Noname manuscript No.
(will be inserted by the editor)

Fluid-based Hatching for Tone Mapping in Line Illustrations

Afonso Paiva · Emilio Vital Brazil · Fabiano Petronetto · Mario Costa Sousa

Received: date / Accepted: date

Abstract This paper presents a novel meshless, physically-based framework for line art rendering of surfaces with complex geometry and arbitrary topology. We apply an inviscid fluid flow simulation using Smoothed Particles Hydrodynamics to compute the global velocity and cross fields over the surface model. These fields guide the automatic placement of strokes while extracting the geometric and topological coherence of the model. Target tones are matched by tonal value maps allowing different hatching and cross-hatching effects. We demonstrate the simplicity and effectiveness of our method with sample renderings obtained for a variety of models.

Keywords Non-Photorealistic Rendering · Pen and Ink Hatching · Direction Fields · Smoothed Particles Hydrodynamics · Computational Fluid Dynamics.

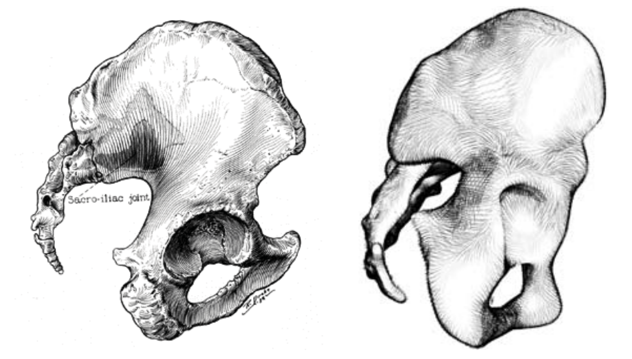


Fig. 1 (left) Traditional medical illustration ("Lumbosacral and Sacroiliac Fusion"), by Russell Drake. Copyrighted and used with permission of Mayo Foundation for Medical Education and Research. (right) Model rendered with our system.

1 Introduction

Line illustrations, produced by using either traditional pen and ink techniques or digital media, allow high-levels of abstraction when depicting shapes, textures and tones of different subjects [1,2]. One of the simplest techniques used to create quantized tonal or shading effects is hatching, in which closely spaced parallel lines are carefully arranged and drawn to match specific target tones. An example is given in Figure 1(left), showing a traditional medical illustration using hatching to depict tone and shape features of the pelvis.

The study of vector fields is broadly used in Computer Graphics, including texture synthesis [3,4], fluid simulation [5,6] and non-photorealistic rendering (NPR) [7–9]. In NPR, traditional hatching methods based on vector fields [4,7–10] are limited to deal only with cases where the input model can be considered as a smooth surface (at least class C^2), because, in these methods, the vector fields are generated using differential quantities like principal curvatures, normal field and geodesic paths from classical differential geometry. However, the smooth surfaces are represented by a smooth polygonal mesh, i.e., these meshes have a well-defined topological map (connectivity) between its polygons (triangles).

We propose a novel physically-based framework for line art rendering of surfaces with complex geometry and arbi-

Afonso Paiva
ICMC (Institute of Mathematics and Computer Science), USP, Brazil
E-mail: apneto@icmc.usp.br

Emilio Vital Brazil
IMPA (Institute of Pure and Applied Mathematics), Brazil
E-mail: emilio@impa.br

Fabiano Petronetto
Department of Mathematics, PUC-Rio, Brazil
E-mail: fbipetro@mat.puc-rio.br

Mario Costa Sousa
Department of Computer Science, University of Calgary, Canada
E-mail: mario@cpsc.ucalgary.ca

trary topology inspired by a particle-based fluid simulation. Unlike the traditional methods, we replace the usual differential geometry approach by a physical approach, in other words, the smooth vector fields are computed using differential quantities (e.g. velocity) of a fluid flow simulation, instead of a curvature field and normal field approximations. These approximations are strongly dependent on the geometry and the connectivity of the input mesh model. Therefore, the proposed framework allows us to take the input model as a non-meshed model, i.e., an unordered collection of triangles which is often called a *triangle soup*.

In our method, the line placement is achieved using an inviscid fluid model to compute the direction fields over the surfaces. One of the seminal works on fluid flow simulation on surfaces was presented by Stam in [6], which uses a mesh-based method to simulate fluid flow defined on Catmull-Clark subdivision surfaces. Instead, in our approach, the computational fluid dynamics is performed using the *Smoothed Particle Hydrodynamics* (SPH) method [11].

1.1 Related work

Different approaches for placing pen-and-ink hatching over 3D models have been proposed. Our main focus is on works aiming at mapping tonal values using hatching.

Winkenbach and Salesin [12] introduces *stroke textures*, allowing procedural generation of strokes for hatching a quantized set of tonal values. Building on this work, Praun *et al.* [13] introduces *tonal art maps*, organizing pre-rendered strokes as a sequence of mip-mapped hatching images. These images are then mapped to the model using lapped textures. Hertzmann and Zorin [9] presents an algorithm for line-art rendering of smooth surfaces. They use local curvature of the object to derive a cross field and place hatches and cross hatches on it. They use modified piecewise-smooth subdivision to make the curvature well-defined and nonzero at extraordinary vertices. In in our method, no curvature approximation is needed. Also, instead of computing silhouette curves in several steps, we directly identify particles on the silhouette.

Other approaches use particle systems for modeling and rendering hatching. Elber [7] describes a method for rendering implicit surfaces based on particle systems for both parametric and implicit surfaces. Foster *et al.* [14] and Jepp *et al.* [15] present methods rendering complex implicit objects using techniques which resemble traditional pen-and-ink illustrations, including hatching. Their methods employ a particle system to depict shape features and tonal values on the surface.

Researchers have also explored the use of line drawing to depict and trace shape measures. In NPR, the focus has been mainly on principal direction of curvature to guide the stroke placement process [13,8–10]. Zhang *et al.* [4] presents a vector field design system allowing the user to create a wide variety of vector fields with control of their features based on concepts of geodesic polar maps and parallel transport.

1.2 Approach and contributions

By using a physical model, we create a global field on the surface allowing to extract information about its geometry and topology; therefore, we can obtain a control over the stroke line tracing/marking, and always maintain the geometric and topological coherence with the model. Such control is given by only changing the gravity acceleration.

One advantage on using our method is the generality of the input data representation. Some algorithms are either limited to meshes with a well-defined topological map (mesh connectivity) or to models constructed with specific implicit representations. Besides, in our approach, strokes automatically adapt themselves in a better way to the geometry and topology of the input model due to the physically-based meaning of our method. Also important, our method does not directly depends on the surface geometry. Eventual singularity points on the surface does not create any impediment to our method.

Our work introduces a novel physically-based rendering framework for hatched pen and ink illustrations of surfaces with complex geometry and arbitrary topology. Unlike the previous works, we use an inviscid fluid model to create the hatching line directions over the surface model. Due to the SPH particle approximation of the fluid flow, our hatching framework becomes a meshless method, i.e., there is no need for a topological map between particles and neither between the triangles of the input non-meshed model. For this reason, as far as we know, our hatching framework is the first hatching method based on vector fields purely geometry & topology-free of the input model. Moreover, our framework is well suited to illustrate arbitrary models independent of their representation: manifold or non-manifold, meshed or non-meshed, simple topology or arbitrary topology, single or multiple connected components, smooth surfaces or surfaces with complex geometry (sharp-features).

The paper is organized as follows. We introduce the physical model of the proposed method in the next section. In Section 3, we give a brief overview of the SPH method. In Section 4, we present the novel physically-based method of NPR, pen and ink illustration. Next, we show the illustrations made by our method. Finally, we finish the paper with a discussion of results and future works.

2 Physics Formulation

Traditionally, hatching lines are generated by using either vector fields computed from the discrete principal curvatures on the model's surface [7–9,4] or by using image gradients [16]. We propose a physically-based framework to create direction fields for hatching lines using the velocity vector field of an inviscid fluid flow.

An intuitive physical interpretation of our method is as follows: imagine, given a complete white-color model, one places drops of black ink on its surface, letting the ink from the droplets to slip and slide in parallel along the surface, thus creating, by its velocity vector field, the ink stroke marks for the hatching. Note that we do not store previous information on the particle's position; instead, the intuition is as if we would freeze the image and take a picture of the vector field of this given ink-drop interpretation. In short, hatching lines are given by the velocity of the ink drops slipping and sliding along the object's surface.

The physical laws of an inviscid fluid flow are given by the *Euler equations*, which are correspondent to the Navier-Stokes equations without the viscous term. In this work, we chose the Lagrangian formulation of the Euler equations. Lagrange's approach describes the governing equations from the viewpoint of a moving particle, i.e., the coordinate system moves with the flow, and it can be formulated by the following two equations:

$$\frac{d\rho}{dt} = -\rho\nabla\cdot\mathbf{v} \qquad \frac{d\mathbf{v}}{dt} = -\frac{1}{\rho}\nabla p + \mathbf{g} \tag{1, 2}$$

where t denotes the time, $\mathbf{v}$ the velocity vector field, ρ the fluid density, p the fluid pressure and $\mathbf{g}$ the gravity acceleration vector.

3 SPH approximation scheme

The SPH method is a numerical tool used in the meshless discretization of the governing equations of a physical system. There are many Computer Graphics applications using SPH, such as deformable bodies [17,18], lava flow [19] and fluid flow simulation [5].

The key idea of SPH method in fluid flow simulations, is to discretize the fluid by a set of particles where each particle represents a fluid element and carries physical *attributes* like velocity, pressure, mass, density. These attributes, and their derivatives at point location $\mathbf{x}$, are updated through of discrete convolutions with a compact support kernel function W as follows:

$$f(\mathbf{x}) = \sum_{j\in N(\mathbf{x})} \frac{m_j}{\rho_j} f(\mathbf{x}_j) W(\mathbf{x}-\mathbf{x}_j, h)$$
$$\nabla f(\mathbf{x}) = \sum_{j\in N(\mathbf{x})} \frac{m_j}{\rho_j} f(\mathbf{x}_j) \nabla_{\mathbf{x}} W(\mathbf{x}-\mathbf{x}_j, h)$$
$$\nabla\cdot\mathbf{f}(\mathbf{x}) = \sum_{j\in N(\mathbf{x})} \frac{m_j}{\rho_j} \mathbf{f}(\mathbf{x}_j)\cdot\nabla_{\mathbf{x}} W(\mathbf{x}-\mathbf{x}_j, h)$$

where the set $N(\mathbf{x})$ contains all the particles at distance below the *smoothing length* h from $\mathbf{x}$, j is the particle index, $\mathbf{x}_j$ the particle position, m_j the particle mass and ρ_j the particle density. In this work, we use a piecewise quartic smoothing kernel [11]. Next, we descibe the Euler equations used in our approximation scheme.

3.1 SPH density approximation

The particle approximation of the derivative density is made by the following SPH version of continuity (equation (1)):

$$\frac{d\rho_i}{dt} = \rho_i \sum_{j\in N(\mathbf{x}_i)} \frac{m_j}{\rho_j} (\mathbf{v}_i - \mathbf{v}_j)\cdot\nabla_i W(\mathbf{x}_{ij}, h),$$

where $\mathbf{v}_i$ and $\mathbf{v}_j$ are velocities at particles i and j, respectively, and $\mathbf{x}_{ij} = \mathbf{x}_i - \mathbf{x}_j$. We then update the density ρ_i at particle i using the Euler integration scheme at each time step δt:

$$\rho_i(t+\delta t) = \rho_i(t) + \delta t\frac{d\rho_i}{dt}.$$

3.2 SPH velocity approximation

Since SPH is better suited for compressible fluid, we approximate the incompressible fluid by a weakly compressible fluid through of an equation of state [20] for the pressure

$$p_i = c^2(\rho_i - \rho_0), \tag{3}$$

where p_i is the pressure at particle i, c the speed of sound, which represents the fastest velocity of a wave propagation in that medium, and ρ_0 is a reference density.

After computing the pressure at all particles using equation (3), we can update the pressure term in momentum (equation (2)) at each particle:

$$\frac{1}{\rho_i}\nabla p_i = \sum_{j\in N(\mathbf{x}_i)} m_j\left(\frac{p_i}{\rho_i^2} + \frac{p_j}{\rho_j^2}\right)\nabla_i W(\mathbf{x}_{ij}, h).$$

Finally, we again utilize the Euler scheme to integrate the acceleration of each particle, to obtain the new particle velocity and to update the particle position

$$\mathbf{v}_i(t+\delta t) = \mathbf{v}_i(t) + \delta t \frac{d\mathbf{v}_i}{dt}$$

$$\mathbf{x}_i(t+\delta t) = \mathbf{x}_i(t) + \delta t\, \mathbf{v}_i(t+\delta t).$$

4 Fluid-based hatching field

This section provides details on the four main stages of our physically-based framework to build the two main hatch direction fields (2): initializing the particles, computing the velocity vector field, building the cross field and approximating the artistic rules for illustration.

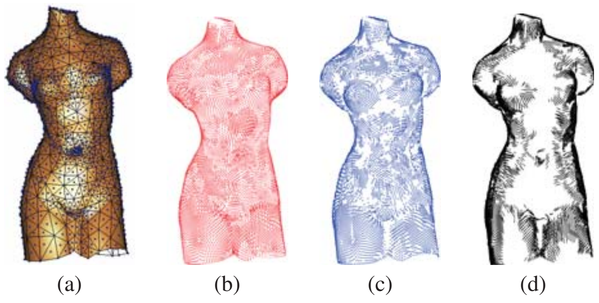
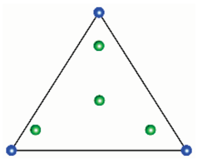


Fig. 2 The main hatch direction fields on the Venus model. (a) Particle sampling; (b) velocity field generated by the SPH fluid equations; (c) cross field produced by the binormal vector of each particle; (d) final result.

4.1 Initialization

This first stage consists on creating a sampling SPH particle set over the input model. We take the sampling set to be at the vertices and the interior points on the triangles of the model.


The creation of the interior points is based on Gaussian quadrature, i.e., the position of the interior points is given by the Gauss points of each triangle as shown in the figure on the left. For particle generation, the sampling SPH particle set is created in each triangle of the input model, taking its vertices (blue) and zero, one or four Gauss points (green), depending on the triangle density. Then, given a triangle defined by its three vertices V_1, V_2 and V_3, a Gauss point $\mathbf{x}$ inside that triangle can be written using barycentric coordinates β_1, β_2 and $\beta_3 = 1-\beta_1-\beta_2$ as $\mathbf{x} = \beta_1 V_1 + \beta_2 V_2 + \beta_3 V_3$. The barycentric coordinates of the interior sampling particles are as follows: for 1 Gauss point, $(\beta_1, \beta_2) = (0.33, 0.33)$, and for 4 Gauss points, $(\beta_1, \beta_2) = (0.33, 0.33), (0.73, 0.13), (0.13, 0.73), (0.13, 0.13)$.

4.2 Velocity field on the surfaces

After we initialize our system, we need a method to create the hatching line directions for the visible portion of the surface model. Traditionally, this is done using a *direction field* [9]. Unlike the traditional vector fields, the direction field does not have any sense of orientation and magnitude.

In our method, we compute the direction field for each particle i by (1) using the SPH fluid equations (Section 3) and (2) taking the direction information using the tangential component of the velocity $\mathbf{v}_i$ over the surface model.

The tangential component is obtained through a collision test between the particles and the triangles of the surface model. Therefore, the tangential velocity $\mathbf{v}_i^{tan}$ is computed by the projection of the velocity $\mathbf{v}_i$ on the plane defined by the triangle T, in which the particle i collides with, as follows:

$$\mathbf{v}_i^{tan} = \mathbf{v}_i - \frac{\langle \mathbf{v}_i, \mathbf{n}_i^T \rangle}{\langle \mathbf{n}_i^T, \mathbf{n}_i^T \rangle} \mathbf{n}_i^T,$$

where $\mathbf{n}_i^T$ is the normal to the triangle T. (The normal of each triangle is computed using the simple right-hand rule.)

We accelerate the above process by storing triangles in the same structure used to search neighboring particles. According to the storage of each particle, this strategy guarantees a constant search time of each triangle for a geometric intersection test with the related particle (sphere). The intersection between particles and triangles is performed by a simplified version of the algorithm proposed by Karabassi *et al.* [21].

Since the hatching lines on the model should not follow arbitrary directions, we use a particle velocity correction to maintain a more ordered motion of particles in absence of viscosity; such correction is called *XSPH velocity correction* (X of unknown) [22]. The XSPH correction consists on computing an average velocity from the velocities of the neighboring particles in the following way:

$$\mathbf{v}_i \leftarrow \mathbf{v}_i + \sum_{j \in N(\mathbf{x}_i)} \frac{m_j}{\rho_i + \rho_j} (\mathbf{v}_j - \mathbf{v}_i)\, W(\mathbf{x}_{ij}, h).$$

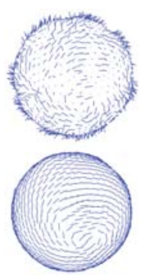
We use an efficient grid hash-based spatial data structure to search neighboring particles called *linked-list search algorithm* [11].

4.3 Constructing the cross field

At this stage, we generate the cross-hatching through another main direction field called *cross field*. The cross field consists on assigning a perpendicular direction to each particle. For this reason, we compute the *binormal vector* of each particle i as $\mathbf{b}_i = \mathbf{v}_i^{tan} \times \mathbf{n}_i$, where $\mathbf{n}_i$ is the particle's surface normal.

In our method, the critical point for constructing cross fields remains on estimating the particle's surface normal. We used a SPH approximation proposed by Müller *et al.* [5]:

$$\mathbf{n}_i = \sum_{j \in N(\mathbf{x}_i)} \frac{m_j}{\rho_j} \nabla_i W(\mathbf{x}_{ij}, h) .$$



However, due to the particles deficiency on the surface model, the above approximation leads to invalid results in the cross field (top sphere on the left). To avoid this problem, we replace the particle's surface normal by the normal to the surface where the particle collides (bottom sphere). Next, to increase the coherence between these normals, we perform the normal smoothing using the SPH approximation as follows

$$\mathbf{n}_i^S = \sum_{j \in N(\mathbf{x}_i)} \frac{m_j}{\rho_j} \mathbf{n}_j^T W(\mathbf{x}_{ij}, h) .$$

Finally, the particle approximation for the binormal vector at particle i is given by $\mathbf{b}_i = \mathbf{v}_i^{tan} \times \mathbf{n}_i^S$.

5 Line-based tone mapping

The art style rules of our rendering method are similar of the method proposed by Hertzmann and Zorin [9]. The hatching lines placement is separated into four levels: highlights (no hatching), midtones (single hatching), shadowed parts (cross-hatching) and silhouettes (thick cross-hatching). These rules are illustrated in Figure 3.

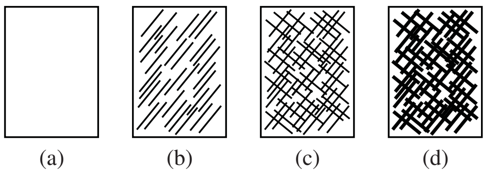


Fig. 3 Hatching rules of the tonal art-map: (a) Highlights; (b) Midtones; (c) Shadows; (d) Silhouettes enhancement.

Once we have the velocity and cross fields, we can apply the hatching rules through a view-dependent vector selection along these fields. This is performed by computing, for each particle i, the dot product between the light direction vector $\mathbf{l}_{dir}$ and the smoothed particle surface normal $\mathbf{n}_i^S$:

$$d_i = -\frac{\langle \mathbf{l}_{dir}, \mathbf{n}_i^S \rangle}{\|\mathbf{l}_{dir}\| \, \|\mathbf{n}_i^S\|} . \qquad (4)$$

Note, the values of d_i are within the range $[-1, 1]$. Moreover, we can use the equation (4) to increase the variety of the maps between tones shown in Figure 3(b) and (c), by solely changing the stroke thickness according to the relation $t_i = 2 - d_i$. Furthermore, our method uses two user-tunable thresholds θ_1 and θ_2, which separates the different hatching levels in the following way:

effect	*condition*	*action*
Highlights:	$d_i > \theta_1$	remove vectors $\mathbf{v}_i^{tan}$ and $\mathbf{b}_i$ (3(a))
Midtones:	$\theta_2 \le d_i < \theta_1$	draw vector $\mathbf{v}_i^{tan}$ (3(b))
Shadows:	$0 < d_i < \theta_2$	draw vectors $\mathbf{v}_i^{tan}$ and $\mathbf{b}_i$ (3(c))
Silhouettes:	$d_i = 0$	draw vectors $\mathbf{v}_i^{tan}$ and $\mathbf{b}_i$ increasing their thickness (3(d))

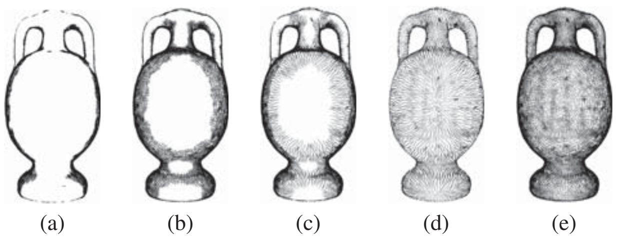


Fig. 4 Our method provides different renderings of amphora model by only changing two parameters. (a) $\theta_1 = 0.3$, $\theta_2 = 0.3$, (b) $\theta_1 = 0.5$, $\theta_2 = 0.5$, (c) $\theta_1 = 0.7$, $\theta_2 = 0.3$, (d) $\theta_1 = 1$, $\theta_2 = 0$, (e) $\theta_1 = 1$, $\theta_2 = 1$

Our method can generate different illustrations by using parameters θ_1 and θ_2, as shown in Figure 4. To improve the performance of our algorithm, we also compute the particle visibility based on the view frustum culling. In this case, we remove occluded particles from the framebuffer when $d_i < 0$.

Finally, the light position is very important in art drawings, as it reveals reveals important features of the model (see Figure 8). For simplicity, we consider the light direction vector $\mathbf{l}_{dir}$ with the same direction and orientation of the gravity vector $\mathbf{g}$.

6 Results and discussion

Results were generated on a 1.86GHz Centrino with 2GB of RAM and a Intel 915GM Express 128MB video card. The results show that our approach produces models depicting their shapes with proper hatching lines arrangements and density placements. Based on our experiments and observations, most of the time spent rendering a model is due to the fluid solver, collision test and the particle density.

We selected 22 three-dimensional models representing a variety of subjects. Our models have, on average, 48,000 triangles, 31,000 sampled particles with average rendering

model	*fig*	*triangles*	*particles*	*render*
Bitorus	*5*	2k	2k	0.01s
Venus	*2*	1k	10k	0.26s
Lungs	*13*	8k	12k	0.42s
Casting	*7*	37k	18k	0.49s
Fertility	*11*	15k	22k	0.50s
Amphora	*4*	19k	15k	0.61s
Twirl	*10*	32k	16k	1.45s
Heart surface	*9*	50k	25k	1.86s
Skull	*13*	57k	29k	2.01s
Hand	*11*	50k	25k	2.04s
Jaw	*13*	48k	24k	2.19s
Inner ear	*12*	52k	26k	2.51s
David	*11*	21k	31k	2.52s
Julius Caesar	*6*	40k	20k	2.75s
Pelvis	*1*	67k	34k	2.82s
Heptoroid	*10*	100k	50k	3.00s
Knot	*7*	54k	27k	3.24s
Bunny	*11*	100k	50k	3.27s
Gargoyle	*11*	104k	50k	3.42s
Horse	*8*	70k	35k	4.16s
Heart model	*13*	12k	85k	7.07s
Ear	*12*	126k	64k	8.92s

Table 1 Results sorted by rendering times.

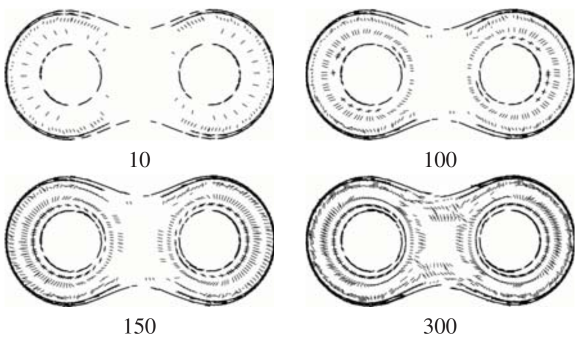


Fig. 5 Evolution of the bitorus surface illustration according to our fluid solver. Number of iterations are shown below each result.

time of 2 seconds (see Table 1). The results show our proposed technique produces images approximating traditional hand-drawn line drawings as found in artistic and scientific illustrations. Our results evaluation is based on observing the visual anesthetics (approximating traditional hatching illustrations), correct tonal value mapping and geometrical/topological coherence of the hatching lines on the model.

The illustrations generated by our method are computed interactively with a few time steps of the fluid solver, which provides a quickly preview of the hatched renderings (Figure 5). The performance and the time-consuming of our algorithm depend exclusively of the number of fluid particles and the number of the triangles of the input model. Rendering time is directly influenced by the particle distribution density, and not by simply the number of particles (Table 1).

The physical attributes of the fluid particles are initialized on the following way: null velocity field, initial fluid density given by $\rho_0 = 1000$, the speed of sound $c = 20$

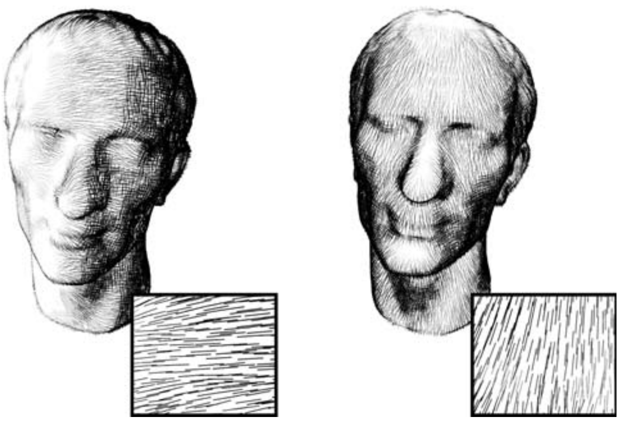


Fig. 6 Controlling the drawing directions by using the gravity acceleration. We create two different illustration styles choosing $\mathbf{g} = (1, 0, 0)$ (left) and $\mathbf{g} = (0, -1, 0)$ (right).

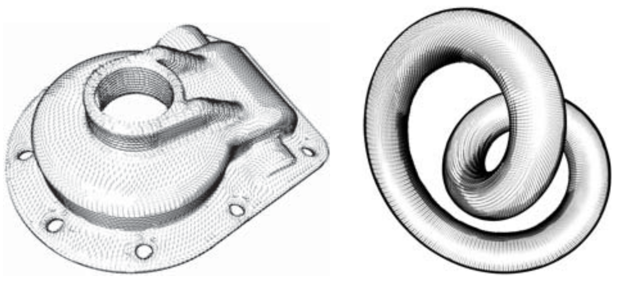


Fig. 7 Casting and Knot models rendered with our technique.

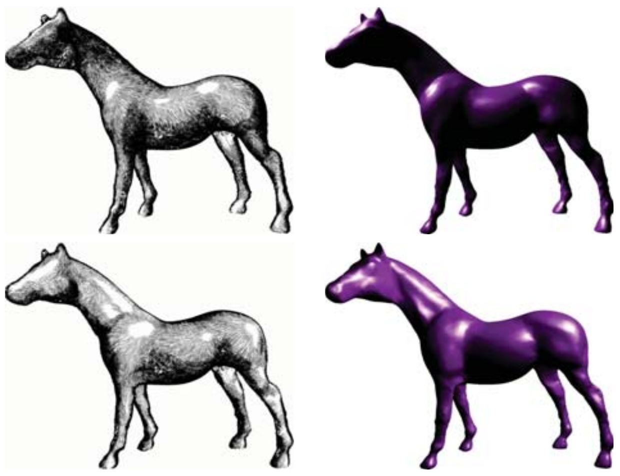


Fig. 8 Changing the spotlight's direction: hatched (left) and shaded (right) horse model

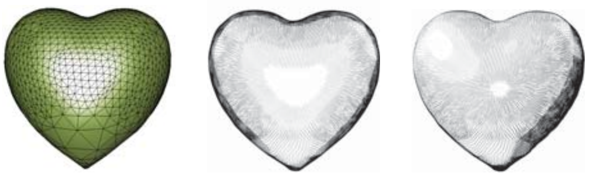


Fig. 9 Heart generated by the implicit function $f(x, y, z) = (2x^2 + y^2 + z^2 - 1)^6 - (0.1x^2 + y^2)z^3$. Left from right: triangulated with the method described in [23] and hatched using different spotlight's directions.

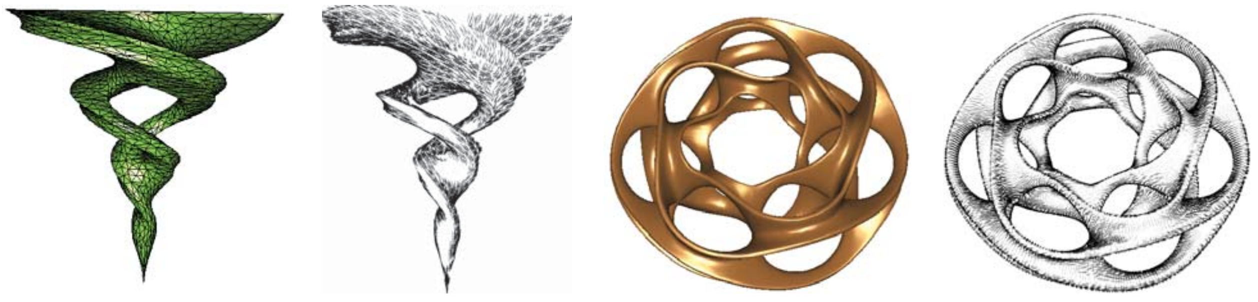


Fig. 10 Twirl and minimal Scherk heptoroid showing shaded and hatched results. Twirl is triangulated with the method described in [24]

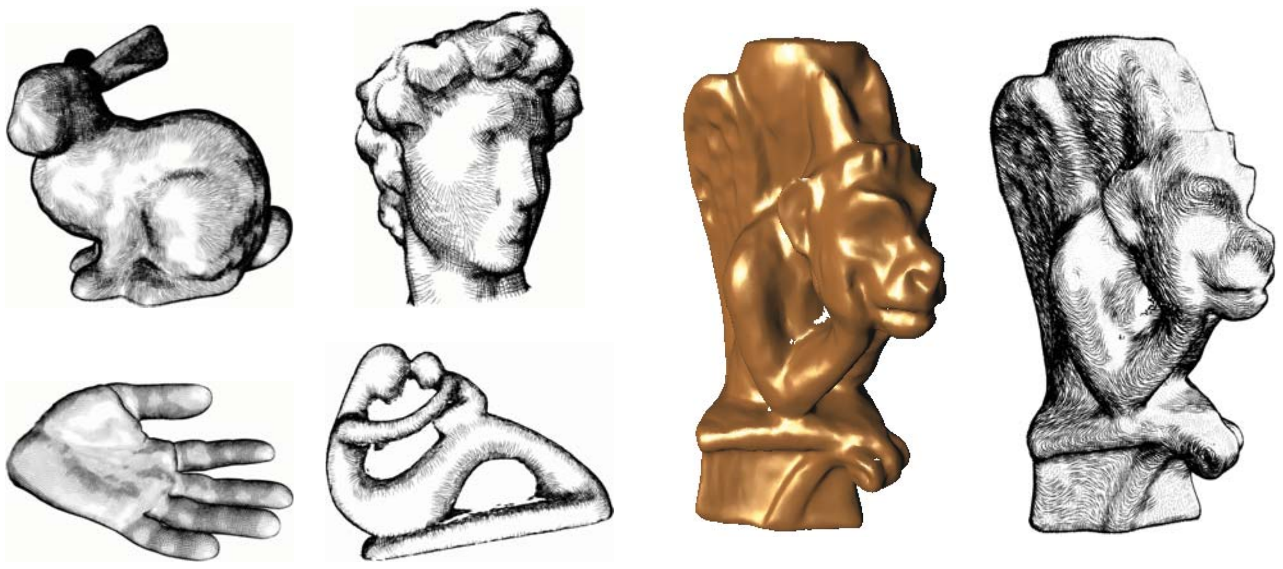


Fig. 11 Five models rendered with our technique. Gargoyle: shaded 3D scanned model (left) and rendered with our technique (right).

and the total mass of the system taken as 20% of the volume of the bounding box of the input model. Note that particle attributes evolution are calculated through SPH, which depends on the neighborhood of each particle; therefore, if we have a dense particle sampling, each particle will, of course, have more neighbors, thus increasing the rendering time.

The proposed method is able to illustrate models with arbitrary topology (Figures 13, 11, 7, 10), multiple connected components (Figures 13), implicit algebraic surfaces (Fig. 9), minimal mathematical surfaces (Fig. 10), tubular structures (Figures 7), non-manifold meshes (Figure 4), large variation of curvature (Figure 11) and sharp features (Figures 10, 7). The information about the models and their particle discretization is given in Table 1.

Our approach allows an efficient generation of hatching directions by simply changing the direction of the gravity acceleration (Figure 6). Our experiments also show we are able to capture those singularity points well, as shown in the example for the Twirl model as well as for topological cases such as the minimal Scherk heptoroid (Figure 10).

The system bottleneck is based on the differential operators discretized in the SPH method through a time-varying, local distribution of particles; searching for neighboring particles is also very important to our method. The dynamic updating of the neighboring particles search structure is a computationally costly task.

7 Conclusions and future work

This paper presents a new NPR technique for drawing pen and ink illustrations using a fluid-based method to compute direction fields on surfaces. Our method relies on the SPH meshless framework, used in the discretization of Euler equation terms. The effectiveness of the method is showed on models with varying geometry and topology complexity. In essence, our approach does not depend on the representation of the input model. This is because we only require a local information on how to construct a tangent plane to be used in the collision test. Therefore, it is not necessary to have any global information about the geometry and topology of the object.

Future improvements include extending our system with support for more artistic and intuitive control over the fluid

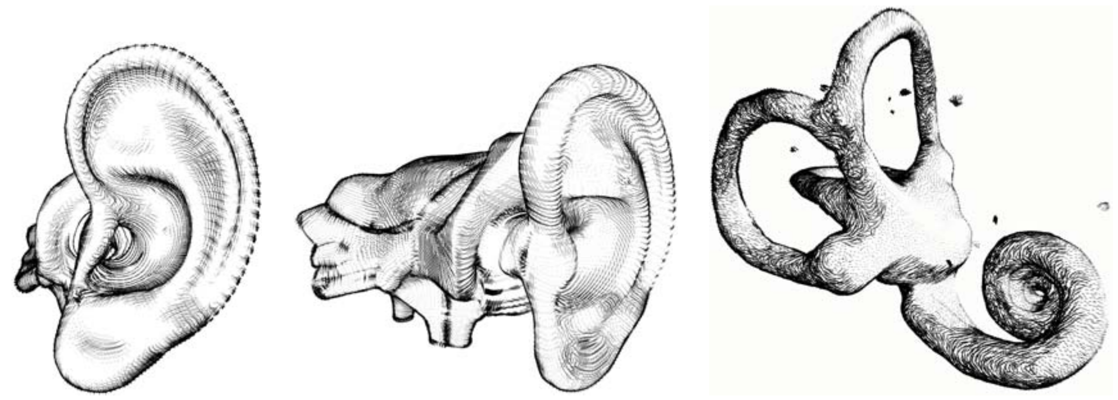


Fig. 12 Ear model in two views and the inner ear structure.

simulation as well as a toolbox with different line art styles. It would also be useful to experiment with different fluid models, adding new physical attributes to each particle, such as ink viscosity/surface tension and pen pressure modulation.

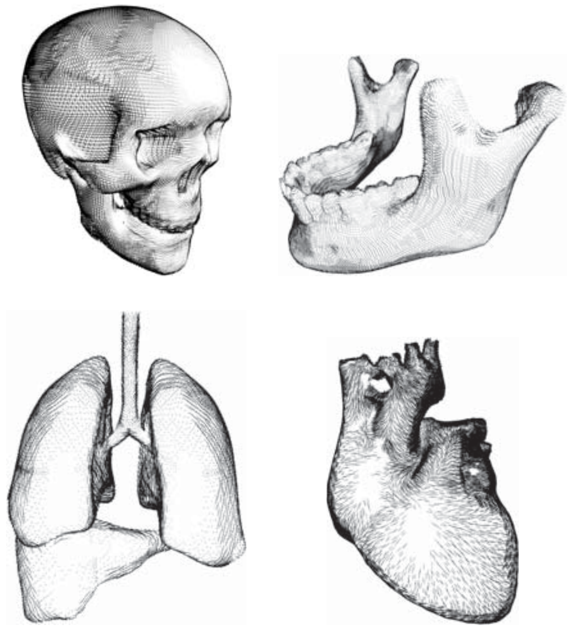


Fig. 13 3D anatomy models rendered with our technique.

Acknowledgements We would like to thank the anonymous reviewers for their careful and valuable comments and suggestions. This research was supported in part by FAPESP (State of São Paulo Research Foundation) under grant n. 2008/00093-0, CNPq (National Council for Scientific Research and Development of Brazil), PETROBRAS (Brazilian Oil Company) and Discovery Grants from NSERC (Natural Sciences and Engineering Research Council of Canada).

References

1. Rawson, P.: Drawing. University of Pennsylvania Press (1987)
2. Hodges, E.: The Guild Handbook of Scientific Illustration, 2nd edn. John Wiley & Sons Inc (2003)
3. Turk, G.: Texture synthesis on surfaces. In: Proc. of SIGGRAPH '01, pp. 347–354 (2001)
4. Zhang, E., Mischaikow, K., Turk, G.: Vector field design on surfaces. ACM Transactions on Graphics **25**(4), 1294–1326 (2006)
5. Müller, M., Charypar, D., Gross, M.: Particle-based fluid simulation for interactive applications. In: Proc. of Symposium on Computer Animation, pp. 154–159 (2003)
6. Stam, J.: Flows on surfaces of arbitrary topology. ACM Transactions on Graphics **22**(3), 724–731 (2003)
7. Elber, G.: Line art illustrations of parametric and implicit forms. IEEE Trans. Vis. Comp. Graph. **4**(1), 71–81 (1998)
8. Girshick, A., Interrante, V., Haker, S., Lemoine, T.: Line direction matters: an argument for the use of principal directions in 3D line drawings. In: Proc. of NPAR '00, pp. 43–52 (2000)
9. Hertzmann, A., Zorin, D.: Illustrating smooth surfaces. In: Proc. of SIGGRAPH '00, pp. 517–526 (2000)
10. Rössl, C., Kobbelt, L., Seidel, H.P.: Line art rendering of triangulated surfaces using discrete lines of curvature. In: Proc. of Winter School of Computer Graphics (WSCG '00), pp. 168–175 (2000)
11. Liu, G.R., Liu, M.B.: Smoothed Particle Hydrodynamics. World Science (2005)
12. Winkenbach, G., Salesin, D.H.: Computer-generated pen-and-ink illustration. In: Proc. of SIGGRAPH '94, pp. 91–100 (1994)
13. Praun, E., Hoppe, H., Webb, M., Finkelstein, A.: Real-time hatching. In: Proc. of SIGGRAPH '01, pp. 579–584 (2001)
14. Foster, K., Jepp, P., Wyvill, B., Sousa, M.C., Galbraith, C., Jorge, J.A.: Pen-and-ink for BlobTree implicit models. Computer Graphics Forum **24**(3), 267–276 (2005)
15. Jepp, P., Wyvill, B., Sousa, M.: Smarticles for sampling and rendering implicit models. In: Proc. of Theory and Practice of Computer Graphics (EG-UK TPCG '06), pp. 39–46 (2006)
16. Secord, A., Heidrich, W., Streit, L.: Fast primitive distribution for illustration. In: Proc. of 13th Eurographics Workshop on Rendering, pp. 215–226 (2002)
17. Keiser, R., Adams, B., Gasser, D., Bazzi, P., Dutré, P., Gross, M.: A unified lagrangian approach to solid-fluid animation. In: Symposium on Point-Based Graphics '05, pp. 125–134 (2005)
18. Paiva, A., Petronetto, F., Lewiner, T., Tavares, G.: Particle-based non-Newtonian fluid animation for melting objects. In: Brazilian Symposium on of Computer Graphics and Image Processing (SIBGRAPI '06), pp. 78–85 (2006)
19. Stora, D., Agliati, P.O., Cani, M.P., Neyret, F., Gascuel, J.D.: Animating lava flows. In: Proc. of Graphics Interface '99, pp. 203–210 (1999)

20. Morris, J.P., Fox, P.J., Zhu, Y.: Modeling low Reynolds number incompressible flows using SPH. Journal of Computational Physics **136**(1), 214–226 (1997)
21. Karabassi, E.A., Papaioannou, G., Theoharis, T., Boehm, A.: Intersection test for collision detection in particle systems. Journal of Graphics Tools **4**(1), 25–37 (1999)
22. Monaghan, J.J.: On the problem of penetration in particle methods. Journal of Computational Physics **82**(1), 1–15 (1989)
23. Paiva, A., Lopes, H., Lewiner, T., de Figueiredo, L.H.: Robust adaptive meshes for implicit surfaces. Brazilian Symposium on Computer Graphics and Image Processing (SIBGRAPI '06) pp. 205–212 (2006)
24. Ohtake, Y., Belyaev, A.: Dual-primal mesh optimization for polygonized implicit surfaces with sharp features. Journal of Computing and Information Science in Engineering **2**(4), 277–284 (2002)