Computers & Graphics 35 (2011) 43–53

Contents lists available at ScienceDirect

Computers & Graphics

journal homepage: www.elsevier.com/locate/cag

Technical Section

Shape and tone depiction for implicit surfaces

Emilio Vital Brazil [a,b,*], Ives Macêdo [a,b], Mario Costa Sousa [b], Luiz Velho [a], Luiz Henrique de Figueiredo [a]

[a] IMPA - Instituto Nacional de Matemática Pura e Aplicada, Brazil
[b] University of Calgary, Canada

ARTICLE INFO

Available online 12 November 2010

Keywords:
Point-based NPR
Computer-generated stippling
Non-photorealistic rendering (NPR)
HRBF Implicits
Variational implicit surfaces
Hermite interpolation
Radial basis functions

ABSTRACT

We present techniques for rendering implicit surfaces in different pen-and-ink styles. The implicit models are rendered using point-based primitives to depict shape and tone using silhouettes with hidden-line attenuation, drawing directions, and stippling. We present sample renderings obtained for a variety of models. Furthermore, we describe simple and novel methods to control point placement and rendering style. Our approach is implemented using HRBF Implicits, a simple and compact representation, that has three fundamental qualities: a small number of point-normal samples as input for surface reconstruction, good projection of points near the surface, and smoothness of the gradient field. These qualities of HRBF Implicits are used to generate a robust distribution of points to position the drawing primitives.

1. Introduction

Pen and ink illustrations, whether with traditional or computer-generated techniques, provide several important perceptual cues such as relationships between light and dark, shape, pattern and edge depiction, drawing direction, focus, and gradients of detail and texture. Three key elements are essential for effectively conveying these perceptual cues: *where* to place drawing primitives, *how many* to place, and *how to draw* them [1–5]. In this paper, we present methods that approximate traditional ink-based rendering techniques for depicting shape and tone perceptual cues, suitable for non-photorealistic rendering (NPR) applications using implicit surfaces as the primary object representation (Fig. 1).

Implicit surfaces provide important, mathematically precise information about surface properties, useful for answering *where* and *how many* primitives to draw across the surface. Implicit surfaces allow global calculations such as point pertinence (i.e., whether a point is within the surface volume) and distance evaluation, and at the same time, also allow obtaining local differential properties, such as normals and curvature. This brings advantages over other types of geometric models. To instantiate our pipeline, we use the recently introduced Hermite radial basis function (HRBF) Implicits which interpolate point-normal data to reconstruct an implicit surface [6]. HRBF Implicits provide a simple and compact representation, requiring only a few number of point-normal samples to reconstruct quality implicit surfaces. In addition, the good behavior of HRBF Implicits allows performing all the general implicit surface operations using simpler and more efficient algorithms, even for complex models.

The main contribution of this paper is on applying NPR techniques directly over implicit surfaces, bringing important benefits such as consistent and good projection of points, controlled placement and distribution of drawing primitives (for artist-driven shape and tone depiction), simple metaphors for style control in pen and ink renderings of implicits, and real-time interaction with the rendered model.

2. Related work

Different works have proposed NPR techniques for implicit surfaces, addressing the problem of extracting contours (silhouettes, feature curves) and approximating different traditional rendering styles including pen and ink stylized rendering, hatching and stippling [7–12], feature line extraction and drawing [13–19], painterly rendering [20], tone-based clip art [17], and mixed media [21].

Bremer and Hughes [7] presented an approach to extract and trace silhouettes incrementally from analytic implicit functions. Short interior ink-based strokes are also positioned using successive ray intersection tests, including hidden-line removal (HLR). Foster et al. [9] extended these tracing and particle-based techniques by providing additional options for stroke stylization and specific interior stroke placement strategies on complex hierarchical implicit models. Techniques for rendering sudden blends and CSG junctions are also presented. Jepp et al. [10,11] have further extended this NPR framework using flocking techniques to manage particle distribution and render additional surface contours in different pen and ink stippling and curvature-based hatching. Proença et al. [18,19] also extended the approach presented by

* Corresponding author at: IMPA - Instituto Nacional de Matemática Pura e Aplicada, Brazil. Tel.: +55 2125295080.
E-mail addresses: emilio@impa.br (E. Vital Brazil), ijamj@impa.br (I. Macêdo), smcosta@ucalgary.ca (M. Costa Sousa), lvelho@impa.br (L. Velho), lhf@impa.br (L. Henrique de Figueiredo).

0097-8493/$ - see front matter © 2010 Elsevier Ltd. All rights reserved.
doi:10.1016/j.cag.2010.09.017

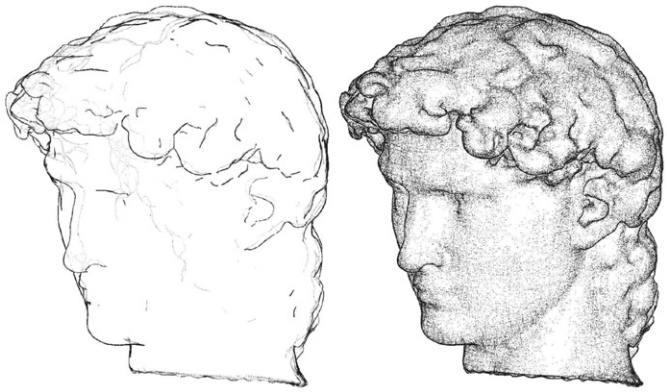

Fig. 1. Drawing steps of David's head with our system. The silhouettes with hidden-line attenuation (left); completing the tone depiction by adding more stippling marks and enhancing interior contours with a white halo (right). The model has 4096 samples, 700 K render points and with CPU rendering at 9 fps.

Foster et al. [9], by extracting and rendering suggestive contours over point-set MPU implicits. One particular strategy is to project particles from a base mesh onto the implicit surface and model the strokes using this particle distribution. This approach was used by Elber [8] who also presented several methods for ink-based stroke rendering effects. More recently, Schmidt et al. [12] adapted an approach where low-resolution silhouette and suggestive contours are extracted from a coarse base mesh approximating the smooth surface and incrementally refined and projected to the implicit surface. Stippling and HLR are also provided by adapting surfel techniques.

In our approach, we use a new representation, Hermite radial basis function (HRBF) Implicits [6]. Our point distribution does not require relaxation techniques, given that HRBF provides a good projection framework and the users can manipulate the seed placement directly. All our rendering primitives (silhouette contours, stippling, hatching) are points. Rendering is performed directly over the implicit model without requiring any intermediate representation. We provide a hidden line attenuation (HLA) method, which approximates some of the visual elements of an artist-generated visual construction, or *scaffolding* [12].

3. System overview

To obtain quality renderings of an implicit surface we require two properties from its representation. First of all, we need the implicit function not to vary too wildly close to the surface and that the surface be at least C^1, with readily available normal information. The first requirement allows us to use simple methods to approximate projections of points onto the surface and obtain good point distributions when those points are close to the surface. From now on, we will assume that we have these properties and that all points on the surface have a well-defined normal vector at them; in Section 7 we discuss the details of the representation which we chose to use.

In the first step of our algorithm (Section 4), we place points onto the surface which will be latter used as seeds to create more points over the surface. The number and position of these seeds have a direct impact on the quality of the final image. After we have obtained a good distribution of seeds on the surface, a refinement phase begins (Section 5), consisting of increasing two sets of points. The first set contains stippling points, which do not follow any specific direction and result in a well-distributed coverage of the surface. The second set follows particular directions to create the perception of short strokes. Finally, given a fixed camera position, we classify the generated points (Section 6) as either front, back, or silhouette and use this classification to define how the points are rendered. Fig. 2 illustrates our pipeline.

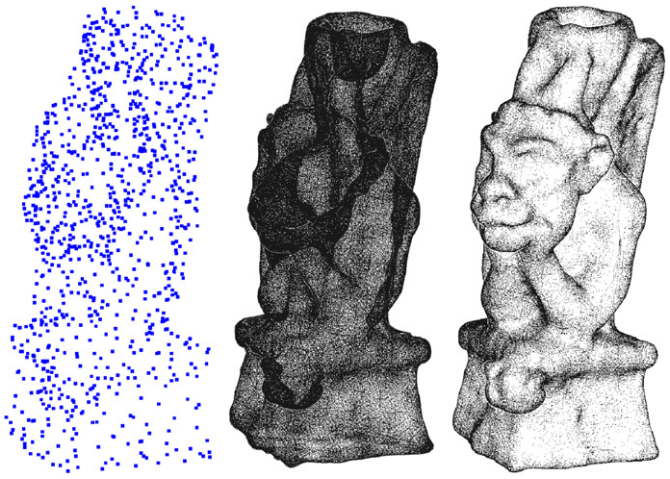

Fig. 2. Overview of our pipeline (left to right), seed points are placed across the surface enabling further placement of render points, which are subsequently used to modulate tone and shape depictions.

4. Seed placement

The seed placement step is important to define the quality of the final drawing, since the positioning of all subsequent points is derived from the seeds. We employ a semi-automatic approach that starts with a certain set of seeds that can be positioned with or without direct user's interaction. Before we continue talking about seed placement, we need to discuss the representation for the implicit surfaces. There are many ways to define implicit surfaces and notable examples are the BlobTree [22], piecewise algebraic surface patches [23] and convolution surfaces [24]. One compact representation for implicit surfaces can be derived by choosing a suitable interpolation method and use samples to define the surface. Our system employs Hermite samples (points and normals) and radial basis functions for interpolation (Section 7). In order to automatically create a set of seed points, we employ two different techniques. The first requires samples on the surface, while the other is more general and can be used for an implicit that does not have previous points on it.

The strategy that uses the samples themselves as seeds works well because it does not require projection of points onto the surface (which may be an expensive process). However, this strategy is only recommended when the samples are well distributed over the surface, which is typically the case when data is sampled from a regular parametrization or a mesh. On the other hand, when we do not have samples over the surface or when these points are not well distributed, we need a different strategy to obtain a good seed placement and capture the surface's overall shape. The second seed placement technique relies on an implicit surface with properties that allow good point projections. We fill the bounding box of the initial samples with random points and then project them onto the surface. In order to do that, we define the resolution R which will be used in its largest dimension, defining the resolution of the other two dimensions to create a regular grid with cubic cells. In the center of each cell, we place a point which is randomly perturbed to a distance up to αL, where L is the side length of the cell (in this work, we use $R=8$ and $\alpha = 0.25$). This random displacement reproduces the effect of jittering and reduces sampling artifacts. Fig. 3 illustrates one advantage of using the bounding box strategy (right) instead of the sample-based approach (left): the complete model is depicted and can be inspected.

However, both approaches may fail even when we have samples well distributed over the surface, because certain regions can have too few points, then adding a few points in the right place will improve the final image. The bounding-box strategy can cluster points over some part of the surface or create lack of points, in the same model.

Tools to manipulate seeds directly are very important to achieve better results (Fig. 4). Our approach is to allow the user to sketch a region in the screen space to select visible points, then delete all or 50% of the number of points are randomly chosen and deleted in these regions. We have the option to place points directly on the model. The user clicks where she/he wishes to place the new point but if we just project this point it could stop far away from the chosen position. In order to place the point below the mouse, we perform a line search with constant stepsize to find an approximation to the first surface intersection with a ray originating from the camera (Fig. 5), then we use this position to start the projection. As a robust and automatic alternative to our user-assisted approach, the relaxation-based method presented by Meyer et al. [25] can be used either to sample the surface from scratch or as a post-processing step for distributing seeds already on the surface in a curvature-dependent manner.

To approximate the projection of one point $\mathbf{p}$ onto the surface, we use the unconstrained optimization method of steepest descent with Armijo Rule to minimize the function $\frac{1}{2}(f(\mathbf{x}))^2$, in which $\mathbf{p}^0 = \mathbf{p}$ is used as the initial iterate (for more details, see Appendix B). It is worth noticing that this simple method provides a good enough approximation to the projection of $\mathbf{p}$ onto the implicit surfaces as long as the function f has properties similar to those of a signed distance function in $\mathbf{p}$.

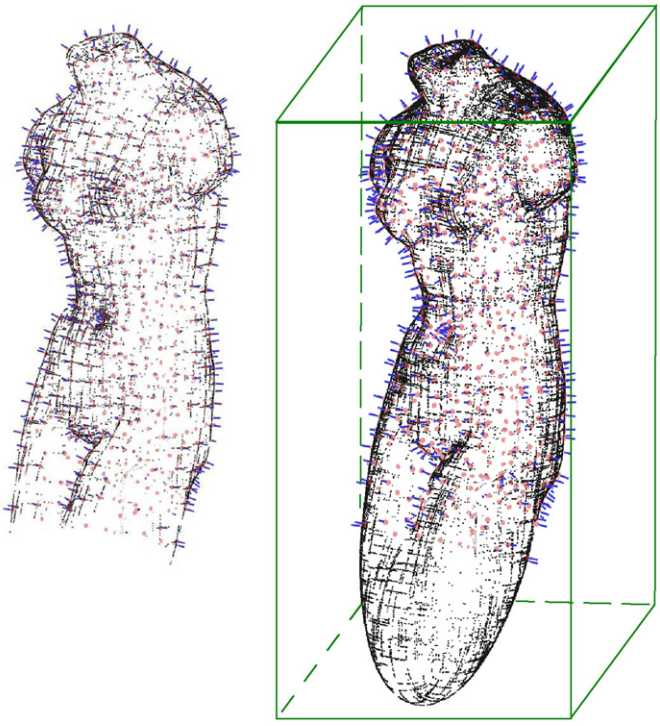

Fig. 3. Two approaches to place seeds. Left: exploiting the samples used to define the surface. Right: a bounding box filled with jittered grid points subsequently projected onto the implicit surface.

5. Multi-level sample refinement

After the seeds have been placed on the surface, their positions are ready to be used to generate render points. We divide the render points into three groups: stippling, principal directions of curvature, and combing directions. Stippling points are placed in a scattered fashion, focusing on covering $\mathcal{S}$ uniformly, while the two other groups provide linear mark depictions by clustering points along a directional field. All three groups share the same recursion idea. We use the actual seed position to place a new point near the surface, located in space at a distance ρ from its seed. After that, we project the new points onto the surface using the same method described in Section 4. In the next step, all the recently generated points become seeds of their own group, and we set $\rho = \rho/3$ (Fig. 6). The process goes on until the desired visual effect is achieved. It is important to notice that, by using this 1/3 rule, the distance between a seed and all its descendants is limited; in fact, after k steps, the distance between the original seed and any descendant will be less than $1.5\rho_0$ and two points with the same original seed will be at most $1/3^k\rho_0$ apart.

5.1. Sampling near the surface

Since the projection method is faster and more precise when the point is near the surface, we try to place new points as close as

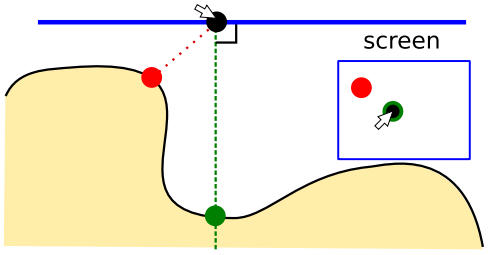

Fig. 5. Placing a new point (black) using the mouse as input. Red point is projection point if it is placed without any approximation to start the projection, green point is the final point position using a line-search (along the green line) to choose an approximation. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

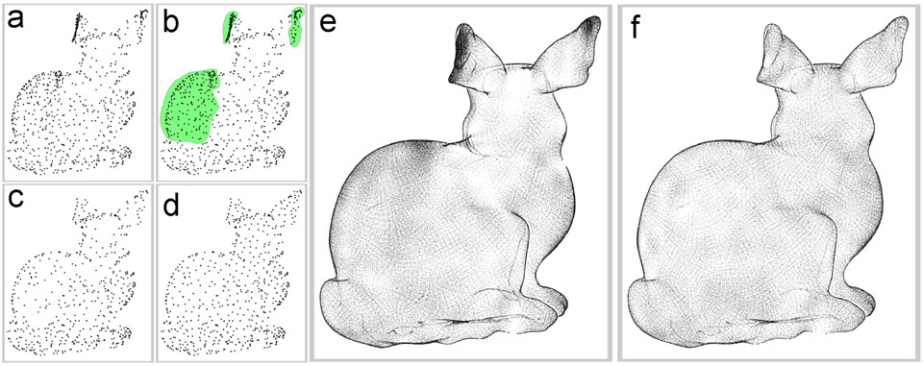

Fig. 4. Tools to improve the quality of the seed distribution. After computing seed points (a), the user can select areas to remove (b) and (c) as well as to insert points in order to obtain a good distribution of seeds on the surface (d). Finally, we can compare the results after 3 rounds of refinement using the original seed points (e) and using the manipulated seed points (f).

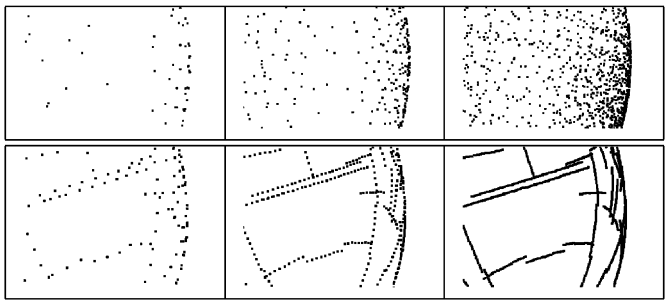

Fig. 6. Multi-level sample refinement. Left to right, levels of refinement: one, two, and three.

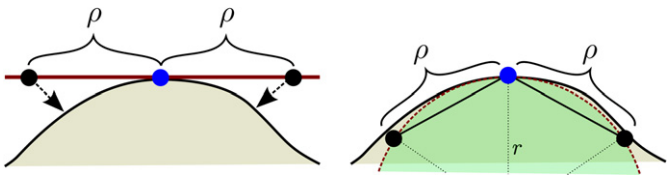

Fig. 7. Placement of render points (black) near the surface using the given seed (blue). Left: using the tangent plane. Right: using the osculating circle. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

possible to it. We have two approaches to place the new points: one uses the tangent plane of the surface, the other uses curvature estimation (Fig. 7). Finding the plane tangent to the surface at a projected or sample point of the implicit surface is virtually costless, since these points already have their gradients calculated. In contrast, in our approach, the use of curvatures to estimate the new point position may be an expensive process, with a higher computational cost than to project a point a bit farther from the surface. As a result, this approach is only used when we place render points at distances ρ along one of the principal directions of curvature on the osculating circle (Section 5.3).

The initial step size ρ_0 defines whether the points are well spread or clustered over the surface. We use two semi-automatic approaches to pick a good estimate of ρ_0. In the first approach, when placing seeds from the bounding box, the first ρ_0 approximation is the voxel diameter. In the second approach, when seeds are placed directly from the samples, we use the average of their empty ball diameter. However, these two approaches can either underestimate or overestimate ρ_0, thus creating clusters or visually broken lines, respectively. To avoid these cases, our system allows the user to explicitly set ρ_0. In order to provide good visual feedback, our system randomly places square patches with sides equal to $2\rho_0$ over the surface (Fig. 8).

5.2. Stippling points

To generate the stippling points we use the seeds' tangent planes. We need to create a basis to the affine plane to place these points. There are many possibilities for building the basis using the normal vector as input, with all choices having at least one point of discontinuity [26]. For this step in our pipeline, we selected a method that has two points of discontinuity, after observing they avoid patterns (Fig. 9(d)). To create our basis, we rotate the normal to a fixed axis $\mathbf{r}' = R\mathbf{n}$ and then compute the cross products $\mathbf{u} = \mathbf{n} \times \mathbf{r}'$ and $\mathbf{r} = \mathbf{n} \times \mathbf{u}$. After that, $\mathbf{n}$, $\mathbf{u}$ and $\mathbf{r}$ are normalized. Observe that, on the fixed axis, this method is not well defined; however, we observed that this was not a problem when placing stippling points. Four points are placed near the surface by using the local coordinates of the affine plane: $p_{i,j} = i \cdot \rho\mathbf{r} + j \cdot \rho\mathbf{u}$, where $i,j = \pm 1 + u$, and

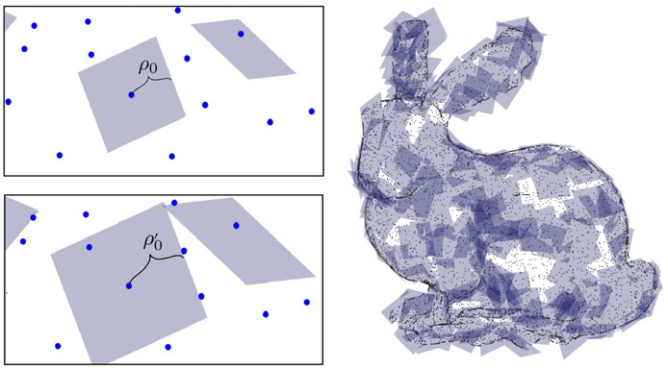

Fig. 8. Using square patches to choose ρ_0. Upper left, the first approximation to ρ_0; bottom left, the user's choice; right, the visual feedback after 2 steps of subdivision.

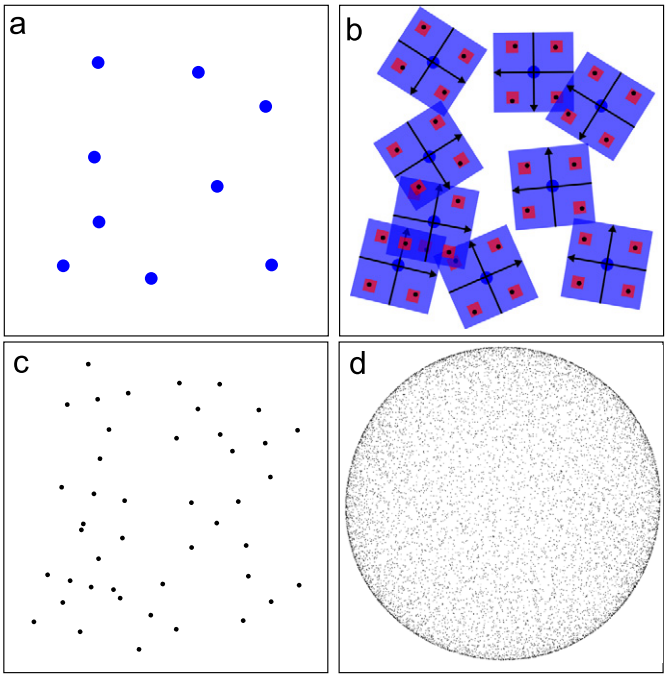

Fig. 9. Placement of stippling points: top view (a–c) and covering the entire model (d). (a) Seeds over the surface; (b) points generated, with red square representing the jitter range; (c) results of one subdivision step; (d) starting with 6 seed points, after 5 subdivision steps, the final results over a sphere. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

$u \sim \mathcal{U}([-0.125, 0.125])$, i.e., u is a random variable with uniform distribution within the interval $[-0.125, 0.125]$ (Fig. 9).

5.3. Drawing direction

To create the perception of short continuous lines, we use the smoothness property of the implicit surface and a method to create smooth directions. The idea is as follows: since we have a smooth variation of normals and a piecewise smooth function $F : \mathcal{S} \times \mathbb{S}^2 \to \mathbb{S}^2 \times \mathbb{S}^2$, $F(p,\mathbf{n}) = (\mathbf{r},\mathbf{u})$, we use these properties to create a sequence of points $p_i = \pm\rho\mathbf{w}$, where $\mathbf{w}$ could be either $\mathbf{u}$ or $\mathbf{r}$. As previously mentioned, the points will be closer to each other at each step of the subdivision; therefore, after a few steps, we have the visual perception of a line being defined (Fig. 6, bottom row). At the first subdivision step, the original seeds throw points along both directions ($\mathbf{r}$ and $\mathbf{u}$). After this first subdivision step, we separate the points into two sets, generated using $\mathbf{r}$ and $\mathbf{u}$,

respectively. As a result, each set will only generate points along its original direction.

We work with two different functions F to create the perception of lines. The first one is the principal directions of curvatures, using the method described by [27] to calculate a reduced Hessian matrix, followed by its eigenvalues and eigenvectors to get the principal directions and values of curvature. The second function is the *combing* directions, using the method described by [26] to create the basis. This approach splits the sphere into 12 regions of directions. This partition creates a pattern that is curvature-independent. These "combing directions" provide another way of distributing line directions across the model. Qualitatively, they seem to depict the overall perception of 'mass volume' of the object. However, further work is necessary to evaluate such perceptual cues and the possible combination with curvature lines and other line directions. In Fig. 10, we compare combing directions with principal directions of curvature.

6. Rendering

At this stage, we are ready to use the render points already placed over the surface to visualize the implicit model in different styles. The render points are classified into three sets: front, back, and silhouette. After that, they are assigned a point size and an alpha value and are subsequently sent to the standard graphics pipeline. We calculate $v = \mathbf{n} \cdot \mathbf{v}$, where $\mathbf{n}$ is the normal at the point and $\mathbf{v}$ is the viewing vector. Using a small threshold $\delta > 0$, we identify front points when $v < -\delta$, back points when $v > \delta$, and silhouette points otherwise. After classifying all points, different rendering effects are created, as described next.

6.1. Silhouettes and hidden-line attenuation

In our system, the silhouette points are always displayed; however, occluded points could appear, thus creating artifacts. We would like to provide different visual effects instead of simply removing occluded points (Fig. 11, bottom). We attenuate hidden-lines by displaying the back and front points in the same color as the background and with an opacity value $\alpha \in [0,1]$ (Fig. 12). The tone of the silhouette point will be closer to the background's as much as its depth-complexity. To be sure the silhouette points will not be occluded by other points, we use a decay function for α. The α values of the front points have a quadratic decay function $\alpha_f = v^2\lambda$, and we use $\alpha_b = 0.2(\log(v-.05)+5)\lambda$ for the back points (Fig. 11, left), where λ is a parameter controlled by the user. If $\lambda = 0$, all silhouettes are displayed (Fig. 11, top right); if $\lambda > 0$, we have line attenuation (Fig. 11, bottom right). The size of the back points and their α- decay function allows to create a halo effect on the silhouette cusp points, but we need to control the point size to achieve the same visual effect independent of scale. In order to enhance the silhouettes the user has the option to draw a thicker line in the tangent direction of the silhouette, i.e. a line with

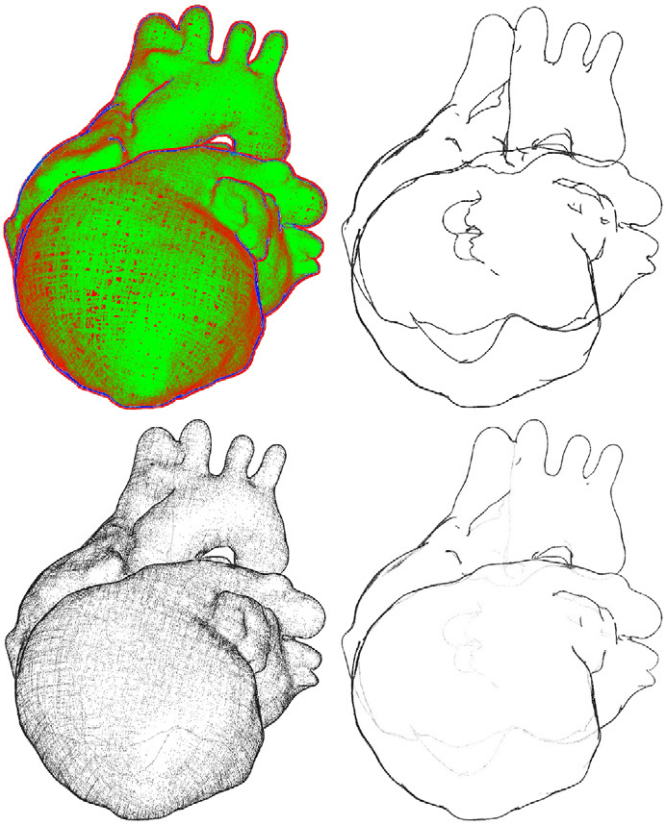

Fig. 11. Top-left: the α decay when the point gets closer to the silhouette: back points (red) and front points (green). Top-right: without hidden line attenuation. Bottom row: final results. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

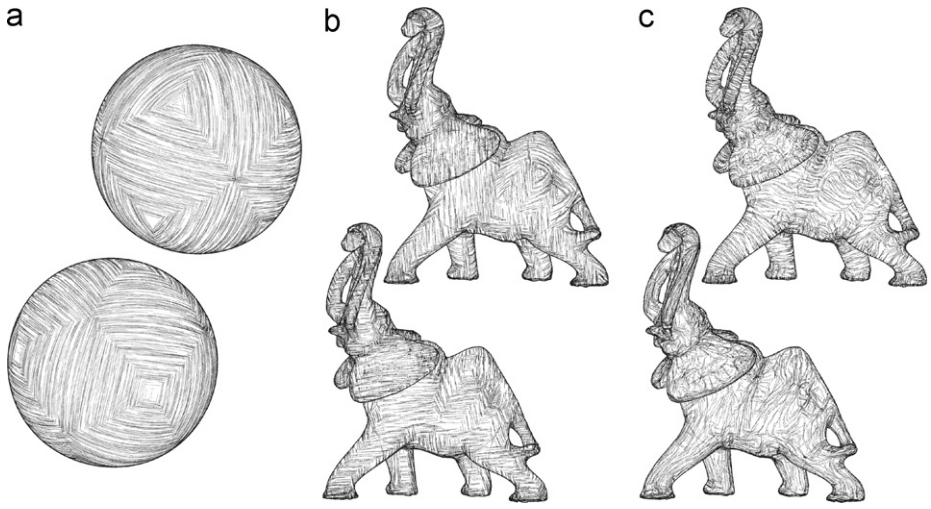

Fig. 10. Comparing drawing directions. Combing directions over (a) the sphere and (b) the elephant model. (c) First and second principal directions of curvature (top and bottom, respectively).

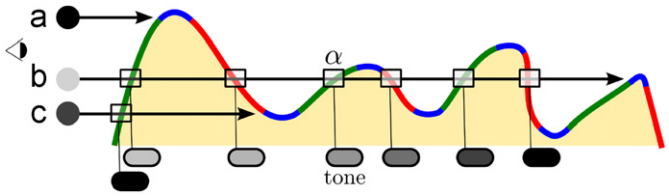

Fig. 12. Different levels of hidden-line attenuation accumulated along the viewing direction: (a) none (b) full (c) partial. Ellipses correspond to the tone value at the intersection point (between surface and viewing directions). Boxes correspond to the alpha attenuation at the point. Line colors correspond to front-faces, back-faces and silhouettes (red, green and blue, respectively). (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

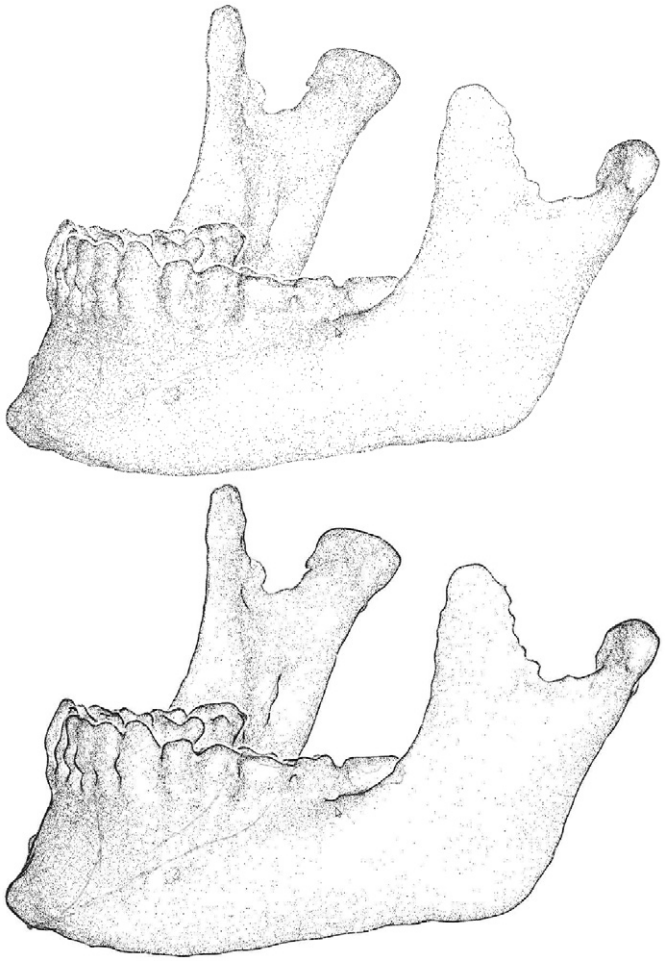

Fig. 13. The Jaw model: without and with silhouette enhancement.

direction $\mathbf{n} \times \mathbf{v}$. Fig. 13 illustrates this effect. In the next section, we provide more details regarding how to control the scaling effect.

6.2. Tone depiction

In our approach, tone is depicted by removing front-points from the surface to create three main types of effects: shading, depth attenuation and tone scaling. The front points are removed randomly, using different probability density functions. The back points are plotted following the same rules of the previous section. Lighting effects are achieved by calculating the tone $\tau \in [0,1]$ at the point, using any choice of illumination model (Fig. 14). Let us define a random variable $u \sim \mathcal{U}[0,1]$. A render point is displayed only if $u \geq \tau$. In this work, lighting effects were generated using $\tau = (\mathbf{n} \cdot \mathbf{l})^{\lambda}$, where $\mathbf{l}$ is the unit light vector and the parameter λ allows the user to control the light intensity. Depth attenuation is achieved by removing both silhouette and front points. As for the lighting effect, points with $\tau \geq u$ are removed. We use the approach presented in [28], $\tau = 1 - \log(d/d_{min})/\log(d_{max}/d_{min})$, where d is the distance between the point and the camera and d_{min} and d_{max} are the depth where we start the attenuation and the far visible depth, respectively (Fig. 22). Tone scaling preserves shading coherence when the model is scaled up or down due to camera motion (Fig. 15). The chance of a front-point being displayed has as an exponential probability density function with the zoom factor as a variable. To control the Halo-effect, we use the same function, but now to affect the size of the back-point.

6.3. Local style control

All effects described earlier are global, meaning that they affect all points on the surface. To give more control and achieve different rendering styles, the system allows the user to select regions to choose different kinds of points to be drawn. Inside the regions will be draw only: (i) silhouettes, (ii) stippling, (iii) first or (iv) second direction, (v) both directions or (iv) all points. The user can in addition choose a decay function to create a smooth transition between these regions (Fig. 16).

Regions are selected by placing spheres in scene-space and adjusting their radii and style. The interface is very simple: the user freezes the camera position and drags the mouse to set the sphere position. There is no limit on the number of spheres for a model. The

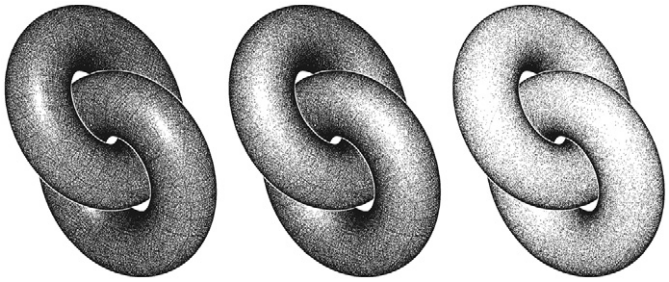

Fig. 14. Tone depiction by removing render points proportionally to the light intensity.

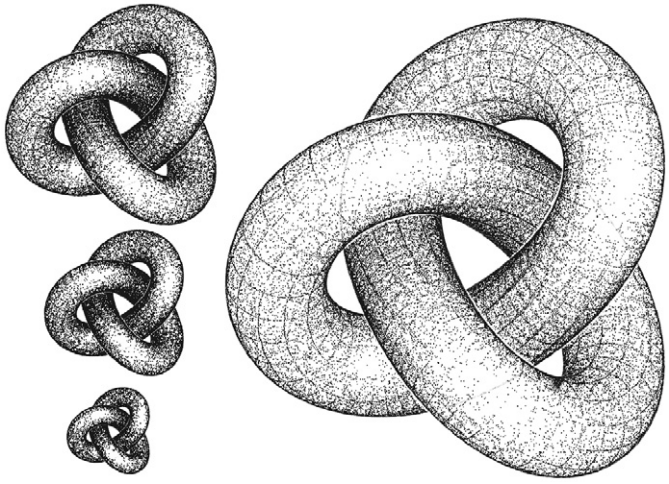

Fig. 15. Scaling a shaded knot model by removing render points.

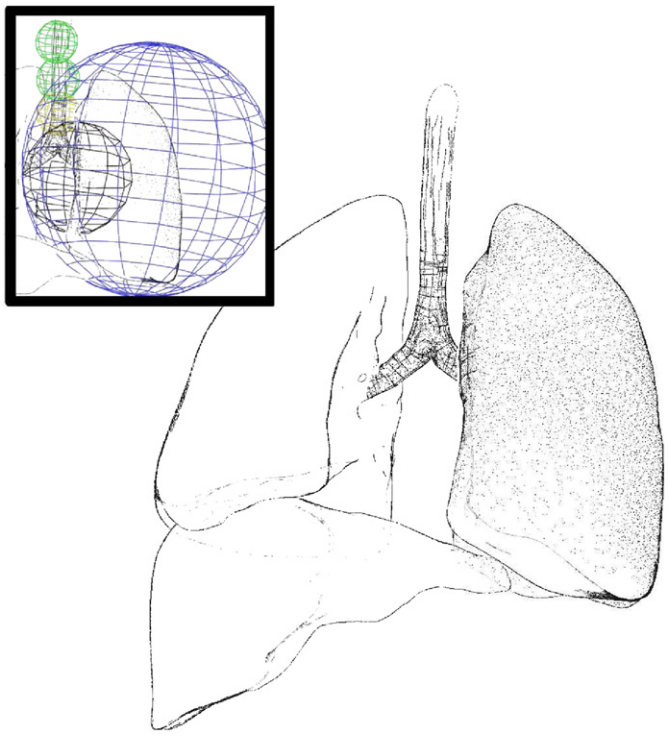

Fig. 16. Employing local style control to edit point placement and density to direct the viewer's attention to the regions of interest.

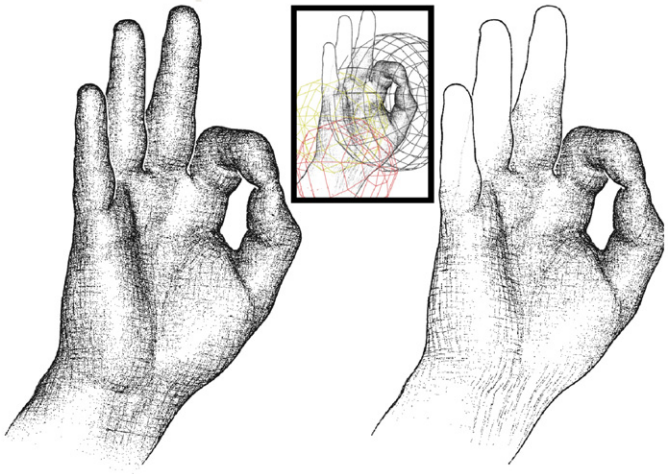

Fig. 17. Comparing the results of global point placement (left) and localized style control (right). In the figure, we illustrate an effect of blending attenuated silhouettes, a single combing-direction (red), both combing-directions in cross-patterns (yellow) and uniform point distribution (black) under local illumination. (For interpretation of the references to colour in this figure legend, the reader is referred to the web version of this article.)

feedback is interactive and allows to compose complex rendering styles. We implemented two decay functions, both depending on the distance to the sphere center. The first one is a quadratic $G(d)=1-(d/r)^2$, the other is an exponential $G(d)=e^{(-4(d/r)^2)}$. In these expressions, d is the distance of the point to the center of the sphere, r is the radius of the sphere and $G(d)=0$ when $d>r$. As before, the probability that the point will be plotted is going to be the function value $G(d)$. Fig. 17 illustrates the effect of controlled blending among different rendering styles.

7. Implicit representation

In order to place our point-based primitives over a surface, we rely on a few basic geometric operators, namely, projection of a point onto a surface and the computation of normals, curvatures and principal directions. By employing a suitable representation, these operations can be made fast and implemented using simple approximate algorithms, yet still giving very good results.

Our representation of choice is based on implicitly defined surfaces computed from points and normals using the variational extension of the HRBF Implicits method of [6] also presented in [29]. This representation has many desirable properties that allow us to employ off-the-shelf linear algebra packages together with simple iterative algorithms to both compute the implicit function and implement our basic geometric operators robustly enough.

We now briefly review HRBF Implicits and our scheme for placing points onto a surface.

7.1. Hermite RBFs

Recently introduced in [6], HRBF Implicits provide a powerful tool to reconstruct implicitly defined surfaces from points and normals. They present many desirable properties of which we take advantage in implementing our pipeline. For instance, the reconstructed surface is guaranteed to interpolate the given points; in addition, the unit normal at those points equals the gradient of the function without having to artificial offset samples. Since the gradient of the implicit function has unit norm at the samples, the function does not vary too wildly close to the surface, a property useful for simple iterative projection algorithms. Also, the implicit function is guaranteed to be at least $\mathcal{C}^1$ at the sample points and, by properly choosing the RBF, $\mathcal{C}^\infty$ everywhere else; the reconstructed surface is therefore typically $\mathcal{C}^1$ at the samples and $\mathcal{C}^\infty$ otherwise. This is a useful property in estimating the local curvatures and principal directions at a given point on the surface. Experiments indicate that their Hermite interpolation property allows good behavior of both the reconstructed surface and the implicit function even under nonuniform and coarse samplings, thus filling holes and recovering local geometric details captured with the first-order information provided by normals.

Since the main focus of this work is on rendering, we use a HRBF Implicits fitter as a black-box whose input is the points $\{\mathbf{x}^j\}_{j=1}^N \subset \mathbb{R}^3$ and normals $\{\mathbf{n}^j\}_{j=1}^N \subset \mathbb{S}^2$ and which outputs a function $f:\mathbb{R}^3\to\mathbb{R}$, implicitly defining a surface by $\mathcal{S}=f^{-1}(0)$ with the properties mentioned above. For the sake of completeness, we briefly review the form of a HRBF Implicits interpolant and how to fit its coefficients from the given points and normals. In Appendix A, we provide more details in order to ease its implementation.

7.1.1. The HRBF implicit interpolant

Macêdo et al. [6] introduced HRBF Implicits as an interpolatory method for recovering implicitly defined surfaces from points and normals. By making use of a theoretical framework for generalized interpolation using radial basis functions, a concrete expression for the implicit function $f:\mathbb{R}^3\to\mathbb{R}$ was derived as follows:

$$f(\mathbf{x})=\sum_{j=1}^{N}\{\alpha_j\psi(\mathbf{x}-\mathbf{x}^j)-\langle\boldsymbol{\beta}^j,\nabla\psi(\mathbf{x}-\mathbf{x}^j)\rangle\}+p(\mathbf{x}), \tag{1}$$

where $\alpha_j\in\mathbb{R}$, $\boldsymbol{\beta}^j\in\mathbb{R}^3$, $p:\mathbb{R}^3\to\mathbb{R}$ is a trivariate polynomial and the scalar field $\psi:\mathbb{R}^3\to\mathbb{R}$ is defined by a radial basis function $\phi:\mathbb{R}_+\to\mathbb{R}$ as $\psi(\mathbf{x}):=\phi(\|\mathbf{x}\|)$. Although the original paper does not contain the polynomial term, this augmentation is possible by introducing appropriate side-constraints, as we shall explain later.

Macêdo et al. [6] provide sufficient conditions and examples of suitable choices for ϕ satisfying the assumptions made in their

theoretical considerations. Most notably, the Gaussians $\phi_\sigma(r) = \exp(-r^2/2\sigma^2)$ and suitable Wendland's compactly supported functions [30], of which $\phi_\rho(r) = (1-r/\rho)_+^4(4r/\rho+1)$ is the one they employed. It was shown that, by enforcing the interpolation conditions $f(\mathbf{x}^j) = 0$ and $\nabla f(\mathbf{x}^j) = \mathbf{n}^j$ at each sample point, the coefficients in the expression above (without the polynomial term) are uniquely determined and can be recovered by solving the induced symmetric positive definite linear system.

In order to introduce augmenting polynomial terms, which is useful when employing compactly supported RBFs, we need to fix a basis $p_1,\ldots,p_M : \mathbb{R}^3 \to \mathbb{R}$ for these trivariate polynomials, where $M = \binom{d+3}{3}$ and d is their degree; in addition, we need to be sure the only polynomial with at most that degree whose value and gradient are zero at all sample points is the constant zero; we also need the additional side-constraints on the coefficients α_j and $\boldsymbol{\beta}^j$:

$$\sum_{j=1}^{N} \{\alpha_j p_k(\mathbf{x}^j) + \langle \boldsymbol{\beta}^j, \nabla p_k(\mathbf{x}^j) \rangle\} = 0, \quad \forall k = 1,\ldots,M. \tag{2}$$

Together with the interpolation conditions, these constraints result in a symmetric (indefinite) linear system with $4N+M$ variables that is guaranteed to have a unique solution for every (pairwise-different) sample points and any prescribed normals.

In this work, we use a radial function that does not satisfy the strict conditions presented in [6], the triharmonic $\phi(r) \coloneqq r^3$. However, it was shown by Duchon in his seminal paper [31] (and exploited in [29]) that, for this choice of basis function and linear augmenting polynomials ($d = 1$), the resulting Hermite interpolation system is well posed for any set of (pairwise-different) sample points. Moreover, the recovered implicit function above will minimize a suitable generalization of the *thin-plate energy* for Hermite problems in $\mathbb{R}^3$.

8. Results and discussion

Our NPR techniques successfully depict shape and tone of HRBF Implicits by extracting and rendering silhouettes with hidden-line attenuation, stippling, and hatching following principal curvatures and combing directions. All the results were generated on an 2.67 GHz Intel i7 920, 6 gigabyte of RAM and OpenGL/nVIDIA GeForce GTX 295 graphics. Timings are presented in Table 1 for models representing a variety of subjects. Our results were generated with point samples from standard 3D meshes (*Stanford bunny, Heart, Gargoyle, Lamp, Elephant, Horse, Hand, Jaw, Venus* and *David*), parametric surfaces (*Tori* and *Knot*), height-maps (*Terrain*) and implicit surfaces (*Sphere* and *Duck*). All pre-processing and run-time rendering were computed on the CPU only. No GPU programming was used.

Table 1
Time (in seconds) given a drawing direction (C for combing, P for principal directions of curvature); number of samples, stippling and hatching marks, and frames per second.

Model	Dir.	Samples	Stippling	Hatching	FPS
Sphere (Fig. 10)	C	6 (0.0s)	3.7K (0.1s)	600K (2.7s)	11
Bunny (Fig. 4)	–	256 (0.1s)	180K (34s)	–	33
Terrain (Fig. 22)	P	512 (0.9s)	1,600K (807s)	410K (168s)	3
Duck (Fig. 21)	P	1021 (5s)	6K (6s)	343K (284s)	16
Jaw (Fig. 13)	P	1023 (5s)	131K (120s)	55K (48s)	32
Gargoyle (Fig. 2)	C	1024 (5s)	638K (409s)	163K (55s)	19
Tori (Fig. 14)	P	1024 (5s)	639K (506s)	53K (59s)	45
Heart (Fig. 11)	C	1024 (5s)	127K (130s)	266K (162s)	14
Horse (Fig. 18)	C	1025 (5s)	128K (110s)	266K (145s)	16
Lungs (Fig. 16)	P	1035 (5s)	128K (100s)	165K (118s)	41
Head (Fig. 19)	P	1038 (5s)	8K (10s)	418K (3118s)	15
Knot (Fig. 15)	P	1440 (16s)	179K (207s)	75K (102s)	22
Elephant (Fig. 10)	C	2048 (38s)	10K (15s)	532K (506s)	11
Elephant (Fig. 10)	P	2048 (38s)	10K (15s)	532K (803s)	11
Hand (Fig. 17)	C	2062 (38s)	255K (438s)	330K (425s)	13
Lamp (Fig. 20)	C	2469 (60s)	308K (592s)	128K (206s)	16
David (Fig. 1)	C	4096 (283s)	520k (1789s)	216K (1747s)	9

Table 1 shows that all models used in our experiments are rendered interactively. Also, the pre-processing time depends on the number of points and samples. As expected, more samples result in more complex HRBF computations. In addition, placing render points along principal directions of curvature takes longer than along combing directions.

Our approach produces promising results approximating pen-and-ink styles as found in line drawings executed by hand on models reconstructed from a given small set of point-normal samples. We evaluated our results by observing how close they approximate traditional pen and ink drawings.

Figs. 1 and 19 illustrate the steps for rendering pen–ink drawings using our system in a similar way as found in traditional drawing production (i.e., from initial sketch to finished rendering). Given a set of point-normal samples, our system initially

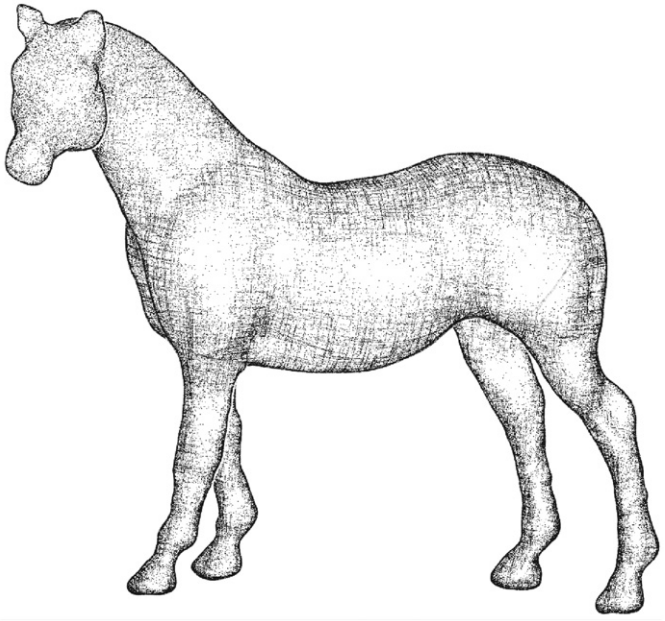

Fig. 18. Horse model rendered using style control: both uniform stippling and combing directions (body) and just uniformly distributed stippling points (head and neck).

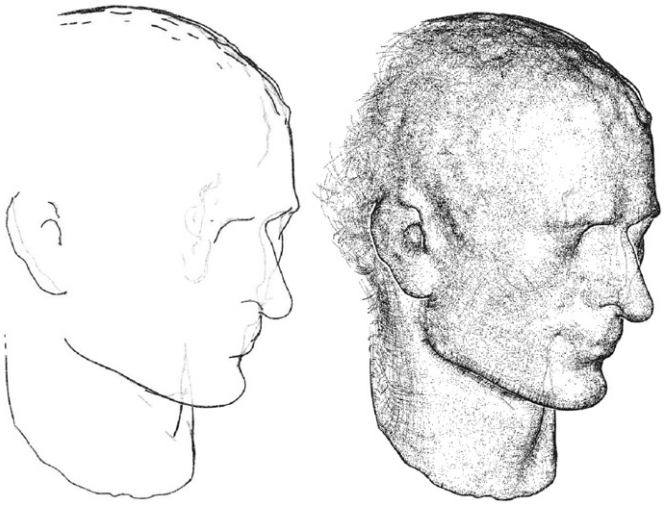

Fig. 19. A head rendered in two different styles.

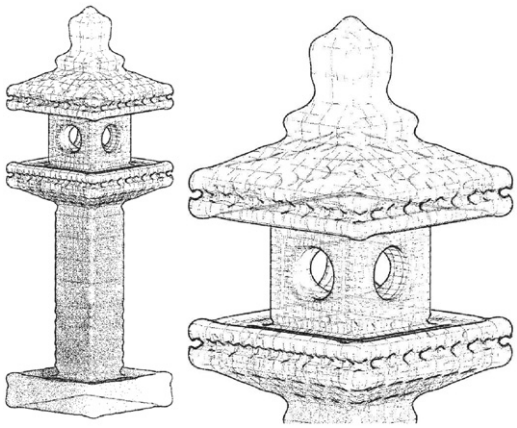

Fig. 20. General and detailed rendering of a garden lamp.

reconstructs the model and allows the user to select rendering techniques providing different levels of visual abstractions for shape and tone depiction using lighting. Fig. 10 provides a comparison between hatching along the combing directions and the principal directions of curvature on the model of an elephant fitted from 2048 samples. Observe the different shape depiction abstractions. Fig. 15 illustrates tone depiction of a knot model rendered using a combination of curvature-based hatching, stippling and a slight attenuation of hidden-lines (enhancing the shape depiction). Fig. 16 shows a blend among uniform point stippling and some combinations between the principal directions of curvature rendered over a lung model. Notice that strokes placed along the first principal direction of curvature depict volumes, while strokes placed along the second direction of curvature direct our eyes along the length of the model [32–35]. Fig. 17 illustrates how a local style control allows the user to direct the focus of attention to a specific region. Fig. 18 depicts a horse model rendered using a style control where both uniform stippling and combing directions are employed on the body and only uniformly distributed stippling points are placed above the neck. Fig. 20 illustrates a garden lamp model with planar regions. Notice that sharp features are adequately rendered. Fig. 21 shows a sketch-based model of a duck [29] fitted from coarsely sampled sparse 3D curves and rendered using our system. Fig. 22 shows a Canyon terrain model (512 point-normal samples) rendered in different ink-based styles, properly depicting both shape and tone.

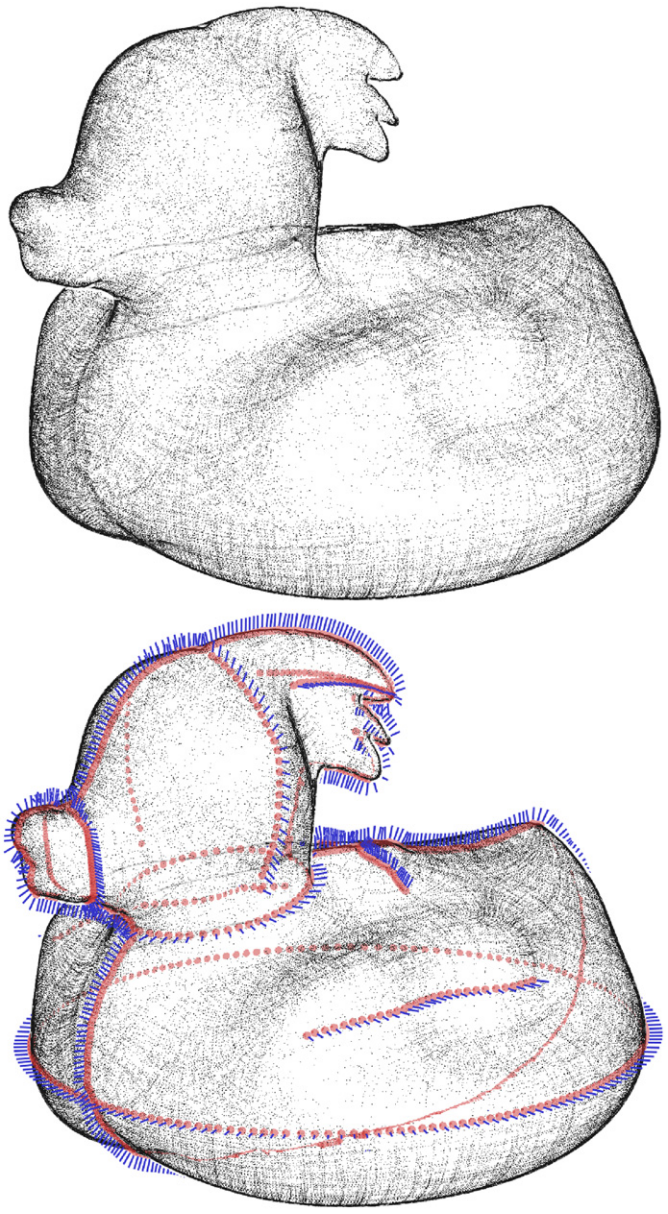

Fig. 21. Sketch-based model of a duck rendered using our system.

9. Conclusions and future work

We presented a completely point-based approach for depicting shape and tone in models represented as implicit surfaces. For this representation, we employ the recently introduced HRBF Implicits for their good properties in reconstructing implicit models from few samples consisting of points and their associated normals. Among the features of our approach, the most salient issues regard our strategies for placing initial seeds, a multilevel refinement of points over the surface, choices of refinement directions suggesting lines drawn along principal directions of curvature, a tiled direction field (the combing direction), a smoother curvature-independent choice of directions, and also new approaches to depict shape and tone by implementing, in a simple manner and directly over the HRBF Implicit model, ink-based NPR techniques including silhouettes, attenuated hidden-lines and lighting tones by stippling and hatching in which all of these can be combined into complex renderings with artist-controlled tools and operators. Our approach demands an initial preprocessing that densely samples the implicit surface; however, after this step, both camera and lighting parameters can be changed still at interactive rates on a single core of modern CPUs.

There are still many avenues for further improvement one may explore: the pre-processing step may be made faster by exploiting the parallelizability of the seed placement and point refinement. Both the evaluation of the HRBF Implicits interpolant and the remotion of render points could be implemented in graphics hardware while the HRBF fitting could be made in the CPU (even exploiting domain decomposition for parallel processing of large sample sets). Also, even though the basic geometric operators on which we rely in designing our pipeline are general enough to be implemented for different representations, we have only experimented with models based on HRBF Implicits. We plan to experiment with data and representations from different application domains to further evaluate the suitability of our method. Also, specific stylization effects for individual drawing primitives is an interesting and important topics to investigate and integrate in our system [36,5]. Finally, a more formal evaluation with trained artists

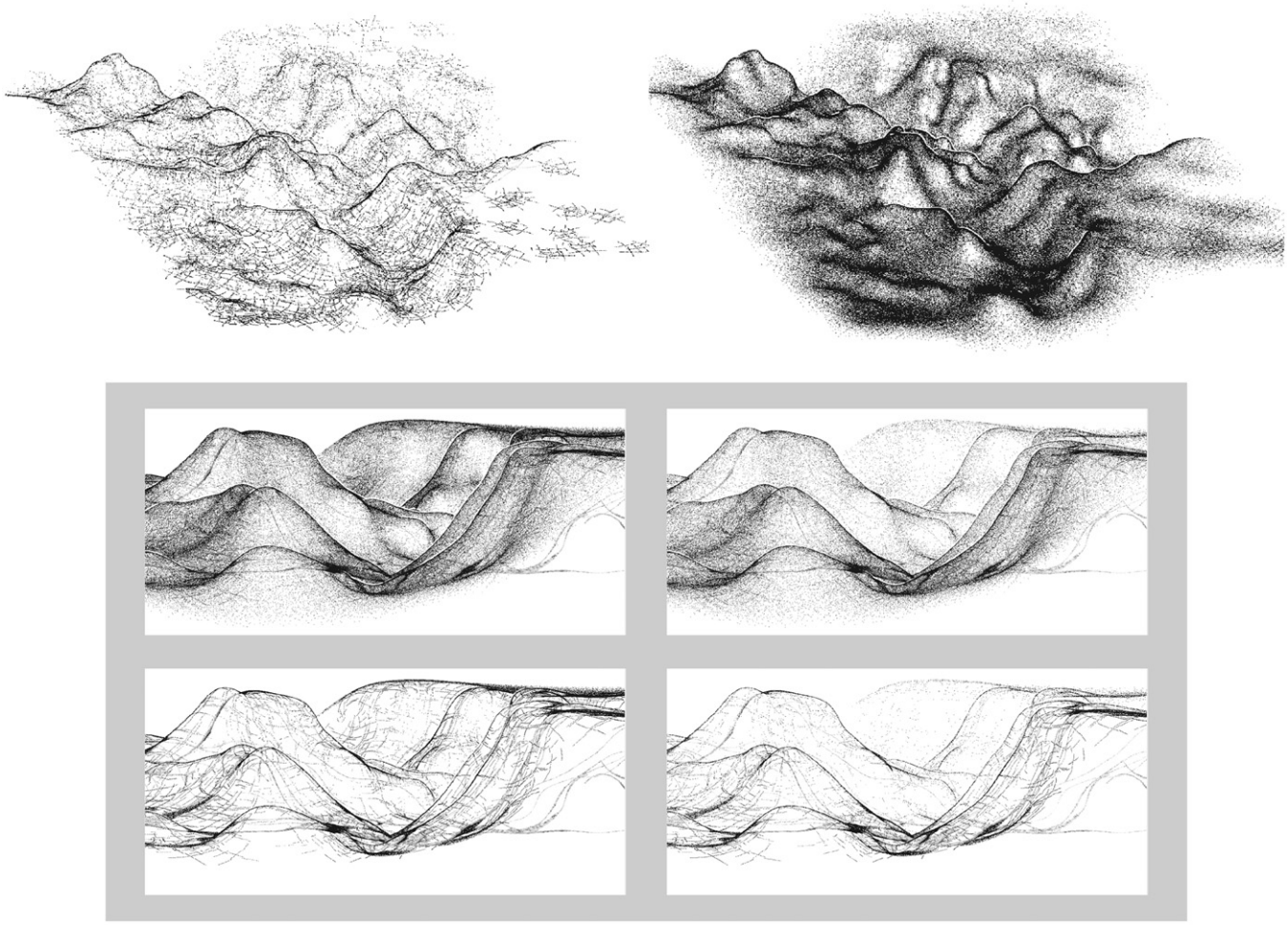

Fig. 22. Shape and tone depiction of the Canyon terrain model in different pen and ink styles using our system. In the bottom frame we compare the light effect (left) with the light and depth attenuation effect (right). The model has 512 samples, 2M render points and with CPU rendering at 8 fps.

and illustrators should be performed and might indicate directions for further usability investigation.

Acknowledgements

We would like to thank the Digital Michelangelo Project, Stanford University for the David model, the Stanford Computer Graphics Laboratory, NTU 3D Model Database, the AIM@SHAPE project, and Rich Pito (University of Pennsylvania, GRASP Lab) who made the other models available for use in this paper. Many thanks to Nicole Sultanum and Patricia Rebolo Medici for their useful discussions and advice. We also thank the anonymous reviewers for their careful and valuable comments and suggestions. This research was supported in part by the iCORE/Foundation CMG Industrial Research Chair in Scalable Reservoir Visualization, by Discovery Grants Program from the Natural Sciences and Engineering Research Council of Canada, and grants from the Brazilian funding agencies CNPq and CAPES/PDEE.

Appendix A. Variational HRBF implicits

In this work, we employ the variational HRBF Implicits presented in [29] to instantiate our pipeline for rendering implicit surfaces. For the sake of completeness and ease of reference, we provide below all the formulas needed to assemble the interpolation system and to evaluate the implicit function as well as its gradient.

In order to assemble the interpolation system, we employ a direct implementation of the block matrix defined by a samplewise grouping of the conditions $f(\mathbf{x}^i)=0$ and $\nabla f(\mathbf{x}^i)=\mathbf{n}^i$, where f is given in (1), and by the side-conditions for degree-one polynomials in (2). This results in the following set of equations:

$$\begin{bmatrix} 0 \\ \mathbf{n}^i \end{bmatrix} = \sum_{j=1}^{N} \begin{bmatrix} \psi(\mathbf{x}^i-\mathbf{x}^j) & -\nabla\psi(\mathbf{x}^i-\mathbf{x}^j)^T \\ \nabla\psi(\mathbf{x}^i-\mathbf{x}^j) & -H\psi(\mathbf{x}^i-\mathbf{x}^j) \end{bmatrix} \begin{bmatrix} \alpha_j \\ \boldsymbol{\beta}^j \end{bmatrix} + \begin{bmatrix} 1 & (\mathbf{x}^i)^T \\ \mathbf{0} & I_{3\times 3} \end{bmatrix} \begin{bmatrix} b \\ \mathbf{a} \end{bmatrix}$$

$$\begin{bmatrix} 0 \\ \mathbf{0} \end{bmatrix} = \sum_{j=1}^{N} \begin{bmatrix} 1 & \mathbf{0}^T \\ \mathbf{x}^j & I_{3\times 3} \end{bmatrix} \begin{bmatrix} \alpha_j \\ \boldsymbol{\beta}^j \end{bmatrix}$$

where the unknowns $\{\alpha_j,\boldsymbol{\beta}^j\}_{j=1}^N$ are computed after an LDL^T-factorization of the resulting symmetric (indefinite) matrix and subsequent forward- and backward-substitutions as implemented in the DSYSV routine from the LAPACK library [37]. The concrete formulas for the functions used in the subblocks above are given by

$$\psi(\mathbf{x}) = \|\mathbf{x}\|^3, \quad \nabla\psi(\mathbf{x}) = 3\mathbf{x}\|\mathbf{x}\|,$$

$$H\psi(\mathbf{x}) = \frac{3}{\|\mathbf{x}\|}(\|\mathbf{x}\|^2 I_{3\times 3} + \mathbf{x}\mathbf{x}^T),$$

where $H\psi(\mathbf{0}) \coloneqq \mathbf{0}_{3\times 3}$ to ensure its continuity.

Appendix B. Approximate projection operator

In order to compute an approximation to the projection of a point $\mathbf{p} \in \mathbb{R}^3$ onto the implicit surface $\mathcal{S} = f^{-1}(0)$, we employ a simple gradient-based iterative method with backtracking. Our iteration is a specialization of the method of *steepest descent with successive stepsize reduction* [38] applied to the problem

$$\min_{\mathbf{x} \in \mathbb{R}^3} \frac{1}{2}(f(\mathbf{x}))^2.$$

This iterative method has two parameters $\delta \in (0,1)$ and $\sigma \in (0, \frac{1}{2})$ and is fully determined by the sequence

$$\mathbf{p}^0 = \mathbf{p}, \quad \mathbf{p}^{k+1} = \mathbf{p}^k - \delta^{i_k} \frac{f(\mathbf{p}^k)}{\|\nabla f(\mathbf{p}^k)\|^2} \nabla f(\mathbf{p}^k),$$

where i_k is the smallest nonnegative integer such that

$$(f(\mathbf{p}^{k+1}))^2 \leq (f(\mathbf{p}^k))^2 [1 - 2\sigma\delta^{i_k}],$$

which defines the backtracking and specializes the *Armijo Rule* for successive stepsize reduction (with $1/\|\nabla f(\mathbf{p}^k)\|^2$ as a first approximation to the stepsize) [38].

In our experiments, we used $\delta = 0.1$ and $\sigma = 10^{-2}$ and we say the projection finished successfully as soon as $|f(\mathbf{p}^k)| < 10^{-6}$ and in failure when that condition is not satisfied up to 16 iterations or $\|\nabla f(\mathbf{p}^k)\| < 10^{-6}$, in which case we simply discard the point. It must be noted that we have not experienced failure with the choice of parameters above.

References

[1] Andrews WM. Introduction to perceptual principles in medical illustration: lines and illusions. Tutorial notes, illustrative visualization for medicine and science (Eurographics '06) 2006.
[2] Lohan FJ. Pen & ink techniques. Contemporary Books, Inc.; 1978.
[3] Smith JA. The pen and ink book: materials and techniques for today's artist. Watson-Guptill Publications; 1992.
[4] Deussen O. Aesthetic placement of points using generalized Lloyd relaxation. In: Proceedings of 5th international symposium on computational aesthetics in graphics, visualization and imaging (CAe'09). Eurographics Association; 2009. p. 123–8.
[5] Kim SY, Maciejewski R, Isenberg T, Andrews WM, Chen W, Sousa MC, et al. Stippling by example. In: Proceedings of 7th international symposium on non-photorealistic animation and rendering (NPAR'09); 2009. p. 41–50.
[6] Macêdo I, Gois JP, Velho L. Hermite interpolation of implicit surfaces with radial basis functions. In: Proceedings of XXII Brazilian symposium on computer graphics and image processing (SIBGRAPI '09); 2009. p. 1–8.
[7] Bremer D, Hughes JF. Rapid approximate silhouette rendering of implicit surfaces. In: Proceedings of implicit surfaces '98; 1998. p. 155–64.
[8] Elber G. Line art illustrations of parametric and implicit forms. IEEE Transactions on Visualization and Computer Graphics 1998;4(1):71–81.
[9] Foster K, Jepp P, Wyvill B, Sousa MC, Galbraith C, Jorge JA. Pen-and-ink for blobtree implicit models. Computer Graphics Forum (Eurographics '05) 2005;24(3):267–76.
[10] Jepp P, Wyvill B, Sousa MC. Smarticles for sampling and rendering implicit models. In: Proceedings of 4th theory and practice of computer graphics (TPCG '06). Berlin: Springer; 2006. p. 39–46.
[11] Jepp P, Denzinger J, Wyvill B, Sousa MC. Using multi-agent systems for sampling and rendering implicit surfaces. In: Proceedings of XXI Brazilian symposium on computer graphics and image processing (SIBGRAPI '08); 2008. p. 255–62.
[12] Schmidt R, Isenberg T, Jepp P, Singh K, Wyvill B. Sketching, scaffolding, and inking: a visual history for interactive 3D modeling. In: Proceedings of the 5th international symposium on non-photorealistic animation and rendering (NPAR '07); 2007. p. 23–32.
[13] Ricci A. A constructive geometry for computer graphics. The Computer Journal 1973;16(2):157–60.
[14] Rosten E, Drummond T. Rapid rendering of apparent contours of implicit surfaces for realtime tracking. British Machine Vision Conference, 2003. p. 719–28.
[15] Burns M, Klawe J, Rusinkiewicz S, Finkelstein A, DeCarlo D. Line drawings from volume data. ACM Transactions on Graphics (SIGGRAPH '05) 2005;24(3): 512–8.
[16] Plantinga S, Vegter G. Computing contour generators of evolving implicit surfaces. ACM Transactions on Graphics 2006;25(4):1243–80.
[17] Stroila M, Eisemann E, Hart J. Clip art rendering of smooth isosurfaces. IEEE Transactions on Visualization and Computer Graphics 2008;14(1):135–45.
[18] Proença J, Jorge J, Sousa M. Sampling point-set implicits. In: Proceedings of 4th IEEE/eurographics symposium on point-based graphics (PBG '07); 2007. p. 11–8.
[19] Proença J, Jorge J, Sousa M. Suggestive contours over point-set implicits. In: Proceedings of 3rd international conference on computer graphics theory and applications (GRAPP '08); 2008. p. 171–80.
[20] Akleman E. Implicit painting of CSG solids. In: Proceedings of CSG '98 set-theoretic solid modelling; 1998. p. 99–113.
[21] Jepp P, Araujo BD, Jorge J, Wyvill B, Sousa MC. Style nodes for hierarchical tree-based implicit surface modelling. In: Proceedings of 5th international symposium on computational aesthetics in graphics, visualization and imaging (CA'09); 2009. p. 41–8.
[22] Wyvill B, Guy A, Galin E. Extending the csg tree-warping, blending, and boolean operations in an implicit surface modeling system. Computer Graphics Forum 1999;18(2):149–58.
[23] Sederberg TW. Piecewise algebraic surface patches. Computer Aided Geometric Design 1985;2(1):53–9.
[24] Bloomenthal J, Shoemake K. Convolution surfaces. SIGGRAPH Comput Graph 1991;25(4):251–6.
[25] Meyer MD, Georgel P, Whitaker RT. Robust particle systems for curvature dependent sampling of implicit surfaces. In: International conference on shape modeling and applications, 2005. p. 124–33.
[26] Stark MM. Efficient construction of perpendicular vectors without branching. Journal of Graphics, GPU,& Game Tools 2009;14(1):55–62.
[27] Kindlmann G, Whitaker R, Tasdizen T, Moller T. Curvature-based transfer functions for direct volume rendering: methods and applications. In: Proceedings of 14th IEEE visualization (VIS' 03); 2003. p. 513–20.
[28] Barla P, Thollot J, Markosian L. X-toon: an extended toon shader. In: Proceedings of 4th international symposium on non-photorealistic animation and rendering (NPAR '06); 2006. p. 127–32.
[29] Vital Brazil E, Macêdo I, Sousa MC, de Figueiredo LH, Velho L. Sketching variational Hermite-RBF Implicits. In: Proceedings of the sketch based interfaces and modeling; 2010. p. 1–8.
[30] Wendland H. Piecewise polynomial positive definite and compactly supported radial functions of minimal degree. Advances in Computational Mathematics 1995;4(1):389–96. doi:10.1007/BF02123482.
[31] Duchon J. Splines minimizing rotation-invariant semi-norms in Sobolev spaces. In: Constructive theory of functions of several variables, Lecture notes in mathematics, vol. 571. Berlin, Heidelberg: Springer; 1977. p. 85–100.
[32] Rawson P. Drawing. University of Pennsylvania Press; 1987.
[33] Goldstein N. The art of responsive drawing. Prentice-Hall; 1999.
[34] Hertzmann A, Zorin D. Illustrating smooth surfaces. In: Proceedings of SIGGRAPH '00; 2000. p. 517–26.
[35] Sousa M, Samavati F, Brunn M. Depicting shape features with directional strokes and spotlighting. In: Proceedings of computer graphics international CGI '04; 2004. p. 214–21.
[36] Xu H, Chen B. Stylized rendering of 3D scanned real world environments. In: Proceedings of the 3rd international symposium on non-photorealistic animation and rendering (NPAR '04); 2004. p. 25–34.
[37] LAPACK—Linear Algebra PACKage. Last visit in August 2010 ⟨http://netlib.org/lapack/⟩.
[38] Bertsekas DP. Nonlinear programming. 2nd ed. Athena Scientific; 1999.