Physically-based hatching for pen-and-ink illustration

Afonso Paiva
ICMC – USP

Emilio Vital Brazil
IMPA

Fabiano Petronetto
Matmídia / PUC–Rio

Mario Costa Sousa
University of Calgary

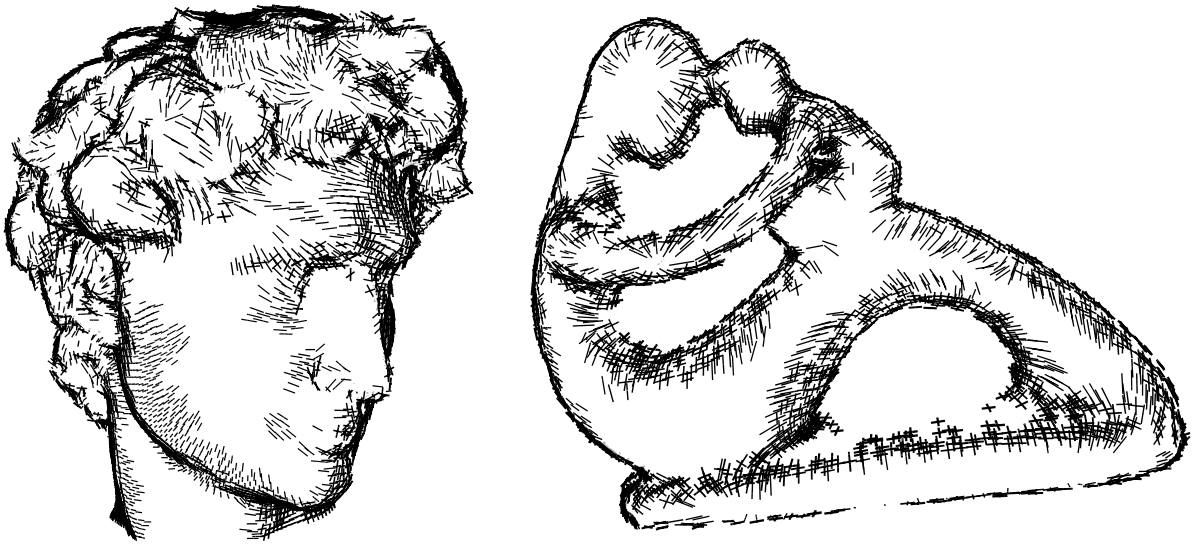


Figure 1. Hatched pen-and-ink illustrations using our fluid-based framework.

Abstract

This paper presents a novel meshless framework for line art rendering of surfaces with complex geometry and arbitrary topology. Unlike the usual frameworks, the line placement is achieved using an inviscid fluid flow simulation to compute the direction fields over the surface model. To perform the computational fluid dynamics, we utilize the Smoothed Particles Hydrodynamics (SPH) method. We demonstrate the simplicity and effectiveness of our method illustrating a variety of complex surfaces.

Keywords: *Pen-and-Ink Illustration, Hatching, Direction Fields, Non-photorealistic Rendering, Smoothed Particles Hydrodynamics, Inviscid Fluids.*

1. Introduction

Pen-and-ink line drawing techniques can simultaneously convey lighting, reveal shapes, suggest material properties, and direct attention using a limited palette of tones. Hatching uses groups of strokes and spatial coherence to create different intensities and tones (see Figure 2).

The study of vector fields is very popular in Computer Graphics and it has many applications, such as texture synthesis [20, 22], fluid simulation [13, 18, 1] and non-photorealistic rendering (NPR) [3, 5, 6]. The traditional hatching methods based on vector fields in NPR literature [3, 5, 6, 22] 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 arbitrary topology inspired by a particle-based fluid simulation (Figure 1). 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

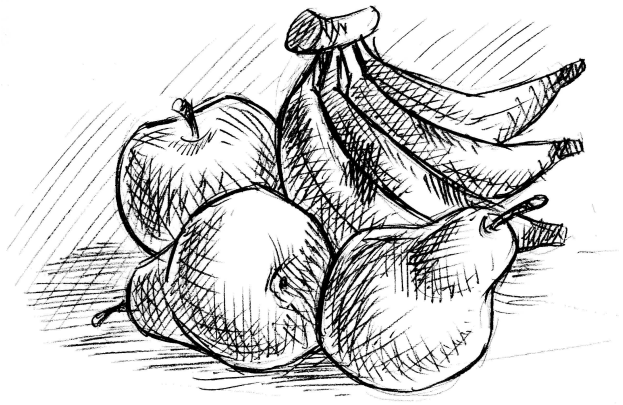


Figure 2. Exemple of real pen-and-ink artistic (courtesy of Fernando Miller).

flow simulation instead of curvature field and normal field approximations. These approximations are strongly dependent of the geometry and the connectivity of the input mesh model. Thus, the proposed framework allows us to take the input model as non-meshed model (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 [18], which uses a mesh-based method to simulate fluid flow defined on Catmull-Clark subdivision surfaces. Instead of this work, the computational fluid dynamics is performed using the *Smoothed Particle Hydrodynamics* (SPH) method.

Related works. Winkenbach *et al.* [16] and Salisbury *et al.* [21] described the principles of traditional pen-and-ink illustration and showed an interactive system that address hatching in still images from 3D scenes. The method presented in this paper works on the object space. Hertzmann and Zorin [6] presented 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. Zhang *et al.* [22] showed a vector field design system that allows 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. Praun *et al.* [15] described a real-time hatching system which allows for smooth transitions between art maps using different texture tone maps. Elber [3] describes a method for rendering implicit surfaces based on particle systems for both parametric and implicit surfaces. Foster *et al.* [4] draws complex implicit objects using techniques that resemble traditional pen-and-ink illustrations, including hatching. Their method employs a particle system to find the interesting areas on the surface. The previous particle-based methods [4, 3] do not have any kind of physical concepts.

Contributions. This 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 not a topological map between particles and neither between the triangles of the input non-meshed model. For this reason, we believe that our hatching framework is the first hatching method based on vector fields purely free of geometry and topology of the input model. Moreover, our framework is well suited to illustrate arbitrary models independent of its representation: manifold or non-manifold, meshed or non-meshed, simple topology or arbitrary topology, smooth surfaces or surfaces with complex geometry (sharp-features).

Paper outline. 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. Section 4 presents the novel physically-based method of illustration. Next, we show the illustrations made by our method. Finally, we finish the paper with a discussion of results and glimpse on future works.

2. Physics formulation

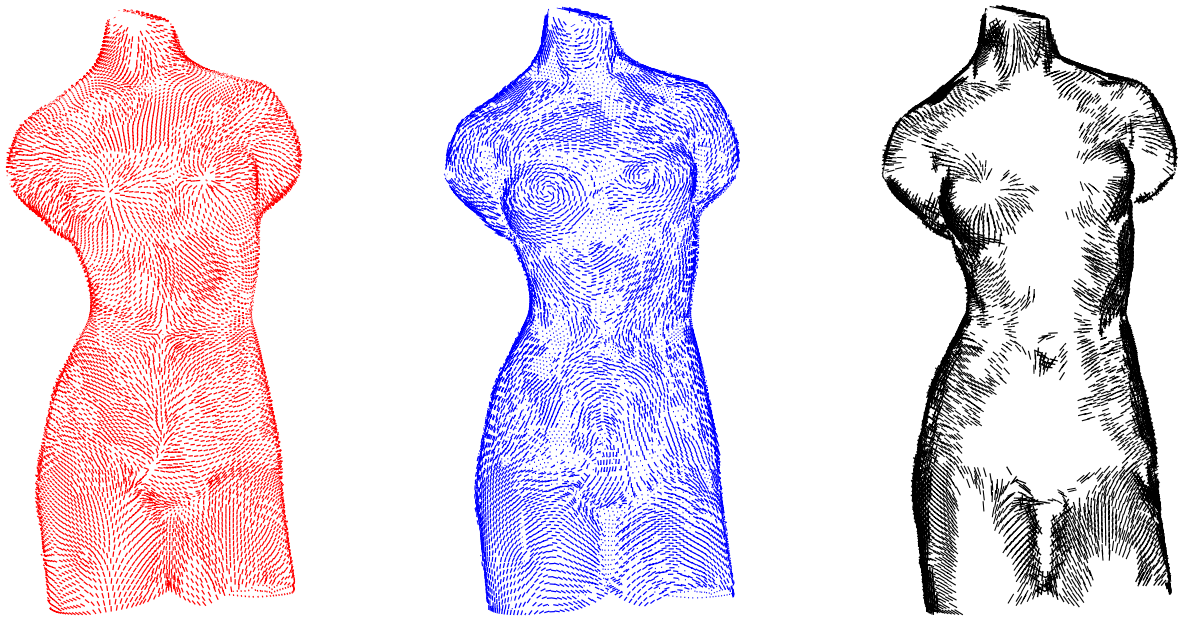
Traditionally, hatching lines are generated through of vector fields computed from the discrete principal curvatures on the model's surface [3, 5, 6, 22] or using image gradient [17]. We propose a physically-based framework to create direction fields for hatching lines using the velocity vector field of an inviscid fluid flow.

The physical laws of an inviscid fluid flow are given by the *Euler equations*, these equations 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., coordinate system moves with the flow, and they can be formulated by the following two equations:

$$\frac{d\rho}{dt} = -\rho\nabla \cdot \mathbf{v} \tag{1}$$

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

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.



(a) Velocity field. (b) Cross field. (c) Final illustration.

Figure 3. The main hatch direction fields on the Venus. (a) Velocity field generated by the SPH fluid equations. (b) Cross field produced by the binormal vector of each particle. (c) Final result.

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 [2, 8], lava flow [19] and fluid flow simulation [13, 14, 1].

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 [1] W as follows:

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

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

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

where the set $N(\mathbf{x})$ contains all the particles at distance below 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.

1 We choose a piecewise quartic smoothing kernel [9].

3.1. SPH Euler equations

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

$$\frac{d\rho_i}{dt} = \rho_i \sum_{j \in N(\mathbf{x}_i)} \left(\mathbf{v}_i - \mathbf{v}_j\right) \cdot \nabla_i W\left(\mathbf{x}_{ij}, h\right) \frac{m_j}{\rho_j}, \qquad (3)$$

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 update the density ρ_i at particle i using the Euler integration scheme in each time step δt as follows:

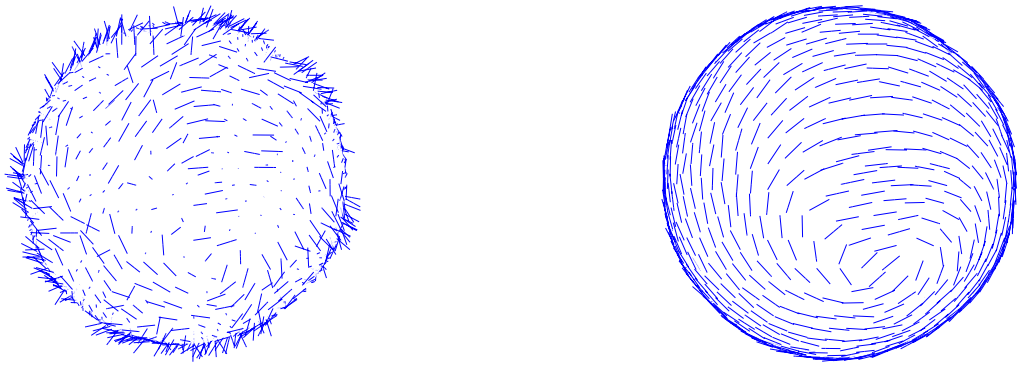
$$\rho_i(t + \delta t) = \rho_i(t) + \delta t \frac{d\rho_i}{dt}.$$

SPH velocity approximation. Since SPH suits better for compressible fluid, we approximate the incompressible fluid by a weakly compressible fluid through of an equation of state [11] for the pressure. In this work, we use an equation of state proposed by Morris *et al.* [12]:

$$p_i = c^2 \left(\rho_i - \rho_0\right) \qquad (4)$$

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 (4), we can update the pressure term in momentum



(a) Particle approximation. (b) Particle-triangle approximation.

Figure 4. Comparing the cross field on the sphere. (a) Cross field produced by the particle approximation. (b) Smooth cross field produced by the particle-triangle approximation.

equation (2) at each particle:

$$\frac{1}{\rho_i}\nabla p_i = \sum_{j \in N(\mathbf{x}_i)} m_j \left(\frac{p_i}{\rho_i^2} + \frac{p_j}{\rho_j^2} \right) \nabla_i W\left(\mathbf{x}_{ij}, h\right). \quad (5)$$

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

$$\mathbf{v}_i(t + \delta t) = \mathbf{v}_i(t) + \delta t \frac{d\mathbf{v}_i}{dt}. \quad (6)$$

4. Fluid-based surface hatching

This section details the four main stages of our physical framework to build the two main hatch direction fields for pen-and-ink illustration (Figure 3): the particle initialization, computing the velocity vector field, building the cross field and the artistic rules for illustration.

Initialization. This first stage consists in to create a sampling SPH particle set over the input model. To perform this task we consider the input model as base mesh and we use Loop subdivision scheme to refine this base mesh. Then, we take vertices of refined mesh as our initial SPH particle set.

Velocity field on surfaces. After we initializate our system, we need a method to create the hatching line directions for the visible portion of the surface model. Traditionally, this is made using a *direction field* [6]. In opposite of 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 using the SPH fluid equations (Section 3.1) and taking the direction information using the projection of the velocity $\mathbf{v}_i$ over the surface model.

This projection is obtained through of a collision test between the particles and the triangles of the surface model. The projected velocity $\mathbf{v}_i^{tan}$ is taken as the tangential component of $\mathbf{v}_i$ provided by the collision response. The intersection between particles and triangles is performed by a simplified version of the algorithm proposed by Karabassi *et al.* [7].

Since the hatching lines on the model should not follow arbitrary directions, we use a particle velocity correction to maintain a more ordered move of particles in absence of viscosity, this correction is called *XSPH velocity correction* (X of unknown) [10]. The XSPH correction consists in computing an average velocity from the velocities of the neighboring particles in the following way:

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

Constructing cross field. In this stage, we generate the cross-hatching trough of another main direction field called *cross field*. The cross field consists in assigning a perpendicular direction to each particle. For this reason, we compute the *binormal vector* for each particle i

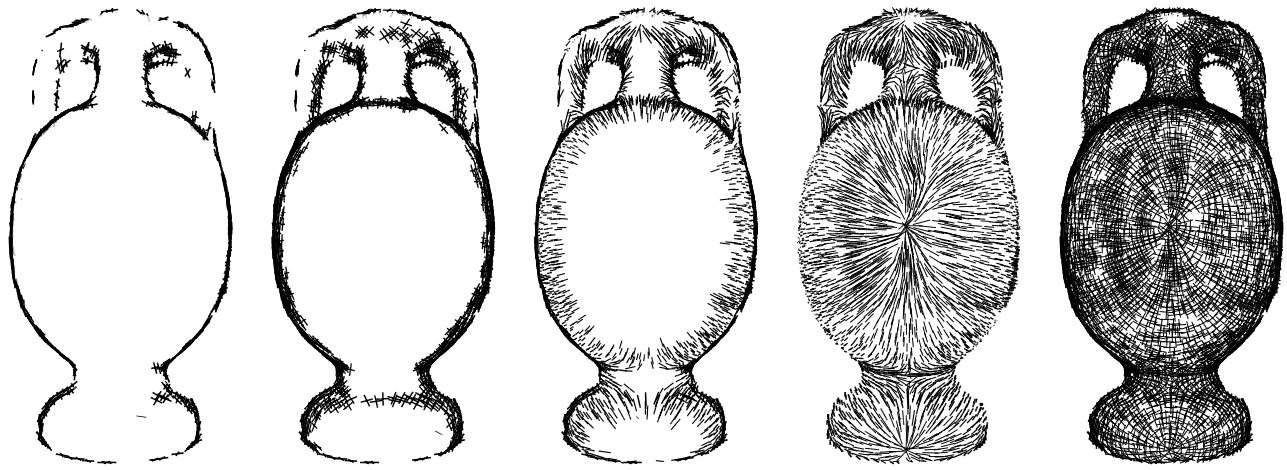
$$\mathbf{b}_i = \mathbf{v}_i^{tan} \times \mathbf{n}_i,$$

where $\mathbf{n}_i$ is the particle surface normal.

The delicate point for constructing cross fields in our method remains in estimating the particle surface normal. It is natural that we utilize a SPH approximation for particle surface normal proposed by Müller *et al.* [13]:

$$\mathbf{n}_i = \sum_{j \in N(\mathbf{x}_i)} \nabla_i W\left(\mathbf{x}_{ij}, h\right) \frac{m_j}{\rho_j}. \quad (7)$$

However, due to the particles deficiency in the surface model the approximation above leads us to spurious results (Figure 4). To avoid this problem, we replace the particle



(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) $\theta_1 = 1, \theta_2 = 1$

Figure 5. Our method provides different drawings of amphora model changing just two parameters.

surface normal by the normal of the surface's triangle wich the particle collides. We compute the normal of each triangle using the simple right-hand rule. Finally, the particle-triangle approximation for the binormal vector at the particle i is given by

$$\mathbf{b}_i = \mathbf{v}_i^{tan} \times \mathbf{n}_i^T,$$

where $\mathbf{n}_i^T$ is the normal of triangle T which the particle i touches.

Surface art style rules. The art style rules of our rendering method are similar of the method proposed by Hertzmann and Zorin [6]. 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 6.

Once we have the velocity field and cross field, we can perform the hatching rules through of a view-dependent vector selection along these fields. This is made computing the dot product between the view vector $\mathbf{d}_{view}$ and the particle surface normal $\mathbf{n}_i^T$ for each particle i:

$$d_i = -\frac{\langle \mathbf{d}_{view}, \mathbf{n}_i^T \rangle}{\|\mathbf{d}_{view}\| \left\|\mathbf{n}_i^T\right\|}.$$

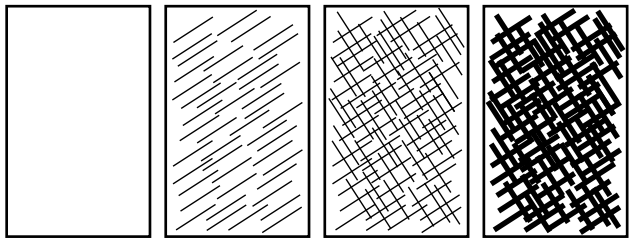
Note that, the values of d_i are in the range $[-1, 1]$.

The light position is very important in art drawings, because the light reveals important features of the model. For simplicity and to keep the natural light position for art illustrators, we consider the light vector with the same direction and orientation of $\mathbf{d}_{view}$.

Furthermore, our method uses two user-tunable thresholds θ_1 and θ_2 that separates the different hatching levels in the following way:

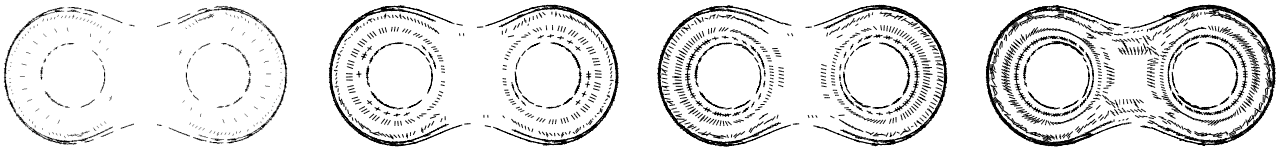
- Highlights: if $d_i > \theta_1$, we remove the vectors $\mathbf{v}_i^{tan}$ and $\mathbf{b}_i$ (Figure 6(a)).
- Midtones: if $\theta_2 \leq d_i < \theta_1$, we draw the vector $\mathbf{v}_i^{tan}$ (Figure 6(b)).
- Shadows: if $0 < d_i < \theta_2$, we draw the vectors $\mathbf{v}_i^{tan}$ and $\mathbf{b}_i$ (Figure 6(c)).
- Silhouettes: if $d_i = 0$, we draw the vectors $\mathbf{v}_i^{tan}$ and $\mathbf{b}_i$ and increasing their thickness (Figure 6(d)).

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 $d_i < 0$.



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

Figure 6. Hatching rules. (a) Highlights. (b) Midtones. (c) Shadowed parts. (d) Silhouettes.



(a) 10 iterations. (b) 100 iterations. (c) 150 iterations. (d) 300 iterations.

Figure 7. Evolution of the bitorus surface illustration according our fluid solver.

5. Results

The illustrations in this paper were generated using our fluid-based framework. The versatility of the our method allows us to create illustrations from arbitrary models (Figures 8(b) and 8(c)) to mathematical implicit surfaces (Figures 7 and 8(a)).

The proposed method is able to illustrate complex models with sharp features (Figure 8(d)) and deals graceful with the complex topology of Chair surface (Figure 8(a)). Note that, the cross-hatching at Kitten model (Figure 8(c)) causes a fur effect at the ears.

The information about the models and their particle discretization is provided by Table 1. In particular, we take the gravity vector $\mathbf{g}$ with the same direction and orientation of $\mathbf{d}_{view}$ in our examples.

Illustration	Number of particles	Number of triangles
David	15000	5000
Fertility	15000	7000
Amphora	10000	5000
Chair	15000	7000
Bitorus	12000	800
Cow	12000	4000
Kitty	16000	8000
Venus	20000	700
Twirl	2600	1300

Table 1. Number of particles and triangles used in the examples.

6. Conclusion & Future Works

In this paper, we have presented a new non-photorealistic rendering method 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 a wide variety of complex examples.

The illustrations generated by our method are computed interactively in a few time steps of the fluid solver, this provides a quickly preview of the drawings (Figure 7). The performance and the time-consuming of our algorithm depend exclusively of the number of fluid particles and the number of the triangles at the surface model.

This work can be extend and improved in three main directions:

Artistic side. We can include in our framework an interactivity with design artist in toward of the fluid simulation becomes controllable. Moreover, we can adding new line styles to create different drawing styles.

Physical side. We can use another fluid model and adding new physical attributes in each particle, such as: ink's viscosity, pen's pressure produced by the artist and ink's surface tension.

Animation side. Finally, we can utilize the time dependency of our fluid framework to produce a coherence between the strokes along the time.

Acknowledgments

The David's head model was courtesy of Stanford Digital Michelangelo Project, the others models used in this paper were provided by the shape repository AIM@SHAPE project (`http://shapes.aim-at-shape.net`).

A. Paiva is member of LCAD lab. at USP – São Carlos, which is sponsored by (Fundação de Amparo a Pesquisa do Estado de São Paulo) under grant n. 2008/00093-0. E.V. Brazil, of Visgraf lab. at IMPA, is sponsored by CNPq (Conselho de Desenvolvimento Científico e Tecnológico). F. Petronetto, of the Matmídia lab. at PUC–Rio, is sponsored by PETROBRAS. M.C. Sousa is member of Department of Computer Science at University of Calgary.

References

[1] S. Clavet, P. Beaudoin, and P. Poulin. Particle-based viscoelastic fluid simulation. In *Symposium on Computer Animation*, pages 219–228, 2005.

[2] M. Desbrun and M. P. Cani. Smoothed particles: A new paradigm for animating highly deformable bodies. In *7th International Workshop on Computer Animation and Simulation*, pages 61–76, 1996.

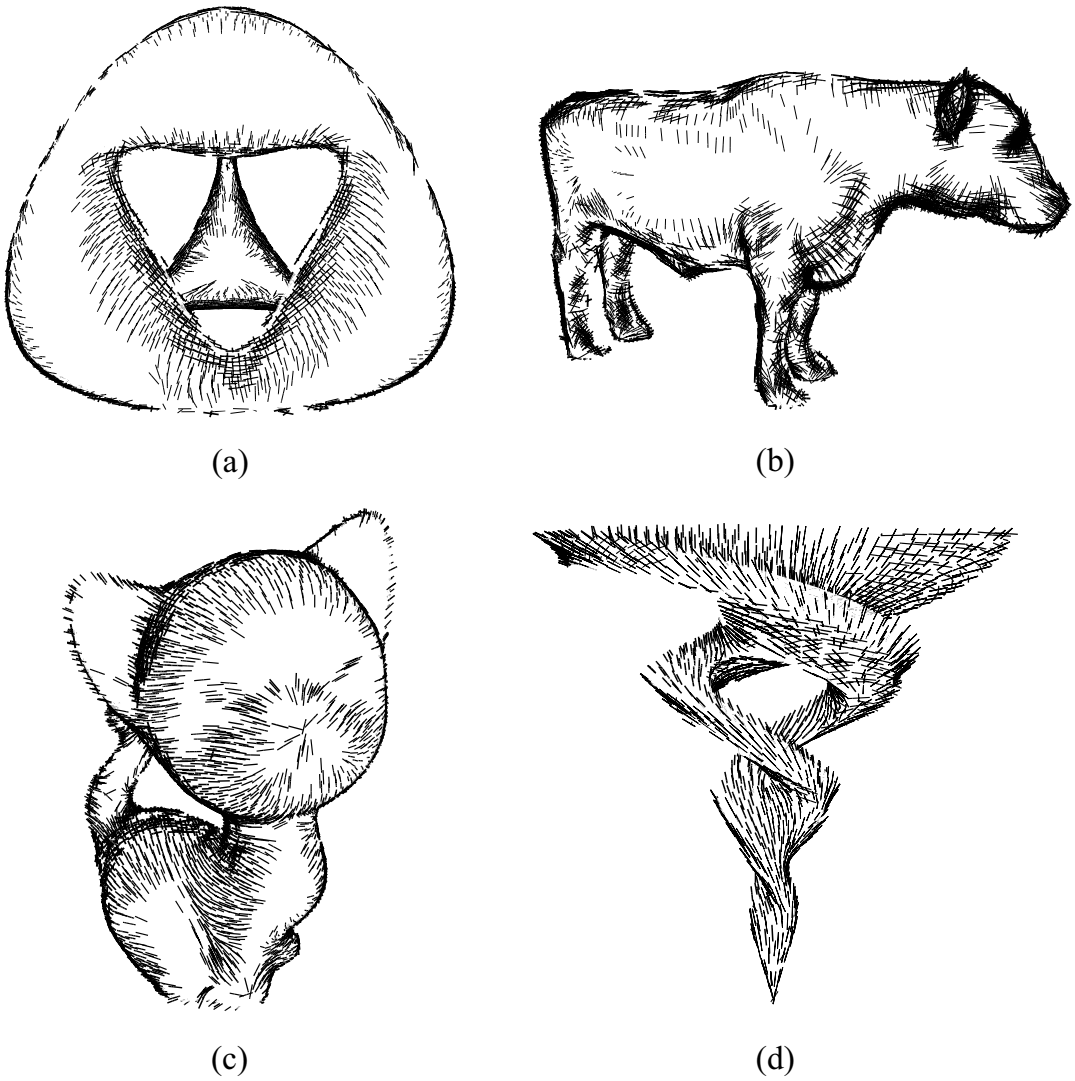


Figure 8. Illustration created by our method. (a) Chair surface. (b) Cow model. (c) Kitten. (d) Twirl.

[3] G. Elber. Line art illustrations of parametric and implicit forms. *IEEE Trans. Vis. Comp. Graph.*, 4(1):71–81, 1998.

[4] K. Foster, P. Jepp, B. Wyvill, M. C. Sousa, C. Galbraith, and J. A. Jorge. Pen-and-ink for BlobTree implicit models. *Computer Graphics Forum*, 24(3):267–276, 2005.

[5] A. Girshick, V. Interrante, S. Haker, and T. Lemoine. Line direction matters: An argument for the use of principal directions in 3d line drawings. In *Proceedings of NPAR 2000*, pages 43–52, 2000.

[6] A. Hertzmann and D. Zorin. Illustrating smooth surfaces. In *Proceedings of ACM SIGGRAPH 2000*, pages 517–526, 2000.

[7] E.-A. Karabassi, G. Papaioannou, T. Theoharis, and A. Boehm. Intersection test for collision detection in particle systems. *Journal of Graphics Tools*, 4(1):25–37, 1999.

[8] R. Keiser, B. Adams, D. Gasser, P. Bazzi, P. Dutré, and M. Gross. A unified lagrangian approach to solid-fluid animation. In *Symposium on Point-Based Graphics 2005*, pages 125–134, 2005.

[9] G. R. Liu and M. B. Liu. *Smoothed Particle Hydrodynamics*. World Science, 2005.

[10] J. J. Monaghan. On the problem of penetration in particle methods. *J. Comput. Phys.*, 82:1–15, 1989.

[11] J. J. Monaghan. Simulating free surface flow with SPH. *J. Comput. Phys.*, 110:399–406, 1994.

[12] J. P. Morris, P. J. Fox, and Y. Zhu. Modeling low reynolds number for incompressible flows using SPH. *J. Comput. Phys.*, 136:214–226, 1997.

[13] M. Müller, D. Charypar, and M. Gross. Particle-based fluid simulation for interactive applications. In *Symposium on Computer Animation*, pages 154–159, 2003.

[14] M. Müller, B. Solenthaler, R. Keiser, and M. Gross. Particle-based fluid-fluid interaction. In *Symposium on Computer Animation*, pages 237–244, 2005.

[15] E. Praun, H. Hoppe, M. Webb, and A. Finkelstein. Real-time hatching. In *Proceedings of ACM SIGGRAPH 2001*, pages 579–584, 2001.

[16] M. P. Salisbury, S. E. Anderson, R. Barzel, and D. H. Salesin. Interactive pen-and-ink illustration. In *Proceedings of ACM SIGGRAPH '94*, pages 101–108, 1994.

[17] A. Secord, W. Heidrich, and L. Streit. Fast primitive distribution for illustration. In *13th Eurographics Workshop on Rendering*, pages 215–226, 2002.

[18] J. Stam. Flows on surfaces of arbitrary topology. In *Proceedings of ACM SIGGRAPH 2003*, pages 724–731, 2003.

[19] D. Stora, P.-O. Agliati, M.-P. Cani, F. Neyret, and J.-D. Gascuel. Animating lava flows. In *Graphics Interface '99*, pages 203–210, 1999.

[20] G. Turk. Texture synthesis on surfaces. In *Proceedings of ACM SIGGRAPH 2001*, pages 347–354, 2001.

[21] G. Winkenbach and D. H. Salesin. Computer-generated pen-and-ink illustration. In *Proceedings of ACM SIGGRAPH '94*, pages 91–100, 1994.

[22] E. Zhang, K. Mischaikow, and G. Turk. Vector field design on surfaces. *ACM Trans. Graph.*, 25(4):1294–1326, 2006.