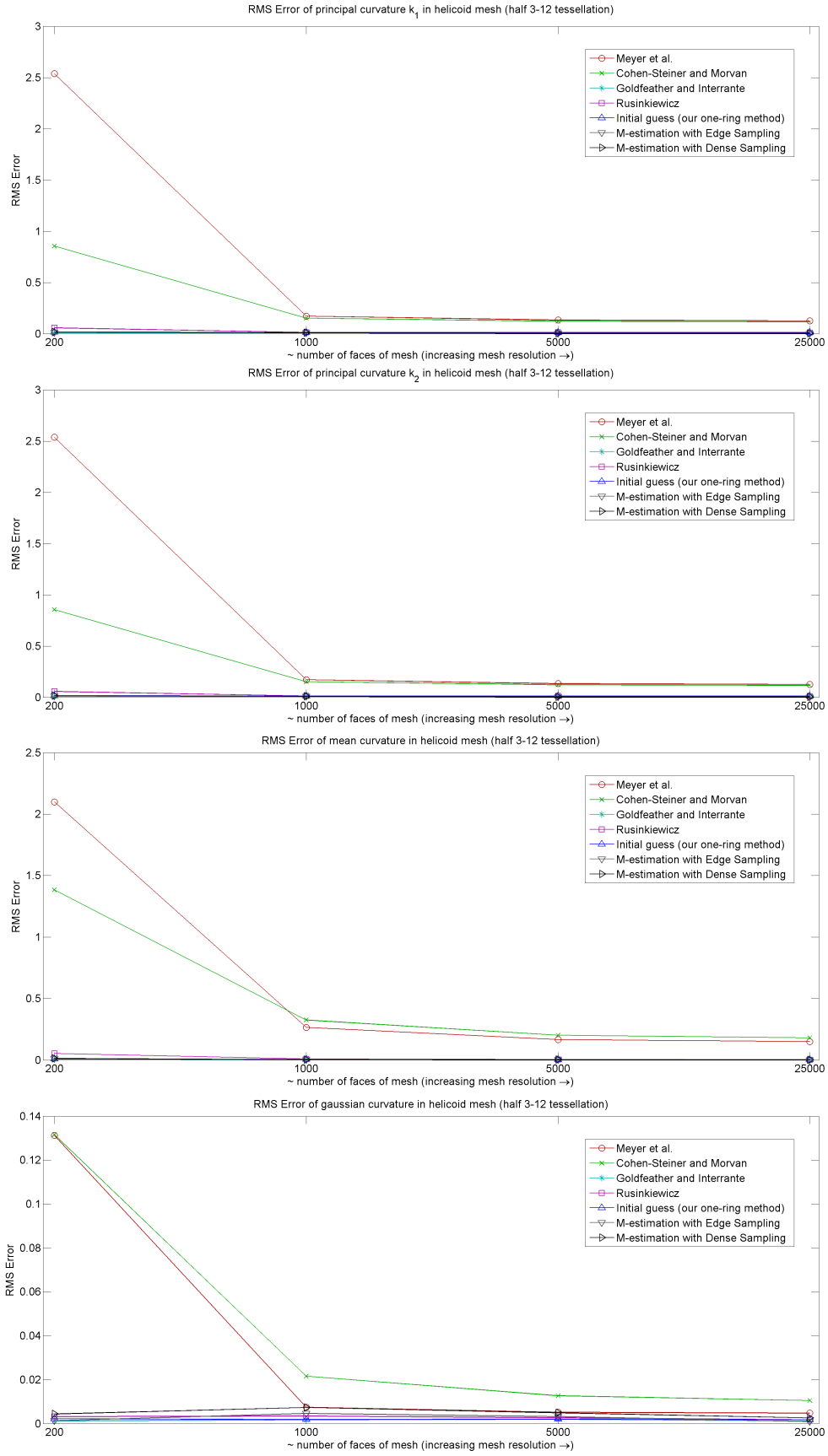

Figure 6.15: Plots of RMS Error for the half 3-12 tessellations of helicoid

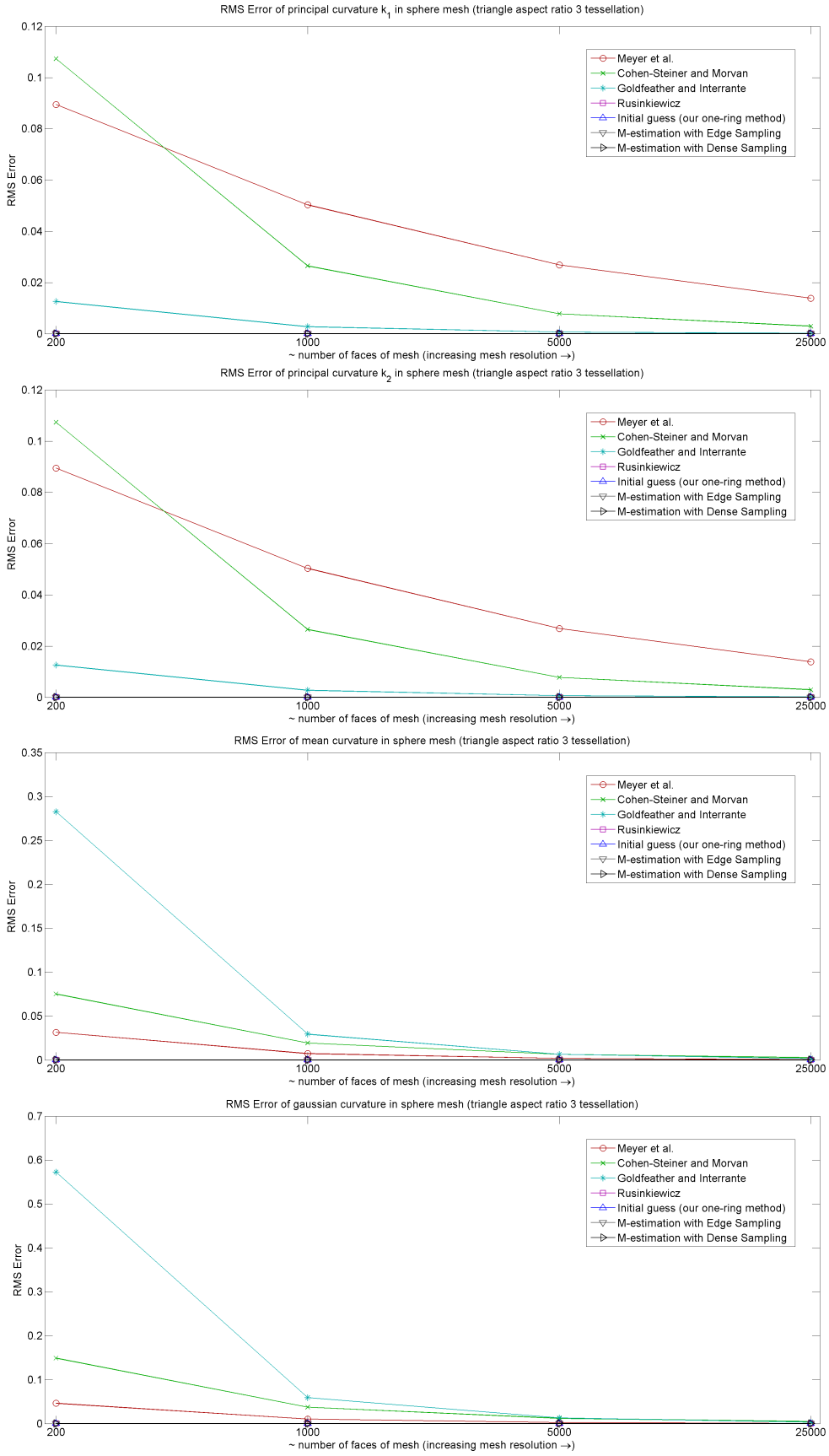

Figure 6.16: Plots of RMS Error for the aspect ratio 3 tessellations of sphere

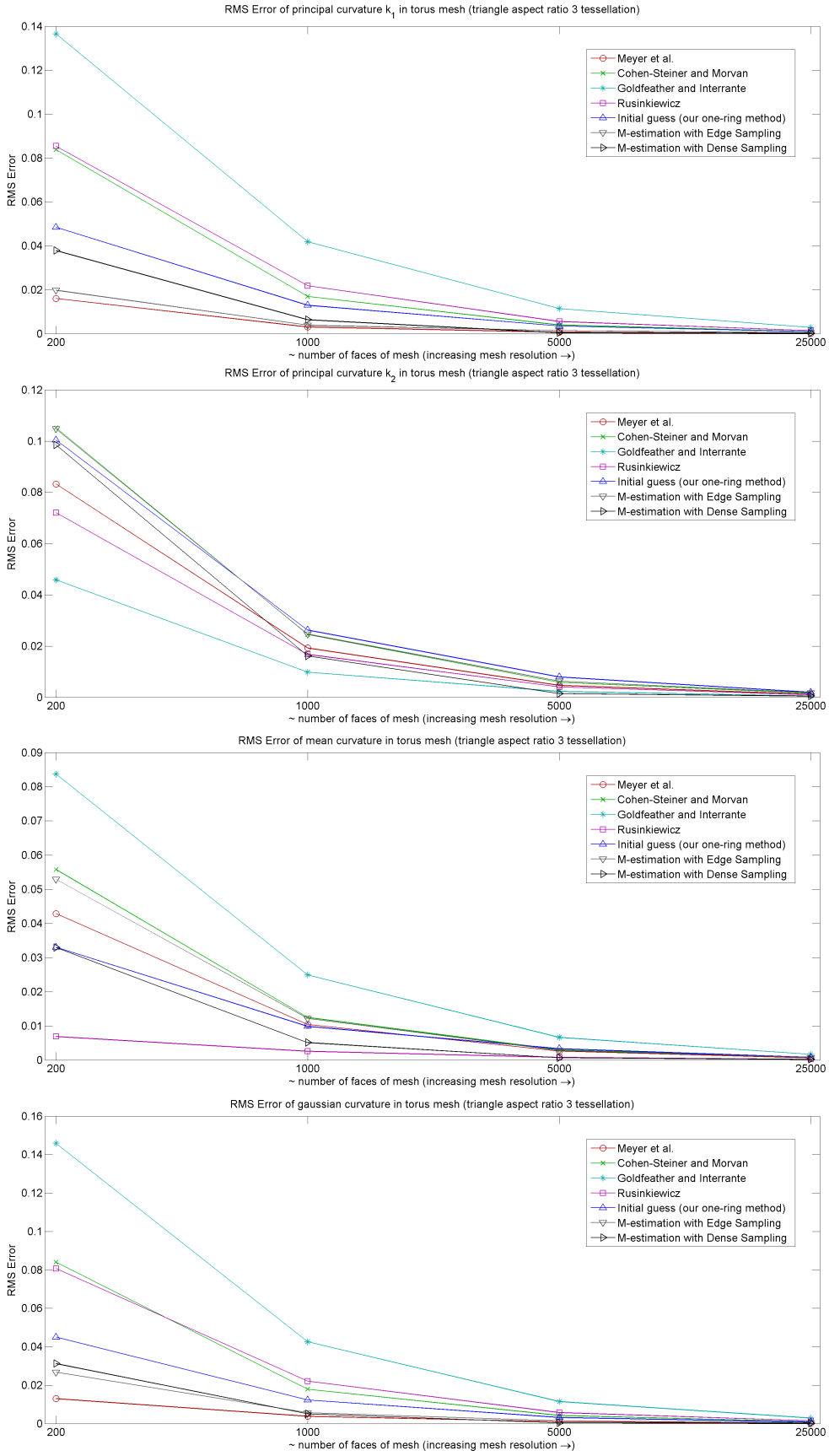

Figure 6.17: Plots of RMS Error for the aspect ratio 3 tessellations of torus

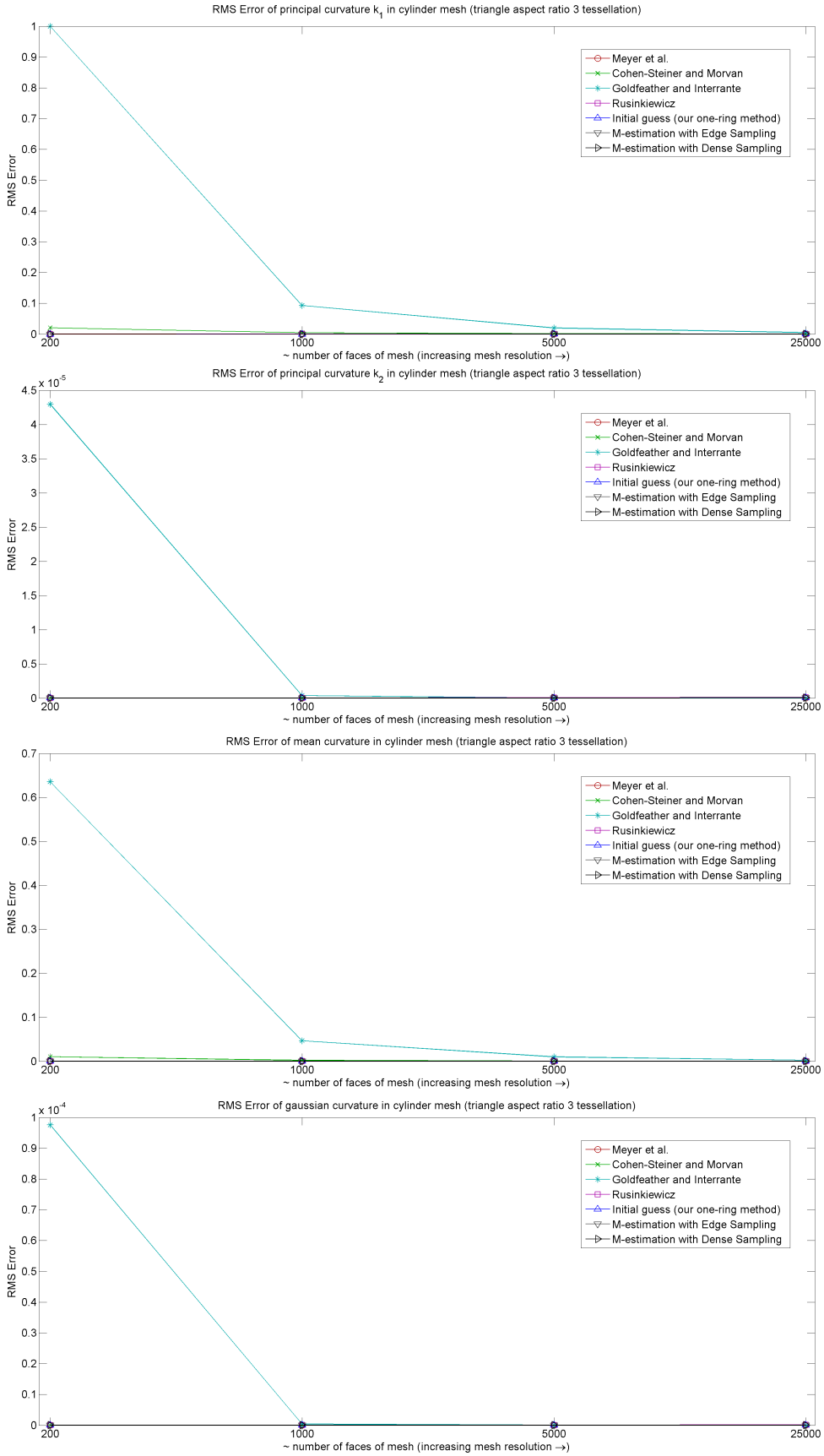

Figure 6.18: Plots of RMS Error for the aspect ratio 3 tessellations of cylinder

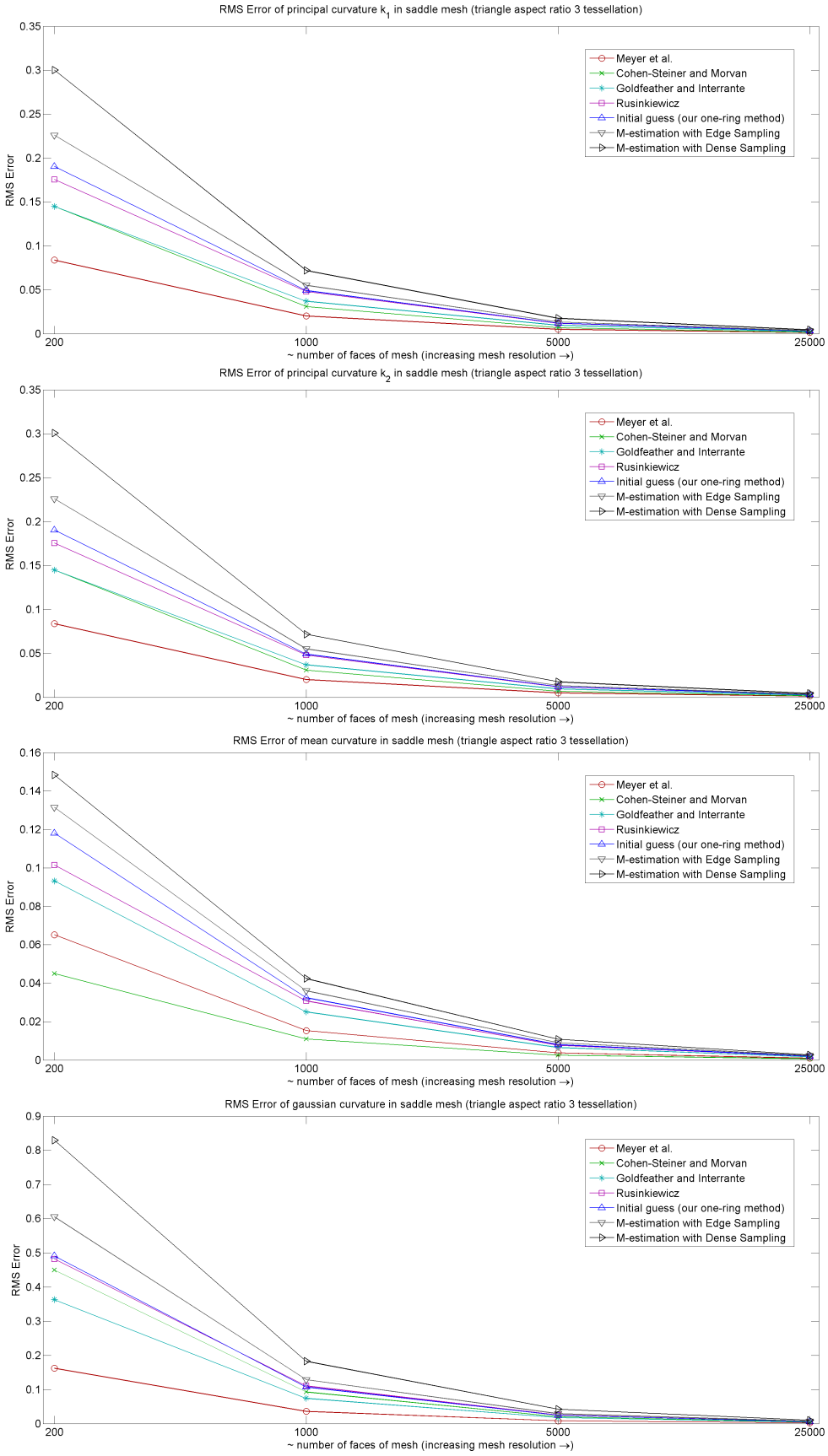

Figure 6.19: Plots of RMS Error for the aspect ratio 3 tessellations of saddle

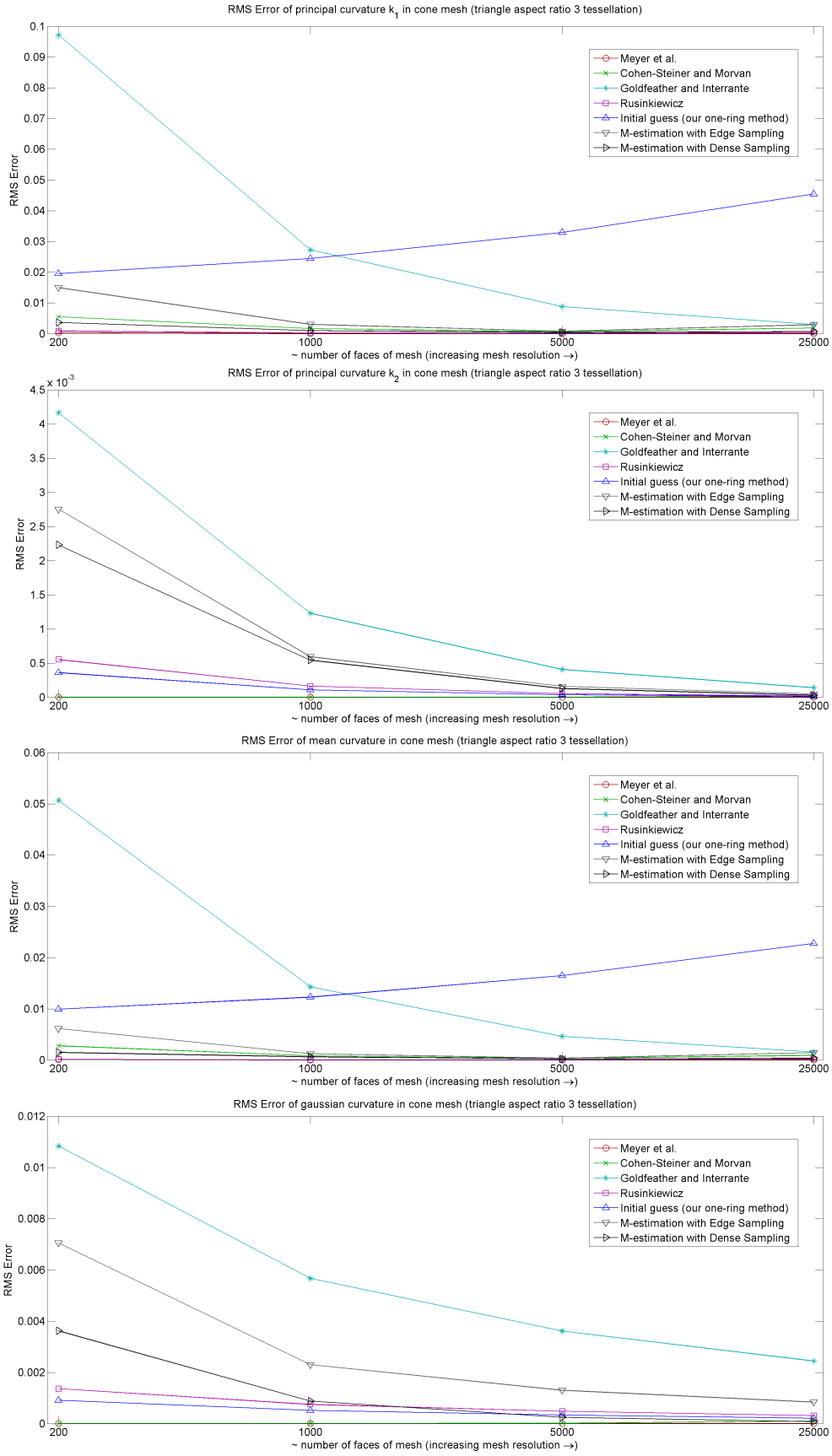

Figure 6.20: Plots of RMS Error for the aspect ratio 3 tessellations of cone

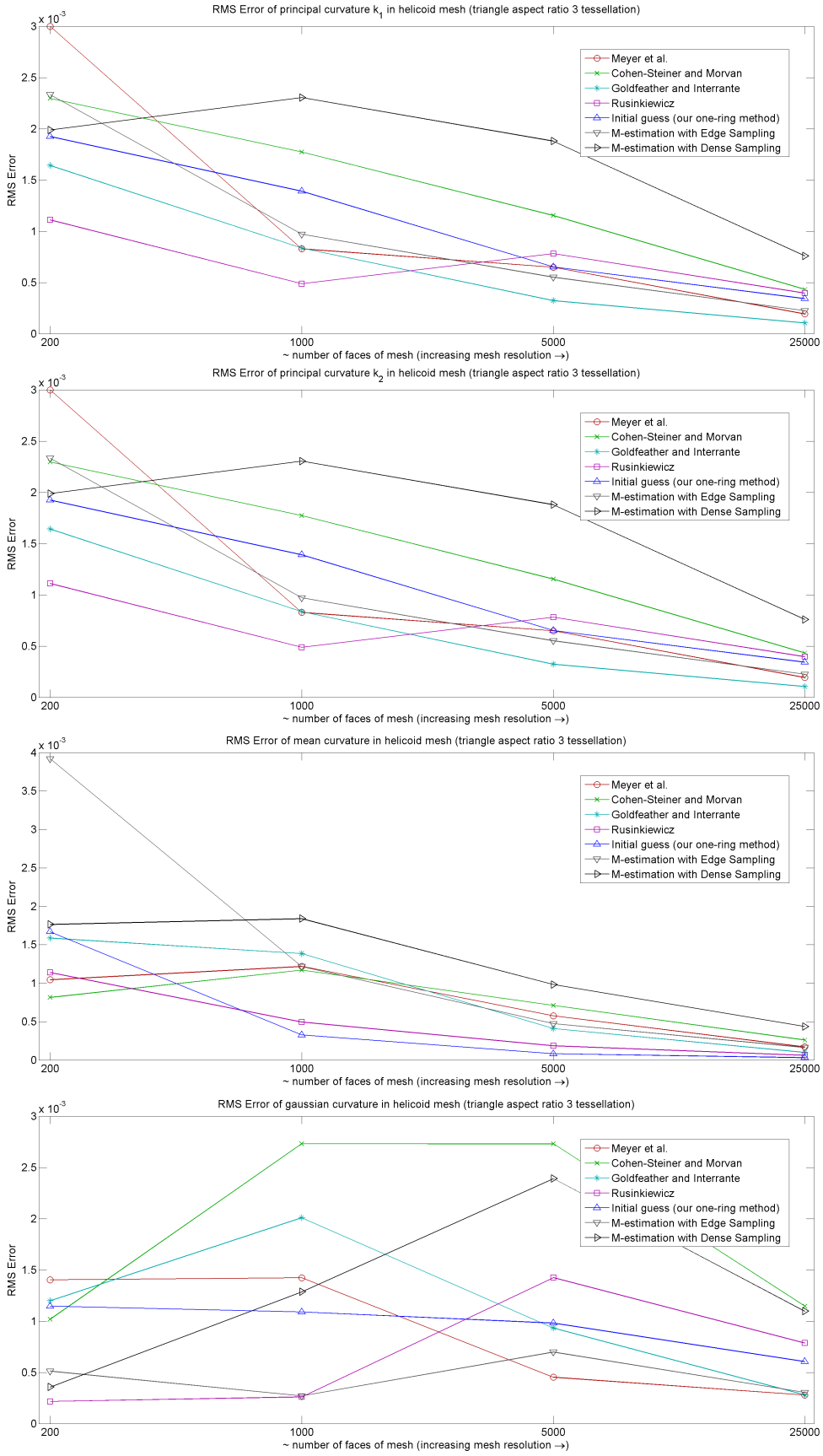

Figure 6.21: Plots of RMS Error for the aspect ratio 3 tessellations of helicoid

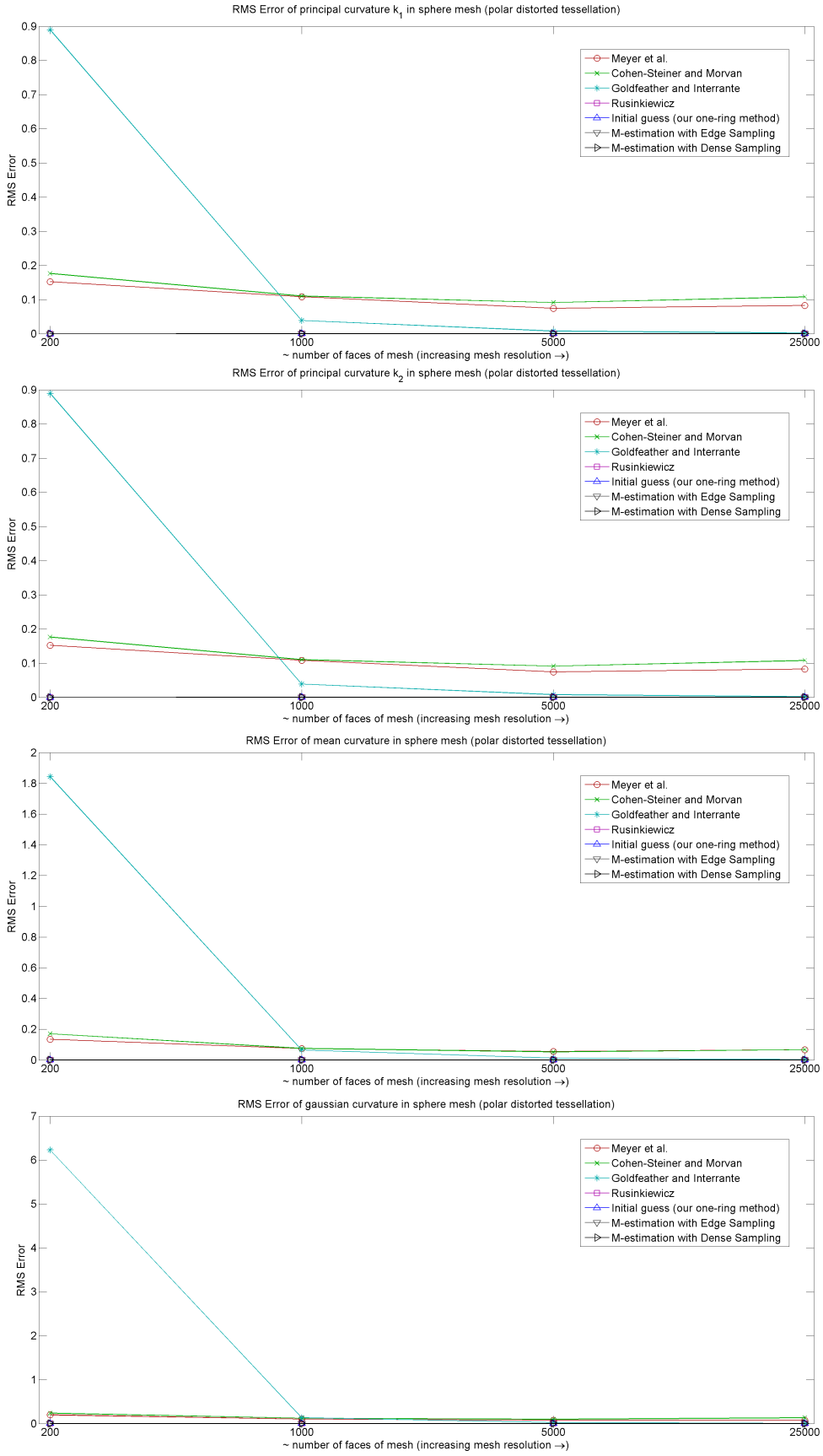

Figure 6.22: Plots of RMS Error for the polar distorted tessellations of sphere

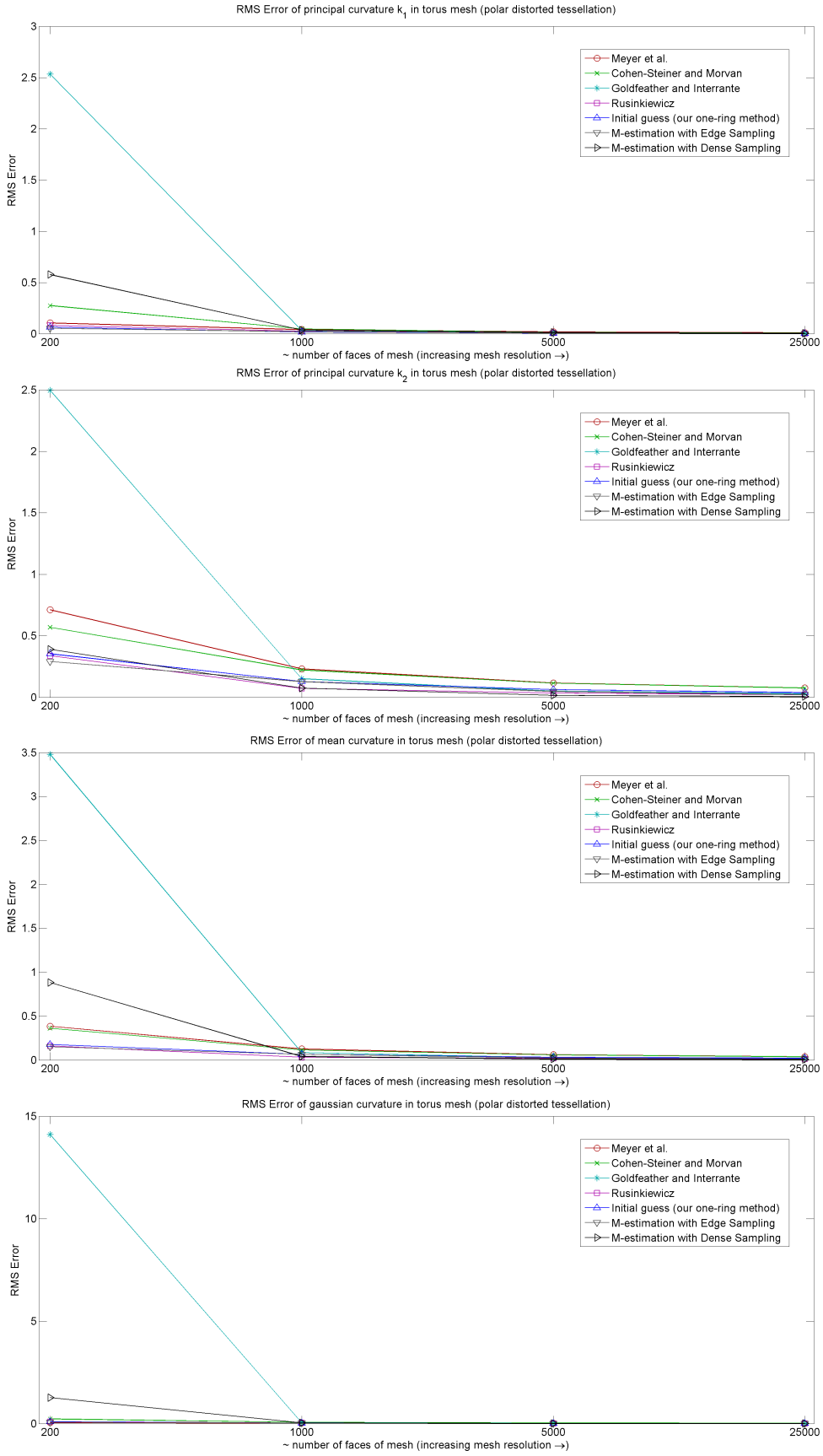

Figure 6.23: Plots of RMS Error for the polar distorted tessellations of torus

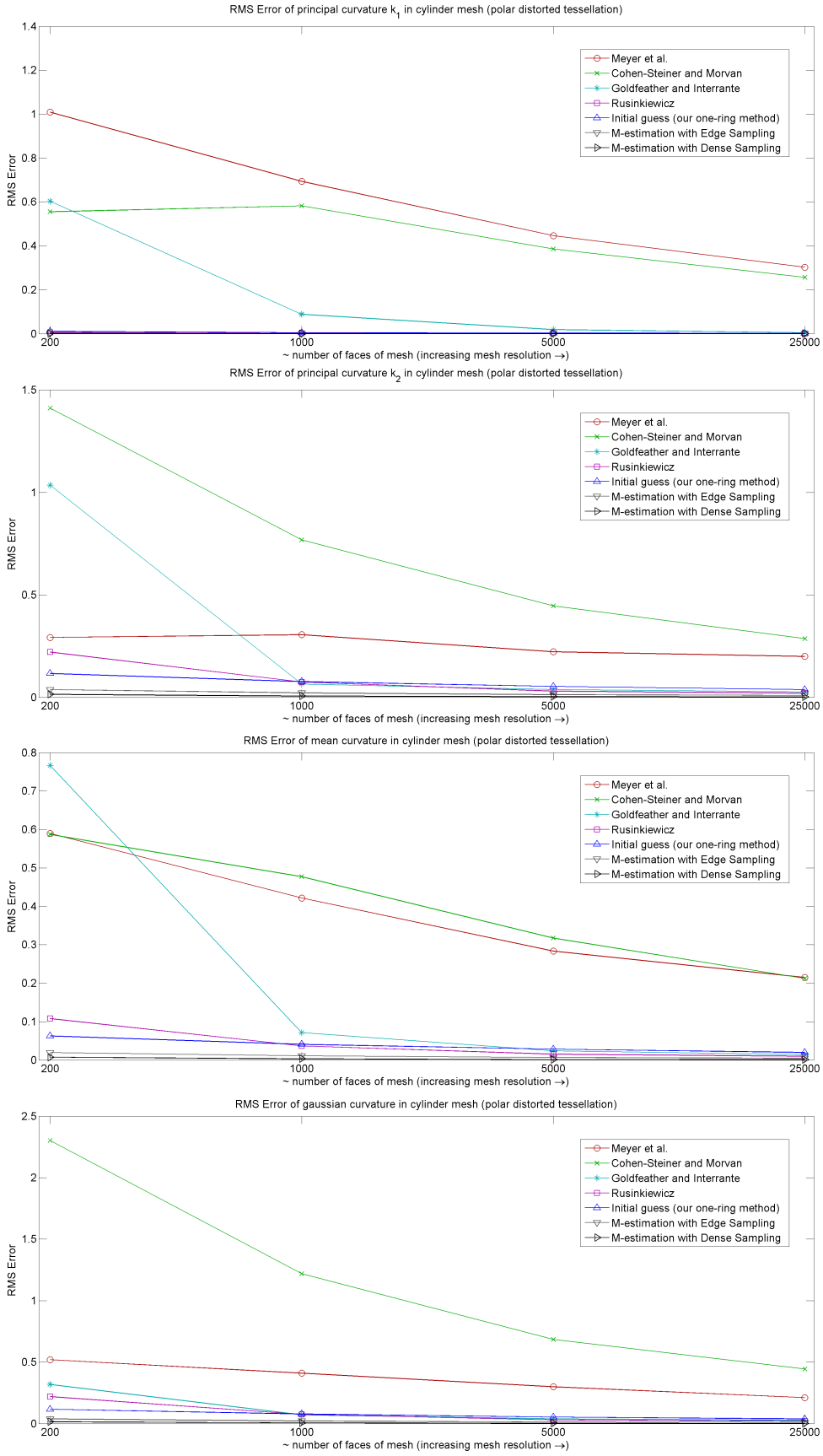

Figure 6.24: Plots of RMS Error for the polar distorted tessellations of cylinder

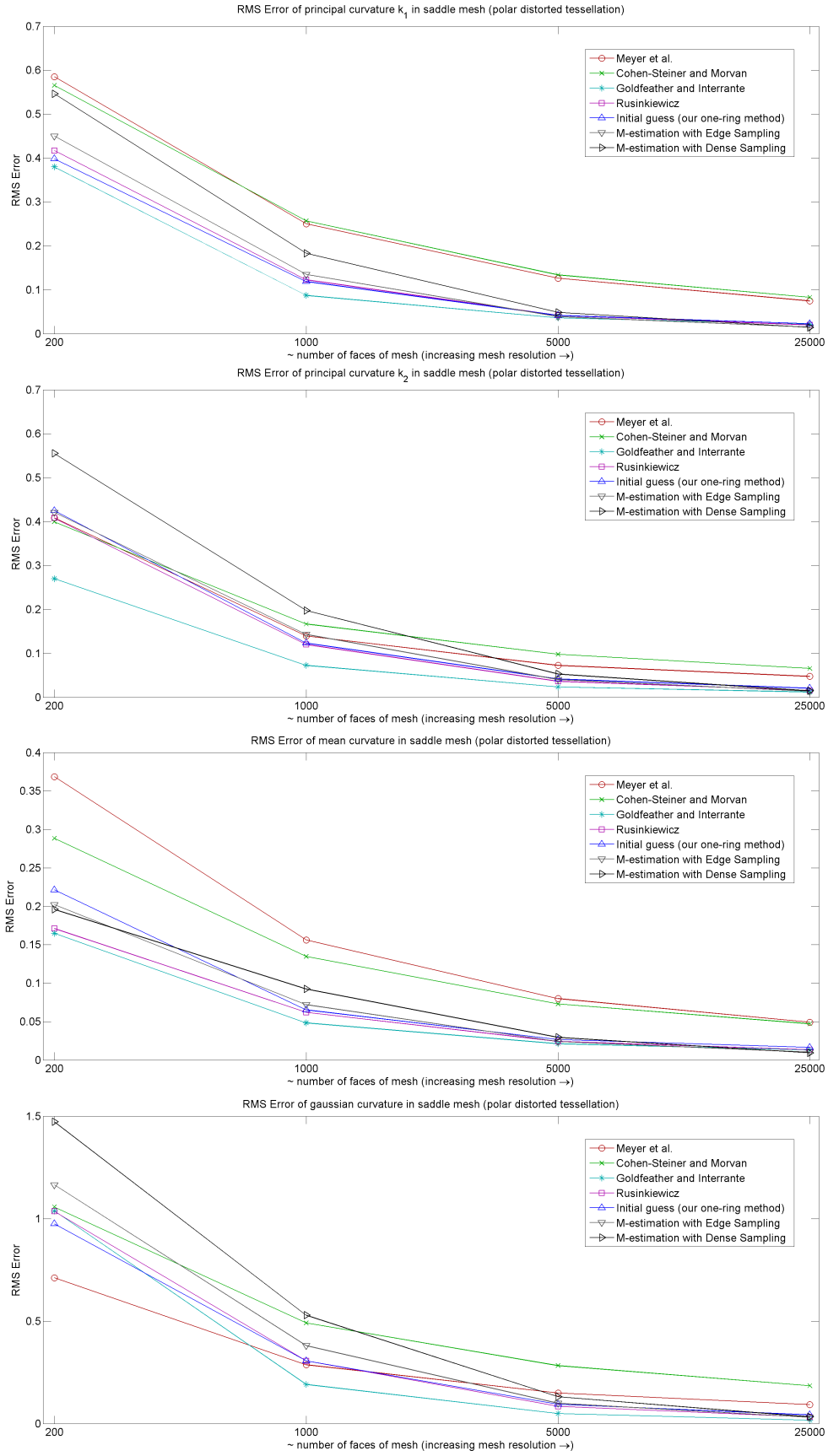

Figure 6.25: Plots of RMS Error for the polar distorted tessellations of saddle

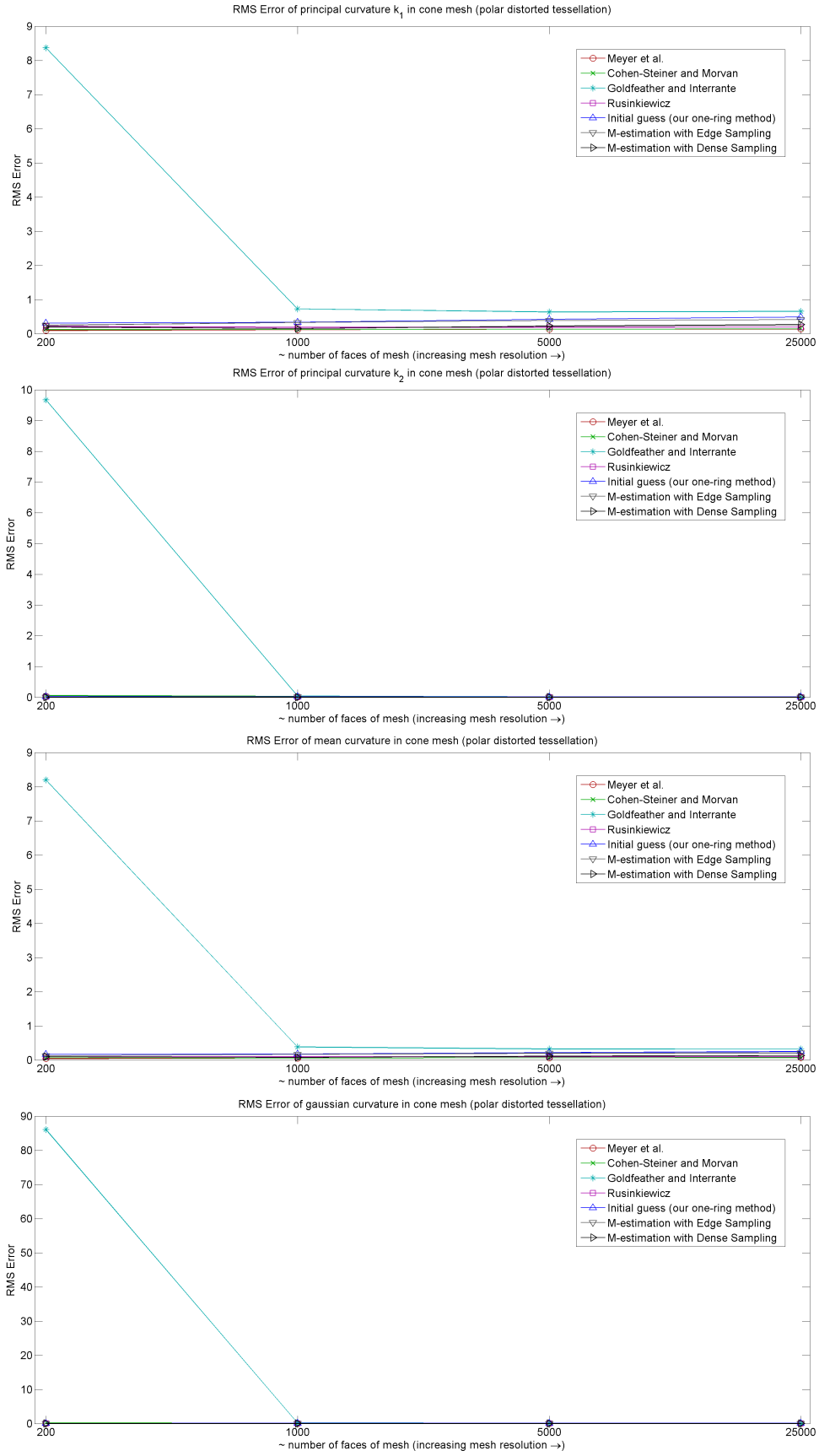

Figure 6.26: Plots of RMS Error for the polar distorted tessellations of cone

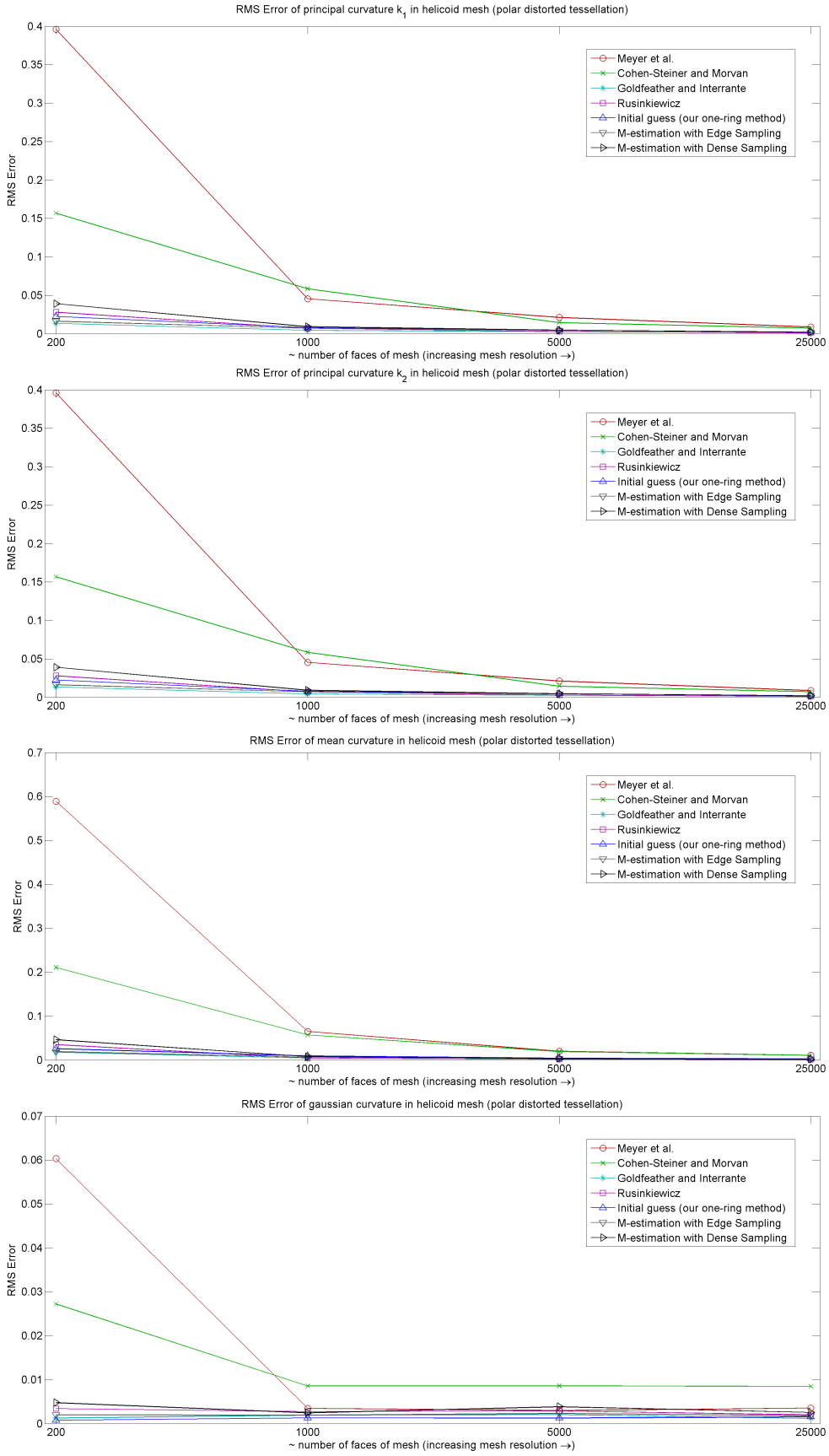

Figure 6.27: Plots of RMS Error for the polar distorted tessellations of helicoid

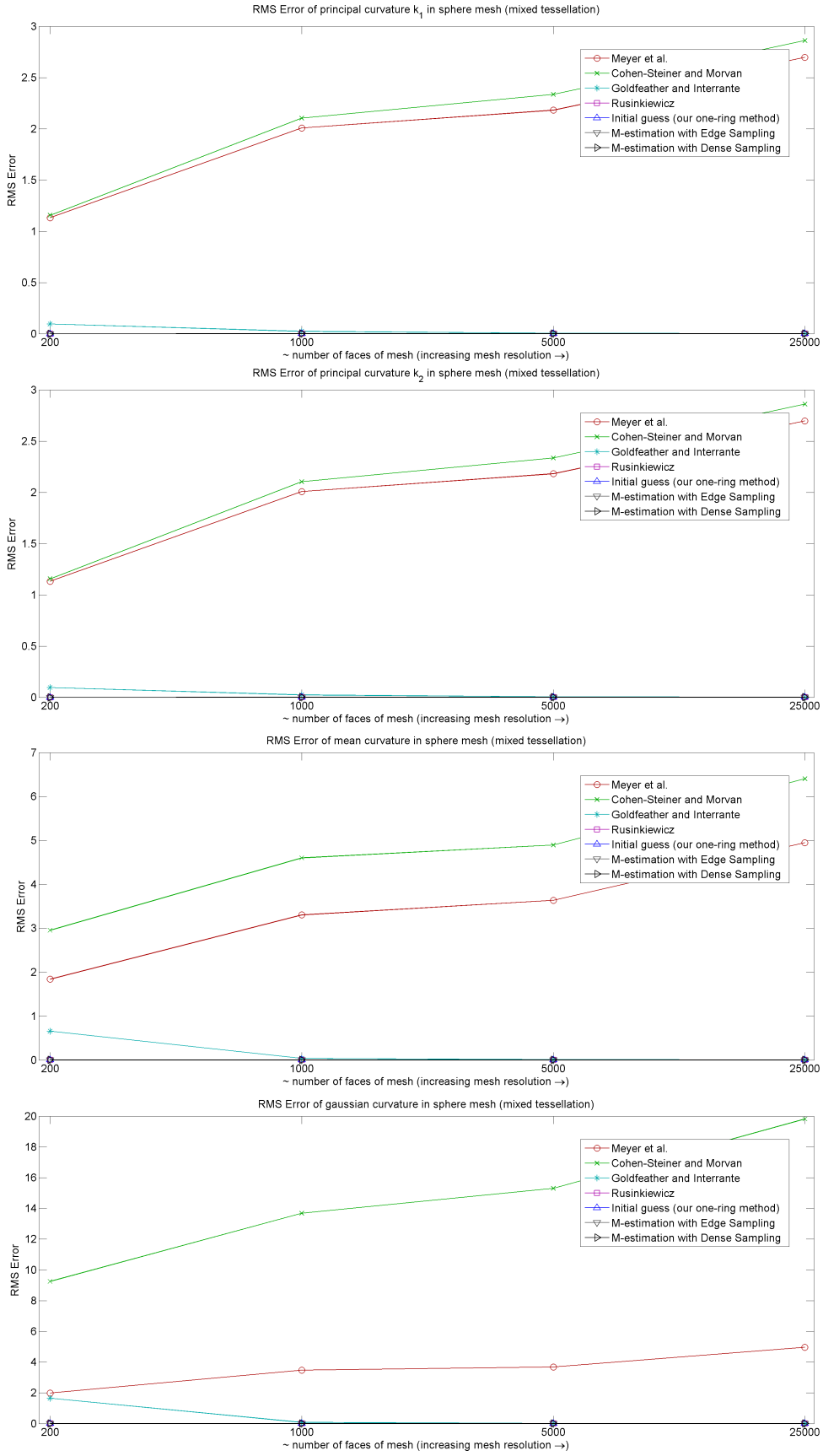

Figure 6.28: Plots of RMS Error for the mixed tessellations of sphere

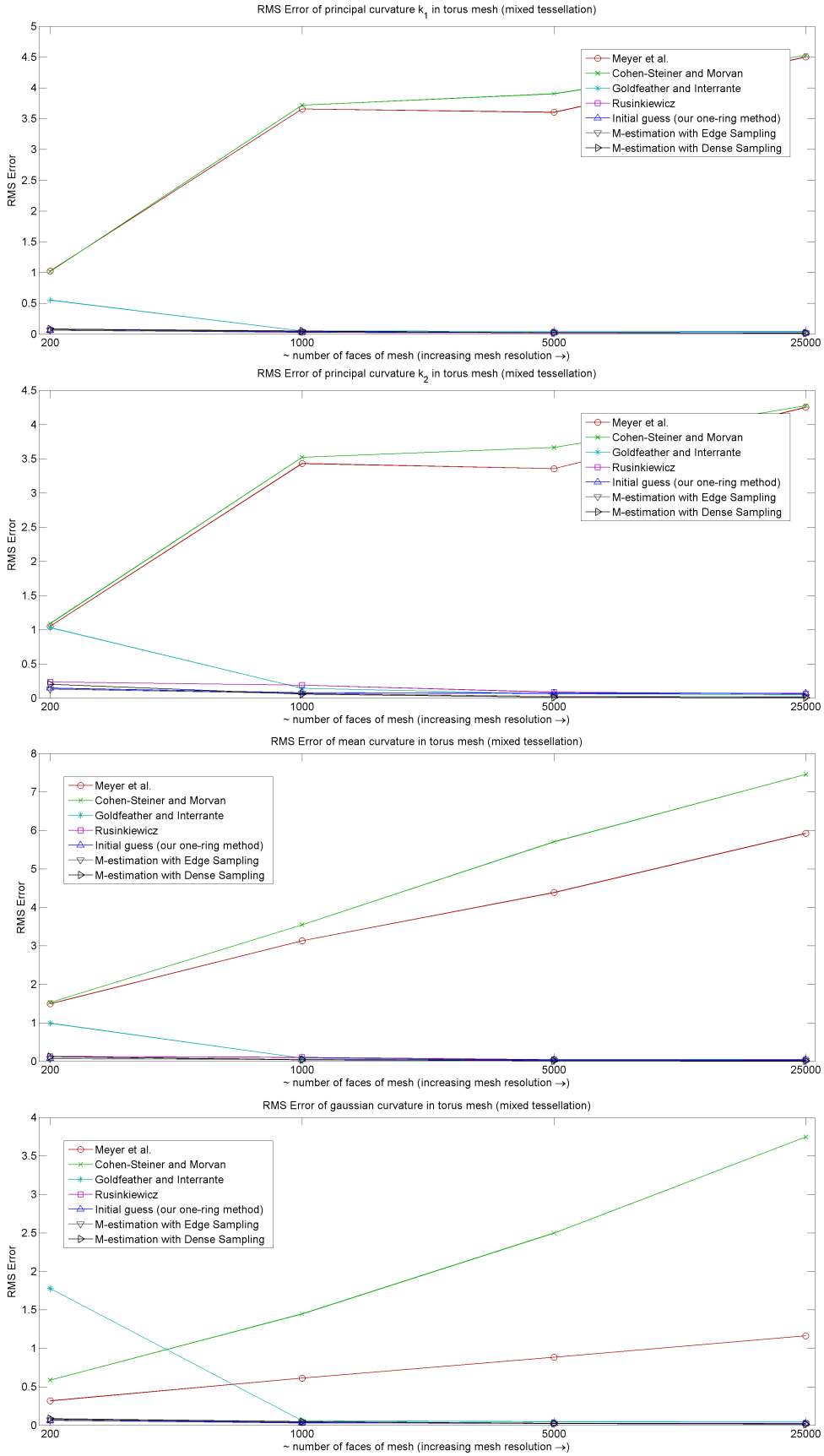

Figure 6.29: Plots of RMS Error for the mixed tessellations of torus

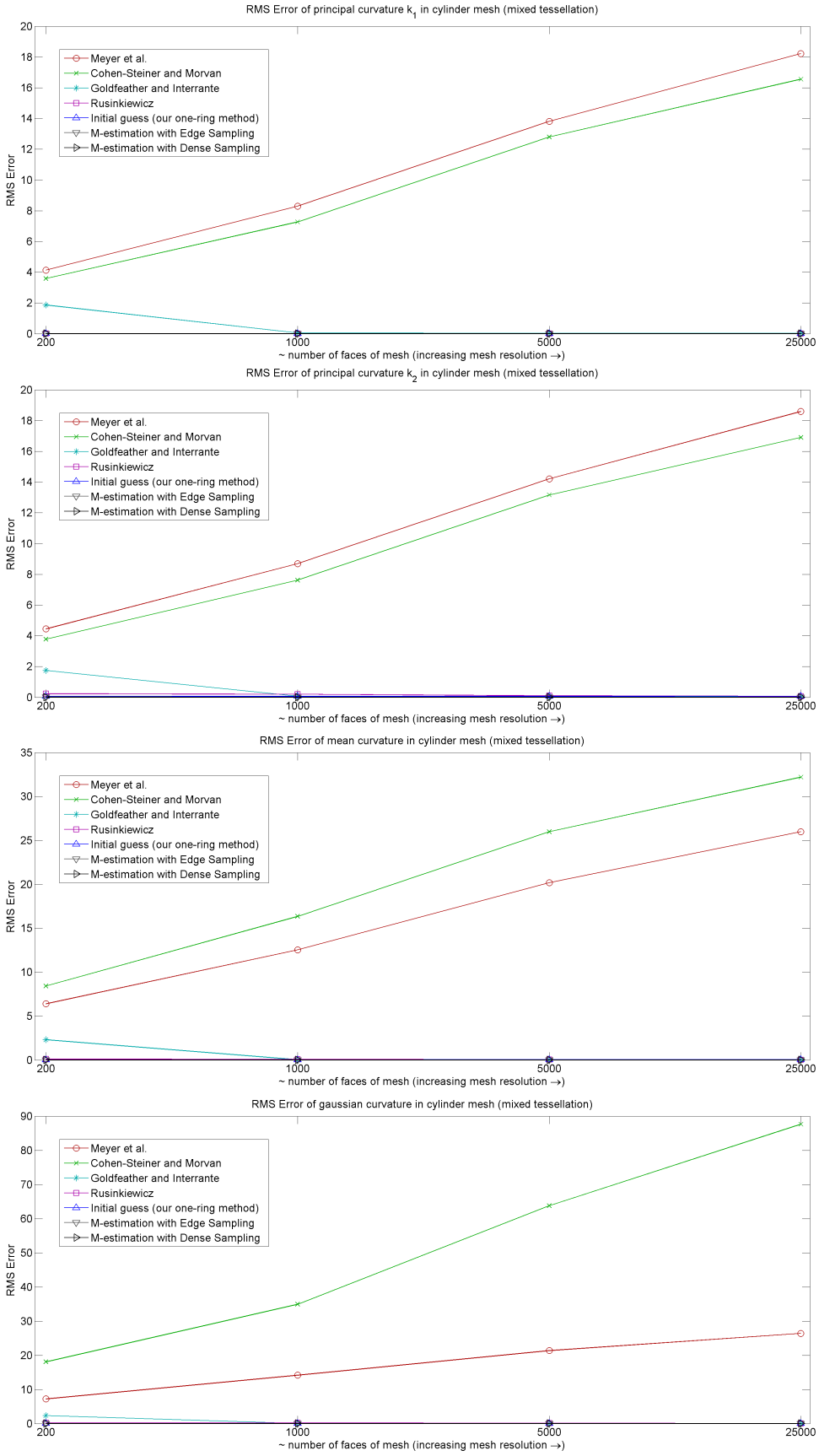

Figure 6.30: Plots of RMS Error for the mixed tessellations of cylinder

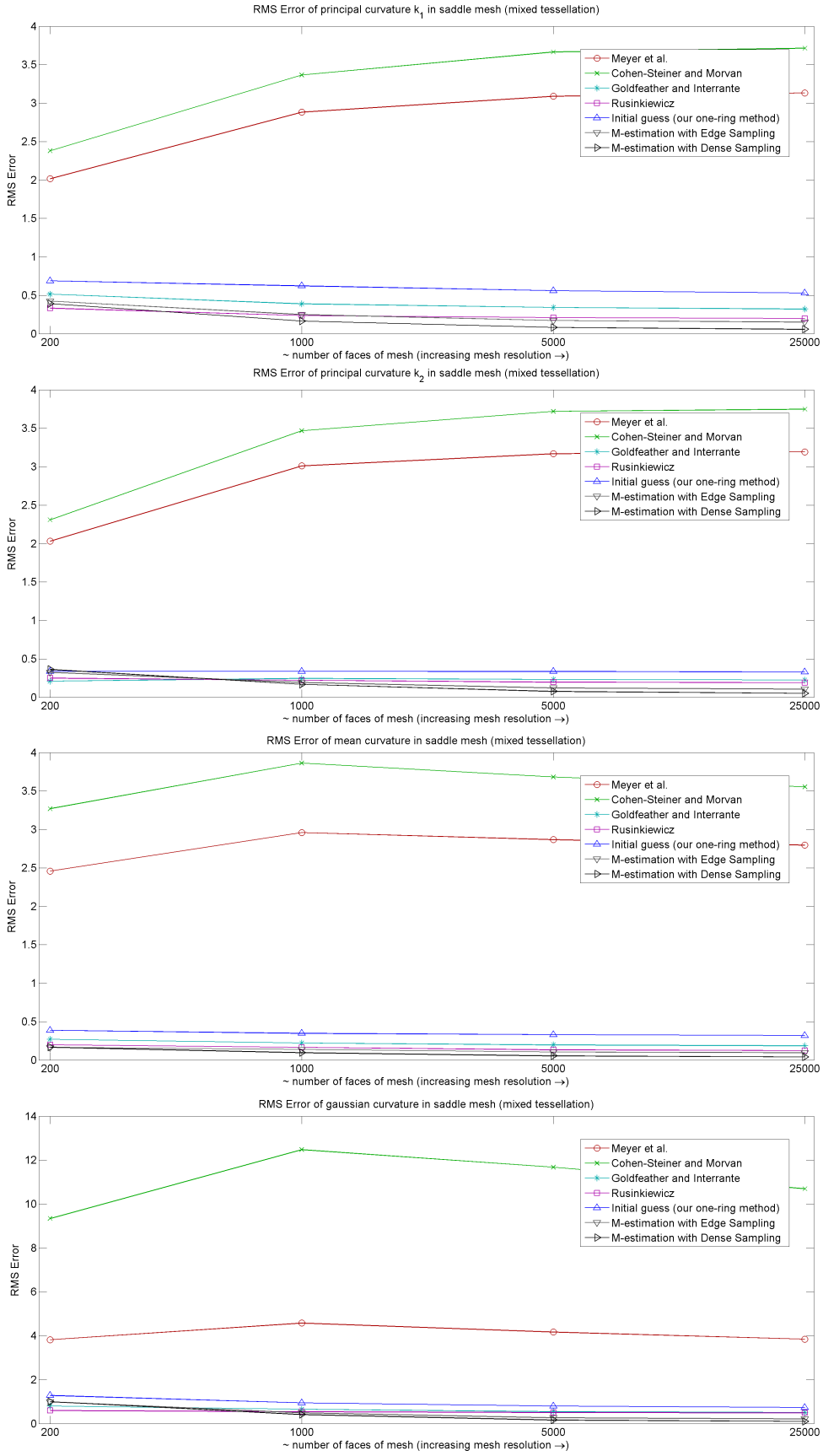

Figure 6.31: Plots of RMS Error for themixed tessellations of saddle

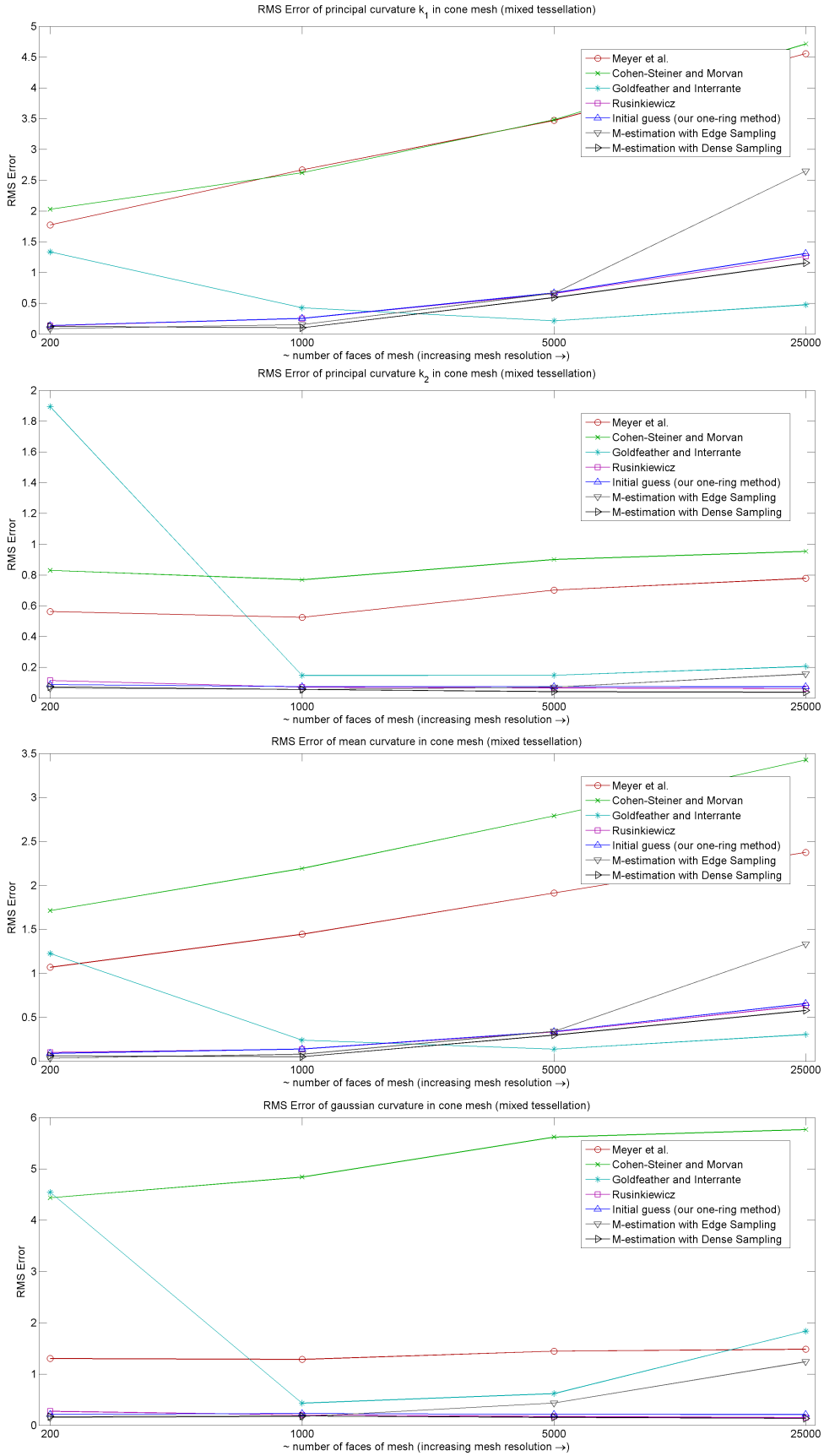

Figure 6.32: Plots of RMS Error for the mixed tessellations of cone

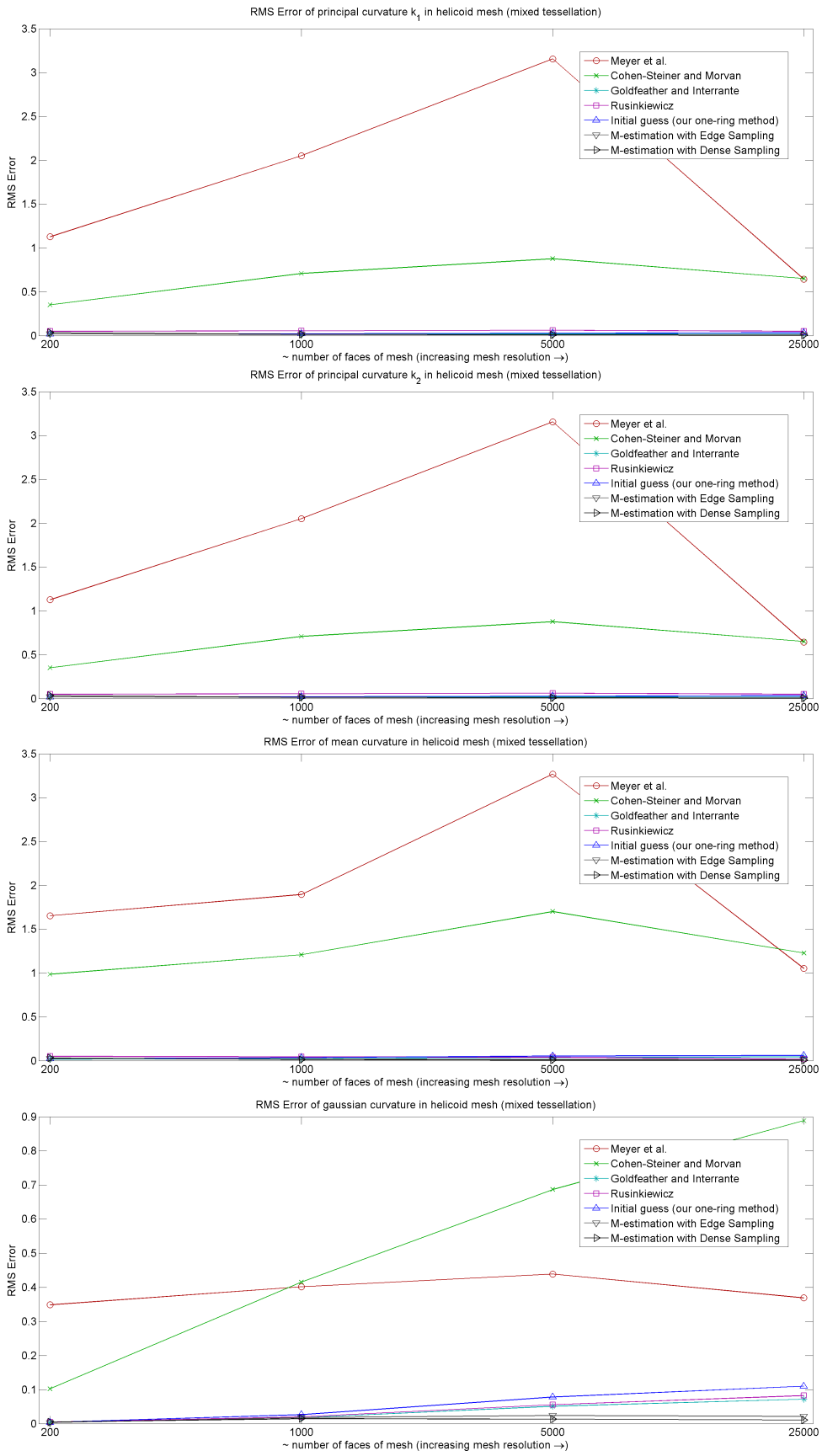

Figure 6.33: Plots of RMS Error for the mixed tessellations of helicoid

6.3 Irregular, coarse and noisy tessellations

In the last test suite, we created irregularly tessellated surfaces with some added light noise on the normal direction of the vertices (with variance 2% of the median edge length). Starting from regularly and densely tessellated initial mesh of our test surfaces, we progressively generate coarser and coarser tessellations, by removing faces so that the shape of the surface is preserved. More specifically, we used the 'reducePatch' command of Matlab. The progressive removal of faces causes irregular tessellations to appear as shown in figure 6.34 for our test surfaces. We chose this test suite so that we examine the combined effect of irregular tessellation and noise in different resolutions, which is a common case in several real world meshes. We note that we repeated the tests 100 times and took the average RMS errors.

Our M-estimation method again seems to outperform in all our tests. The dense sampling mode of the algorithm achieves lower errors than the existing edge sampling. Also our fixed region method has satisfactory results, except in the case of 90% reduced torus, where in principal curvature k_1, a larger error appears (due to an almost degenerate triangle in an irregularly shaped one-ring neighborhood). On the other hand, the M-estimation method, even from a relatively bad initial guess, again fixes that estimation. The plots of RMS errors for these test surfaces, having the combined effect of noise and irregularity are shown in figures 6.35 - 6.40.

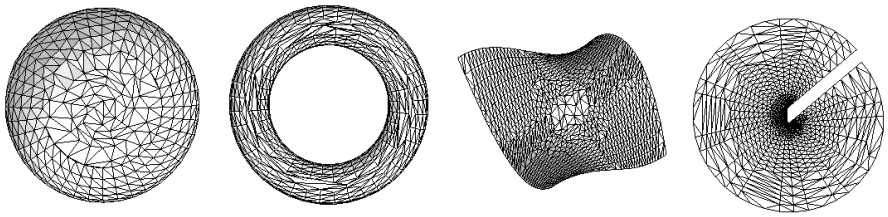

Figure 6.34: Examples of 50% reduced test surfaces: sphere, torus, monkey saddle, helicoid

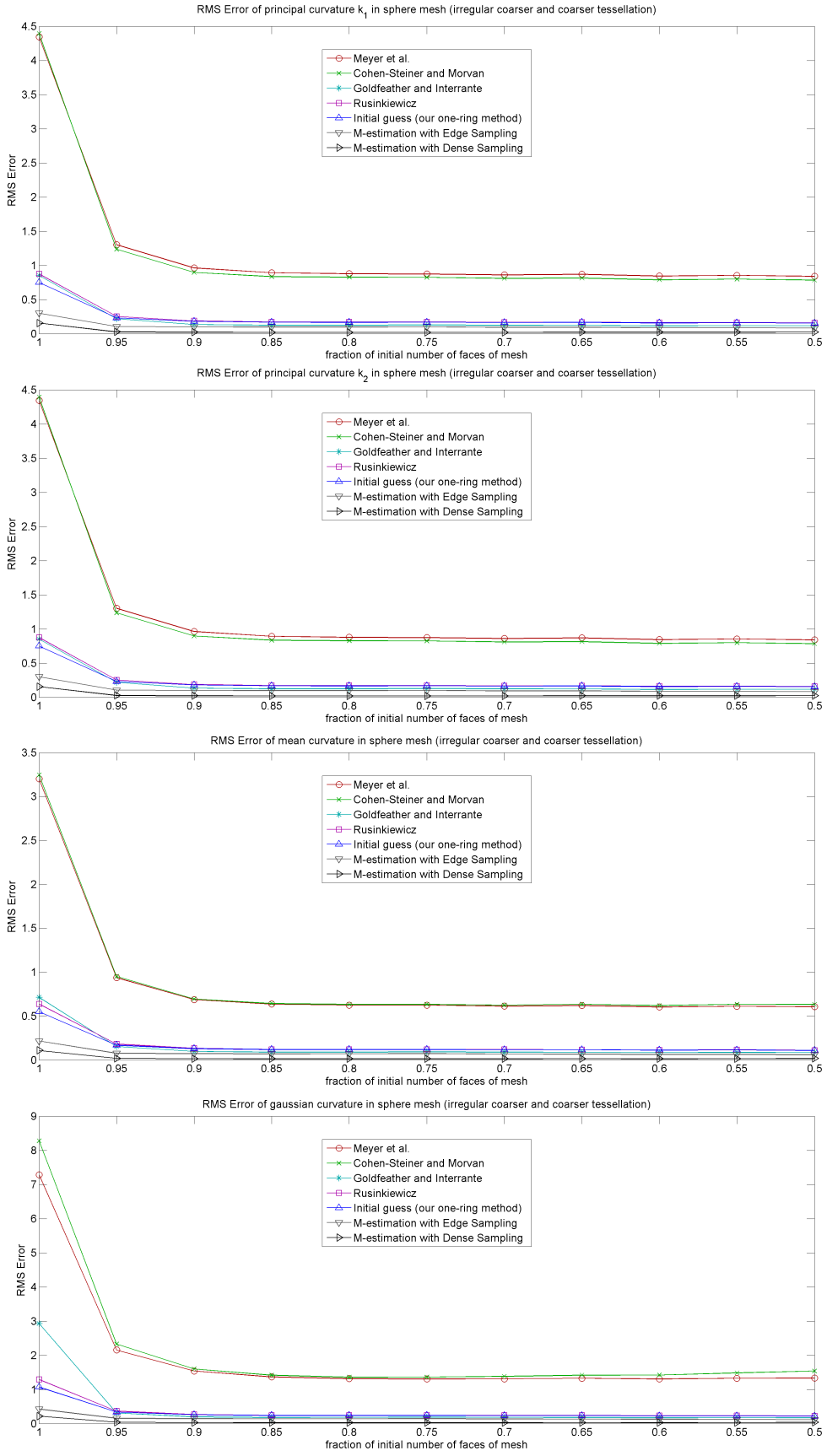

Figure 6.35: Plots of RMS Error for the reduced and noisy tessellations of sphere

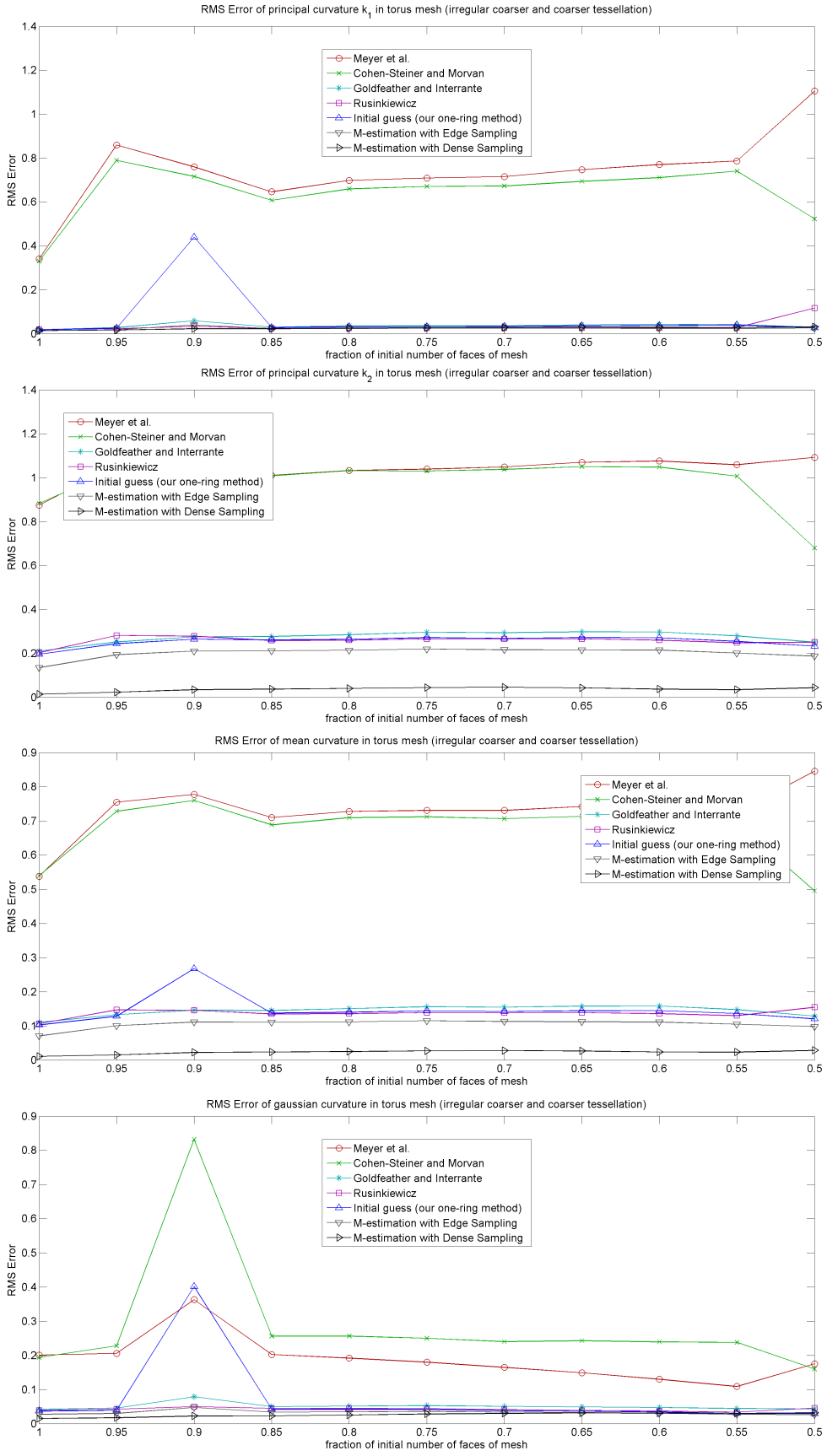

Figure 6.36: Plots of RMS Error for the reduced and noisy tessellations of torus

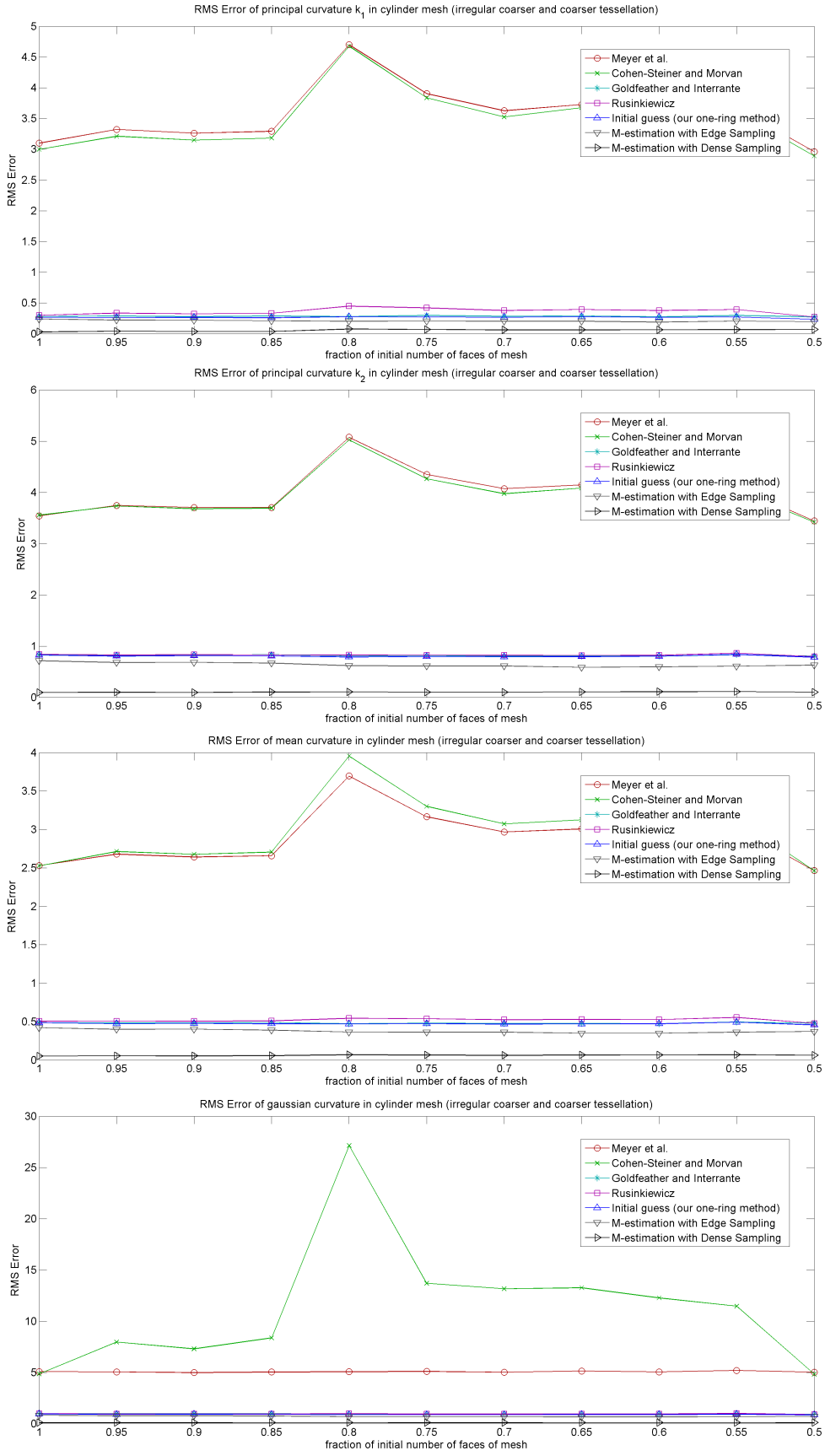

Figure 6.37: Plots of RMS Error for the reduced and noisy tessellations of cylinder

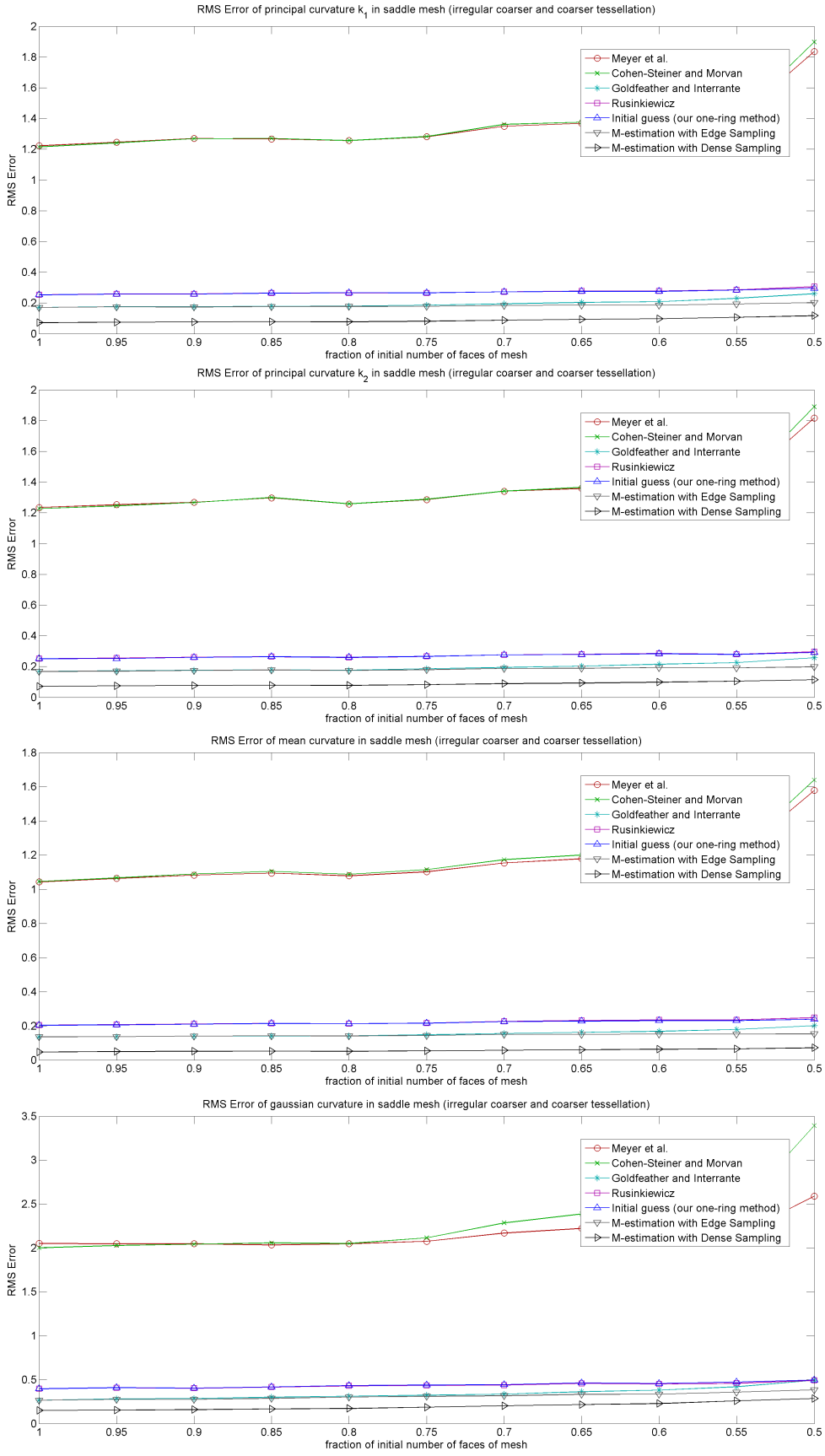

Figure 6.38: Plots of RMS Error for the reduced and noisy tessellations of saddle

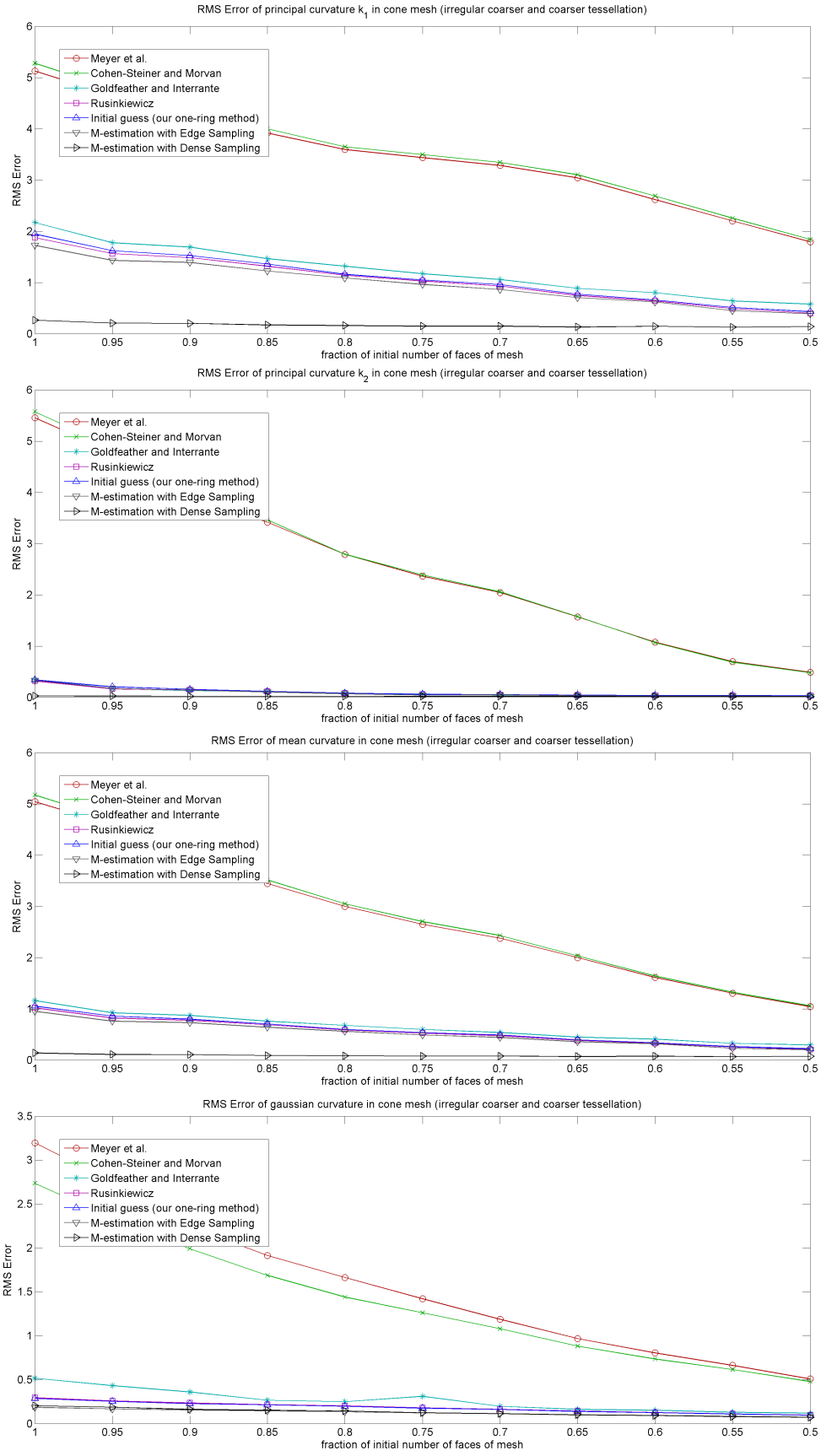

Figure 6.39: Plots of RMS Error for the reduced and noisy tessellations of cone

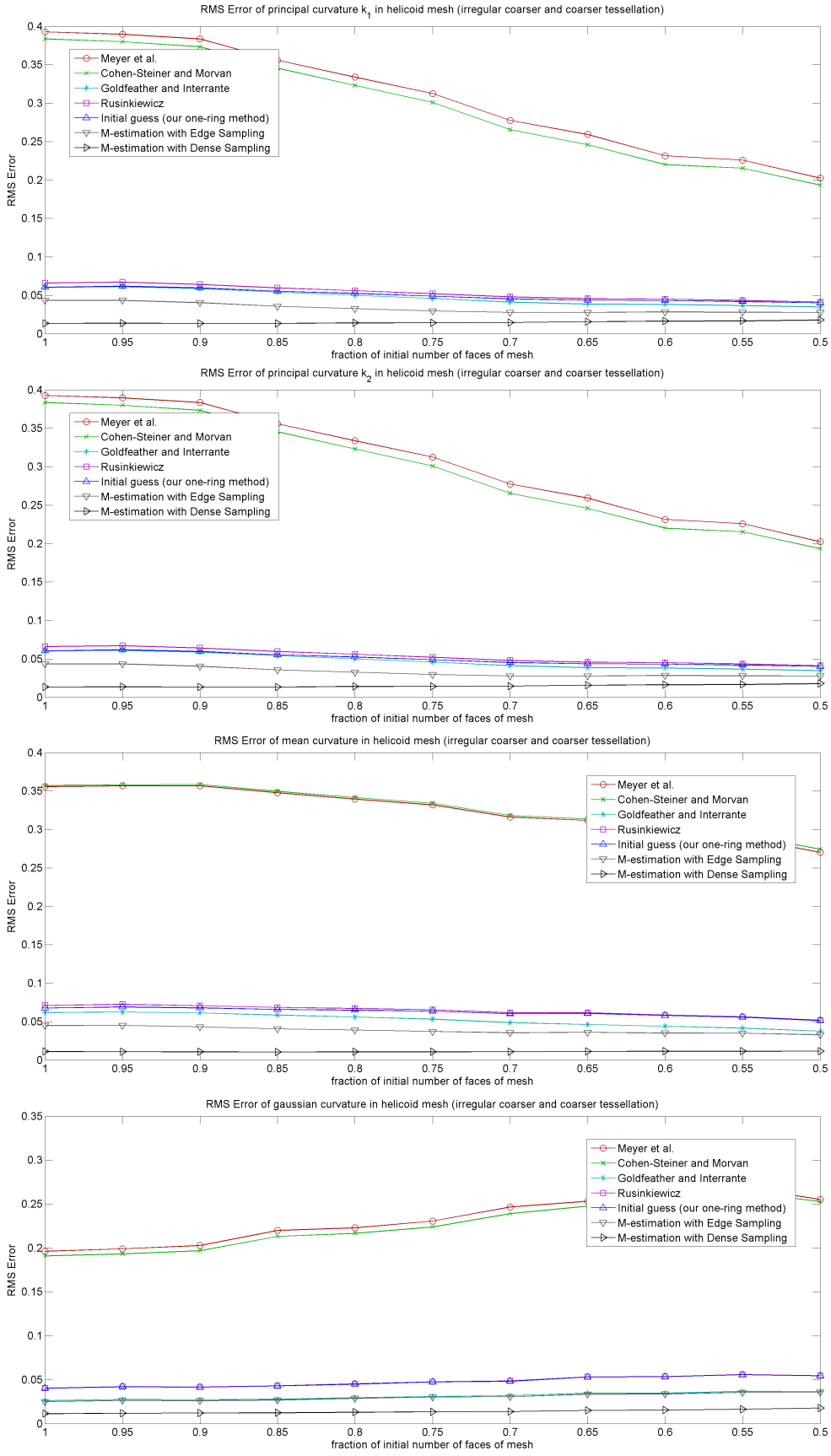

Figure 6.40: Plots of RMS Error for the reduced and noisy tessellations of helicoid

6.4 Results on some real world meshes

Apart from the test surfaces that we showed in the previous sections, it would be interesting to check how the above methods behave in some real world example meshes, generated by artists and modelers in 3D professional applications. Thus, we present four example meshes, where we know that the surfaces are smooth and C^2 continuous everywhere. For the visualization of the curvature tensor on the example meshes, we have used Rusinkiewicz's trimesh2 library [58]. The color assignment to the principal curvature values is shown in figure 6.41.

In the figures 6.42 - 6.45, we show the visualization of the curvature field on a mesh (30K faces), representing the bones of a left human lip, as produced by the one-ring normal cycle method, cubic patch fitting method, Rusinkiewicz's approach and our M-estimation algorithm with dense sampling. The mesh is designed in the 3D Studio Max. Our M-estimation seems to produce more consistent and continuous curvature field especially on the joints and the bottom of the foot. In the figures 6.46 - 6.49 (20K faces), we show the same visualizations on an ear. Our M-estimation method clearly produces a more continuous non-noisy curvature field. In figures 6.50 - 6.53 (150K faces), we show the same visualizations on a dog, which was created with ShapeShop [61] by irregularly tessellating an implicit surface (thus, the surface is C^2 continuous everywhere and the curvature field should be smooth). However, all the methods, except to our M-estimation one, produces considerably noisy curvature fields. Finally, we show results on the scanned David's head (300K faces), where our M-estimation method seems to produce less noisy results (see figures 6.54 - 6.57).

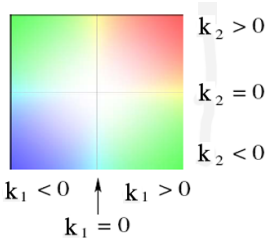

Figure 6.41: Color-coded visualization of curvature [59]

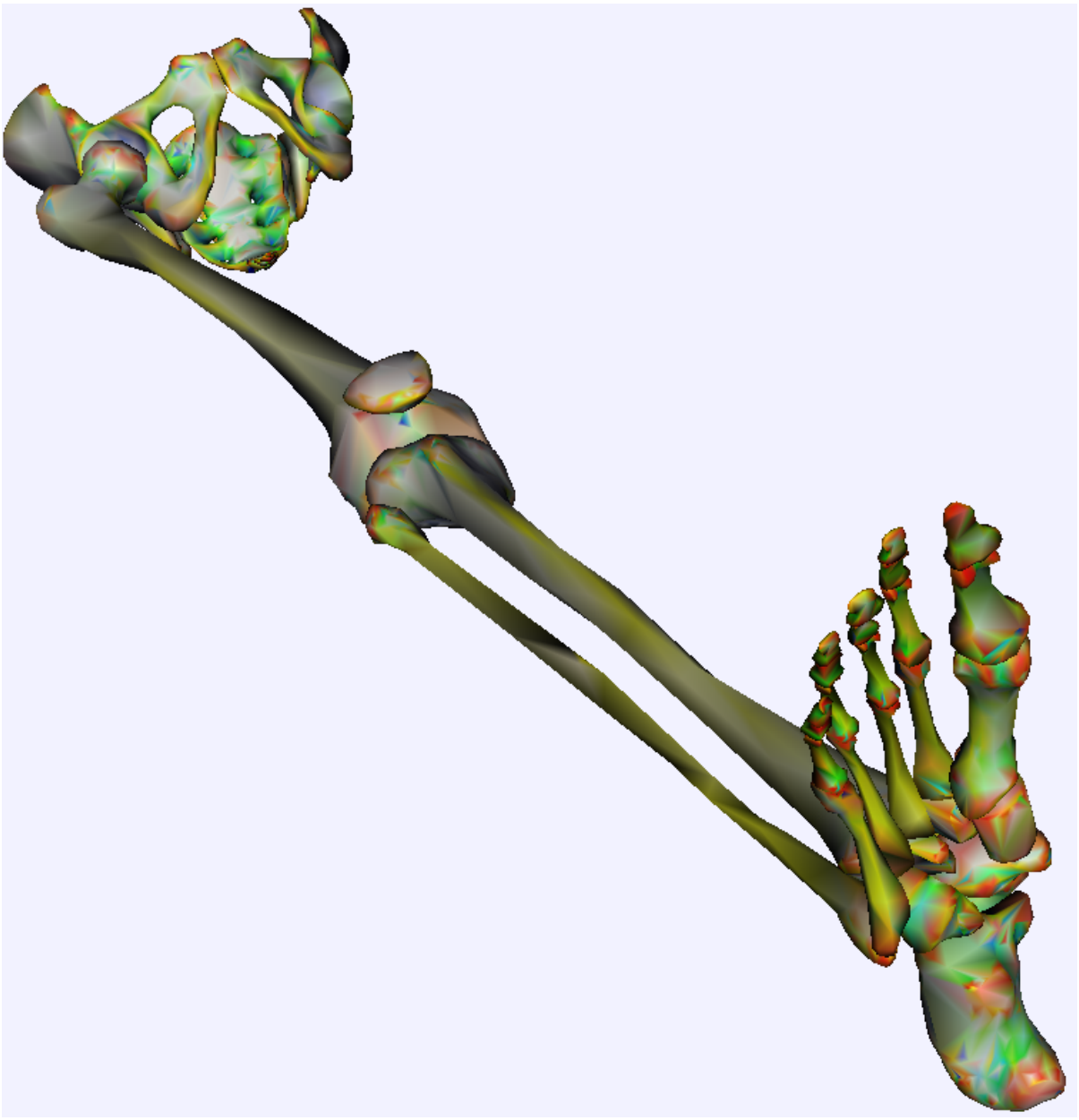

Figure 6.42: Steiner's and Morvan's curvature field visualization on the bones of a left extreme

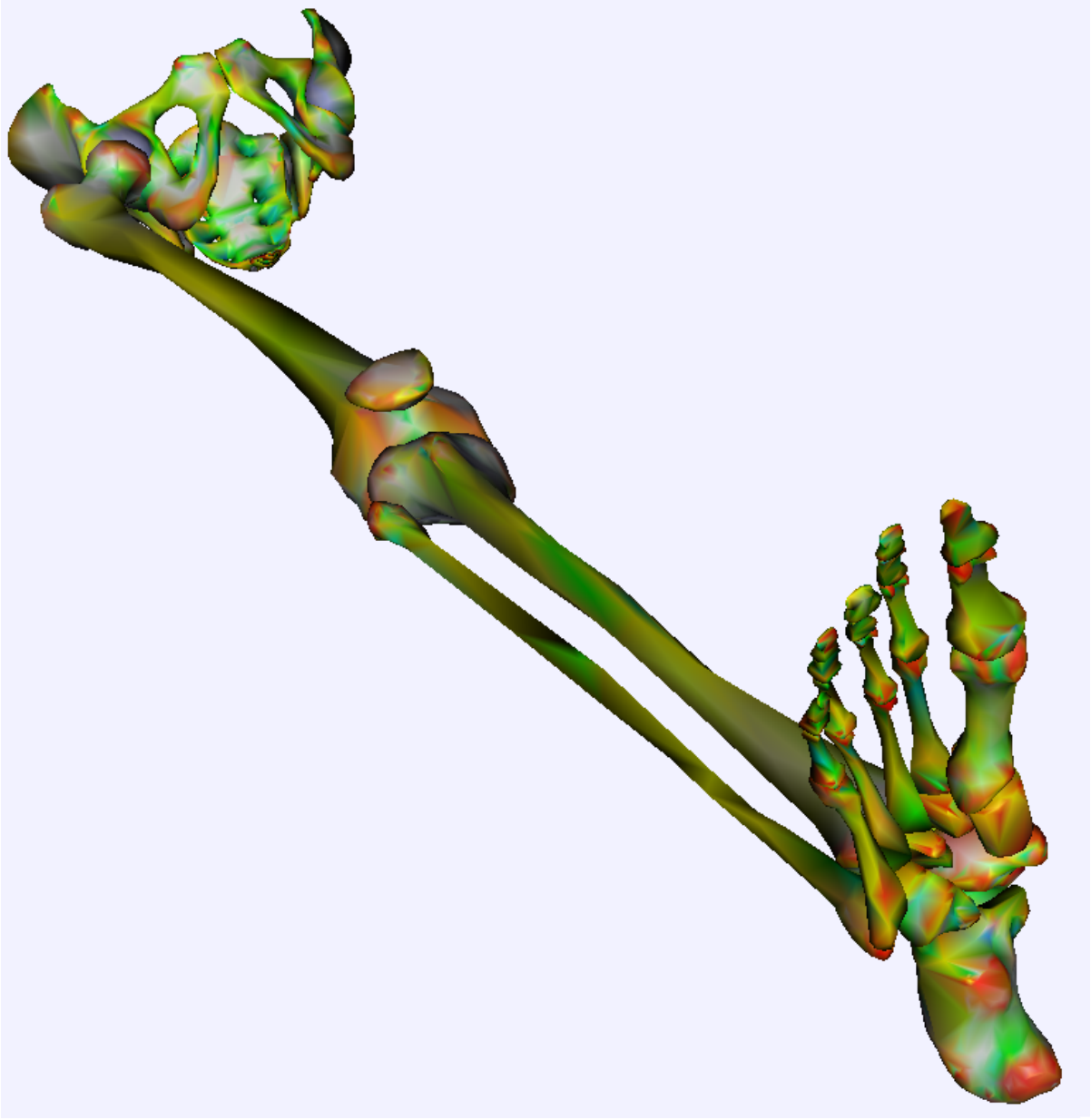

Figure 6.43: Goldfeather's and Interrante's curvature field visualization on the bones of a left extreme

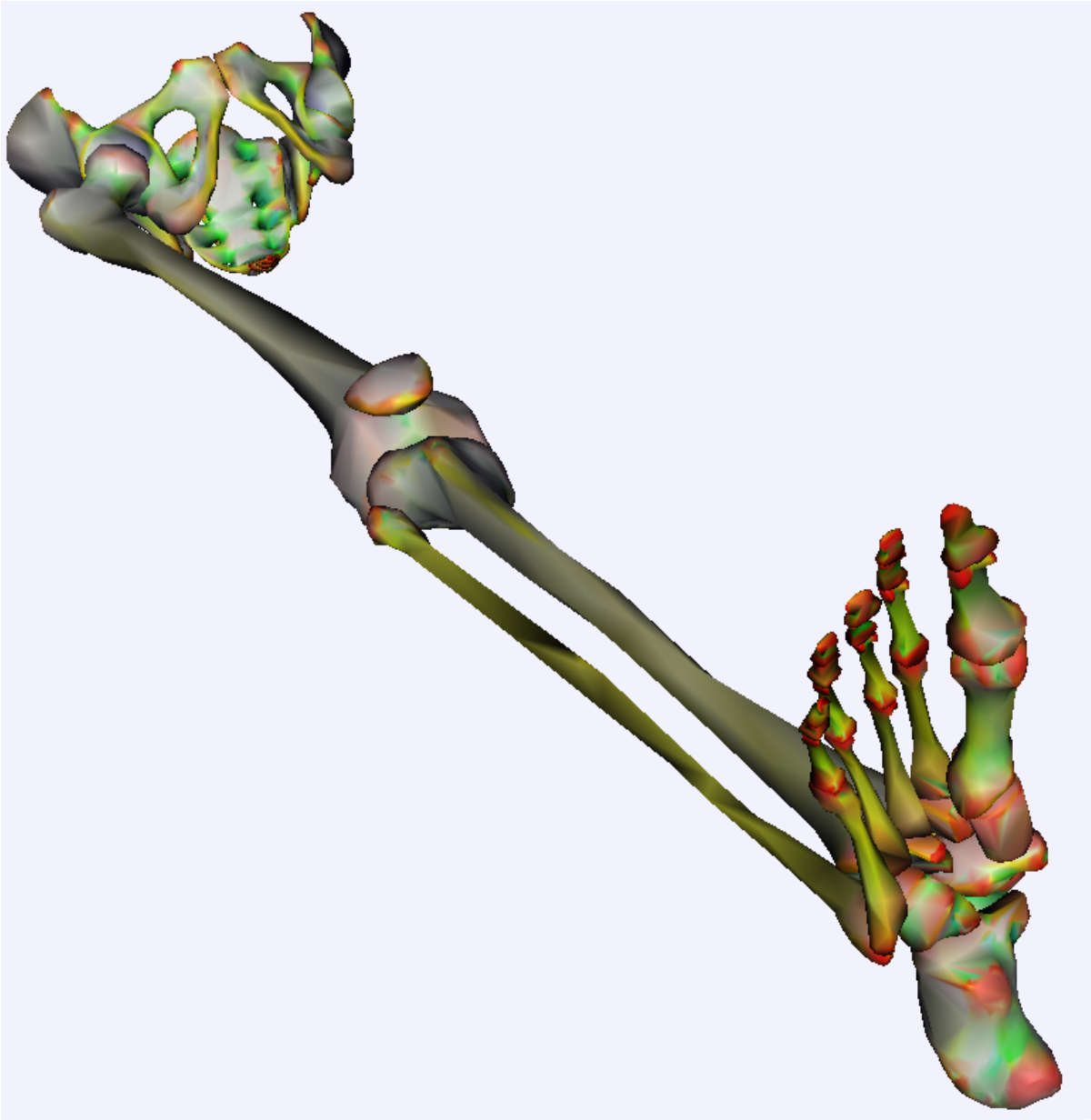

Figure 6.44: Rusinkiewicz's curvature field visualization on the bones of a left extreme

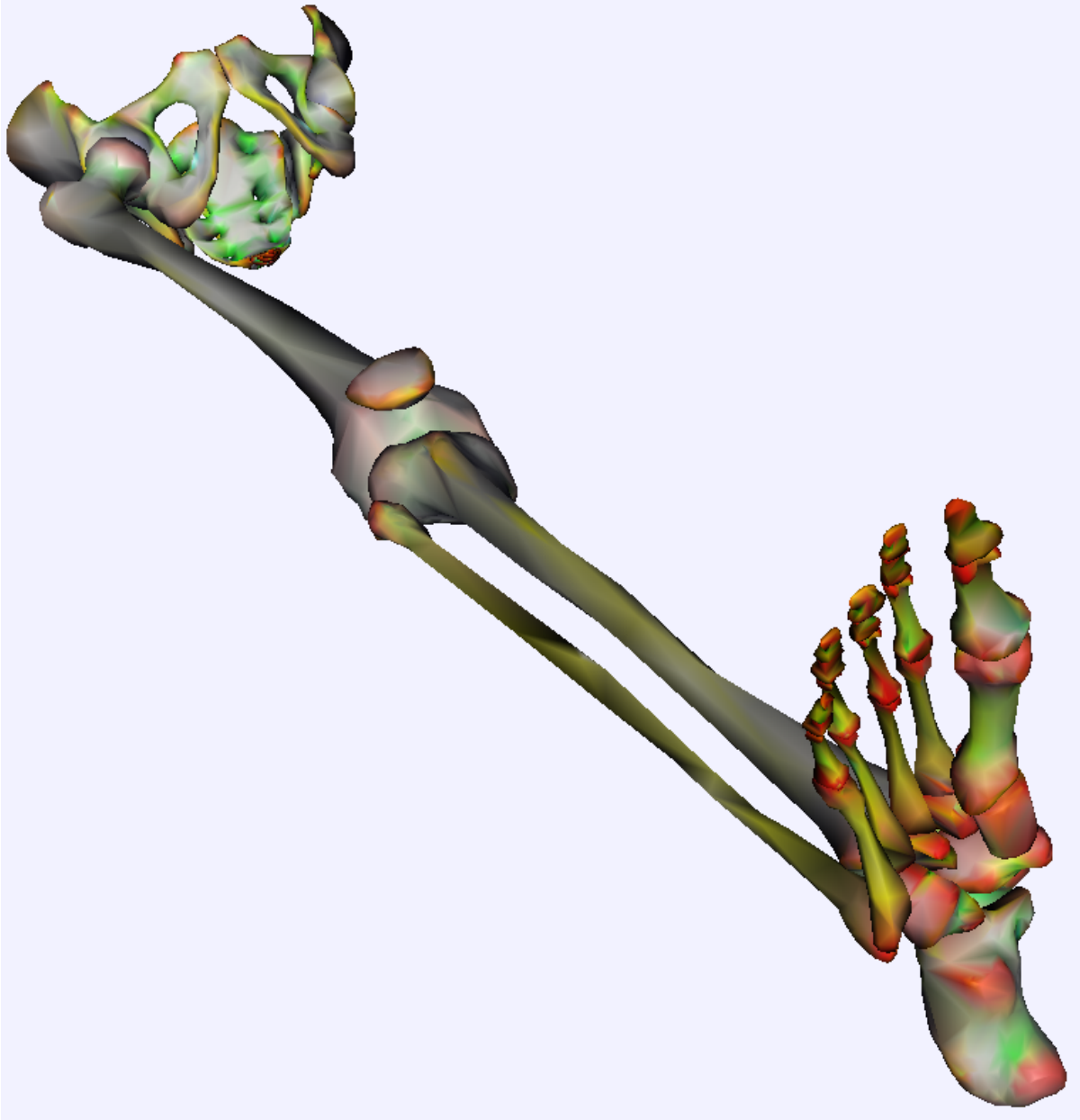

Figure 6.45: Our M-estimation curvature field visualization on the bones of a left extreme

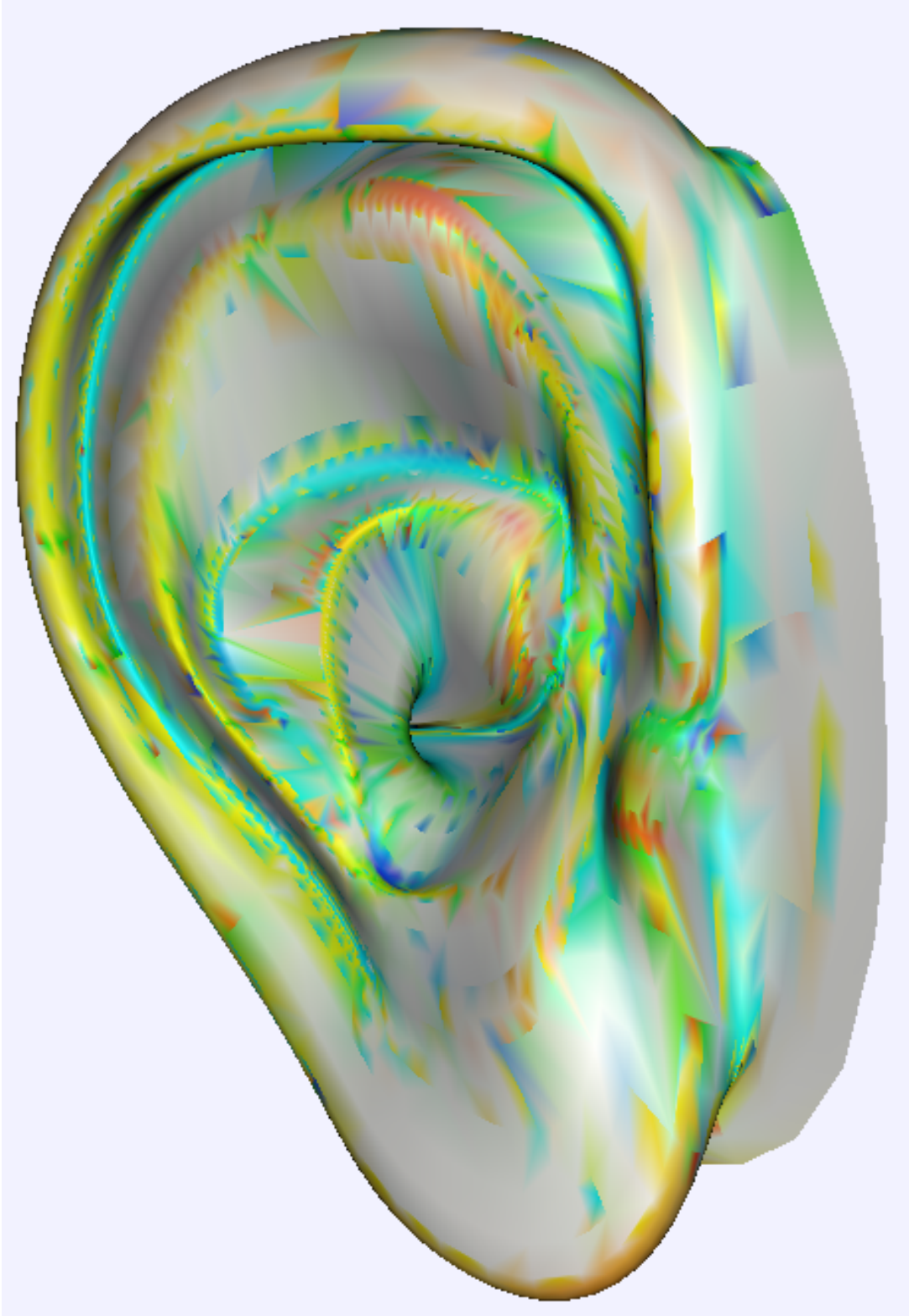

Figure 6.46: Steiner's and Morvan's curvature field visualization on an ear

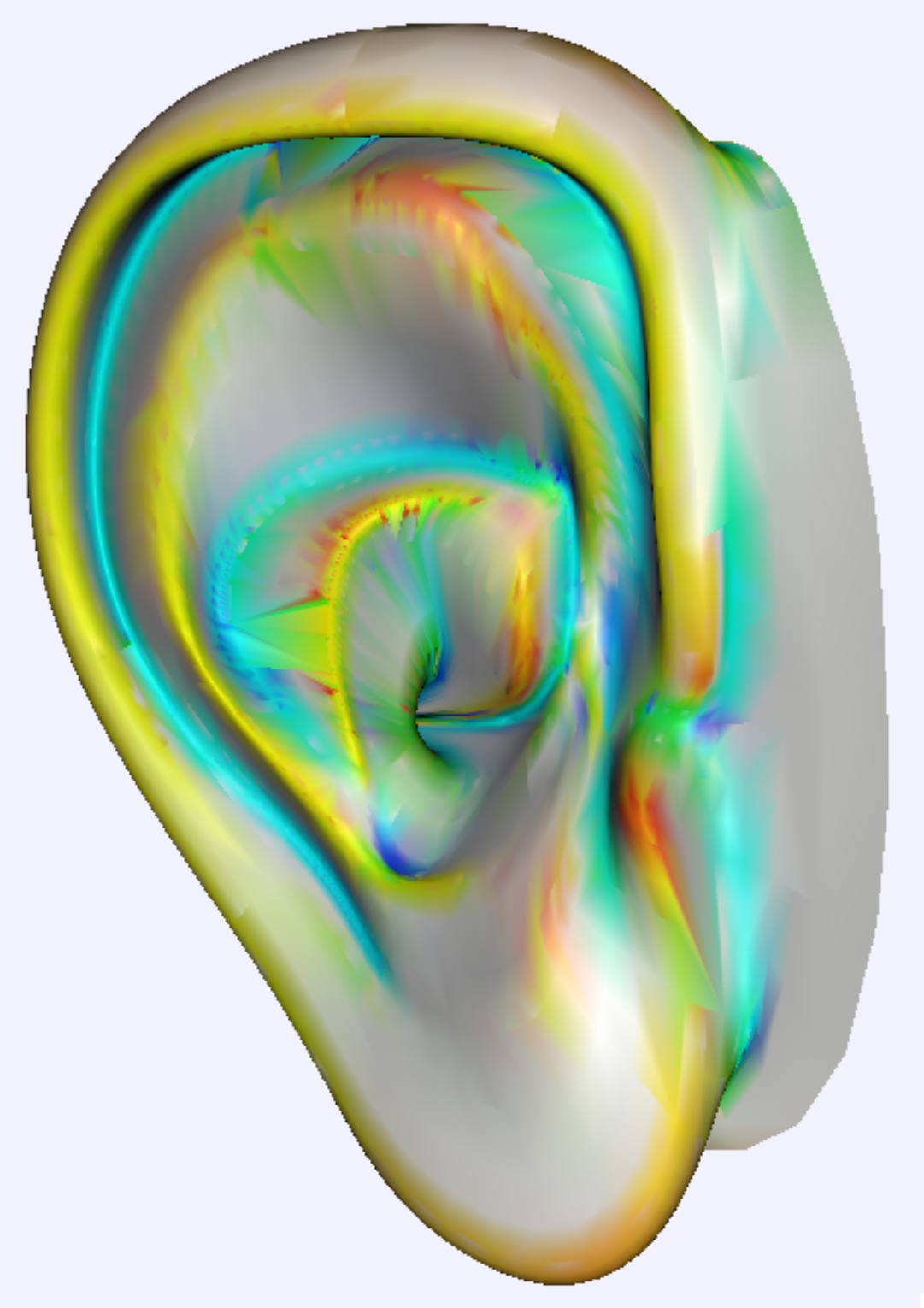

Figure 6.47: Goldfeather's and Interrante's curvature field visualization on an ear

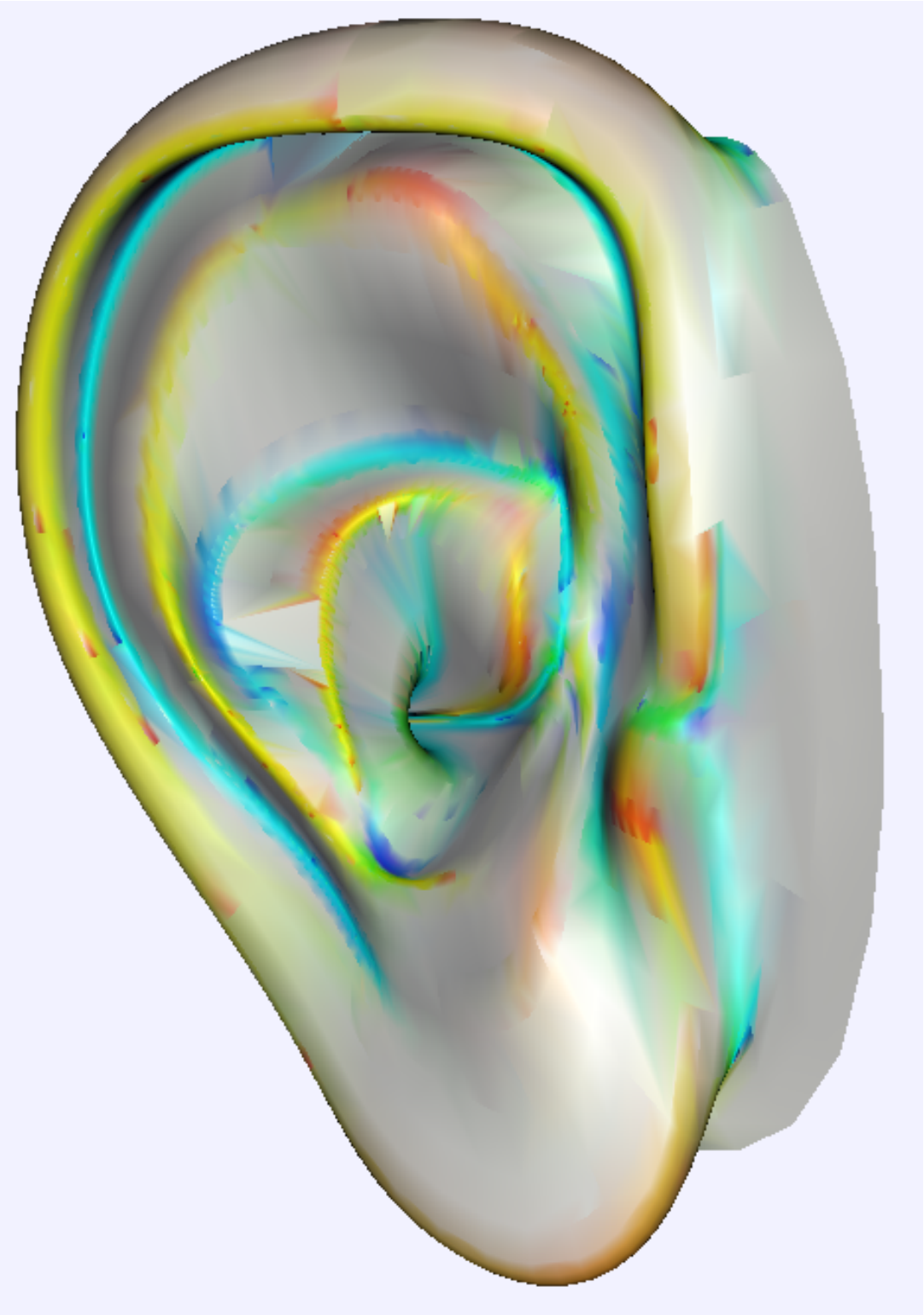

Figure 6.48: Rusinkiewicz's curvature field visualization on an ear

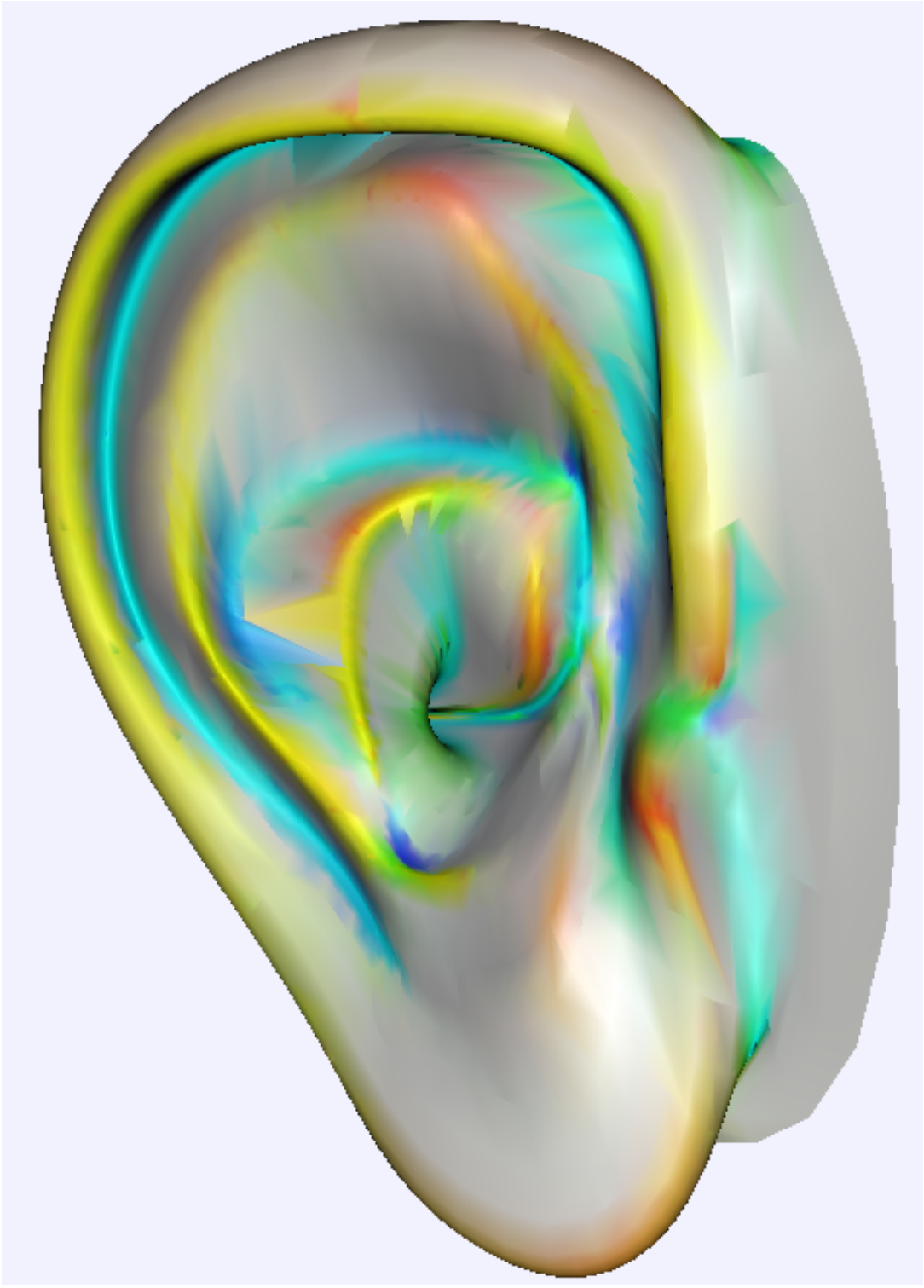

Figure 6.49: Our M-estimation curvature field visualization on an ear

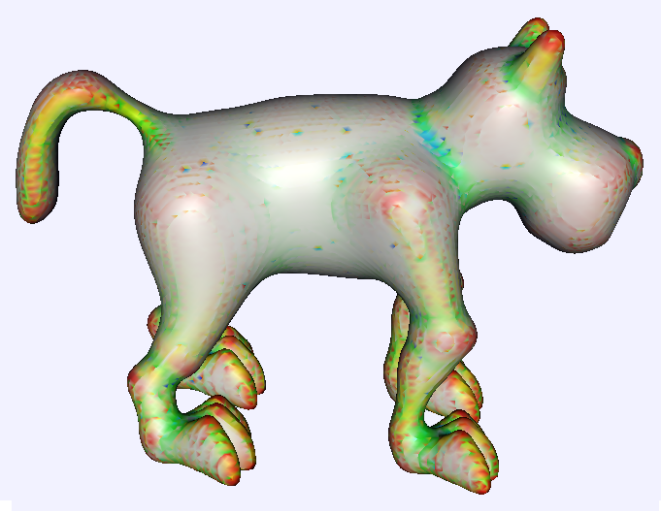

Figure 6.50: Steiner's and Morvan's curvature field visualization on a C^2 dog

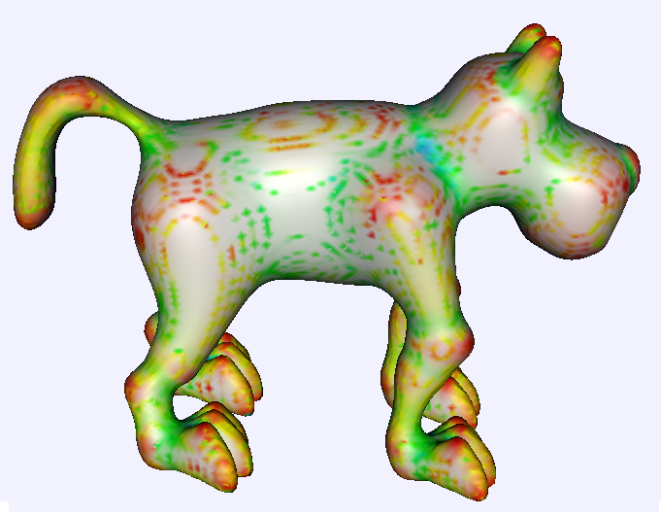

Figure 6.51: Goldfeather's and Interrante's curvature field visualization on a C^2 dog

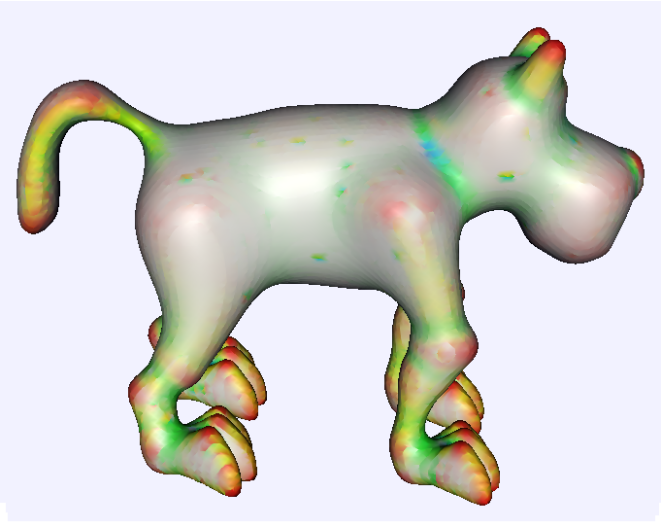

Figure 6.52: Rusinkiewicz's curvature field visualization on a C^2 dog

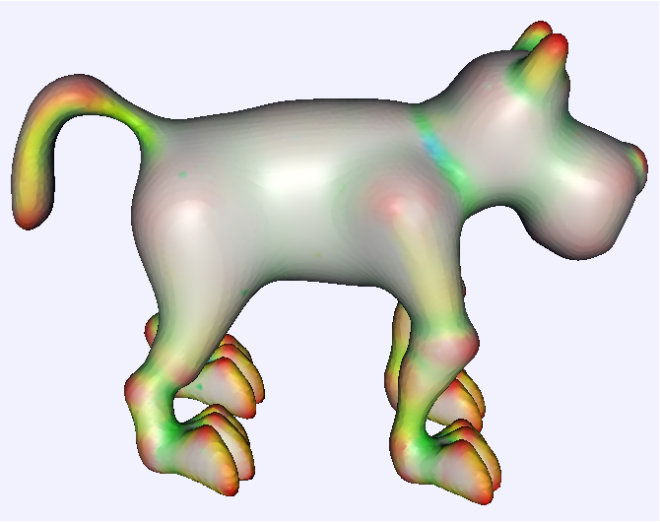

Figure 6.53: Our M-estimation curvature field visualization on a C^2 dog

Figure 6.54: Steiner's and Morvan's curvature field visualization on the David's head

Figure 6.55: Goldfeather and Interrante's curvature field visualization on the David's head

Figure 6.56: Rusinkiewicz's curvature field visualization on the David's head

Figure 6.57: Our M-estimation curvature field visualization on the David's head

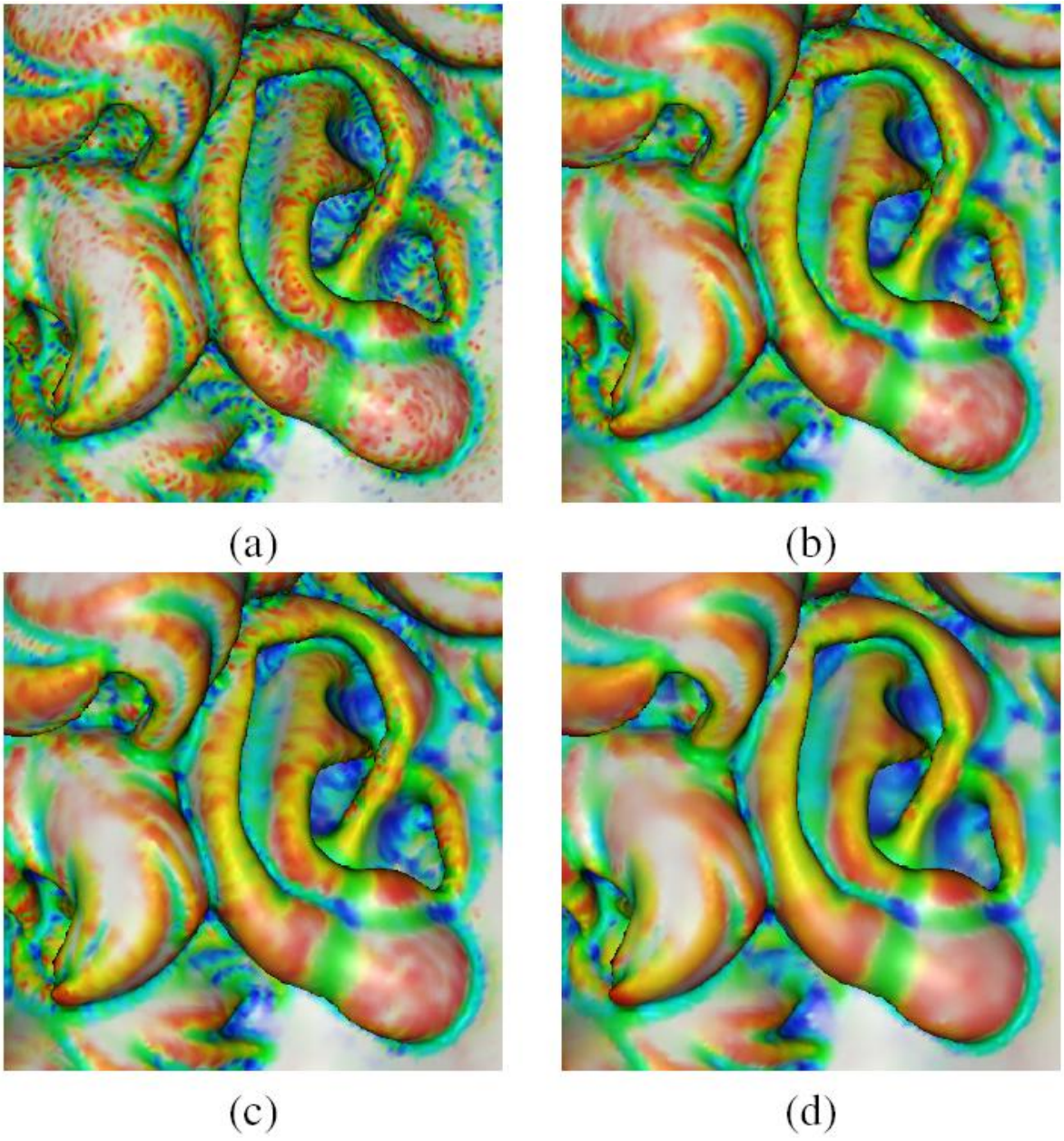

Figure 6.58: curvature field visualization on David's ear: a) Steiner's and Morvan's curvature estimation b) Goldfeather and Interrante's approach c) Rusinkiewicz's approximation d) our M-estimation technique

6.5 Results for the updated normal sections method

In the section 3.2 of chapter 3, we presented an updated version of the algorithm with parabola fitting on normal sections. This method seems a bit more accurate in some cases than the one-ring methods, but it can easily become unstable like them (see figure 6.59).

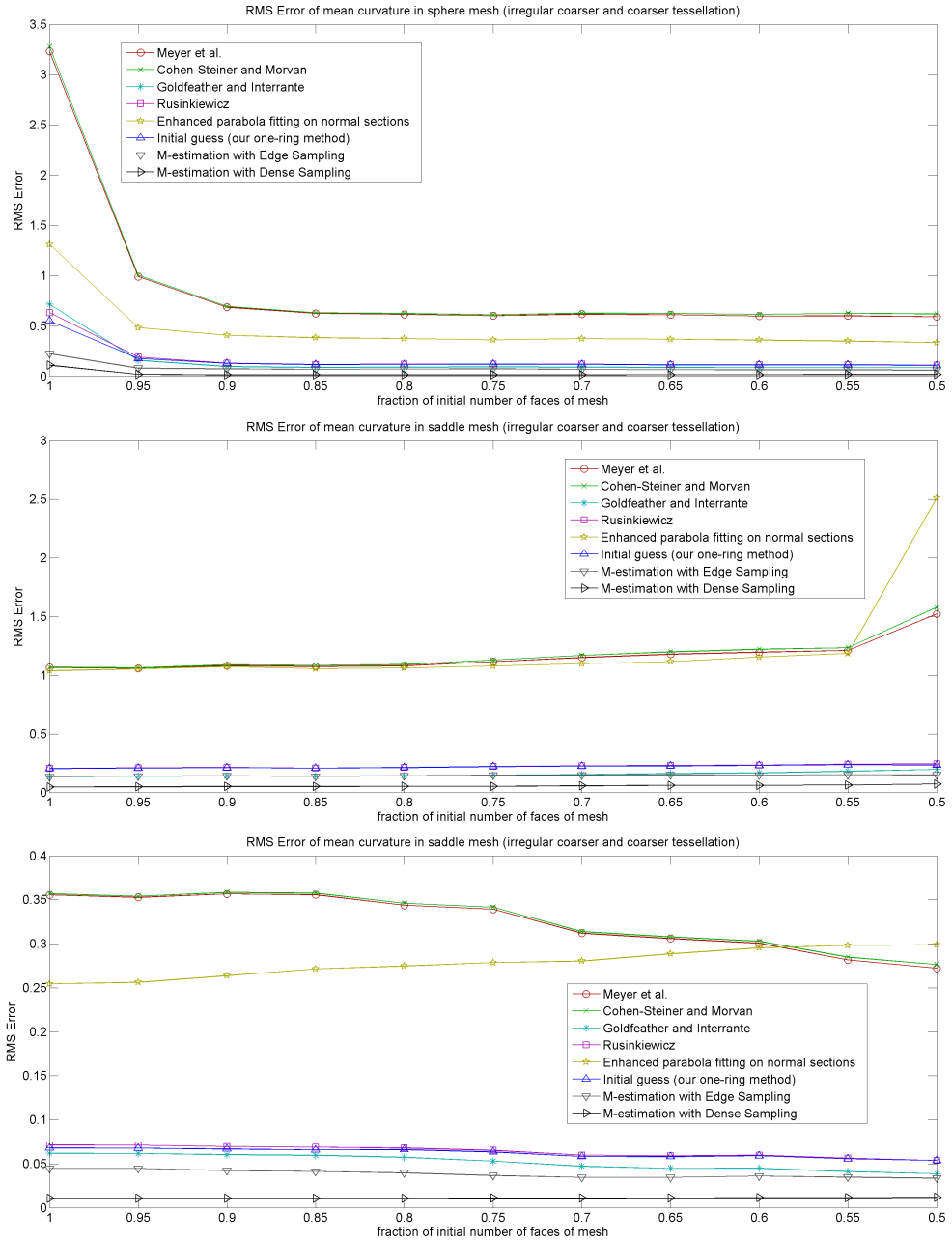

Figure 6.59: Some plots of RMS Error including the normal section fitting method

Chapter 7

Conclusion and future work

In this thesis, we have presented a robust statistical approach for curvature estimation in discretized surfaces. Firstly, in chapter 4, we showed a fast one-ring neighborhood approach, whose framework was extended in order to develop the robust statistics algorithm (chapter 5), based on the M-estimation literature. This algorithm is capable of robustly estimating the curvature tensor, by fitting a linear model to the normal variations samples in varying regions around each point of interest. In the vast majority of the test cases, presented in the previous chapter, which included noisy, irregular or non-uniformly sampled surfaces or a combination of these bad conditions, as well as real-world meshes, our M-estimation method exhibited the best accuracy and stability compared to existing methods in the literature.

We believe that our contribution is two-fold: firstly, the M-estimation robust statistics approach that is able to handle structured outliers and noise that other existing methods cannot, while at the same time, it can automatically converge to an anisotropic area for an accurate and stable curvature estimation with minimum user intervention. Secondly, we showed that sampling normal variations beyond the existing edges, in the cases of polygon meshes, produces more stable and accurate results, than relying only on the existing edges, especially, when the meshes

are highly noisy and irregular.

We acknowledge the fact that our method is computationally more expensive than the existing approaches in the literature until now, due to its iterative local optimization nature. Note, however, that its overall complexity remains linear in the number of points or vertices of the discrete surface. We also notice that curvature estimation is an off-line operation for many applications, like non-photorealistic rendering and shape analysis. Also, other applications, like de-noising, symmetry detection, segmentation etc, that usually require an accurate curvature field, that must be as consistent as possible, are not real-time dependent. On the other hand, another last contribution of ours is the development of a fast fixed region approach for curvature estimation, which can very quickly produce quite decent results, similar or a bit better than the other fixed region approaches. Moreover, when time constraints become important in an application, the M-estimation approach can be accelerated by limiting the number of maximum iterations (e.g. 30 or 20), as we have shown that convergence to a local minima of the objective function is achieved relatively quickly after a few iterations. In addition, limiting the normal variation sampling to existing edges, can be three or four times faster, and still often produces much better results than the other existing methods.

An interesting area of future research, is the study of the theoretical properties of convergence of such statistical methods for curvature estimation. More specifically, the choice of the theoretically most appropriate geometric weights on the samples is an open issue. Another question is if and how the minimization of the residual cost function guarantees convergence to the true estimate of the curvature tensor at each point of interest on the surface.

Furthermore, another intriguing research topic would be how the M-estimation approach can be extended in other problems of geometric optimization, like robust normal computation. For example, the calculated M-estimation weights of the normal variation samples reveal which edges and points are considered inliers or outliers in the estimation of the differential properties at a surface point. Thus, these weights could also be used to recalculate and correct the normals

of the discretized surface at each point, preserving the fine features of the normal field. The M-estimation IRLS process could potentially be used for feature-preserving surface smoothing and reconstruction.

Other areas of future work include the development of new applications or the refinement of existing applications that will be highly benefited from our robust curvature estimation algorithm. For example, applications like automatic mesh segmentation, non-photorealistic rendering and local symmetry detection and surface registration could highly be improved by our algorithm as they rely on robust curvature computation. In general, we believe that the M-estimation framework, that was presented in this thesis, can open interesting areas of research, concerning robust estimates of properties in discretized surfaces, used in geometric applications, that involve optimization problems like automatic rejection of noise and structured outliers.

Bibliography

[1] Jain A.K. and P.J. Flynn. Error bounds and optimal neighborhoods for mls approximation. In *Computer Vision and Pattern Recognition, CVPR '89*, pages 110–116, 1989.

[2] Pierre Alliez, David Cohen-Steiner, Olivier Devillers, Bruno Lévy, and Mathieu Desbrun. Anisotropic polygonal remeshing. *ACM Trans. Graph.*, 22(3):485–493, 2003.

[3] David Baraff and Andrew Witkin. Large steps in cloth simulation. In *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 43–54, New York, NY, USA, 1998. ACM Press.

[4] P.J. Besl, J.B. Birch, and L.T. Watson. Robust window operators. In *Second International Conference on Computer Vision (ICCV 1988)*, pages 591–600, Washington, USA, 1988. IEEE Computer Society.

[5] M. J. Black and P. Anandan. The robust estimation of multiple motions: parametric and piecewise-smooth flow fields. *Computer Vision and Image Understanding*, 63(1):75–104, 1996.

[6] R. Bridson, S. Marino, and R. Fedkiw. Simulation of clothing with folds and wrinkles. In *SCA '03: Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 28–36. Eurographics Association, 2003.

[7] R.J. Carroll and D. Ruppert. *Transformation and Weighting in Regression*. Chapman and Hall, New York, NY, USA, 1988.

[8] F. Cazals and M. Pouget. Estimating differential quantities using polynomial fitting of osculating jets. In *SGP '03: Proceedings of the 2003 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, pages 177–187. Eurographics Association, 2003.

[9] Xin Chen and Francis Schmitt. Intrinsic surface properties from surface triangulation. In *ECCV '92: Proceedings of the Second European Conference on Computer Vision*, pages 739–743, London, UK, 1992. Springer-Verlag.

[10] David Cohen-Steiner and Jean-Marie Morvan. Restricted delaunay triangulations and normal cycle. In *SCG '03: Proceedings of the nineteenth annual symposium on Computational geometry*, pages 312–321, 2003.

[11] Doug DeCarlo, Adam Finkelstein, Szymon Rusinkiewicz, and Anthony Santella. Suggestive contours for conveying shape. *ACM Trans. Graph.*, 22(3):848–855, 2003.

[12] Mathieu Desbrun, Mark Meyer, Peter Schroder, and Alan H. Barr. Implicit fairing of irregular meshes using diffusion and curvature flow. In *SIGGRAPH '99: Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, pages 317–324, New York, NY, USA, 1999. ACM Press/Addison-Wesley Publishing Co.

[13] Ting Jun Fan, Gerard Medioni, and Ramakant Nevatia. Recognizing 3d objects using surface descriptions. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 11(11):1140–1157, 1989.

[14] S. Fleishman, M. Alexa, D. Cohen-Or, and C.T. Silva. Progressive point set surfaces. *ACM Transactions on Graphics*, 22:997–1011, 2003.

[15] Shachar Fleishman, Daniel Cohen-Or, and Claudio T. Silva. Robust moving least-squares fitting with sharp features. *ACM Trans. Graph.*, 24(3):544–552, 2005.

[16] D. A. Forsyth and J. Ponce. *Computer Vision: A Modern Approach*. Prentice Hall, first edition, 2002.

[17] Timothy Gatzke and Cindy Grimm. Estimating curvature on triangular meshes. *International Journal of Shape Modeling*, 12(1), 2006.

[18] Timothy Gatzke, Steve Zelinka, Cindy Grimm, and Michael Garland. Curvature maps for local shape comparison. In *Shape Modeling International*, pages 244–256, June 2005.

[19] J. Goldfeather and V. Interrante. A novel cubic-order algorithm for approximating principal direction vectors. *ACM Transactions on Graphics*, 23(1), 2004.

[20] Gabriele Gorla, Victoria Interrante, and Guillermo Sapiro. Texture synthesis for 3d shape representation. *IEEE Transactions on Visualization and Computer Graphics*, 9(4):512–524, October-December 2003.

[21] Eitan Grinspun, Yotam Gingold, Jason Reisman, and Denis Zorin. Computing discrete shape operators on general meshes. *Eurographics (Computer Graphics Forum)*, 25(3), 2006.

[22] Eitan Grinspun, Anil N. Hirani, Mathieu Desbrun, and Peter Schröder. Discrete shells. In *SCA '03: Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on Computer animation*, pages 62–67, Aire-la-Ville, Switzerland, Switzerland, 2003. Eurographics Association.

[23] Xianfeng Gu, Steven J. Gortler, and Hugues Hoppe. Geometry images. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 355–361, 2002.

[24] B. Hamann. Curvature approximation for triangulated surfaces. *Geometric modelling*, pages 139–153, 1993.

[25] Eyal Hameiri and Ilan Shimshoni. Estimating the principal curvatures and the darboux frame from real 3d range data. *3dpvt*, 00:258, 2002.

[26] F. R. Hampel, E. M. Ronchetti, P. J. Rousseeuw, and W. A. Stahel. *Robust Statistics: The Approach Based on Influence Functions*. Wiley-Interscience, New York, USA, 1986.

[27] Paul S. Heckbert and Michael Garland. Optimal triangulation and quadric-based surface simplification. *Comput. Geom. Theory Appl.*, 14(1-3):49–65, 1999.

[28] Aaron Hertzmann and Denis Zorin. Illustrating smooth surfaces. In *SIGGRAPH '00: Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 517–526, 2000.

[29] Klaus Hildebrandt and Konrad Polthier. Anisotropic filtering of non-linear surface features. In *Eurographics 2004*, volume 23, pages 04–25, 2004.

[30] Hugues Hoppe, Tony DeRose, Tom Duchamp, John McDonald, and Werner Stuetzle. Surface reconstruction from unorganized points. In *SIGGRAPH '92: Proceedings of the 19th annual conference on Computer graphics and interactive techniques*, pages 71–78, New York, NY, USA, 1992. ACM Press.

[31] Qi-Xing Huang, Simon Flöry, Natasha Gelfand, Michael Hofer, and Helmut Pottmann. Reassembling fractured objects by geometric matching. *ACM Trans. Graph.*, 25(3):569–578, 2006.

[32] P. J. Huber. *Robust Statistics*. Wiley-Interscience editions, New York, USA, 1981.

[33] Victoria Interrante. Illustrating surface shape in volume data via principal direction-driven 3d line integral convolution. In *SIGGRAPH '97: Proceedings of the 24th annual confer-*

ence on Computer graphics and interactive techniques, pages 109–116, 1997.

[34] Victoria Interrante, Henry Fuchs, and Stephen Pizer. Enhancing transparent skin surfaces with ridge and valley lines. In *VIS '95: Proceedings of the 6th conference on Visualization '95*, page 52, 1995.

[35] S. Katz and A. Tal. Hierarchical mesh decomposition using fuzzy clustering and cuts. *ACM Transactions on Graphics*, 22(3):954–961, 2003.

[36] Torsten Langer, Alexander Belyaev, and Hans Peiter Seidel. Exact and interpolary quadratures for curvature tensor estimation. *To appear in the Special Issue of Discrete Differential Geometry, Computer Aided Geometric Design*, 2007.

[37] Guillaume Lavoue, Florent Dupont, and Atilla Baskurt. A new cad mesh segmentation method, based on curvature tensor analysis. *Computer-Aided Design (CAD)*, 37(10):975–987, 2005.

[38] R. Lengagne, P. Fua, and O. Monga. Using crest lines to guide surface reconstruction from stereo. In *ICPR '96: Proceedings of the 1996 International Conference on Pattern Recognition (ICPR '96) Volume I*, page 9, Washington, DC, USA, 1996. IEEE Computer Society.

[39] Bruno Lévy, Sylvain Petitjean, Nicolas Ray, and Jérome Maillot. Least squares conformal maps for automatic texture atlas generation. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 362–371, New York, NY, USA, 2002. ACM Press.

[40] A. Leyton and M. Leyton. 3d symmetry-curvature duality theorems. *Comput. Vision Graph. Image Process.*, 52(1):124–140, 1990.

[41] CGAL Libary. Computational geometry algorithms library. http://www.cgal.org/.

[42] Yaron Lipman, Daniel Cohen-Or, and David Levin. Error bounds and optimal neighborhoods for mls approximation. In *SGP '06: Proceedings of the 2006 Eurographics/ACM SIGGRAPH symposium on Geometry processing*, pages 71–80. Eurographics Association, 2006.

[43] L. Liu, B. G. Schunck, and C. C. Meyer. On robust edge detection. In *International Workshop on Robust Computer Vision*, pages 261–286, 1990.

[44] Yang Liu, Helmut Pottmann, Johannes Wallner, Yong-Liang Yang, and Wenping Wang. Geometric modeling with conical meshes and developable surfaces. *ACM Trans. Graph.*, 25(3):681–689, 2006.

[45] Alan P. Mangan and Ross T. Whitaker. Partitioning 3d surface meshes using watershed segmentation. *IEEE Transactions on Visualization and Computer Graphics*, 5(4):308–321, October-December 1999.

[46] Martin Marinov and Leif Kobbelt. Direct anisotropic quad-dominant remeshing. In *PG '04: Proceedings of the Computer Graphics and Applications, 12th Pacific Conference on (PG'04)*, pages 207–216, Washington, DC, USA, 2004. IEEE Computer Society.

[47] Nelson Max. Weights for computing vertex normals from facet normals. *Journal of Graphics Tools*, 4(2):1–6, 1999.

[48] D.S. Meek and D.J. Walton. On surface normal and gaussian curvature approximations given data sampled from a smooth surface. *Computer Aided Geometric Design*, 17:521–543, February 2000.

[49] Mark Meyer, Mathieu Desbrun, Peter Schröder, and Alan H. Barr. Discrete differential-geometry operators for triangulated 2-manifolds. In Hans-Christian Hege and Konrad Polthier, editors, *Visualization and Mathematics III*, pages 35–57. Springer-Verlag, Heidelberg, 2003.

[50] N. J. Mitra and A. Nguyen. Estimating surface normals in noisy point cloud data. In *Symposium on COmputational Geometry*, pages 322–328, 2003.

[51] Niloy J. Mitra, Leonidas J. Guibas, and Mark Pauly. Partial and approximate symmetry detection for 3d geometry. *ACM Trans. Graph.*, 25(3):560–568, 2006.

[52] University of Tennessee. Linear algebra package. http://www.netlib.org/lapack/.

[53] Yutaka Ohtake, Alexander Belyaev, and Hans-Peter Seidel. Ridge-valley lines on meshes via implicit surface fitting. *ACM Trans. Graph.*, 23(3):609–612, 2004.

[54] Barrett O'Neill. *Elementary Differential Geometry*. Academic Press, 1966.

[55] D. L. Page, A. F. Koschan, and M. A. Abidi. Perception-based 3d triangle mesh segmentation using fast marching watersheds. In *IEEE Computer Society Conference on Computer Vision and Pattern Recognition - CVPR '03*, volume 2, pages 27–32, June 16-22 2003.

[56] Helmut Pottmann, Qi-Xing Huang, Yong-Liang Yang, and Stephan Kolpl. Integral invariants for robust geometry processing. Technical Report 146, Geometry Preprint, TU Wien, 2005.

[57] P. Preisig and D. Kragic. Robust statistics for 3d object tracking. In *Proceedings 2006 IEEE International Conference on Robotics and Automation, 2006. ICRA 2006.*, pages 2403–2408, Washington, USA, 2006. IEEE Computer Society.

[58] Szymon Rusinkiewicz. Trimesh library. http://www.cs.princeton.edu/gfx/proj/trimesh2/.

[59] Szymon Rusinkiewicz. Estimating curvatures and their derivatives on triangle meshes. *3D Data Processing, Visualization, and Transmission, 2nd International Symposium on (3DPVT'04)*, pages 486–493, 2004.

[60] H. S. Sawhney, S. Ayer, and M. Gorkani. Model-based 2d&3d dominant motion estimation for mosaicing and video representation. In *Proceedings of the Fifth International Conference on Computer Vision*, page 583, Washington, USA, 1995. IEEE Computer Society.

[61] R. Schmidt, B. Wyvill, M. C. Sousa, and J. A. Jorge. Shapeshop: sketch-based solid modeling with blobtrees. In *SIGGRAPH '06: ACM SIGGRAPH 2006 Courses*, page 14, New York, NY, USA, 2006. ACM Press.

[62] Patricio Simari, Evangelos Kalogerakis, and Karan Singh. Folding meshes: Hierarchical mesh segmentation based on planar symmetry. In *Proceedings of the Symposium on Geometry Processing (SGP '06)*, pages 111–119, June 2006.

[63] Patricio Simari, Karan Singh, and Hans Petersen. Spider: A robust curvature estimator for noisy, irregular meshes. Technical Report CSRG-531, DGP, Department of Computer Science, University of Toronto, 2005.

[64] M. Spivak. *A Comprehensive Introduction to Differential Geometry*. Berkeley, CA: Publish or Perish Press, 1979.

[65] C. V. Stewart. Robust parameter estimation in computer vision. *SIAM Rev.*, 41(3):513–537, 1999.

[66] Ernest M. Stokely and Shang You Wu. Surface parametrization and curvature measurement of arbitrary 3-d objects: Five practical methods. *IEEE Trans. Pattern Anal. Mach. Intell.*, 14(8):833–840, 1992.

[67] G. Taubin. Estimating the tensor of curvature of a surface from a polyhedral approximation. In *ICCV '95: Proceedings of the Fifth International Conference on Computer Vision*, page 902, Washington, DC, USA, 1995. IEEE Computer Society.

[68] Demetri Terzopoulos, John Platt, Alan Barr, and Kurt Fleischer. Elastically deformable models. In *SIGGRAPH '87: Proceedings of the 14th annual conference on Computer graphics and interactive techniques*, pages 205–214, New York, NY, USA, 1987. ACM Press.

[69] Holger Theisel, Christian Rossl, Rhaleb Zayer, and Hans-Peter Seidel. Normal based estimation of the curvature tensor for triangular meshes. In *PG '04: Proceedings of the Computer Graphics and Applications, 12th Pacific Conference on (PG'04)*, pages 288–297, Washington, DC, USA, 2004. IEEE Computer Society.

[70] Grit Thurmer and Charles Wuthrich. Computing vertex normals from polygonal facets. *Journal of Graphics Tools*, 3(1):43–46, 1998.

[71] Wai-Shun Tong and Chi-Keung Tang. Robust estimation of adaptive tensors of curvature by tensor voting. *IEEE Trans. Pattern Anal. Mach. Intell.*, 27(3):434–449, 2005.

[72] P. Torr and D. Murray. The development and comparison of robust methods for estimating the fundamental matrix. *Intl. J. of Computer Vision*, 24(3):271–300, 1997.

[73] Emanuele Trucco and Robert B. Fisher. Experiments in curvature-based segmentation of range data. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 17(2):177–182, February 1995.

[74] William Welch and Andrew Witkin. Free-form shape design using triangulated surfaces. In *SIGGRAPH '94: Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, pages 247–256, New York, NY, USA, 1994. ACM Press.

[75] Guoliang Xu. Discrete laplace-beltrami operators and their convergence. *Comput. Aided Geom. Des.*, 21(8):767–784, 2004.

[76] Hitoshi Yamauchi, Stefan Gumhold, Rhaleb Zayer, and Hans-Peter Seidel. Mesh seg-

mentation driven by gaussian curvature. In *Pacific Graphics 2005*, volume 21, pages 649–658, October 2005.

Figure 7.1: Visualization of the principal direction vectors, estimated by our M-estimation method, on David's head