Non-Photorealistic Animation and Rendering
David Mould and Pierre Bénard (Editors)

EXPRESSIVE 2015

Hybrid-Space Localized Stylization Method for View-Dependent Lines Extracted from 3D Models

Luis Cardona[1] and Suguru Saito[2,1]

[1]Tokyo Institute of Technology, Tokyo, Japan
[2]Ochanomizu University, Tokyo, Japan

Abstract

We propose a localized stylization method that combines object-space and image-space techniques to locally stylize view-dependent lines extracted from 3D models. In the input phase, the user can customize a style and draw strokes by tracing over view-dependent feature lines such as occluding contours and suggestive contours. For each stroke drawn, the system stores its style properties as well as its surface location on the underlying polygonal mesh as a data structure referred as registered stroke. In the rendering phase, a new attraction field leads active contours generated from the registered strokes to match current frame feature lines and maintain the style and path coordinates of strokes in nearby viewpoints. For each registered stroke, a limited surface region referred as influence area is used to improve the line matching accuracy and discard obvious mismatches. The proposed stylization system produces uncluttered line drawings that convey additional information such as material properties or feature sharpness and is evaluated by measuring its usability and performance.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Picture/Image Generation—Line and curve generation

1. Introduction

1.1. Background

Computer-generated 3D models have been extensively used in applications that try to replicate realistic imagery. However, there is another branch of computer graphics called Non-Photorealistic rendering (NPR), that focuses on how to imitate the principles of abstraction used by artists. NPR research covers different topics such as line extraction algorithms, stylization and animation coherence. However, there are still many problems to solve in order to generate imagery with the same degree of abstraction and stylization as human-made drawings.

As stated in a recent study [CGL*12], most of the lines drawn by artists to convey 3D shape can be explained by some of the most commonly-known line extraction algorithms. However, artists often make global decisions such as the omission of implicit lines or stylistic decisions which tend to depart from realism. Human-made line drawings usually have a coherent style for the entire scene but also include subtle differences in the style of each individual stroke. Therefore, there is a need for new algorithms that in combination with flexible user interfaces allow designers to customize the appearance of drawings as well as to emphasize or omit some of its details.

1.2. Related Work

Various methods have been proposed in order to extract lines from 3D models. Image-space algorithms focus on image processing methods such as edge detection to extract the lines at each pixel of the output image [ST90, Dec96, Her99, LMLH07]. This type of algorithm generates quite convincing images but suffers from noise problems and are unsuitable for further stylization.

Object-space algorithms are based on the field of differential geometry which analyzes the properties of curves and surfaces. Previous research has successfully characterized various types of lines by making use of derivatives of different order. The main contributions in this category can be classified into view-dependent lines such as occluding contours [K*84], suggestive contours [DFRS03, DFR04] and apparent ridges [JDA07]; view-independent lines such as ridges and valleys [OBS04] and demarcating curves [KST08]; and

© The Eurographics Association 2015.

DOI: 10.2312/exp.20151181

delivered by EUROGRAPHICS DIGITAL LIBRARY www.eg.org diglib.eg.org

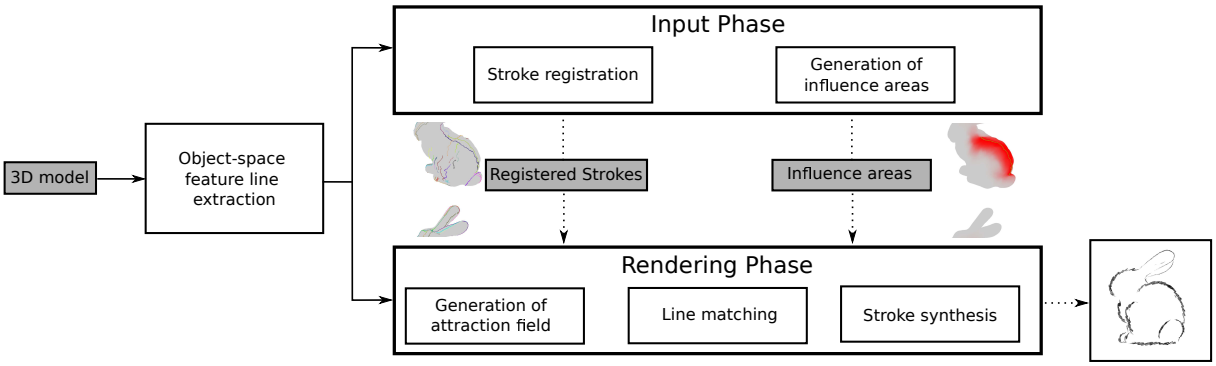


Figure 1: Overview of the proposed stylization system.

illumination-dependent lines such as photic extremum lines [XHT*07]. Some surveys [IFH*03,LBSP14] have compared numerous methods to extract silhouettes or feature lines and analyzed the quality of the resulting lines. A common problem of most line extraction algorithms is that they usually produce either too many lines which would not be drawn by artists or not enough to convey the shape of the object.

Recently, researchers have focused on how to propagate the path coordinates of lines in order to maintain the temporal coherence of stylized strokes as the viewpoint changes or the 3D model is animated [KDMF03, BCGF10, BFP*11, KH11, BLC*12]. Bénard et al. [BCK*13] formalized the problem of temporal coherence in term of three concurrent goals (*flatness*, *motion coherence*, and *temporal continuity*) and compared the methods proposed in recent years.

Earlier works successfully recreate styles found in artistic and technical drawings [MKG*97, NM00] but are limited to a single style for each type of line. Other stylization techniques include methods to express tone, texture or shading from polygonal surfaces through hatching strokes [HZ00], prevent cluttered drawings by adjusting the amount of lines [SP03], locally control stylized shading [TABI07] or generate temporally coherent animation sequences from stylization examples provided by the user [BCK*13].

Only a few previous works have stated the need for user interfaces to provide better control of the style of feature lines [KMM*02, IB06, GTDS10]. These systems are useful to quickly define the style of the entire scene but lack control over the local stylization of view-dependent lines. Therefore, the purpose of our research is to provide a system to stylize view-dependent lines extracted from 3D models that maintains control over individual details. In this paper, we define *localized stylization* as stylistic annotations made by the user to represent local features such as material properties or feature sharpness. Our focus is to solve the following problems related to the *localized stylization*:

- Maintain the style of view-dependent lines as the viewpoint is changed or the 3D model is animated.
- Replicate global decisions made by artists such as emphasizing or omitting certain lines.

In our previous work [CS13], we proposed a *localized stylization* method based on surface segmentation using the Gaussian curvature. This method divides *occluding contours* at inflection points and tracks object-space contours to generate additional surface areas. However, this approach has the following problems and limitations:

- Terminal points are limited to inflection points.
- High dependence on object-space line topology.
- Does not support specifying stroke direction.
- Difficult to stylize individual lines.
- Not suited for *suggestive contours*.

In this paper we propose a *localized stylization* method that solves the above limitations by combining object-space and image-space techniques to locally stylize view-dependent lines extracted from 3D models.

2. Overview

The proposed method allows the user to specify the style, terminal points and drawing direction of individual strokes where surface segmentation methods would fail because multiple lines usually appear in the same area. Storing the stylization information directly as strokes allows for a more intuitive stylization system that replicates the 2-dimensional nature of drawing so that the user can focus on the lines instead of the underlying surface. The algorithm starts by extracting view-dependent feature lines such as *occluding contours* and *suggestive contours*. The stylization system is divided into an input phase where strokes drawn by the user are registered and a rendering phase where the *registered strokes* are matched to current frame feature lines in order to synthesize strokes that maintain the style as specified by the user. The overview of the proposed stylization system is illustrated in Figure 1.

© The Eurographics Association 2015.

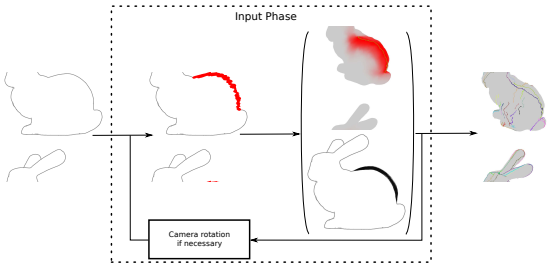


Figure 2: Input phase: interactive stroke registration.

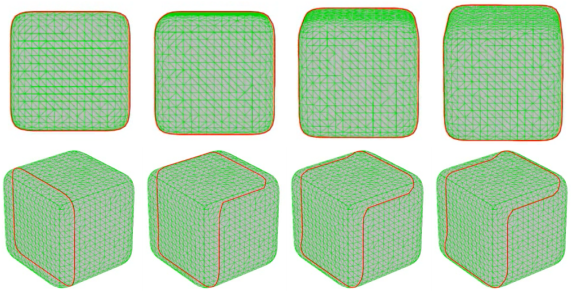
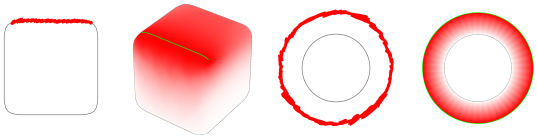


Figure 3: Movement contour lines in image space (top) and object space (bottom).



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

Figure 4: Generation of the *influence area* of a *registered stroke*.

3. Input Phase

3.1. Stroke Registration

A user interface is provided to easily apply a wide range of stroke styles to the lines extracted from a 3D model. The user can customize style properties such as width, color, texture and add line offsets by sketching an example stroke. Figure 2 shows the process to register a stroke: the initial state without stylization shows the extracted feature lines as a visual guide; the user then traces over part of a feature line to specify its style; for each stroke drawn, the system stores its style properties as well as its surface location on the underlying polygonal mesh as a data structure referred as *registered stroke* and generates its corresponding *influence area* as explained in the following section; finally, all *registered strokes* after drawing from multiple viewpoints are used to match current frame feature lines as explained in section 4.1. Once a stroke is registered, the user can select it and make further adjustments through sliders controlling the style properties. The user can change the camera position in order to input new strokes for feature lines appearing at different view directions. This process is repeated until the desired style is achieved for all viewpoints. A local coordinate system defined along each edge of the mesh is used to store the surface location of the *registered strokes* so that it can be updated in the case of animated models.

3.2. Influence Areas

We define an *influence area* as a surface region that delimits the locations where its corresponding *registered stroke* can be matched to current frame feature lines. The *influence area* of a *registered stroke* is generated by diffusion over the surface. When the viewpoint is modified, one problem is that, while a contour line moves smoothly in image-space, it may move abruptly over the surface. In object-space, the distance on surface that the contour line traverses is inversely proportional to the radial curvature (i.e. the curvature in the view direction) of the underlying surface. Figure 3 shows how even for small changes of view direction, the contour line may move long distances when the underlying surface has low radial curvature (e.g. planar surfaces). To solve this problem, we propose a diffusion algorithm that accentuates the contribution of vertices close in screen coordinates. The resulting diffusion is indirectly dependent on the radial curvature and spreads faster in the view direction. Figure 4 (a) shows the registration of a stroke for the contour on the top edge of a rounded cube and Figure 4 (b) shows the same rounded cube from a different viewpoint as the influence values (red) are propagated faster on its top side from the *registered stroke* (green line).

We define a Laplacian approximation with weights equal to the inverse of the lengths of the projected edges connecting a vertex to its one-ring neighbor vertices into the image plane:

$$\Delta I_{n,i} = \sum_{j \in i*} w_{ij}(I_{n,i} - I_{n,j}) \tag{1}$$

where $i*$ is the set of neighbors of vertex i, $I_{n,i}$ and $I_{n,j}$ are the influence values at vertices i and j associated with the registered stroke n and

$$w_{ij} = \frac{1}{|q_i - q_j|} \tag{2}$$

where q_i and q_j are the projected vertices i and j.

The diffusion process is done iteratively using the following conditional explicit Euler scheme:

$$I_{n,i}^{k+1} = \begin{cases} 1 & \text{if } i \in t \mid t \in T_n \\ 0 & \text{if } i \in t \mid t \in T_L \text{ and } t \notin T_n \text{ and } t \notin T_{L_n} \\ I_{n,i}^{k} + \lambda \Delta I_{n,i}^{k} & \text{otherwise} \end{cases} \tag{3}$$

where λ is a constant such that $0 < \lambda \leq 1$, t is a triangle of the 3D mesh, T_n is the set of triangles containing feature line

© The Eurographics Association 2015.

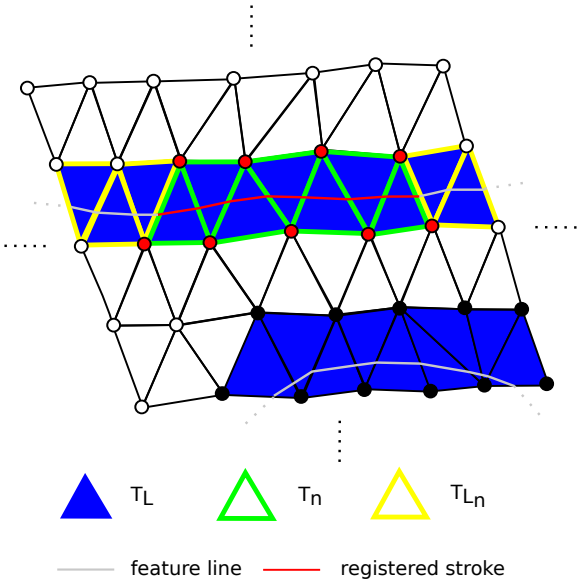


Figure 5: Conditional diffusion and triangle sets. The influence value is set to 1 for vertices of triangles crossed by the *registered stroke* (red dots), set to 0 for vertices of triangles crossed by feature lines other than the one connected to the *registered stroke* (black dots) and updated at each iteration for all other vertices (white dots).

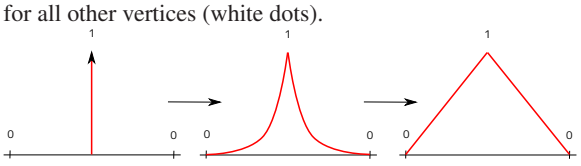


Figure 6: Diffusion convergence to linear interpolation.

segments that belong to the registered stroke n, T_L is the set of triangles containing all line segments in the same view when the registered stroke n was stored and T_{L_n} is the set of triangles containing line segments that belong to the feature line connected to the registered stroke n (Figure 5).

By resetting the initial values to 1 and the boundary conditions to 0 at each iteration, the diffusion process converges to linear interpolation as shown in Figure 6. Setting the boundary conditions stops the diffusion before reaching other feature lines but can be optionally omitted by the user. Figure 4 (c) shows the registration of a stroke on the outer ring contour of a torus and Figure 4 (d) shows how setting the boundary conditions to 0 for the inner ring contour results in a the diffusion process where the influence values fade out before reaching the inner ring contour.

The *registered strokes* and *influence areas* represent two-dimensional information that is mapped into the object-space domain. This approach is motivated by the need to allow all types of affine transformations (e.g. rotations) that would not be possible on the image-space domain. In the rendering phase, this data will eventually be re-projected into the image-space to match current frame feature lines as explained in the following section.

4. Rendering Phase

The rendering pipeline takes as input a 3D model as well as the *registered strokes* and corresponding *influence areas*. View-dependent feature lines such as *occluding contours* and *suggestive contours* are extracted from the 3D model. These feature lines are then rendered as a binary image B that is used as input to build an attraction field. This attraction field leads 2D polylines generated from the *registered strokes* to current frame feature lines. The *influence area* explained in the previous section serves the purpose of discarding obvious mismatches which are consequence of self-occlusion and usually occur at surface locations far away from the *registered stroke*. Figure 7 shows these mismatches and how an *influence area* avoids this problem by limiting the area where a *registered stroke* is matched. For each 2D polyline that successfully reaches current frame feature lines, one or more strokes are synthesized and rendered as triangle strips reflecting the unique style properties stored in the corresponding *registered stroke*. Figure 8 shows the frame buffers used in the proposed method.

4.1. Line Matching

The proposed image-space line matching method leads 2D polylines to current frame feature lines. These 2D polylines are created along the projection of their corresponding *registered stroke* into the image plane as active contours [KWT88] with sample points $v(s) = (x,y)$, $s \in [0;1]$. All sample points of the active contour are updated by minimizing an energy function composed of an internal and an external energy:

$$E = \int_0^1 E_{int}(v(s)) + E_{ext}(v(s))ds \tag{4}$$

We use a common internal energy definition in terms of first and second derivatives to minimize length and curvature:

$$E_{int} = \frac{1}{2}\alpha|v'(s)|^2 + \beta|v''(s)|^2 \tag{5}$$

where α and β are constants ($\alpha = 0.001$ and $\beta = 0.02$ in our implementation).

In contrast to methods that track feature lines frame by frame to achieve temporal coherence [KDMF03, BLC*12], the proposed line matching algorithm needs to lead the active contours from regions relatively far away from current frame feature lines. Consequently, the common external energy gradient is replaced by an attraction field A_n defined for all points on the image near the registered stroke n that are located over the 3D mesh or a feature line. For each registered

© The Eurographics Association 2015.

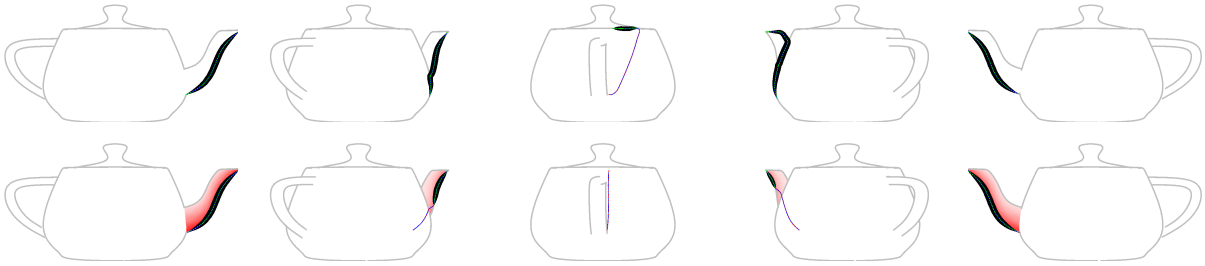


Figure 7: Line mismatches (top) and their avoidance by an influence area (bottom).

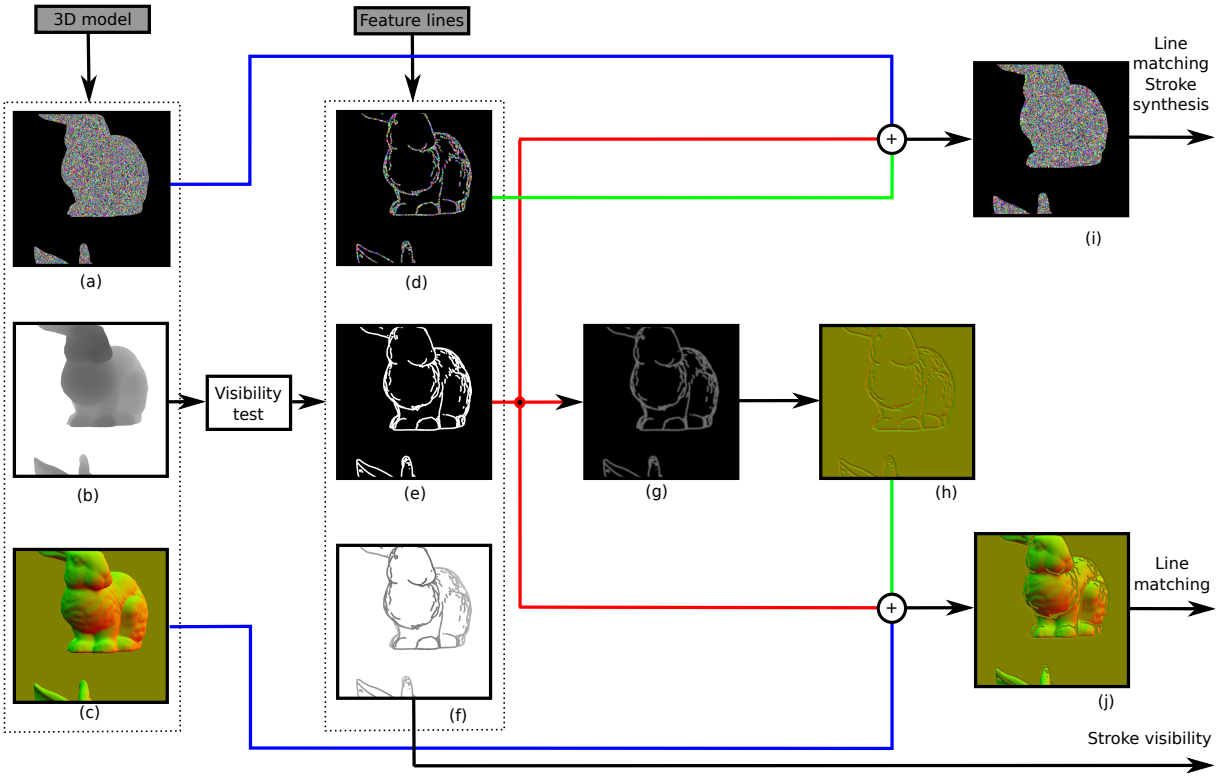


Figure 8: Frame buffers used in the proposed method. The triangle references of the 3D model (a); the depth of the scene (b); the projected normals field A_{normal} (c); the underlying triangle references at feature lines (d); the feature lines binary image B (e); the depth of the feature lines (f); the feature lines binary image B diffused by Gaussian filter (g); the feature line attraction field A_{line} (h); the combined triangle reference image (i); the combined attraction field (j). The triangle references in (a), (d) and (i) are color-coded for visualization purposes.

stroke n, the attraction field is scaled by its *influence area*:

$$\nabla E_{ext} = A_n(v(s)) = \begin{cases} I_n(v(s))A_{line}(v(s)) & \text{if } B(v(s)) = 1 \\ I_n(v(s))A_{normal}(v(s)) & \text{otherwise} \end{cases} \quad (6)$$

where A_{line} is the gradient of the result of filtering the binary image B with a Gaussian kernel such that $A_{line} = \nabla(G_\sigma * B)$ ($\sigma = 3$ in our implementation) and A_{normal} is the projection of surface normals into image-space.

The sample points $v(s)$ of the active contours are defined in the image space and therefore do not have direct access to the influence values stored at the mesh vertices. In order to provide access to the influence values, the triangle references are stored in an intermediary frame buffer defined at every pixel over the feature lines and the 3D model (Figure 8 (i)). All sample points of the active contour are updated by minimizing the energy function using Euler-Lagrange integration method. This update process is repeated until the active

© The Eurographics Association 2015.

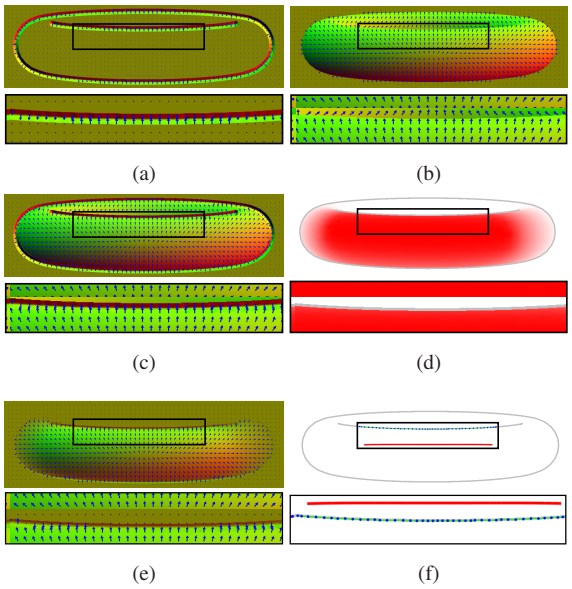


Figure 9: Attraction fields, influence area and line matching. The feature line attraction field A_{line} (a); the projected normals field A_{normal} (b); the combined attraction field before scaling (c); the *influence area* of a registered stroke n (d) and the corresponding attraction field A_n after scaling by the influence values (e); the proposed attraction field leads an active contour from a registered stroke (red line) to a current frame internal feature lines.

contour converges or a maximum number of iterations is exceeded (500 iterations in our implementation). A_{line} is only defined near external and internal feature lines while A_{normal} is defined at almost all points over the 3D model. However, the projected normals field A_{normal} does not provide the correct directions for the active contours to stop at internal feature lines. By combining A_{normal} with the feature line attraction field A_{line}, the proposed attraction field provides a path to current frame feature lines from points far away and provides the correct directions for internal lines. Figure 9 illustrates the relationship between the attraction fields, influence areas and line matching. In practice, the scaled attraction field A_n is only calculated at the active contour sample points and therefore the image in Figure 9 (e) is only provided for illustrative purposes.

5. Stroke Rendering

5.1. Visibility

The spine test method [CF10] is used to check for fragment visibility at the nearest point on the spine of the feature lines. However, the strokes generated from the active contours do not have depth information to perform the spine test. Therefore, the depth values at the spine of the feature lines are stored into an intermediary frame buffer (Figure 8 (f)). This

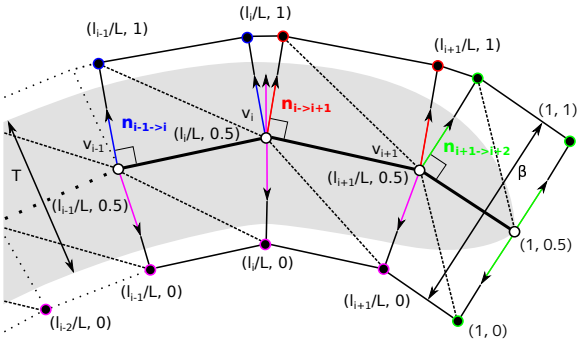


Figure 10: Stroke synthesis and texture coordinates.

frame buffer is used to assign a depth value to every sample point of the active contours that have reached the feature lines. These depth values which correspond to the spine of the feature lines are used to perform the spine test at each sample point before synthesizing the final strokes.

5.2. Synthesis

The final step of the rendering pipeline consists of synthesizing one or more strokes from the active contours that have successfully reached current frame feature lines. The style properties of the synthesized strokes are derived directly from their corresponding *registered stroke*. A stroke is synthesized from an active contour as a 2D polyline with successive visible sample points $v(s) = (x, y)$ that fulfill the following condition:

$$w_n I_n(v(s)) > \lambda \tag{7}$$

where $w_n = \left|1 - \frac{2\theta}{\pi}\right|$ is a view-angle weight such that θ is the angle between the current view direction and the view direction when the registered stroke n was stored, $I_n(v(s))$ is the influence value associated to a registered stroke n at sample point $v(s)$ and λ is a scalar threshold such that $0 < \lambda < 1$ ($\lambda = 0.01$ in our implementation). Additionally, we adopt a one-to-one trimming policy similar to Kalnins et al. [KDMF03] to prevent the overlapping of strokes at intermediary viewpoints by selecting only the parts of the strokes that maximize Eq. 7.

Final stroke rendering is done through a common method that uses a geometry shader in order to extrude triangles strips along the path of the synthesized strokes in the outward and inward directions. Each vertex i of the polygonal stroke is assigned with a texture coordinates $t = (t_u, t_v)$ dependent on the arc length l_i from the start of the stroke to vertex i and the total arc length L (Figure 10). The shape of a stroke can be customized by changing its width as well as through tapering controlled by the α coefficients of a shifted

© The Eurographics Association 2015.

and scaled sigmoid function [SKCN07]:

$$S_\alpha(x) = \frac{2}{1+e^{-\alpha x}} - 1 \tag{8}$$

The stroke thickness is calculated as follows:

$$T = \begin{cases} \beta S_{\alpha_L}(L) S_{\alpha_{start}}(t_u) & \text{if } t_u < 0.5 \\ \beta S_{\alpha_L}(L) S_{\alpha_{end}}(1-t_u)) & \text{otherwise} \end{cases} \tag{9}$$

where β is the width of the triangle strip, α_L is the arc length coefficient that controls the influence of the length of the stroke on its thickness and α_{start}, α_{end} are the tapering coefficients that define the shape of the stroke near the starting and ending terminal points respectively.

In practice, the tapering and arc length dependent width are implemented in a fragment shader that discards all fragments which fulfill the following condition:

$$\beta|2t_v - 1| > T \tag{10}$$

6. Results

The proposed system provides an interactive way to locally stylize view-dependent lines extracted from 3D models. Figure 11 shows results for organic and non-organic models where strokes with different styles are used simultaneously to convey additional information such as material properties or feature sharpness. The process of stylizing models such as the ones shown in Figures 11, 12 and 13 only needs to be done for a small number of viewpoints and is pose invariant for animated models. Figure 14 shows how the style of the strokes is maintained from multiple viewpoints and Figure 12 shows how the proposed line matching method successfully matches current frame feature lines of an animated model by transforming the *registered strokes* in accordance with the underlying surface. Figure 13 shows a comparison of results obtained with the single style approach found in most previous methods and the proposed stylization system. The single style approach produces strokes of different thickness thanks to our tapering and arc length dependent width implementation but is limited to global parameters that control the overall appearance of the extracted feature lines. In contrast, the results obtained with the proposed stylization system show how stylistic annotations and global decisions made by the user such as the omission of unwanted lines produce highly stylized and uncluttered line drawings.

6.1. Usability

One inevitable consequence of *localized stylization* is that the number of user operations and input time required increases. However, the ideal system would maintain the control over all details of line drawings while reducing the burden on the user. Therefore, we evaluated the usability of the proposed stylization system by measuring the input times to stylize various 3D models. Automatic stroke registration of all visible feature lines (e.g. from three orthogonal view directions) can generate an initial state where the style is defined for most viewpoints and therefore greatly reduce the input time required. This initial state is used as a base from where the user can do any of the following operations to make further adjustments: insert a new stroke; delete an existing stroke; modify the style of an existing stroke; cut an existing stroke; undo last operation. Figure 13 (bottom) shows the results obtained for three different 3D models and Table 1 shows the corresponding input times (less than 12 min in all cases).

	Input time
Bunny	9 min 20 s
Buddha	11 min 29 s
Beethoven	7 min 38 s

Table 1: Input times.

6.2. Performance

We evaluated the performance of the proposed method by measuring the pre-computation time and frame rate for various 3D models. The pre-computation phase registers all strokes from three orthonormal view-directions to generate an initial stylization state. The real-time phase consists of the generation of the frame buffers and the attraction field that will be used to lead active contours from the *registered strokes* to current-frame feature lines and render the final strokes. The hardware used to evaluate the performance is composed by an Intel(R) Core(TM) i7-3770 CPU @ 3.40 GH processor, 16 GB of system memory and a GeForce GTX 780 Ti graphics card. Table 2 shows the pre-computing times and frame rates for three different 3D models.

	# Triangles	# Reg. strokes	Pre-comp. time	Frame rate
Bunny	69666	197	7 min 27 s	12 fps
Buddha	25000	492	1 min 36 s	17 fps
Beethoven	30168	329	1 min 5 s	24 fps

Table 2: Pre-computation times and frame rates.

7. Conclusion and Future Work

In this paper we proposed a new *localized stylization* method that enables the user to define the style of individual strokes at view-dependent feature lines generated from polygonal surfaces. The proposed attraction field leads active contours to current frame external or internal feature lines far away from the *registered strokes*. Furthermore, the combination of active contour energy minimization and *influence areas* greatly reduces the number of incorrect matches. Line matching is performed in image-space and consequently is robust to changes in connectivity of the object-space lines. The proposed method assumes that the shape of the surface

© The Eurographics Association 2015.

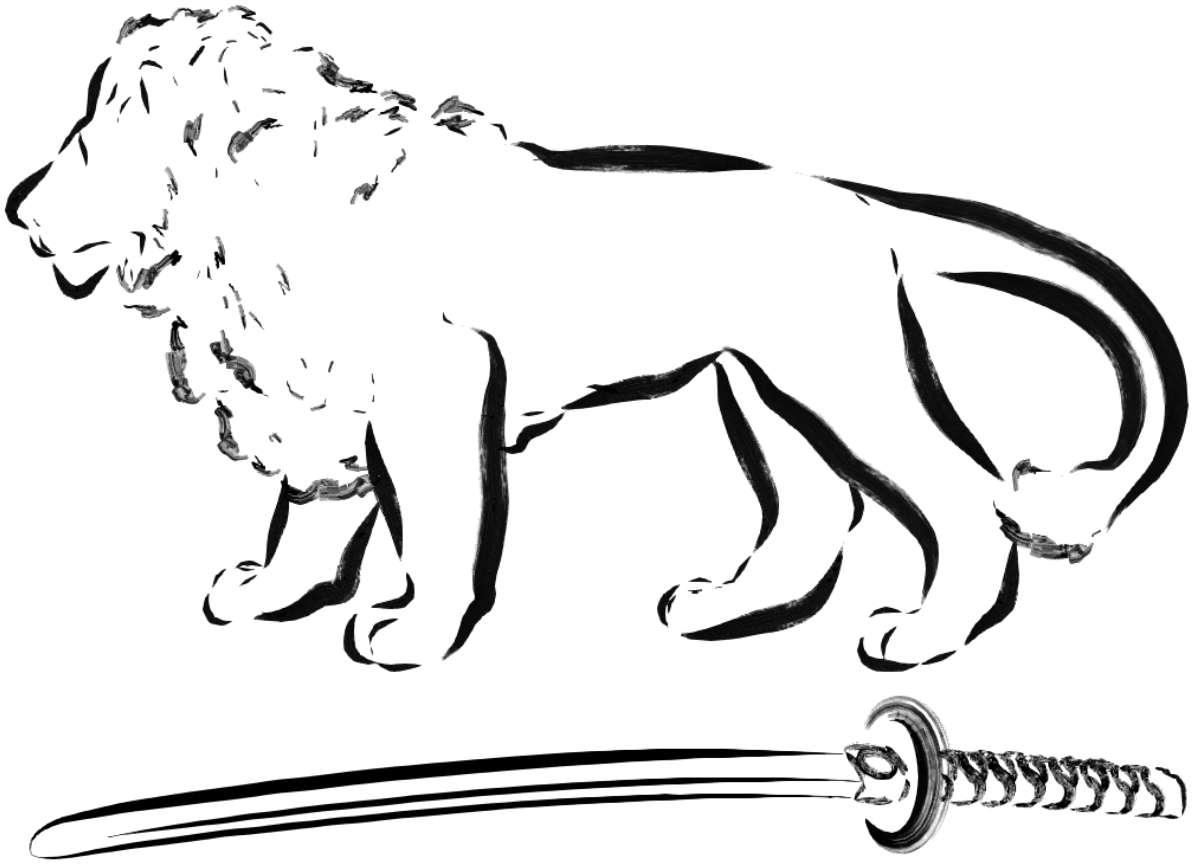


Figure 11: *Localized stylization* using different styles for organic and non-organic models.

remains relatively similar and therefore future work may deal with the problem of enabling complex animations such as morphing. In our implementation we used a one-to-one trimming policy to deal with overlapping strokes but an interesting problem would be how to blend different styles at intermediary viewpoints. Future work may also address other remaining problems such as maintaining temporal coherence to deal with topological events and smooth the transitions for high frame rate render sequences of line drawings containing multiple styles as well as implement a progressive level of detail method to emulate how artists usually depict far objects with higher degree of abstraction and close objects with finer detail.

Acknowledgements

The result images shown in this paper were generated using 3D models courtesy of the Stanford Computer Graphics Laboratory and the MIT Computer Graphics Group.

References

[BCGF10] BÉNARD P., COLE F., GOLOVINSKIY A., FINKELSTEIN A.: Self-similar texture for coherent line stylization. In *Proceedings of the 8th International Symposium on Non-Photorealistic Animation and Rendering* (New York, NY, USA, 2010), NPAR '10, ACM, pp. 91–97. doi:10.1145/1809939.1809950. 2

[BCK*13] BÉNARD P., COLE F., KASS M., MORDATCH I., HEGARTY J., SENN M. S., FLEISCHER K., PESARE D., BREEDEN K.: Stylizing animation by example. *ACM Trans. Graph. 32*, 4 (July 2013), 119:1–119:12. doi:10.1145/2461912.2461929. 2

[BFP*11] BUCHHOLZ B., FARAJ N., PARIS S., EISEMANN E., BOUBEKEUR T.: Spatio-temporal analysis for parameterizing animated lines. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Non-Photorealistic Animation and Rendering* (New York, NY, USA, 2011), NPAR '11, ACM, pp. 85–92. doi:10.1145/2024676.2024690. 2

[BLC*12] BÉNARD P., LU J., COLE F., FINKELSTEIN A., THOLLOT J.: Active strokes: coherent line stylization for animated 3D models. In *Proceedings of the Symposium on Non-Photorealistic Animation and Rendering* (Aire-la-Ville, Switzerland, Switzerland, 2012), NPAR '12, Eurographics Association,

© The Eurographics Association 2015.

pp. 37–46. URL: http://dl.acm.org/citation.cfm?id=2330147.2330156. 2, 4

[CF10] COLE F., FINKELSTEIN A.: Two fast methods for high-quality line visibility. *IEEE Transactions on Visualization and Computer Graphics 16*, 5 (Feb. 2010). 6

[CGL*12] COLE F., GOLOVINSKIY A., LIMPAECHER A., BARROS H. S., FINKELSTEIN A., FUNKHOUSER T., RUSINKIEWICZ S.: Where do people draw lines? *Commun. ACM 55*, 1 (Jan. 2012), 107–115. doi:10.1145/2063176.2063202. 1

[CS13] CARDONA L., SAITO S.: Customizable strokes for localized stylization of view-dependent lines extracted from 3D models. In *International Conference on Cyberworlds (CW) (2013)* (Oct 2013), pp. 140–146. doi:10.1109/CW.2013.62. 2

[Dec96] DECAUDIN P.: *Cartoon Looking Rendering of 3D Scenes*. Research Report 2919, INRIA, June 1996. URL: www.antisphere.com/Research/RR-2919.php. 1

[DFR04] DECARLO D., FINKELSTEIN A., RUSINKIEWICZ S.: Interactive rendering of suggestive contours with temporal coherence. In *Proceedings of the 3rd International Symposium on Non-Photorealistic Animation and Rendering* (New York, NY, USA, 2004), NPAR '04, ACM, pp. 15–145. doi:10.1145/987657.987661. 1

[DFRS03] DECARLO D., FINKELSTEIN A., RUSINKIEWICZ S., SANTELLA A.: Suggestive contours for conveying shape. *ACM Trans. Graph. 22*, 3 (July 2003), 848–855. doi:10.1145/882262.882354. 1

[GTDS10] GRABLI S., TURQUIN E., DURAND F., SILLION F. X.: Programmable rendering of line drawing from 3D scenes. *ACM Trans. Graph. 29*, 2 (Apr. 2010), 18:1–18:20. doi:10.1145/1731047.1731056. 2

[Her99] HERTZMANN A.: Introduction to 3D non-photorealistic rendering: Silhouettes and outlines. *Non-Photorealistic Rendering. SIGGRAPH 99* (1999). 1

[HZ00] HERTZMANN A., ZORIN D.: Illustrating smooth surfaces. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 2000), SIGGRAPH '00, ACM Press/Addison-Wesley Publishing Co., pp. 517–526. doi:10.1145/344779.345074. 2

[IB06] ISENBERG T., BRENNECKE A.: G-strokes: A concept for simplifying line stylization. *Computers & Graphics 30*, 5 (2006), 754 – 766. doi:10.1016/j.cag.2006.07.006. 2

[IFH*03] ISENBERG T., FREUDENBERG B., HALPER N., SCHLECHTWEG S., STROTHOTTE T.: A developer's guide to silhouette algorithms for polygonal models. *Computer Graphics and Applications, IEEE 23*, 4 (July 2003), 28–37. doi:10.1109/MCG.2003.1210862. 2

[JDA07] JUDD T., DURAND F., ADELSON E.: Apparent ridges for line drawing. In *ACM SIGGRAPH 2007 papers* (New York, NY, USA, 2007), SIGGRAPH '07, ACM. doi:10.1145/1275808.1276401. 1

[K*84] KOENDERINK J., ET AL.: What does the occluding contour tell us about solid shape. *Perception 13*, 3 (1984), 321–330. 1

[KDMF03] KALNINS R. D., DAVIDSON P. L., MARKOSIAN L., FINKELSTEIN A.: Coherent stylized silhouettes. In *ACM SIGGRAPH 2003 Papers* (New York, NY, USA, 2003), SIGGRAPH '03, ACM, pp. 856–861. doi:10.1145/1201775.882355. 2, 4, 6

[KH11] KARSCH K., HART J. C.: Snaxels on a plane. In *Proceedings of the ACM SIGGRAPH/Eurographics Symposium on Non-Photorealistic Animation and Rendering* (New York, NY, USA, 2011), NPAR '11, ACM, pp. 35–42. doi:10.1145/2024676.2024683. 2

[KMM*02] KALNINS R. D., MARKOSIAN L., MEIER B. J., KOWALSKI M. A., LEE J. C., DAVIDSON P. L., WEBB M., HUGHES J. F., FINKELSTEIN A.: WYSIWYG NPR: drawing strokes directly on 3D models. *ACM Trans. Graph. 21*, 3 (July 2002), 755–762. doi:10.1145/566654.566648. 2

[KST08] KOLOMENKIN M., SHIMSHONI I., TAL A.: Demarcating curves for shape illustration. *ACM Trans. Graph. 27*, 5 (Dec. 2008), 157:1–157:9. doi:10.1145/1409060.1409110. 1

[KWT88] KASS M., WITKIN A., TERZOPOULOS D.: Snakes: Active contour models. *INTERNATIONAL JOURNAL OF COMPUTER VISION 1*, 4 (1988), 321–331. 4

[LBSP14] LAWONN K., BAER A., SAALFELD P., PREIM B.: Comparative evaluation of feature line techniques for shape depiction. In *Vision, Modeling & Visualization* (2014), Bender J., Kuijper A., von Landesberger T., Theisel H., Urban P., (Eds.), The Eurographics Association. doi:10.2312/vmv.20141273. 2

[LMLH07] LEE Y., MARKOSIAN L., LEE S., HUGHES J. F.: Line drawings via abstracted shading. In *ACM SIGGRAPH 2007 papers* (New York, NY, USA, 2007), SIGGRAPH '07, ACM. doi:10.1145/1275808.1276400. 1

[MKG*97] MARKOSIAN L., KOWALSKI M. A., GOLDSTEIN D., TRYCHIN S. J., HUGHES J. F., BOURDEV L. D.: Real-time nonphotorealistic rendering. In *Proceedings of the 24th Annual Conference on Computer Graphics and Interactive Techniques* (New York, NY, USA, 1997), SIGGRAPH '97, ACM Press/Addison-Wesley Publishing Co., pp. 415–420. doi:10.1145/258734.258894. 2

[NM00] NORTHRUP J. D., MARKOSIAN L.: Artistic silhouettes: A hybrid approach. In *Proceedings of the 1st International Symposium on Non-photorealistic Animation and Rendering* (New York, NY, USA, 2000), NPAR '00, ACM, pp. 31–37. doi:10.1145/340916.340920. 2

[OBS04] OHTAKE Y., BELYAEV A., SEIDEL H.-P.: Ridge-valley lines on meshes via implicit surface fitting. *ACM Trans. Graph. 23*, 3 (Aug. 2004), 609–612. doi:10.1145/1015706.1015768. 1

[SKCN07] SAITO S., KANI A., CHANG Y., NAKAJIMA M.: Curvature-based stroke rendering. *Vis. Comput. 24*, 1 (Nov. 2007), 1–11. doi:10.1007/s00371-007-0165-0. 7

[SP03] SOUSA M. C., PRUSINKIEWICZ P.: A few good lines: Suggestive drawing of 3D models. *Computer Graphics Forum 22*, 3 (2003), 381–390. doi:10.1111/1467-8659.00685. 2

[ST90] SAITO T., TAKAHASHI T.: Comprehensible rendering of 3-d shapes. *SIGGRAPH Comput. Graph. 24*, 4 (Sept. 1990), 197–206. doi:10.1145/97880.97901. 1

[TABI07] TODO H., ANJYO K.-I., BAXTER W., IGARASHI T.: Locally controllable stylized shading. *ACM Trans. Graph. 26*, 3 (July 2007). doi:10.1145/1276377.1276399. 2

[XHT*07] XIE X., HE Y., TIAN F., SEAN H.-S., GU X., QIN H.: An Effective Illustrative Visualization Framework Based on Photic Extremum Lines (PELs). *IEEE Transactions on Visualization and Computer Graphics 13* (2007), 1328–1335. doi:10.1109/TVCG.2007.70538. 2

© The Eurographics Association 2015.

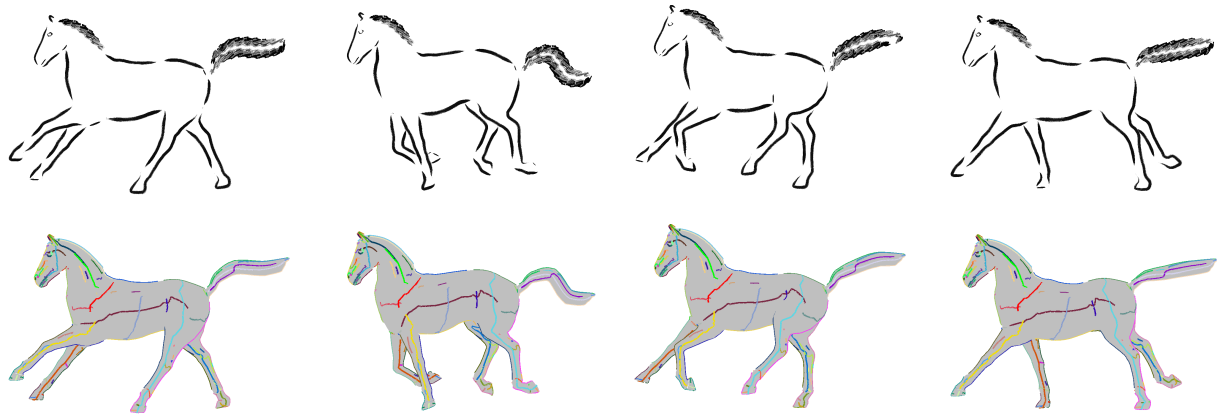


Figure 12: *Localized stylization* using different styles for an animated model.

Figure 13: Comparison of results for three 3D models obtained with single style (top) and with the proposed method (bottom).

© The Eurographics Association 2015.

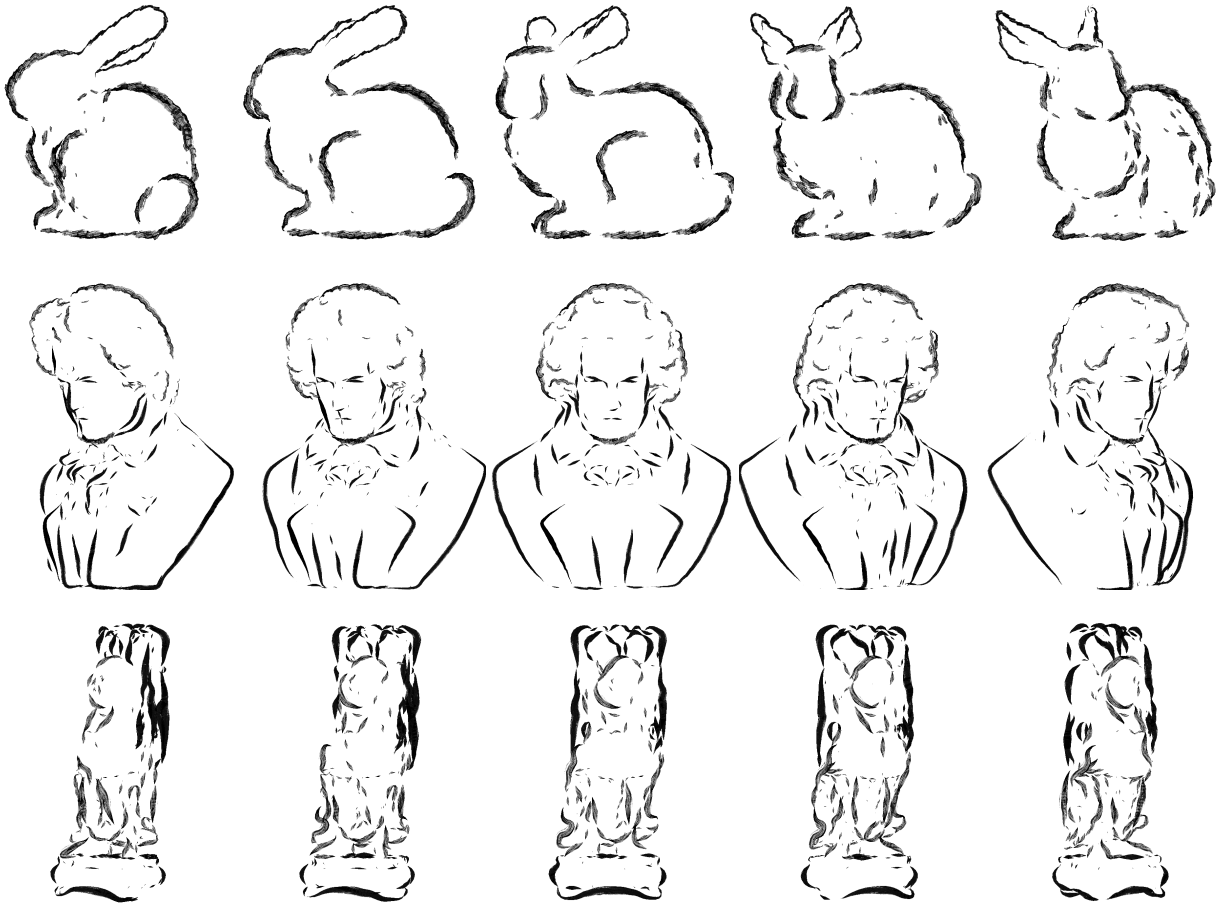


Figure 14: Locally stylized strokes from multiple viewpoints.

© The Eurographics Association 2015.