Refining Procedures on Mesh via Algebraic Fitting

Tibor Stanko*
Supervised by: doc. RNDr. Pavel Chalmovianský, PhD.†

Faculty of Mathematics, Physics and Informatics
Comenius University
Bratislava / Slovakia

Abstract

A new nonlinear refinement algorithm for surfaces is presented in this work. Our scheme operates on triangular meshes and interpolates input data. Each triangle is associated with a small set of neighbouring points and normals. A low degree algebraic surface (quadric) is fitted to this set with respect to the chosen objective function. The new vertex is taken from the computed quadric. Such a setup overcomes the limitations of the linear schemes. Our experiments show the scheme might be capable of reconstructing quadratic surfaces from a coarse approximating mesh. A comparison of the proposed method with the linear schemes is shown, as well as an application to the compression of a large-scale mesh.

Keywords: mesh refinement, subdivision surface, nonlinear scheme, quadric, mesh compression

1 Introduction

The problem of efficient and accurate geometry modelling of solids has been present in computer graphics from the very beginning. A new approach for boundary representation of three-dimensional objects has emerged in the late 1970s. What became known as *subdivision surfaces* is now widely used in domains such as CAGD, geometry modelling for animation, level-of-detail modelling, multiresolution analysis. For more details on subdivision surfaces and related work, see sections 2 and 3.

A novel nonlinear scheme is proposed in our paper. Even though the scheme is nonlinear, it only requires solving a well-formed system of linear equations for each triangle of the subdivided mesh. For details on the method and the implementation, see sections 4 and 5.

In section 6, we experiment with various sets of weights and analyse the influence of the normal vectors on the limit surface generated by our method. We also show how the resulting method can be used to compress triangular mesh obtained from laser scanning. Such a mesh usually consists of large datasets, typically $\sim 10^5 - 10^6$ vertices. Applying our scheme on properly chosen decimation of input mesh, we are able to reconstruct scanned data very accurately.

The proposed scheme can also be used for the reconstruction of quadratic surfaces from a coarse approximating mesh. We provide a demonstration by reconstructing the sphere from the cube.

2 Subdivision Surfaces

Subdivision is a way of representing smooth shapes in computer [1]. The basic idea of subdivision is to define surface $\mathcal{S}$ as a limit of iterative refinement of mesh

$$\mathcal{S} = \lim_{k\to\infty} \mathcal{M}^k, \tag{1}$$

where the mesh $\mathcal{M}^{k+1}$ is obtained by applying set of refinement rules on the mesh $\mathcal{M}^k$, the mesh $\mathcal{M}^0$ is initial.

A subdivision scheme is *interpolating* if the limit surface interpolates the vertices of the initial mesh. Otherwise, the scheme is *approximating*.

Each mesh $\mathcal{M}$ consists of the topological component (vertices, edges, faces) and the geometric component (vertex positions in $\mathbb{R}^3$). Likewise, every subdivision scheme has *topological* step and *geometric* step. In the topological step, the topology of the mesh in the next iteration is determined. New vertices, edges and faces are inserted, and some of the old ones are removed. Typical operations in this step include inserting new vertices and leaving out some of the old, introducing new edges and faces, flipping an edge. In the geometric step, the new positions of the vertices are computed.

Linear schemes use linear combinations of the vertices from the previous iteration to compute the new positions. Consequently, vertices in the arbitrary iteration $\mathcal{M}^k$ (particularly the limit surface $\mathcal{S} = \mathcal{M}^\infty$) can be expressed as linear combinations of initial mesh $\mathcal{M}^0$ in a natural way. For *nonlinear schemes*, this condition does not hold true.

3 Related Work

Early work on the linear refinement of triangular meshes has been done by Loop [2], who designed an approximating scheme. An interpolating scheme was proposed by

*ts@tiborstanko.sk
†pavel.chalmoviansky@fmph.uniba.sk

Dyn et al. [3] and later modified by Zorin et al. [4]. Another approximating scheme was proposed by Kobbelt [5]. For an overview of existing linear schemes, see [6] or [7]. Extensive bibliography on the topic can be found in the latter.

While linear methods for surface refinement have been closely studied in the past decades, nonlinear methods have received little attention. Linear schemes work well in cases when only the positions of vertices are known. The difficulties arise when we try to make use of data coming from derivatives, such as tangent or curvature. Nonlinear schemes seem to be the right mechanism to bridge this gap.

Only a few nonlinear schemes for surface refinement have been introduced so far. Interpolating triangular algorithms were proposed in [8], [9], [10]. The scheme presented in this paper was inspired by the work of Chalmoviansk\'y and Jüttler [11], who introduced a nonlinear circle-preserving algorithm for *curve refinement*.

4 Refinement via Algebraic Fitting

In this paper, we introduce a different approach to nonlinear subdivision of triangular meshes. The basic idea of the proposed method is to look for the new vertices on the quadric surface, which is the best local approximation of the mesh with respect to the chosen objective function. In the text, we talk about *quadric fitting refinement* or simply QFR when referencing our method.

4.1 Topological step

The quadric fitting refinement uses the topological step introduced by Kobbelt [5]. A new vertex is introduced per triangle face and connected to all vertices of the triangle. Old edges are flipped. Figure 1 shows the topological step on the regular grid for better illustration.

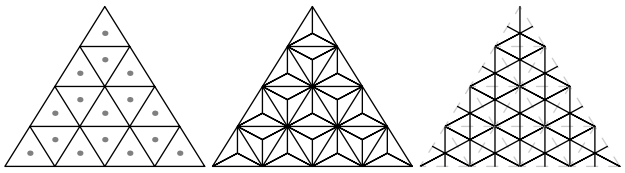


Figure 1: The topological step of Kobbelt's $\sqrt{3}$-subdivision used in our scheme.

4.2 Computing position of the new vertex

Since the proposed scheme is interpolating, the positions of the old vertices remain unchanged. Therefore, the focus of the scheme lies in the computation of the position of the new vertex.

Suppose we want to subdivide the mesh $\mathcal{M}$. Let $\mathcal{V}(\mathcal{M})$ be the set of all vertices of $\mathcal{M}$. We are looking for the position of the new vertex $\mathbf{v}$ introduced in the triangle $T = \mathbf{v}_0\mathbf{v}_1\mathbf{v}_2$. The set $\mathcal{N}_T$ of vertices is called the $m-$neighbourhood of T for some $m \in \mathbb{N}$ if

$$\mathcal{N}_\mathrm{T} \coloneqq \{\mathbf{p} \in \mathcal{V}(\mathcal{M}) \,:\, \mathcal{D}_\mathrm{T}(\mathbf{p}) \leq m\} \tag{2}$$

for some $m \in \mathbb{N}$, where

$$\mathcal{D}_\mathrm{T}(\mathbf{p}) \coloneqq \min_{\tilde{\mathbf{v}} \in \mathcal{V}(\mathrm{T})} \left(\mathcal{D}(\mathbf{p}, \tilde{\mathbf{v}})\right) \tag{3}$$

is a relative distance of vertex $\mathbf{p}$ from the triangle T, $\mathcal{D}(\mathbf{x},\mathbf{y})$ is a graph distance on $\mathcal{M}$ (number of edges in the shortest path connecting $\mathbf{x}$ and $\mathbf{y}$). We choose m to be the smallest natural number, for which $|\mathcal{N}_T| \geq 9$. Typically, this yields $m = 1$ or $m = 2$. The choice of 9 as the minimal cardinality is justified later in the text (in section 4.3). An example of 1-neighbourhood is shown in figure 2.

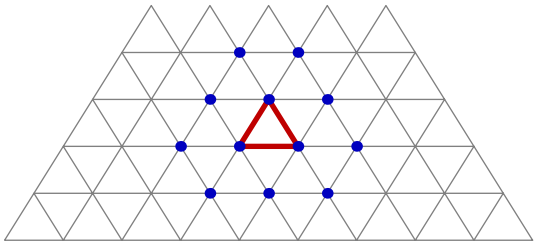


Figure 2: Visualisation of the set $\mathcal{N}_T$ (1-*neighbourhood*) for the triangle T on the regular grid.

The vertex $\mathbf{v}$ is picked out of the quadric surface

$$\mathcal{Q} \coloneqq \left\{(x,y,z) \in \mathbb{E}^3 \,:\, f(x,y,z) = 0\right\}, \tag{4}$$

where

$$\begin{aligned} f(x,y,z) = a_{11}x^2 + a_{22}y^2 + a_{33}z^2 + 2a_{12}xy + 2a_{13}xz + \\ + 2a_{23}yz + 2a_{14}x + 2a_{24}y + 2a_{34}z + a_{44} \end{aligned} \tag{5}$$

is an unknown trivariate polynomial with real coefficients. Using matrix notation, the equation (5) is written down to

$$f(\mathbf{x}) = \tilde{\mathbf{x}}^\top \mathbf{A}\, \tilde{\mathbf{x}}, \tag{6}$$

where

$$\tilde{\mathbf{x}} \coloneqq \begin{pmatrix} x \\ y \\ z \\ 1 \end{pmatrix}, \; \mathbf{A} \coloneqq \begin{pmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{12} & a_{22} & a_{23} & a_{24} \\ a_{13} & a_{23} & a_{33} & a_{34} \\ a_{14} & a_{24} & a_{34} & a_{44} \end{pmatrix},$$

$\tilde{\mathbf{x}}$ stands for the homogenous coordinates of $\mathbf{x} \in \mathbb{E}^3$, $\mathbf{A}$ denotes the symmetric matrix of the coefficients a_{ij} from the equation (5).

4.3 Fitting quadric to vertices

We look for such a quadric $\mathcal{Q}$ that is the best approximation of the mesh $\mathcal{M}$ in a close neighbourhood of the

triangle T. For this purpose, we use the set $\mathcal{N}_\mathrm{T}$ defined in (2). In order to specify $\mathcal{Q}$, exactly 10 unknown coefficients $\{a_{ij},\, 1 \le i \le j \le 4\}$ need to be computed up to a non-zero multiple. This clarifies the required condition on the cardinality of $\mathcal{N}_\mathrm{T}$.

To improve the reading comprehension of this section, we use the following notation for gradient operators:

$$\nabla^\mathrm{a} f = \begin{pmatrix} \frac{\partial f}{\partial a_{11}} & \frac{\partial f}{\partial a_{22}} & \frac{\partial f}{\partial a_{33}} & \frac{\partial f}{\partial a_{12}} & \cdots & \frac{\partial f}{\partial a_{44}} \end{pmatrix} \tag{7}$$

denotes gradient with respect to variables $a_{11},\ldots,a_{44}$, while

$$\nabla^\mathrm{x} f = \begin{pmatrix} \frac{\partial f}{\partial x} & \frac{\partial f}{\partial y} & \frac{\partial f}{\partial z} \end{pmatrix} \tag{8}$$

is the gradient with respect to x,y,z.

Now, suppose $\mathcal{N}_\mathrm{T} = \{\mathbf{p}_i\,,\, i=1,\ldots,n\}$, where $\mathbf{p}_i = (x_i,y_i,z_i)$. Ideally, $\mathcal{Q}$ interpolates $\mathcal{N}_\mathrm{T}$, meaning f vanishes in every point from $\mathcal{N}_\mathrm{T}$. In general case though, such an interpolation cannot be guaranteed. Instead, we compute the vector

$$\vec{a} \coloneqq \begin{pmatrix} a_{11} & \cdots & a_{44} \end{pmatrix}^\top \tag{9}$$

of the unknown parameters such that the *objective function*

$$\mathcal{F}(a_{11},\ldots,a_{44}) = \sum_{i=1}^{n} \tilde{\mathrm{w}}_\mathrm{i}\, f^2(\mathbf{p}_i) + \hat{\mathrm{w}}_\mathrm{i}\, \|\nabla^\mathrm{x} f(\mathbf{p}_i) - \vec{\mathbf{n}}_i\|^2 \tag{10}$$

is minimized and

$$\vec{a}_\mathrm{min} = \underset{a_{11},\ldots,a_{44}}{\arg\min}\, \mathcal{F}(a_{11},\ldots,a_{44})\,. \tag{11}$$

In (10), the vector $\vec{\mathbf{n}}_i = (\hat{x}_i,\hat{y}_i,\hat{z}_i)^\top$ denotes the normal vector of $\mathcal{M}$ at the vertex $\mathbf{p}_i$. The scalars $\tilde{\mathrm{w}}_\mathrm{i}, \hat{\mathrm{w}}_\mathrm{i} > 0$ are the associated real weights, which are specified later in the text (in section 6.1).

The necessary conditions for minima of the function $\mathcal{F}$ give

$$\nabla^\mathrm{a} \mathcal{F}(\vec{a}) = \vec{0}, \tag{12}$$

a system of linear equations

$$\frac{\partial \mathcal{F}}{\partial a_{ij}}(a_{11},\ldots,a_{44}) = 0, \quad 1 \le i \le j \le 4. \tag{13}$$

If we denote

$$\tilde{\mathcal{F}}(\vec{a}) = \sum_{i=1}^{n} \tilde{\mathrm{w}}_\mathrm{i}\, f^2(\mathbf{p}_i)\,, \tag{14}$$

$$\hat{\mathcal{F}}(\vec{a}) = \sum_{i=1}^{n} \hat{\mathrm{w}}_\mathrm{i}\, \|\nabla^\mathrm{x} f(\mathbf{p}_i) - \vec{\mathbf{n}}_i\|^2\,, \tag{15}$$

then $\mathcal{F} = \tilde{\mathcal{F}} + \hat{\mathcal{F}}$. Therefore,

$$\nabla^\mathrm{a} \mathcal{F} = \nabla^\mathrm{a} \tilde{\mathcal{F}} + \nabla^\mathrm{a} \hat{\mathcal{F}}\,. \tag{16}$$

Every vertex $\mathbf{p}_i$ contributes to the objective function in two parts. The function $\tilde{\mathcal{F}}$ measures the distance of $\mathbf{p}_i$ to the quadric $\mathcal{Q}$, weighted by $\tilde{\mathrm{w}}_\mathrm{i}$. The function $\hat{\mathcal{F}}$ measures the deviation of the computed normal from the prescribed normal (at $\mathbf{p}_i$), weighted by $\hat{\mathrm{w}}_\mathrm{i}$.

Let us have a look at the first term on the right side of (16). Denote $\phi_i \coloneqq \phi(\mathbf{p}_i)$, where

$$\phi(\mathbf{p}_i) \coloneqq (\nabla^\mathrm{a} f)(\mathbf{p}_i) = \begin{pmatrix} {x_i}^2 & {y_i}^2 & \cdots & 2z_i & 1 \end{pmatrix}^\top. \tag{17}$$

Using (17) and the fact that $f(\mathbf{p}_i) = \phi_i^\top \vec{a}$, we get

$$\begin{aligned} \nabla^\mathrm{a} \tilde{\mathcal{F}} &= \sum_{i=1}^{n} \tilde{\mathrm{w}}_\mathrm{i}\, 2\,(\nabla^\mathrm{a} f)(\mathbf{p}_i)\, f(\mathbf{p}_i) = \\ &= 2\sum_{i=1}^{n} \tilde{\mathrm{w}}_\mathrm{i}\, \phi_i\, \phi_i^\top \vec{a} = 2\sum_{i=1}^{n} \tilde{\mathrm{w}}_\mathrm{i}\, \Phi_i\, \vec{a} = 2\,\Phi\,\vec{a}. \end{aligned} \tag{18}$$

Here, we have used the notation $\Phi_i \coloneqq \phi_i\phi_i^\top$ and $\Phi \coloneqq \sum_{i=1}^{n} \tilde{\mathrm{w}}_\mathrm{i}\, \Phi_i$ for the corresponding matrices.

Now, we analyse the second term on the right side of (16). First, the gradient of f with respect to x,y,z is computed

$$\nabla^\mathrm{x} f(\mathbf{p}_i) = 2\begin{pmatrix} a_{11}x_i + a_{12}y_i + a_{13}z_i + a_{14} \\ a_{12}x_i + a_{22}y_i + a_{23}z_i + a_{24} \\ a_{13}x_i + a_{23}y_i + a_{33}z_i + a_{34} \end{pmatrix} =: \begin{pmatrix} \alpha_i \\ \beta_i \\ \gamma_i \end{pmatrix}, \tag{19}$$

where $\alpha_i,\beta_i,\gamma_i$ are dependent on $\vec{a}$. Recall the coordinates of normal $\vec{\mathbf{n}}_i$ at the vertex $\mathbf{p}_i$ are $(\hat{x}_i,\hat{y}_i,\hat{z}_i)$. Consequently,

$$\|\nabla^\mathrm{x} f(\mathbf{p}_i) - \vec{\mathbf{n}}_i\|^2 = (\alpha_i - \hat{x}_i)^2 + (\beta_i - \hat{y}_i)^2 + (\gamma_i - \hat{z}_i)^2\,. \tag{20}$$

Applying the gradient operator ∇^a on (20), we have

$$\begin{aligned} &\nabla^\mathrm{a}\left(\|\nabla^\mathrm{x} f(\mathbf{p}_i) - \vec{\mathbf{n}}_i\|^2\right) = \\ &= \nabla^\mathrm{a}\left((\alpha_i - \hat{x}_i)^2 + (\beta_i - \hat{y}_i)^2 + (\gamma_i - \hat{z}_i)^2\right) = \\ &= 2\,(\alpha_i - \hat{x}_i)\,\nabla^\mathrm{a}\alpha_i + 2\,(\beta_i - \hat{y}_i)\,\nabla^\mathrm{a}\beta_i + 2\,(\gamma_i - \hat{z}_i)\,\nabla^\mathrm{a}\gamma_i. \end{aligned} \tag{21}$$

Note that applying ∇^a on α_i,β_i and γ_i, we get the vectors

$$\begin{aligned} \nabla^\mathrm{a}\alpha_i &= 2\begin{pmatrix} x_i & 0 & 0 & y_i & z_i & 0 & 1 & 0 & 0 & 0 \end{pmatrix}^\top, \\ \nabla^\mathrm{a}\beta_i &= 2\begin{pmatrix} 0 & y_i & 0 & x_i & 0 & z_i & 0 & 1 & 0 & 0 \end{pmatrix}^\top, \\ \nabla^\mathrm{a}\gamma_i &= 2\begin{pmatrix} 0 & 0 & z_i & 0 & x_i & y_i & 0 & 0 & 1 & 0 \end{pmatrix}^\top. \end{aligned} \tag{22}$$

Each of these vectors has only four non-zero coordinates. Plugging (22) into (21) and substituting into (15) yields

$$\begin{aligned} \nabla^\mathrm{a}\hat{\mathcal{F}}(\vec{a}) &= \sum_{i=1}^{n} 4\,\hat{\mathrm{w}}_\mathrm{i} \begin{pmatrix} x_i\,(\alpha_i - \hat{x}_i) \\ y_i\,(\beta_i - \hat{y}_i) \\ z_i\,(\gamma_i - \hat{z}_i) \\ (\alpha_i - \hat{x}_i)\,y_i + (\beta_i - \hat{y}_i)\,x_i \\ (\alpha_i - \hat{x}_i)\,z_i + (\gamma_i - \hat{z}_i)\,x_i \\ (\beta_i - \hat{y}_i)\,z_i + (\gamma_i - \hat{z}_i)\,y_i \\ \alpha_i - \hat{x}_i \\ \beta_i - \hat{y}_i \\ \gamma_i - \hat{z}_i \\ 0 \end{pmatrix} \\ &= 4\sum_{i=1}^{n} \hat{\mathrm{w}}_\mathrm{i}\,(2\,\Psi_i\,\vec{a} - \Omega_i)\,, \end{aligned} \tag{23}$$

where

$$\Psi_i := \begin{pmatrix} x_i^2 & 0 & 0 & x_i y_i & x_i z_i & 0 & x_i & 0 & 0 & 0 \\ 0 & y_i^2 & 0 & x_i y_i & 0 & y_i z_i & 0 & y_i & 0 & 0 \\ 0 & 0 & z_i^2 & 0 & x_i z_i & y_i z_i & 0 & 0 & z_i & 0 \\ x_i y_i & x_i y_i & 0 & x_i^2+y_i^2 & y_i z_i & x_i z_i & y_i & x_i & 0 & 0 \\ x_i z_i & 0 & x_i z_i & y_i z_i & x_i^2+z_i^2 & x_i y_i & z_i & 0 & x_i & 0 \\ 0 & y_i z_i & y_i z_i & x_i z_i & x_i y_i & y_i^2+z_i^2 & 0 & z_i & y_i & 0 \\ x_i & 0 & 0 & y_i & z_i & 0 & 1 & 0 & 0 & 0 \\ 0 & y_i & 0 & x_i & 0 & z_i & 0 & 1 & 0 & 0 \\ 0 & 0 & z_i & 0 & x_i & y_i & 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix},$$

$$\Omega_i := \left(x_i\hat{x}_i,\ y_i\hat{y}_i,\ z_i\hat{z}_i,\ x_i\hat{y}_i+y_i\hat{x}_i,\ x_i\hat{z}_i+z_i\hat{x}_i,\ y_i\hat{z}_i+z_i\hat{y}_i,\ \hat{x}_i,\ \hat{y}_i,\ \hat{z}_i,\ 0\right)^\top. \tag{24}$$

Introducing the notation

$$\Psi := \sum_{i=1}^{n} \hat{\mathrm{w}}_\mathrm{i}\, \Psi_i, \quad \Omega := \sum_{i=1}^{n} \hat{\mathrm{w}}_\mathrm{i}\, \Omega_i, \tag{25}$$

the equation (23) becomes

$$\nabla^\mathrm{a} \hat{\mathcal{F}}(\vec{a}) = 8\Psi\vec{a} - 4\,\Omega. \tag{26}$$

Using the equations (12), (18) and (26), we obtain the desired system of linear equations in matrix form

$$\begin{aligned} \nabla^\mathrm{a}\mathcal{F} &= 2\,\Phi\vec{a} + 8\Psi\vec{a} - 4\,\Omega = \\ &= (2\Phi + 8\Psi)\,\vec{a} - 4\,\Omega = \Gamma - 4\Omega = 0, \end{aligned} \tag{27}$$

with its explicit solution

$$\vec{a}_{\min} = 4\,\Gamma^{-1}\,\Omega, \tag{28}$$

provided $\Gamma = 2\Phi + 8\Psi$ is an invertible matrix.

4.4 Picking the new vertex

After the quadric $\mathcal{Q}$ has been found by solving the system (27), we are able to compute the coordinates of the new vertex $\mathbf{v}$. Denote

$$\mathbf{b}_T := \frac{\mathbf{v}_0 + \mathbf{v}_1 + \mathbf{v}_2}{3} \tag{29}$$

to be the barycenter of the triangle T.

4.4.1 Intersection of normal line and quadric

Our first approach is to find the position of $\mathbf{v}$ as the intersection of quadric $\mathcal{Q}$ and the normal line $\bar{n}$ of the triangle T. The line $\bar{n}$ is defined parametrically as

$$\bar{n} \equiv \mathbf{b}_T + t\vec{\mathbf{n}}_T\,,\ t \in \mathbb{R}. \tag{30}$$

The vector $\vec{\mathbf{n}}_T$ is defined as the unit normal of the plane determined by the vertices of T (modulo the vector signum), see fig. 3. Denote the intersection of $\mathcal{Q}$ and $\bar{n}$

$$\mathbf{v}_T = \mathbf{b}_T + t_0\,\vec{\mathbf{n}}_T, \tag{31}$$

or, coordinate-wise,

$$\begin{pmatrix} x_v \\ y_v \\ z_v \end{pmatrix} = \begin{pmatrix} x_b \\ y_b \\ z_b \end{pmatrix} + t_0 \begin{pmatrix} x_n \\ y_n \\ z_n \end{pmatrix} \tag{32}$$

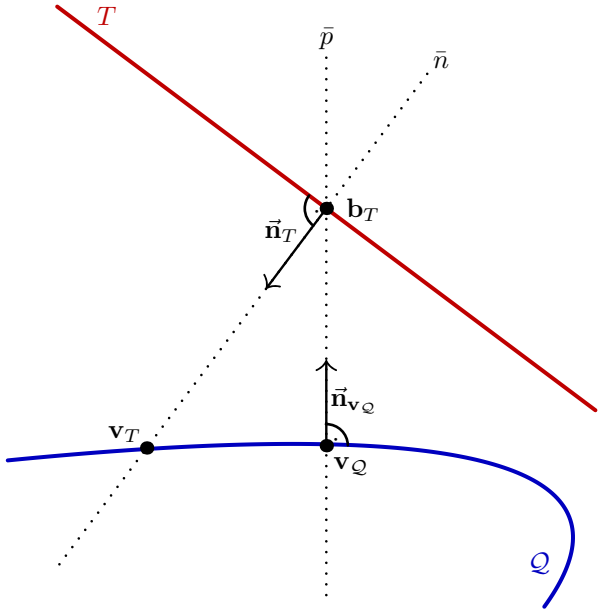


Figure 3: Schematic comparison of the two approaches for picking the new vertex from the quadric $\mathcal{Q}$. The technique described in section 4.4.1 yields $\mathbf{v}_T$, while the technique from 4.4.2 yields $\mathbf{v}_\mathcal{Q}$.

for some t_0. Plugging (31) into (4), (5), we get

$$a_{11}x_v^2 + a_{22}y_v^2 + \cdots + a_{34}z_v + a_{44} = 0. \tag{33}$$

This leads to the quadratic equation in t_0 of the form

$$At_0^2 + 2Bt_0 + C = 0\,, \tag{34}$$

where the coefficients $A, B, C \in \mathbb{R}$ are

$$\begin{aligned} A = {} & x_n\,(a_{11}x_n + a_{12}y_n + a_{13}z_n) + \\ & y_n\,(a_{12}x_n + a_{22}y_n + a_{23}z_n) + \\ & z_n\,(a_{13}x_n + a_{23}y_n + a_{33}z_n)\,, \end{aligned}$$

$$\begin{aligned} B = {} & x_n\,(a_{11}x_b + a_{12}y_b + a_{13}z_b + a_{14}) + \\ & y_n\,(a_{12}x_b + a_{22}y_b + a_{23}z_b + a_{24}) + \\ & z_n\,(a_{13}x_b + a_{23}y_b + a_{33}z_b + a_{34})\,, \end{aligned}$$

$$C = f\,(x_b, y_b, z_b)\,.$$

Denote the roots of (34) as t_1, t_2. The parameter t_0 is chosen as

$$t_0 = \begin{cases} 0, & \text{if } B^2 - 4AC < 0\,; \\ t_1, & \text{if } |t_1| < |t_2|\,; \\ t_2, & \text{otherwise.} \end{cases} \tag{35}$$

If the parameters t_1, t_2 are real, they determine two points on $\bar{n}$. We pick the point which is closer to $\mathbf{b}_T$. If t_1, t_2 are complex, the barycenter $\mathbf{b}_T$ is picked as the new point.

4.4.2 Foot point of barycenter

Although the procedure described in section 4.4.1 is easy to implement, it does not generate optimal choice of the

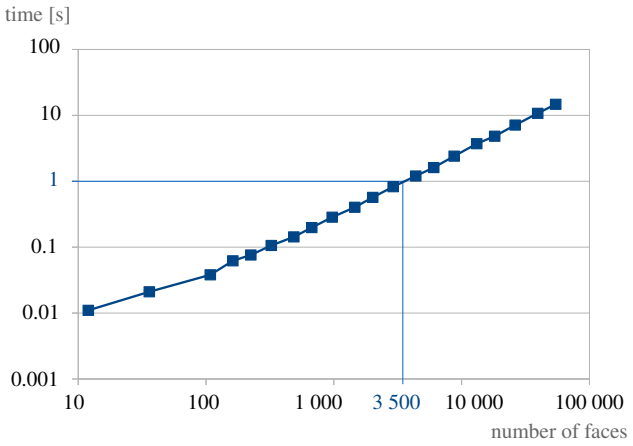


Figure 4: Experimental measurements of the time complexity of our algorithm. In average, we were able to process 3 500 faces per second.

new vertex. In some cases, the intersection of $\mathcal{Q}$ and $\bar{n}$ does not exist or is relatively distant from the mesh. The distant vertices create unwanted local sharpness (spikes). Moreover, if the initial mesh is not closed, the spikes naturally occur around the boundary.

These issues are resolved by picking the new vertex as a foot point of perpendicular line $\bar{p}$ from $\mathbf{b}_T$ onto $\mathcal{Q}$, see figure 3. To find the foot point, we use the algorithm described by Hartmann in [12, section 5.1.2].

5 Implementation

Implementation of our method was done in C++ and compiled under GCC 4.8.1. Both techniques for picking the new vertex from the quadric were implemented. The results in this paper were obtained using the foot point algorithm exclusively.

For mesh manipulation, we decided to use an open-source library *OpenMesh* [13], developed by working group of Leif Kobbelt at RWTH Aachen University. We chose *OpenMesh* for two main reasons:

- ◇ meshes are represented using doubly-connected edge list (DCEL), which allows fast performance of the mesh operations.
- ◇ *OpenMesh* contains application *Subdivider* with built-in framework for subdivision surfaces. *Subdivider* also implements various linear triangular subdivision schemes (Loop, $\sqrt{3}$, Modified Butterfly) overloading abstract base class `SubdividerT`. Simple GUI is provided using Qt and GLUT. User can load and save mesh in popular formats (.obj, .off, .ply) and iteratively apply subdivision operators.

To solve the linear system (27), we used an open-source C++ linear algebra library Armadillo [14].

Computational complexity of the quadric fitting refinement is $\mathcal{O}(F)$, where F is the number of processed faces.

Figure 4 shows the relation between the time needed to perform one iteration of the QFR and the number of triangles in the refined mesh. All the measurements were performed on the PC with Intel Core i7 3517 Ivy Bridge processor running Ubuntu 13.10 Saucy Salamander.

6 Results

6.1 Choosing the weights

Theoretically, any positive real number can be used as a weight $\tilde{w}_i$ of the vertex $\mathbf{p}_i$ or as a weight $\hat{w}_i$ of the normal $\vec{\mathbf{n}}_i$. In practice though, the weights have to be chosen carefully as their choice can influence the result significantly.

The used weights are dependent on the graph distance $\mathcal{D}_T(\mathbf{p}_i) =: \mathcal{D}_T^i$ between the vertex $\mathbf{p}_i \in \mathcal{N}_T$ and the triangle T, see (3). Given the initial values v_i, n_i and factors v_f, n_f, the weights are computed as

$$\tilde{\mathrm{w}}_\mathrm{i} = v_i\, v_f^{\mathcal{D}_T^i}, \quad \hat{\mathrm{w}}_\mathrm{i} = n_i\, n_f^{\mathcal{D}_T^i}. \tag{36}$$

We demonstrate the effect of different sets of weights on the Stanford bunny, see figure 5. The bunny mesh was decimated to 3000 faces and refined using QFR with the initial values and factors

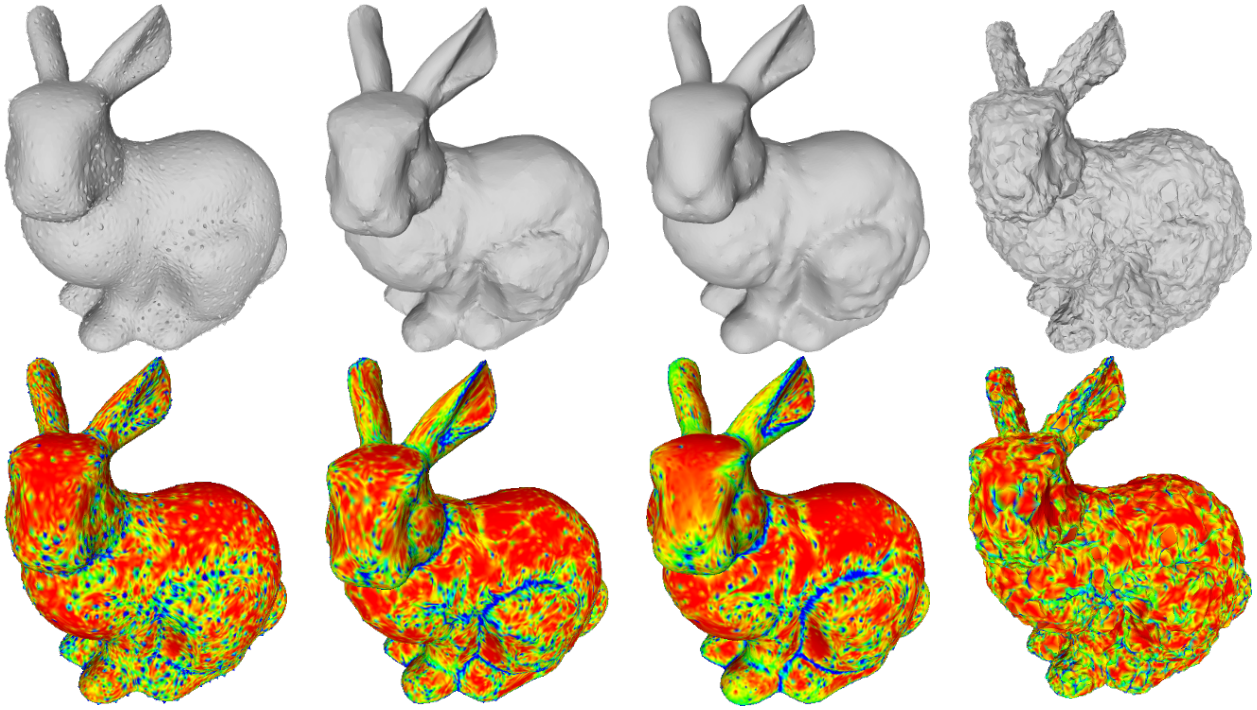
$$v_i = 1, \quad v_f = 1, \quad n_i = 1, \quad n_f = 1; \tag{37a}$$
$$v_i = 1000, v_f = 1, \quad n_i = 0.0001, n_f = 0.0001; \tag{37b}$$
$$v_i = 1000, v_f = 0.0001, n_i = 0.0001, n_f = 0.0001. \tag{37c}$$

It is clear the strategy of taking all data with the same weights as in (37a) does not produce fine results for an irregular mesh such as the Stanford bunny. This is due to the fact that the information carried by normal vectors is very strong and has to be treated gently. Assigning the normals smaller weights as in (37b,c) yields much smoother result. The best results are obtained in (37c), where both $\tilde{\mathrm{w}}_\mathrm{i}, \hat{\mathrm{w}}_\mathrm{i}$ get smaller as the distance from the refined triangle increases.

In our current setup, the weights need to be adjusted case-by-case. One of the possible future improvements of the QFR is to compute the weights algorithmically. The local geometry of the mesh (vertex angles, triangle areas) could be used for this purpose.

6.2 Influence of the normal vectors

The normal vector $\vec{\mathbf{n}}_i$ determines the tangent plane at the vertex $\mathbf{p}_i$. To show how the change in the prescribed normals influences the limit surface, we applied the QFR on the Stanford bunny with the alternative set of vertex normals. This alternative set of normals was generated randomly from a noise function. The results are visualised in figures 5c (original normals) and 5d (random normals). The weights from (37c) were used in both cases. The refined meshes differ dramatically, despite the fact the weights for normals are much smaller comparing to the vertex weights (order of 10^7).



(a) (b) (c) (d)

Figure 5: (a-c) The Stanford bunny with original normals, refined using the sets of weights from (37a-c). (d) The Stanford bunny with randomly generated normals, refined using the weights from (37c). Bottom part shows the visualisation of the discrete ABS curvature.

6.3 Comparison with linear schemes

To compare the proposed algorithm with the linear schemes, we have used the large-scale mesh of the Venus of Dolní Věstonice. This mesh is a discretized version of the small nude female statuette found in Moravia south of Brno. Dated to 29,000-25,000 BCE, it is considered one of the oldest known pieces of ceramic in the world.

The original Venus mesh (131 114 vertices) was decimated with app. 99% compression rate (1 356 vertices). The decimated mesh was refined four times using the QFR, the $\sqrt{3}$-subdivision, the Modified butterfly and the Loop scheme. For the QFR, we have used the weights $(v_i, v_f) = (1, 0.1), (n_i, n_f) = (0.001, 0.01)$. The one-sided Hausdorff distance was used to measure the error and to compare the refined meshes. For reference, the lengths of the sides of the bounding box of the Venus mesh are $108.4, 31.8$, and 42.8 units.

	2nd iteration			4th iteration		
	max.	**mean**	**RMS**	**max.**	**mean**	**RMS**
QFR	1.945	0.092	0.173	1.945	0.093	0.174
$\sqrt{3}$	1.990	0.167	0.226	2.003	0.174	0.233
MB	1.846	0.083	0.163	1.839	0.084	0.164
Loop	2.001	0.170	0.230	2.003	0.175	0.234

Table 1: Performance of the QFR on the Venus mesh comparing to the linear schemes

The results are visualised in figure 7. The numerical values of maximum, mean and RMS error are summarized in table 1.

Using this setup, we are able to obtain a close approximation of the original mesh. The performance of QFR is comparable to the Modified Butterfly. This is related to the fact that both QFR and Butterfly are interpolating schemes. However, the mesh produced by our method is visually smoother. The meshes generated by the approximating schemes ($\sqrt{3}$-subdivision, Loops) are also smooth, but they lack the details of the mesh produced by the QFR.

6.4 Reconstruction of quadratic surfaces

In the context of our refinement method, quadratic surfaces or *quadrics* are an important tool. Our algorithm can also be used for the reconstruction of quadratic surfaces from a coarse, approximating mesh.

Using the weights $\tilde{w}_i = 1000, \hat{w}_i = 0.0001$, the scheme is capable of reconstructing a close approximation of the sphere from the cube, see figure 6. The initial mesh (cube) is shown after $0, 1, 2, 3$ and 9 iterations, together with the color visualisation of the distance of the densest mesh from the sphere. The red color corresponds to zero distance, blue color corresponds to distance ≥ 0.0025, which is fairly small taking into account the sphere has unit radius. The output is also influenced by the initial triangulation of the cube.

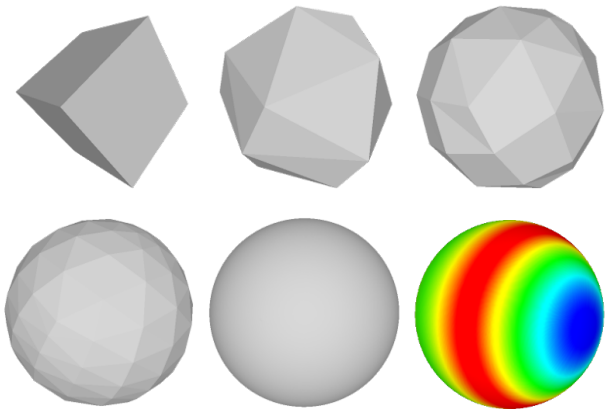


Figure 6: Reconstruction of the sphere $x^2 + y^2 + z^2 = 1$.

In addition to the sphere, the scheme was capable of reconstructing a cylinder, an elliptic paraboloid and a hyperbolic paraboloid. These experimental results allow to make a hypothesis that QFR actually *reproduces* quadratic surfaces. In the future, we want to study this hypothesis from the analytic point of view.

7 Conclusions

We introduce a new approach to nonlinear surface subdivision. While developing the scheme, we encountered problems with spikes, arising in some regions of the mesh and around boundary. These issues are resolved by altering the way the new vertex is picked from the quadric. Although the alternative setup is more complex, it gives more accurate results and is applicable on general input mesh.

In the future, we plan to study the algorithm from the analytical point of view. We want to prove the limit surface is G^1-continuous and confirm the hypothesis about the reproduction of quadratic surfaces. As we have mentioned in section 6.1, we also plan to improve the computation of weights, which should be determined by the local geometry of the mesh.

Even though we assume triangular mesh, the proposed scheme can be extended to quad mesh in a straightforward way. To perform the extension, appropriate topological step has to be chosen.

Acknowledgement

We would like to thank Moravian Museum* and EDICO SK, Inc† for providing the mesh of Venus of Dolní Věstonice we used to test our method. The bunny mesh was kindly provided by the Stanford University Computer Graphics Laboratory‡.

*www.mzm.cz
†www.edico.sk
‡graphics.stanford.edu

References

[1] M. Sabin, *Analysis and Design of Univariate Subdivision Schemes*. Springer Berlin Heidelberg, 2010.

[2] C. Loop, "Smooth subdivision surfaces based on triangles," Master's thesis, University of Utah, August 1987.

[3] N. Dyn, D. Levin, and J. A. Gregory, "A butterfly subdivision scheme for surface interpolation with tension control," *ACM Transactions on Graphics (TOG)*, vol. 9, no. 2, pp. 160–169, 1990.

[4] D. Zorin, P. Schröder, and W. Sweldens, "Interpolating subdivision for meshes with arbitrary topology," in *Proceedings of SIGGRAPH 96, Annual Conference Series*, pp. 189–192, 1996.

[5] L. Kobbelt, "$\sqrt{3}$-subdivision," in *Proceedings of SIGGRAPH 2000, Annual Conference Series*, pp. 103–112, 2000.

[6] T. J. Cashman, "Beyond Catmull–Clark? A survey of advances in subdivision surface methods," *Computer Graphics Forum*, vol. 31, no. 1, pp. 42–61, 2012.

[7] J. Peters and U. Reif, *Subdivision surfaces*. Springer, 2008.

[8] S. Karbacher, S. Seeger, and G. Häusler, "A non-linear subdivision scheme for triangle meshes," in *VMV*, pp. 163–170, 2000.

[9] N. Aspert, T. Ebrahimi, and P. Vandergheynst, "Non-linear subdivision using local spherical coordinates," *Computer Aided Geometric Design*, vol. 20, no. 3, pp. 165–187, 2003.

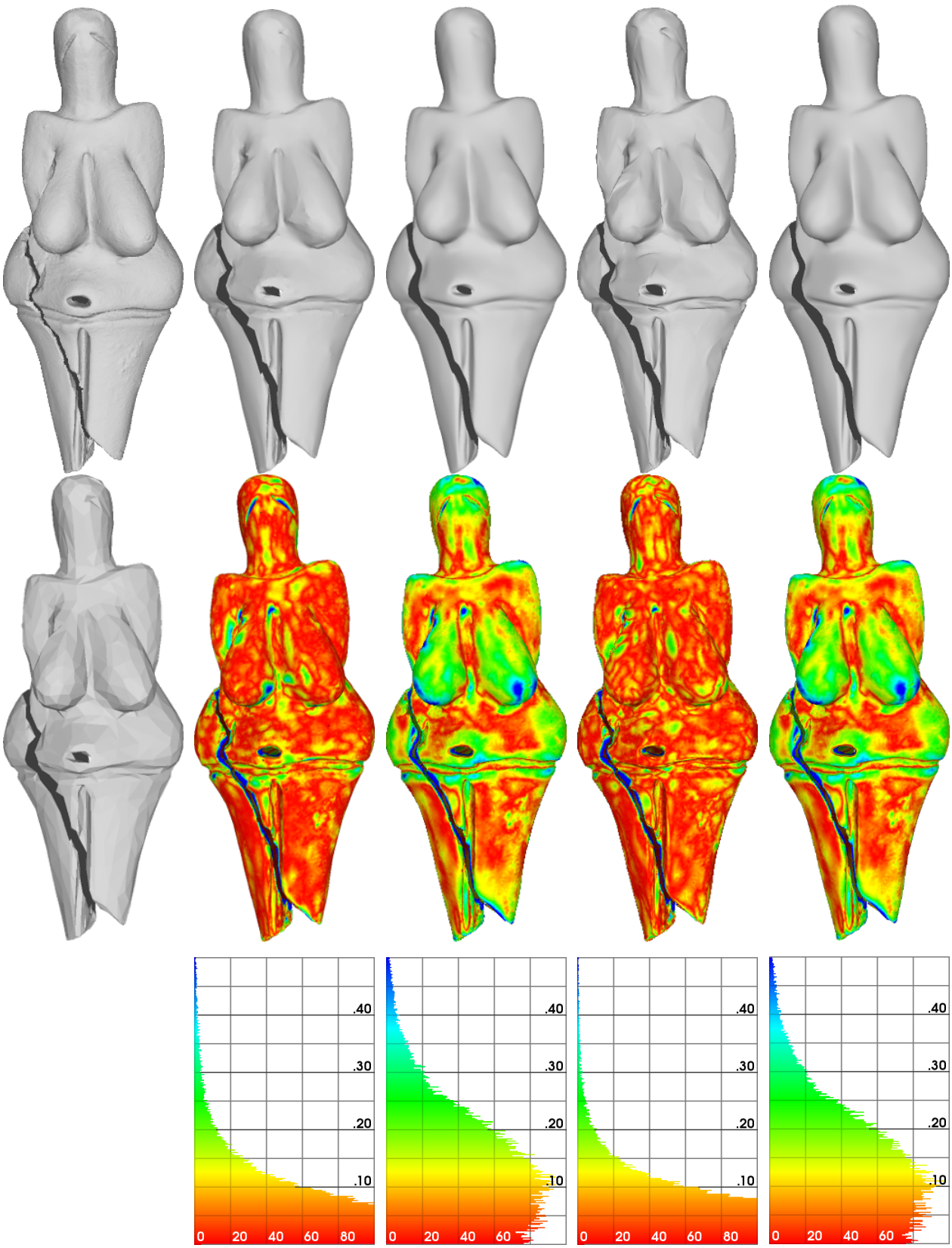
[10] X. Yang, "Surface interpolation of meshes by geometric subdivision," *Computer-Aided Design*, vol. 37, no. 5, pp. 497–508, 2005.

[11] P. Chalmoviansky and B. Jüttler, "A non-linear circle-preserving subdivision scheme," *Advances in Computational Mathematics*, vol. 27, no. 4, pp. 375–400, 2007.

[12] E. Hartmann, "On the curvature of curves and surfaces defined by normalforms," *Computer Aided Geometric Design*, vol. 16, no. 5, pp. 355–376, 1999.

[13] M. Botsch, S. Steinberg, S. Bischoff, and L. Kobbelt, "Openmesh - a generic and efficient polygon mesh data structure," 2002.

[14] C. Sanderson, "Armadillo: An open source C++ linear algebra library for fast prototyping and computationally intensive experiments," tech. rep., NICTA, 2010.



(a) (b) (c) (d) (e)

Figure 7: Comparison of our method with the linear triangular schemes. (a) Original and decimated Venus mesh, (b-e) decimated mesh refined with QFR, $\sqrt{3}$, Modified Butterfly and Loop. Top row in (b-e) shows the mesh after four iterations of given scheme, middle row shows the visualisation of the Hausdorff distance of the original mesh from the refined meshes. Bottom row displays the histograms of the Hausdorff distance.