Non-Photorealistic Rendering from Stereo

A. Bartesaghi G. Sapiro

Electrical and Computer Engineering Department
University of Minnesota
Minneapolis, Minnesota 55455

Abstract

A new paradigm for automatic non-photorealistic rendering is introduced in this paper. Non-photorealistic rendering (NPR) provides an alternative way to render complex scenes by emphasizing high level or salient perceptual features. Particularly, the pen-and-ink rendering style produces sketchy-like drawings that can effectively communicate shape and geometry. This is achieved by combining drawing primitives that mimic ink patterns used by artists. Existing NPR approaches can be categorized in two groups depending on the type of input they use: image based and object based. Image based NPR techniques use 2D images to produce the renderings. Object based techniques work directly on given 3D models and make use of the full volumetric representation. In this paper we propose to enjoy the best of both worlds developing an hybrid model that simultaneously uses information from the image and object domains. These two sources of information are provided by a calibrated stereoscopic system. Given a pair of stereo images and the calibration data we solve the stereo problem in order to extract the normal and principal direction fields, which are fundamental to guide a texture synthesis algorithm that generates the NPR renderings. In particular, normals guide tonal variations, while principal directions determine the orientation of stroke-like texture patterns. We describe a particular, fully automatic, implementation of these ideas and present a number of examples.

1. Introduction

Non-photorealistic rendering can produce schematic representations of scenes that include only the relevant geometric information, clarifying shapes and focusing attention. This means simple sketches that suffice to visually communicate geometry and omit secondary scene information. Mimicking artists abilities involved in the drawing process well defined formation rules can be deducted and then reproduced by automatic computer graphics algorithms. For example, pen-and-ink drawings can be artificially generated by adequately placing individual strokes over the image, see [21, 18, 22] and references therein. Varying the density and direction of strokes a wide range of textures can be achieved that represent different materials, illumination conditions, shapes, etc.

Existing NPR approaches can be categorized in two groups depending on the type of input they use: image or object based.[1] Image based approaches use 2D gray level information to generate the renderings, *e.g.* [18, 19]. As any type of image can be used much more complex models can be considered (*e.g.* real scenes, landscapes, faces, etc.) but no geometric information can be inferred from the images themselves. On the other hand, object based techniques work directly with 3D models, *e.g.* [22, 6, 16, 13, 10, 17]. They have full access to the geometry (which is essential to represent shape) but are limited to computer generated scenes.

We then propose to have the best of both worlds, considering an hybrid system where images *and* the geometry are simultaneously available. These two sources are obtained from a calibrated stereoscopic system that provides the images and the corresponding 3D reconstruction as inputs to guide the rendering. Although stereo reconstruction may not be accurate enough for many applications, we show that the shape information needed to generate pen-and-ink illustrations can be obtained from a stereoscopic system.

Regarding the rendering stage, there are two basic illustration principles that properly combined convey the desired appearance to pen-and-ink drawings: density and orientation of strokes.[2] Varying the hatch density we can represent different tonalities or illumination conditions ranging from bright (no hatching), to dark or shadowed areas (densely crosshatched). Orientation carries the shape information, and stroke directions that follow the projection of principal directions of the surface are known to be appropriate

[1] Another possible categorization of NPR algorithms is according to the degree of user intervention they require. Some just assist the user in the generation process, others (the vast minority) are fully automatic. The one we propose here belongs to the small yet very important class of fully automatic algorithms.

[2] Additional features that further enhance the drawings may include silhouette drawing, edge tracing, etc.

in representing shape [14]. This is the approach taken in [17, 10, 13, 16] and the one we follow here. Given then a target tonal value and a preferred orientation, what the rendering process does is find the appropriate combination of strokes that achieves that particular configuration. The target tone and orientation combinations are obtained from the stereoscopic system.

1.1. Algorithm Overview

The main steps of the proposed NPR from stereo algorithm are the following:

1. Reconstruct the 3D surface given the pair of images and the calibration data. Once we have the surface, normals and principal directions can be readily obtained. This is addressed in Section 2.[3]

2. Compute required tone and orientation combinations from the geometry and image domains provided by the stereoscopic system. These, which are just particular examples of the information that can be extracted from the stereo data, are further described in Section 3.

3. The last step in the pipeline is the rendering itself. Here we use a texture synthesis algorithm guided by the illumination and orientation values computed in the previous step. This is described in Section 4.

Given a calibrated image pair, the reconstructed surface and its differential properties (normals and principal directions) can be obtained in many ways. Likewise, the rendering stage can be performed using various existing techniques (*e.g.* individual stroke placement, texture synthesis, etc). What is important and the main idea we want to present here is the concept of combining both sources of information (2D and 3D) as input to guide the synthesis of non-photorealistic renderings. As an example, we describe in this paper a particular realization of a system of this kind. Various examples are presented in Section 5.

2. Solving the Stereo Problem

Given a stereo pair of images (I_1, I_2) and corresponding calibration data we can reconstruct 3D scene coordinates up to a given transformation. The type of reconstruction (Euclidean, affine, or projective) depends on the calibration data available. We will assume that intrinsic and extrinsic camera parameters are known so Euclidean reconstruction can be performed. See [8, 9, 11] for details. Any 3D space point $\mathbf{X} = (x, y, z)$ projects into both camera retinal planes as $m_1(u, v) \in I_1$ and $m_2(u', v') \in I_2$, where (u, v) and (u', v') are 2D coordinates expressed in the systems attached to the first and second image planes, respectively. Each point $m_1 = (u, v)$ in the first image has its correspondent $m_2 = (u', v') = (u + U(u, v), v + V(u, v))$ in the second image, where $U(u, v)$ and $V(u, v)$ are the so called horizontal and vertical disparities. If disparities are known, 3D coordinates can be analytically obtained by solving the stereo reconstruction formulas. We compute disparities using a variational energy approach that recovers a dense disparity map from a set of two weakly calibrated stereoscopic images [1]. We can then solve the reconstruction formulas and get the surface coordinates in parametrized form: $\mathcal{S}(u, v) = (x(u, v), y(u, v), z(u, v))$. Taking directional derivatives $\mathcal{S}_u = (x_u, y_u, z_u)$ and $\mathcal{S}_v = (x_v, y_v, z_v)$, 3D normals are obtained from:

$$\mathbf{N}(u, v) = \frac{\mathcal{S}_u \wedge \mathcal{S}_v}{|\mathcal{S}_u \wedge \mathcal{S}_v|} \quad (1)$$

Principal curvatures and directions are the eigenvalues and eigenvectors of the shape operator $\mathbf{I}^{-1}\mathbf{II}$, where $\mathbf{I}$ and $\mathbf{II}$ are the first and second fundamental forms of the surface:

$$\mathbf{I} = \begin{bmatrix} \mathcal{S}_u \cdot \mathcal{S}_u & \mathcal{S}_u \cdot \mathcal{S}_v \\ \mathcal{S}_u \cdot \mathcal{S}_v & \mathcal{S}_v \cdot \mathcal{S}_v \end{bmatrix}$$
$$\mathbf{II} = \begin{bmatrix} \mathcal{S}_{uu} \cdot \mathbf{N} & \mathcal{S}_{uv} \cdot \mathbf{N} \\ \mathcal{S}_{uv} \cdot \mathbf{N} & \mathcal{S}_{vv} \cdot \mathbf{N} \end{bmatrix} \quad (2)$$

The one we just described, is the straight forward approach to compute normals and principal directions from stereo. Another approach may be to obtain normals and principal directions directly from the images without actually reconstructing the 3D surface. This was done in [3], but poor principal curvature estimates were obtained. A third alternative is to follow the variational approach in [7] that directly computes the surface (in implicit form) given an arbitrary number of images. Having an implicit representation of the surface makes it easier to obtain normals and principal directions.

3. Guiding the Rendering Stage

This section describes how to obtain tone and orientation values to guide the rendering stage using image and object domain features provided by the stereoscopic system. Before starting, we note that the reconstructed surface $\mathcal{S}(u, v)$ is somewhat noisy, so a smoothing step is required before computing any of the differential quantities presented in the previous section. As pointed out in [13], important scene features are better captured by tone variations, while directions only provide a low resolution field that indicates general object shape. Accordingly, we perform a small amount of smoothing before computing normals (so we keep relevant features), and apply a bigger amount of smoothing before computing principal directions (so we obtain a lower resolution direction field).

[3]Note that in certain scenarios, normals and principal directions can be obtained without a full 3D reconstruction.

3.1. Computing Illuminations

Tone values at each image point are obtained by projecting surface normals into the viewing direction. This gives bright values for fronto-parallel surface points and increasingly dark values for points approaching object silhouettes. We can also exploit the image based side of our approach and extract illumination features from the images themselves. Although we could use any of the techniques reported in image based rendering, for the examples in this paper we simply detect dark image regions by thresholding.[4] Other relevant image features may include texture, edges, object segmentation, etc.[5] Feature extraction can be made more robust by combing estimates from both stereo images, we have not explored this option, but it can certainly improve the results. By merging illumination information extracted from both sources, we obtained the target tone image that will guide the rendering. We now describe how to get the target orientations.

3.2. Computing Orientations

As mentioned in Section 1 directions that follow the projection of principal directions of the surface are known to be appropriate in representing shape. We already know how to compute principal directions from stereo, except that at umbilic points they are not defined (all directions are principal), so solving the eigenvalue problem is ill-posed. In the actual computation we get a condition number that indicates presence or proximity to an umbilic point. We therefore discard the unreliable estimates and provide a method for filling-in the missing values. A similar approach was taken in [13]. In the end we need to project 3D principal directions into the image plane, therefore, we choose to apply the fill-in procedure after the projection step and the particular technique we use is based on the variational vector diffusion approach [20].

Let $\hat{\mathbf{U}}(u, v)$ be the (projected) principal direction field estimate obtained by solving the eigenvalue problem described in Section 2. A regularized $\mathbf{U}$ field is then obtained as the minimizer of the following energy:

$$E(\mathbf{U}) = \frac{1}{2}\int_{\Omega} \omega(\mathbf{U} - \hat{\mathbf{U}})^2 + (1-\omega)||\nabla \mathbf{U}||^2 dudv \quad (3)$$

The first component in this equation is a fidelity term that forces the minimizer to be close to the estimate $\hat{\mathbf{U}}$. The second component is the regularizing term that guarantees smoothness. The weighting factor ω accounts for the accuracy of the estimate $\hat{\mathbf{U}}$, the idea is to leave accurate values unchanged ($\omega = 1$) while allowing those that are unreliable to evolve (ω=0). Computing the Euler-Lagrange for the energy in Equation (3) we obtain the minimizing, unity-preserving, evolution:

$$\begin{cases} \mathbf{U}_t = & \omega(\hat{\mathbf{U}} - (\hat{\mathbf{U}} \cdot \mathbf{U})\mathbf{U}) + \\ & (1-\omega)(\Delta\mathbf{U} - (\Delta\mathbf{U} \cdot \mathbf{U})\mathbf{U}) \\ \mathbf{U}(0) = & \hat{\mathbf{U}} \end{cases} \quad (4)$$

The steady state solution of this system will provide a suitable 2D direction field for guiding the rendering stage.

At this point we have completely determined target tone and orientations by simultaneously using image and object space features. When actually doing the synthesis we will only have a discrete number of different tones (dictated by the particular texture set we are using), so we need to threshold illumination values into the available number of tones.[6] The same happens for orientation values, we only have a discrete number of possible orientations. For the examples in Section 5 we use 6 and 8 different illumination levels (depending on the texture set) and 180 orientation values.

4. Non-photorealistic Rendering

Two methods can be used to generate stroke based image renderings. The first is to individually place each stroke on the target image until the desired target appearance is reached. Bright areas of the image would require fewer strokes, less density, while darker or shadow areas would require denser stroke concentrations. To achieve higher density levels we can decrease the distance among adjacent strokes or we can even superpose them in different directions (cross hatching). Systems that used individual placing of stokes were studied in [18, 21, 22, 13].

The second approach is to use a set of pre-generated stroke textures and apply a texture synthesis algorithm to reproduce those patterns throughout the image. By using different texture sets a variety of rendering styles can be obtained. This is the approach used in [16, 10] for object-space rendering (texture mapping on 3D surfaces). This simpler approach is easy to implement and is the one we use here. Observe that we only need to select a basic set of textures and then run a standard texture synthesis algorithm, we describe the specific algorithm in the next section.

For every possible combination of tone and orientation we need to provide a sample texture that achieves that particular configuration. Textures with different tones are achieved by varying the density of strokes, we then simply pre-rotate each of them to get all the possible orientations. This way we have a 2D bank of texture samples, each representing a particular combination of tone and orientation, see Figure 1. Given a particular target tone and orientation

[4]Gray level encoded features are clearly overseen when reconstructing geometry, but are easily recognized in the image domain.

[5]Through out this work we assumed a good segmentation of the object was given, meaning we can separate the object from the background. Instead, we could have obtained the segmentation automatically from the images.

[6]Thresholds are computed by minimum variance quantization.

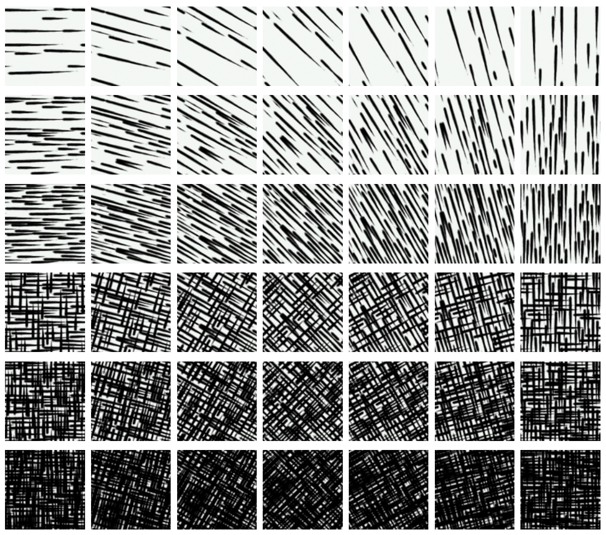


Figure 1: Image bank generated with texture sets taken from [16]. Directions span along the horizontal axis and illumination values along the vertical axis.

we have to choose the corresponding texture sample from the bank to perform the synthesis.

4.1. Texture Synthesis Rendering

Among many texture synthesis algorithms considered [2, 5, 4, 12, 15], patch based algorithms probed to be the most effective for the type of textures we are interested in. Patch-based texture synthesis algorithms generate the output image by pasting together patches of the sample texture. Two such algorithms are [4] and [15], we mainly follow the latter.

Let $I_{d,i}(u, v)$ denote the sample texture corresponding to direction value d and illumination value i (both take integer values).[7] $I_{synth}(u, v)$ will denote the output synthesized image. P_k is a square patch of $I_{d,i}(u, v)$ of size $w \times w$. For each patch P_k we denote its boundary zone by B_{P_k}. E_k denotes a square patch of $I_{synth}(u, v)$ of size $w \times w$ with boundary zone B_{E_k}. The boundary zones represent the overlapping regions between adjacent patches, see Figure 2.

The patch P_k *matches* E_k if the distance between their overlapping boundary zones is below some threshold value ϵ. The set of matches for the patch E_k is then defined as:

$$\phi_k = \{P_k \in I_{d,i}(u, v) / d(B_{P_k}, B_{E_k}) < \epsilon\} \qquad (5)$$

With these definitions, the patch based algorithm works as follows:

[7]Note that unlike classical texture synthesis here we require a *set* of sample textures instead of just one reference texture.

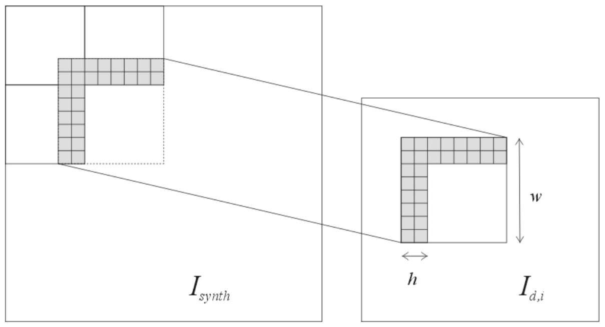


Figure 2: Patch based algorithm. *Left:* three patches were already assigned to the output image $I_{synth}(u, v)$, we now look for the best matching patch to paste in the lower right corner. *Right:* candidate patches are searched in the sample image $I_{d,i}(u, v)$ by matching the overlapping regions (shaded areas).

- Randomly select a patch from the corresponding sample image $I_{d,i}(u, v)$ and paste it in the upper left corner of $I_{synth}(u, v)$.
- Build the set ϕ_k of all patches $P_k \in I_{d,i}(u, v)$ whose boundary zone B_{P_k} matches the current position patch boundary B_{E_k}. Select one patch from ϕ_k at random and paste it into the output image $I_{synth}(u, v)$.
- Repeat until the whole image $I_{synth}(u, v)$ is covered.

To build the set ϕ_k the search is done using an approximate p-nearest neighbor algorithm for which fast structures and routines are readily available. The best candidate is chosen at random from these p-nearest neighbors and pasted into the output $I_{synth}(u, v)$. The search space dimension is the number of pixels in B_{P_k} and the whole sample image space is considered.

5. Data Sets and Examples

We used three calibrated stereo data sets. The first two are real faces and the third one is a concrete bust of Buffalo Bill. Stereo pairs and corresponding renderings are shown in Figures 4, 5 and 6.

The synthesis stage was run with two different texture sets. Set 1 was obtained from [16] (see Figure 1), the second set is shown in Figure 3. These sets have 6 and 8 different tone values respectively, and were pre-rotated to 180 different directions.

6. Summary and Discussion

We presented an automatic system for the generation of pen-and-ink illustrations given a calibrated stereoscopic system. We simultaneously use gray level information from the images as well as geometric features extracted from the

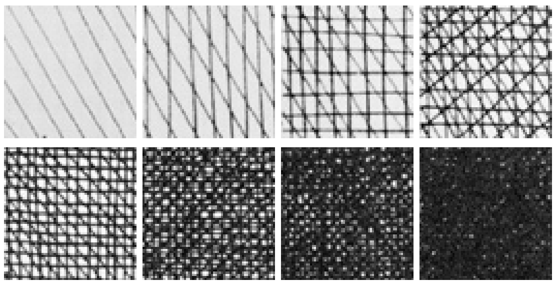


Figure 3: Texture set 2.

stereo reconstruction to guide a texture synthesis algorithm that generates the renderings.

The main idea is general in the sense that we could have used any stereo reconstruction procedure and any rendering technique (*e.g.* individual placement of strokes, other textures sets for the synthesis, etc.) to obtain the results. The emphasis is in the idea of using both sources of information to guide the rendering stage and the fact that a stereoscopic system can provide them both with enough accuracy for this particular task. Although the required minimum to perform the 3D stereo reconstruction are two images, if more are available, better 3D models can be obtained, therefore better geometry estimates would guide the rendering stage.

Let's conclude by mentioning that the approach in [19] is similar in nature to ours, the difference being that a single image is used. Therefore, external user action was needed to provide the directional fields that guide the rendering. We propose to get those fields from the stereo reconstruction without requiring any user input, thus presenting a fully automatic approach.

Acknowledgments

This work is is partially supported by a grant from the Office of Naval Research ONR-N00014-97-1-0509, the Presidential Early Career Award for Scientists and Engineers (PECASE), and a National Science Foundation CAREER Award.

References

[1] L. Alvarez, R. Deriche, J. Sánchez, and J. Weickert. Dense disparity map estimation respecting image discontinuities: A pde and scale-space based approach. Technical Report 3874, INRIA, January 2000.

[2] Michael Ashikhmin. Synthesizing natural textures. In *Symposium on Interactive 3D Graphics*, pages 217–226, 2001.

[3] F. Devernay and O. Faugeras. Computing differential properties of 3-d shapes from stereotopic images without 3-d models. Technical Report 2304, INRIA, July 1994.

[4] Alexei A. Efros and William T. Freeman. Image quilting for texture synthesis and transfer. *Proceedings of SIGGRAPH 2001*, pages 341–346, August 2001.

[5] Alexei A. Efros and Thomas K. Leung. Texture synthesis by non-parametric sampling. In *IEEE International Conference on Computer Vision*, pages 1033–1038, Corfu, Greece, September 1999.

[6] G. Elber. Line art illustrations of parametric and implicit forms. *IEEE Transactions on visualization and computer graphics*, 4(1):71–81, January-March 1998.

[7] O. Faugeras and R. Keriven. Variational principles, surface evolution, pde's, level set methods and the stereo problem. Technical Report 3021, INRIA, October 1996.

[8] Olivier Faugeras. *Three-Dimensional Computer Vision*. MIT Press, 1993.

[9] Olivier Faugeras and Quang-Tuan Luong. *The Geometry of Multiple Images*. MIT Press, 2001.

[10] G. Gorla, V. Iterrante, and G. Sapiro. Texture synthesis for 3d shape representation. To appear in IEEE Transactions on Visualization and Computer Graphics.

[11] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge, 2000.

[12] Aaron Hertzmann, Charles E. Jacobs, Nuria Oliver, Brian Curless, and David H. Salesin. Image analogies. In Eugene Fiume, editor, *SIGGRAPH 2001, Computer Graphics Proceedings*, pages 327–340. ACM Press / ACM SIGGRAPH, 2001.

[13] Aaron Hertzmann and Denis Zorin. Illustrating smooth surfaces. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 517–526. ACM Press/Addison-Wesley Publishing Co., 2000.

[14] Victoria L. Interrante. Illustrating surface shape in volume data via principal direction-driven 3D line integral convolution. *Computer Graphics*, 31(Annual Conference Series):109–116, 1997.

[15] Lin Liang, Ce Liu, Ying-Qing Xu, Baining Guo, and Heung-Yeung Shum. Real-time texture synthesis by patch-based sampling. *ACM Transactions on Graphics (TOG)*, 20(3), 2001.

[16] Emil Praun, Hugues Hoppe, Matthew Webb, and Adam Finkelstein. Real-time hatching. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, page 581. ACM Press, 2001.

[17] C. Rossl and L. Kobbelt. Line-art rendering of 3d models. In *Proceedings of Pacific Graphics 2000*, Hong Kong, China, October 2000.

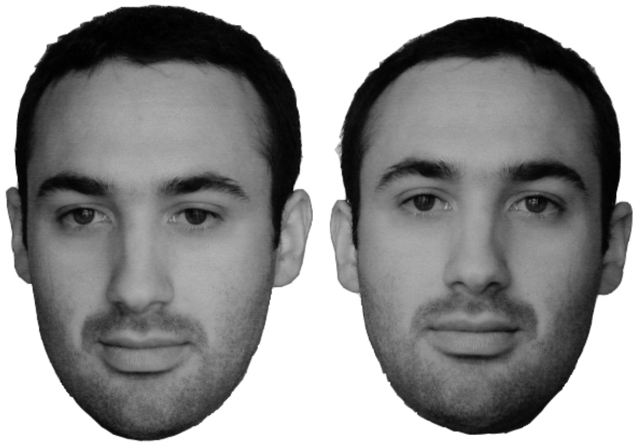
[18] M. Salisburry, S. Anderson, R. Barzel, and D. Salesin. Interactive pen-and-ink illustration. Technical Report 94-01-07b, University of Washington, Seattle, Washington 98195, April 1994.

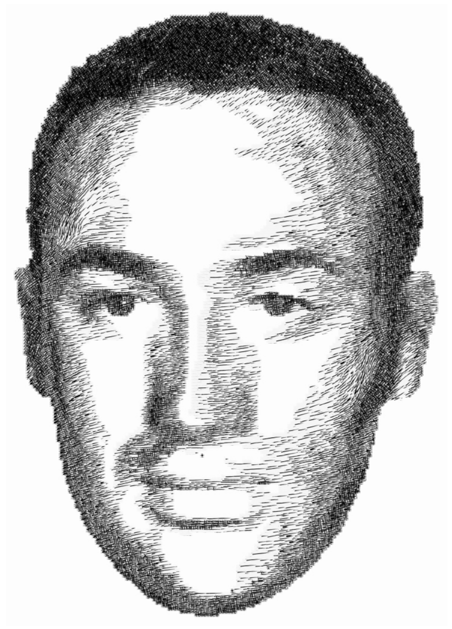
[19] Michael P. Salisbury, Michael T. Wong, John F. Hughes, and David H. Salesin. Orientable textures for image-based pen-and-ink illustration. In *Proceedings of the 24th annual conference on Computer graphics and interactive techniques*, 1997.

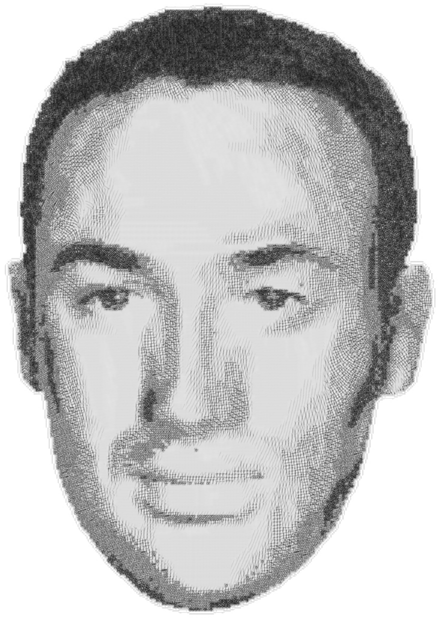
[20] Bei Tang, Guillermo Sapiro, and Vicent Caselles. Direction diffusion. In *ICCV (2)*, pages 1245–1252, 1999.

[21] G. Winkenbach. *Computer-Generated Pen-and-Ink Illustration*. PhD thesis, University of Washington, 1996.

[22] G. Winkenbach and D. Salesin. Rendering parametric surfaces in pen and ink. Technical Report 96-01-05b, University of Washington, Seattle, Washington 98195, May 1996.







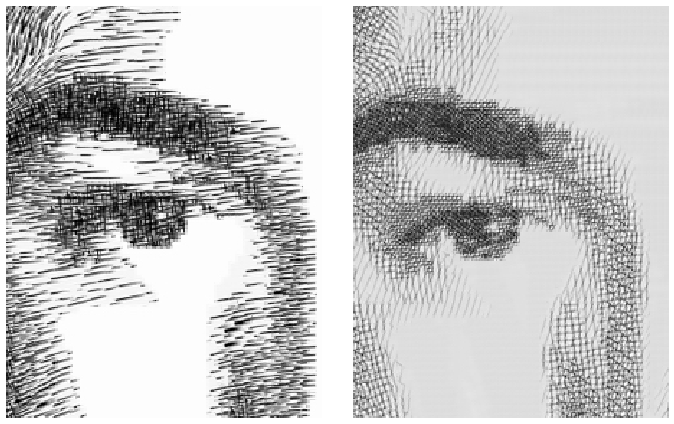


Figure 4: First face data set. The stereo pair, renderings using the first and second texture sets and two corresponding close-ups views are shown. This data set is distributed with the `StereoFlow` software package available at `http://serdis.dis.ulpgc.es/∼jsanchez/research/software/StereoFl`

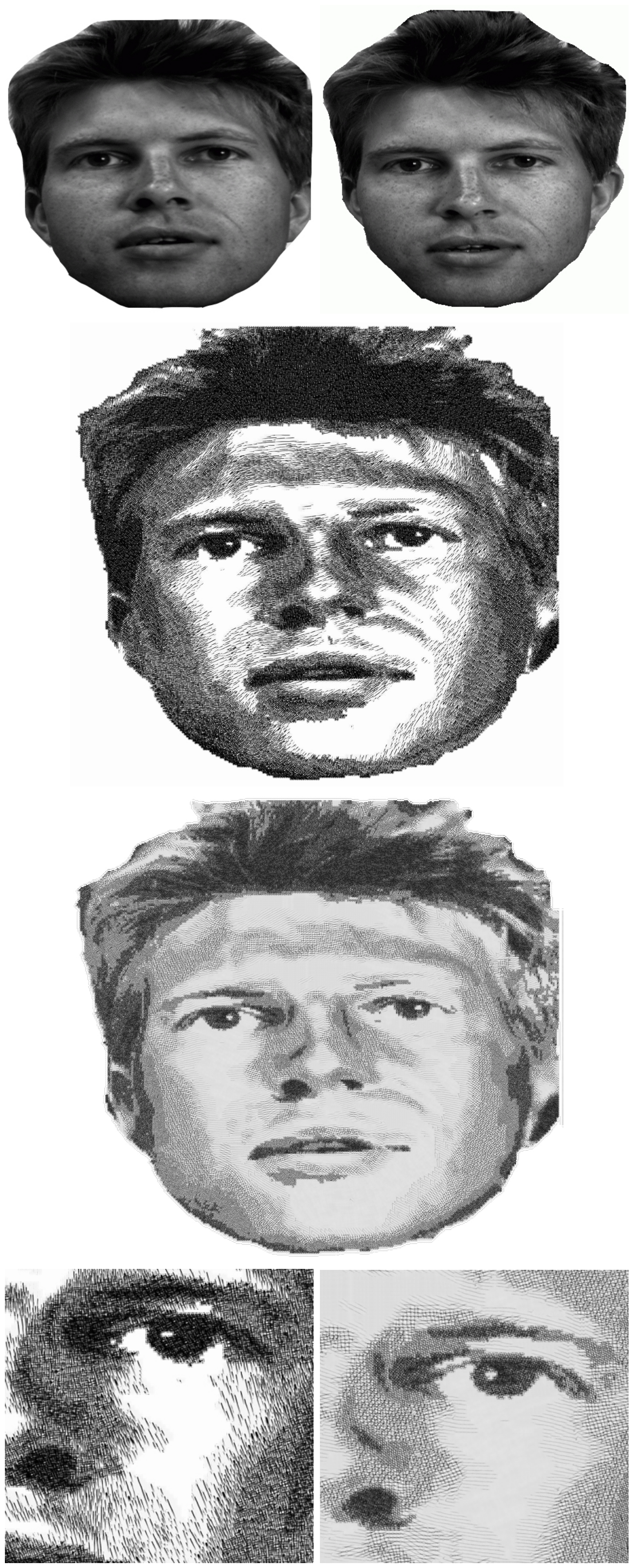


Figure 5: Second face data set. The stereo pair, renderings using the first and second texture sets and two corresponding close-ups views are shown. This data set is available from the RobotVis project at the INRIA website.

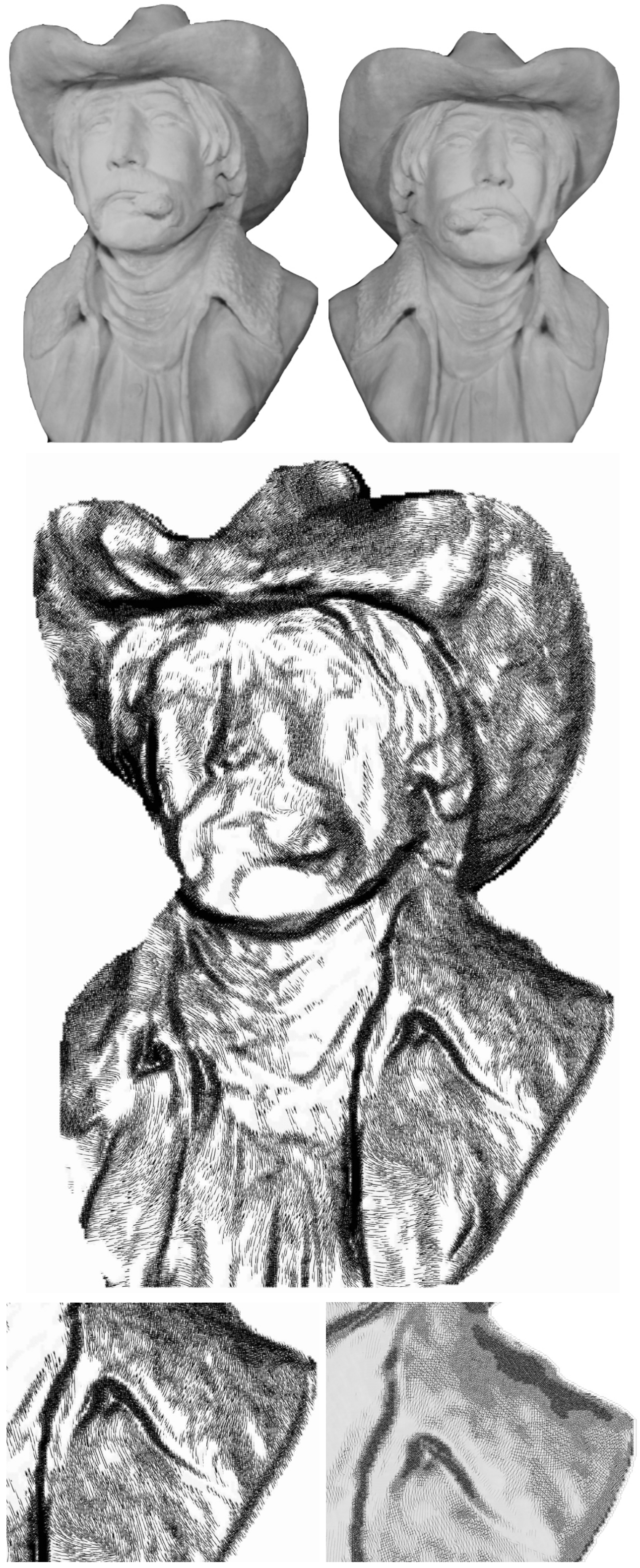


Figure 6: Buffalo Bill bust. The stereo pair, renderings using the first and second texture sets and two corresponding close-ups views are shown. This data set is available from the GRASP Laboratory at `http://www.cis.upenn.edu/~janem/rsrch/dataset/ReadMe.html`.