Non-Photorealistic Rendering

Anna Vilanova

1 Introduction

Rendering in computer graphics means the process by which a virtual scene is converted into an image. Pioneer work in computer graphics was done to achieve photorealism. The goal was to create synthetic images that can be mistaken by photographs.

It is rather absurd that a field of study is named after what it is not. Stanislaw Ulam said once that "The study of non-linear physics is like the study of non-elephant biology".

Non-photorealistic rendering is the means of generate imaging that does not aspire to realism. It is difficult to express what exactly it is, therefore it is easier to express what it is not.

Photorealism has a lot of applications (special effects, design visualization and so on). But simulating reality is not as important as creating the illusion of reality in the mind of the observer. There are several motivations for generating images that does not pretend to simulate reality.

Figure 1: Van Gogh, The Starry Night(June 1889)

Artistic

Realism is not an invention of the computer graphics. Already painters, like Jan Vemeer, tried to reproduce the reality through their paintings. It is believed that Vermeer experiment with the camera obscura, an optical device that could project the image of sunlit objects placed before it with extraordinary realism. Although his high quality realism some critics were less appreciative since accused him from being cold and inartistic. When photography came, realism became less important in art and it has given place to more abstract representation. For example "The Starry Night" painted by van Gogh (see figure 1). A description of the painting could be: It was painted furiously and vibrates with rockets of burning yellow while planets gyrate like cartwheels. The hills quake and heave, yet the cosmic gold fireworks that swirl against the blue sky are somehow restful. The criticism to Vermeer has been applied to photorealistic rendering,too. It produced a rich set of tools in the hands of the artist but they are to perfect and lack on feeling. It is clear that the panacea of an artist is not and nor will be photorealism. An artist has to choose the medium to better communicate each job (aesthetics, effectiveness and so on). Non-photorealistic rendering combine computer graphics with artistic techniques. It is a computer assisted art form.

In animations (e.g. cartoons) where a representation of an imaginary world is used, the rendering style does not need to be restricted to photorealism. It should be a choice of the artist. Using 3D computer graphics helps for producing animations (e.g. the characters movements).

NPR techniques can produce animations which look like a succession of impressionist paintings. This is difficult (or impossible) to achieve using hand made picture.

Comprehensibility and clarity

Human-drawn technical illustration are still used in manuals and to communicate technical information, better than photographies. They has the ability to communicate the shape and the structure of complex models with much more clarity than real pictures. Images convey information better by omitting extraneous detail, focussing the attention in relevant features. Medical texts almost always use hand drawn illustrations, since tiny and hidden structures are better described. Mechanical manuals for example, Boeing, although CAD models exists the high quality manuals are still generated by hand. NPR includes the techniques that imitated the illustration style of 3D modeled objects.

Aesthetics

In various market segments there is the necessity of providing forms of visualization which are non-photorealistic to attract people. Photorealism has its place there too, but sometimes the images are too precise and sterile for some applications.

Figure 2: Photorealistic and artist impression of a kitchen

For example, a sale systems used for design of kitchens and bathrooms in retail stores. This operate by observing prescribed rules that describe how units connect together. The designer then places the units within the space plan, and the rules are applied to snap the location of the units to the appropriate boundaries. This allows an assembly of predefined 3D objects to be designed very quickly and accurately. This assembly can then be provided to a rendering system from which photorealistic images can be generated (see figure 2). When selling a kitchen, it is usual to invest more in presenting the design for a client who is prepared to pay for high value units and appliances. Consequently, the services of an artist will often be employed to create a stylized depiction of the design.

Techniques overview

Initially NPR was based in 2D interactive paint systems (i.e., synthetic artist drawing objects: air brushes and pencil). The user applied these to a canvas to create pixel-based effects. Then 2D brush-oriented painting involving more sophisticated models for brush, canvas, strokes, etc..

2D/2$\frac{1}{2}$D post-processing systems appear as next step where raw or augmented image data is used as the basis for image processing. This include techniques that can be applied to photographic images to synthesize painterly renderings of those images.

The classic 3D computer graphics rendering pipeline exposes a number of opportunities within which NPR techniques can be applied to manipulate data in both 3D and 2D forms.

One of the key approaches that separate branches of research in NPR is the degree to which user intervention is required: some researchers have favored automatic techniques that require no or very limited user input, while others use the computer to place strokes at the guidance of the artist. It should be noted that any claim that an automatic process can produce ”art” or a ”painting” should be regarded as suspect. Making art is a creative and thoughtful process. It may even be uniquely human. The possibility of artificial creativity, let alone artificial intelligence, are open questions. None of the techniques described are candidates for true artificial creativity

2 Outlines and Silhouettes Extraction

One of the most basic and simple image enhancement is using silhouettes or outlines. The silhouettes provide a great deal of shape information. Lines provide better shape information that just shaded images. Humans are good in inferring shape from line drawings (e.g. children books are usually drawn like this).

The techniques to generate outlines can be divided in image space techniques and object space techniques. The image space techniques use as an input an already rendered image or enhanced image of the scene. The object space techniques generate outlines using directly the 3D model.

2.1 Image Space

An easy image space technique is to let a 3D graphics package render the image and then apply an edge detection algorithm (e.g., sobel filter). Unfortunately, the edges usually do not correspond to the silhouette of an object. Furthermore, there are

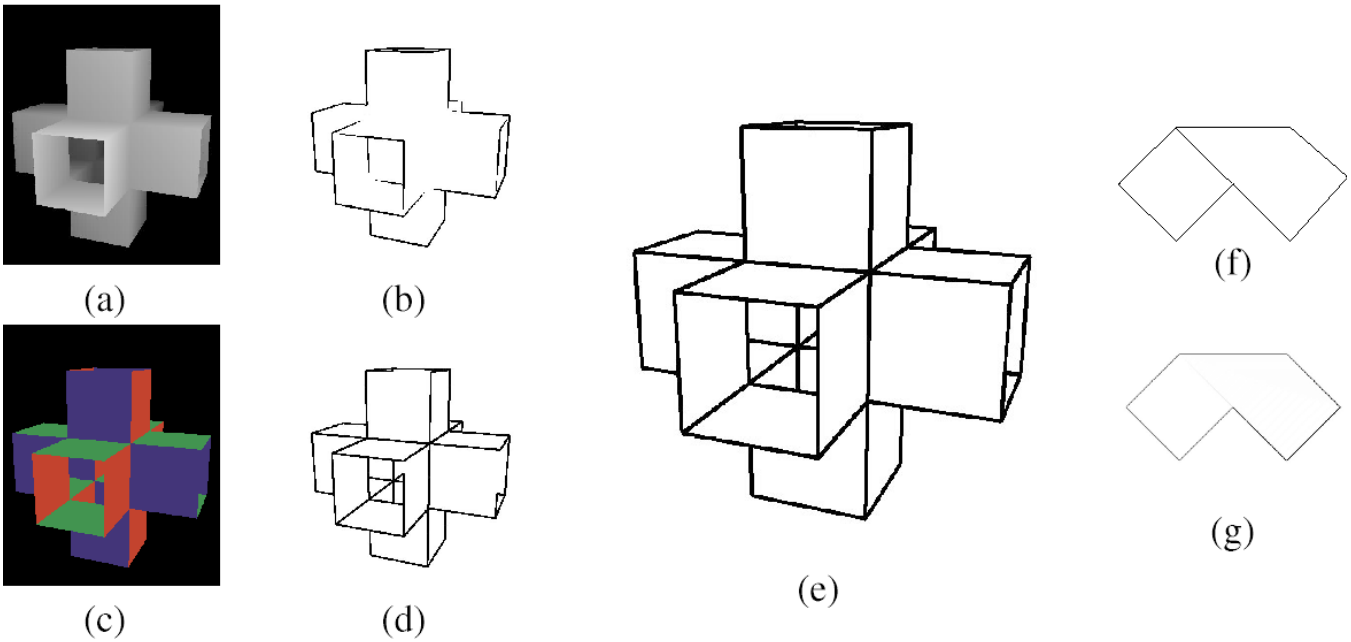

Figure 3: Outline drawing with image processing. (a) Depth map. (b) Edges of the depth map.(c) Normal map. (d) Edges of the normal map. (e) The combined edge images. (f) A difficult case: folded piece of paper (g) Depth edges.

problems with highly textured surfaces and there is no edge detected between overlapping objects of the same color.

Another technique is using a depth map image where the intensity of the image proportional to the depth. It uses the fact that the variation in depth between adjacent pixels is usually small over a single object but large between objects. The depth map can be obtained by reading the Z-buffer once the model has been rendered using a 3D graphics package. It detect C_0 surface discontinuities (see figure 3a and b). The drawback of this technique is that different objects at the same depth or creases in the same object are not detected.

A technique to detect C_1 discontinuities in a scene (i.e. changes in the surface orientation) is using image processing techniques like a second order differential filter to the depth map. Unfortunately these techniques are quite sensitive to noise and some modifications are needed to obtain reasonable results.

A possibility is calculating a surface normal map. The normal map contains for each pixel the normal vector of the surface rendered in the pixel. It can be generated using a general graphics package. The object color is set white and just diffuse. There are defined directional lights in the direction of X with red color, in the Y direction a green light and in the Z direction blue. Negative intensity lights are set in the negative direction of the axis (if negative intensity is not allowed we can do 2 rendering steps). Then the model is rendered. In the obtained image the (R,G,B) color in each pixel encodes the normal vector (x,y,z). Then applying an edge detection filter to the image the C_1 discontinuities are found (see figure 3c and d).

Together with the depth map we can detect C_0 and C_1 discontinuities (see figure 3e). Anyway this method does not solve all the cases like distinguish discuntinuities from large continuous changes and fold surfaces.

Important information of the 3D scene is discarted during rendering and this information cannot be obtained from a 2D image alone. The folded structure in figure 3f will never be detected in the presented image space methods. Also undesirable artifacts can be solved using object space instead of the enhanced image space.

2.2 Object Space

In this section we discuss how to find outlines of models in 3D. These methods are more involved than the image space methods but can produce curves with much higher precision. Furthermore these curves are also more suitable for additional processing. The curves that are interesting are: silhouettes, surface boundaries, creases and self-intersections. We will just deal with detection of silhouettes. Detecting boundaries and creases is straightforward, and can be done in advance for each model.

For triangular meshes, the silhouette consists of all the edges that connect back-facing polygons with front-facing polygons.

For smooth surfaces the silhouette can be defined as those surface points x_i with a surface normal n_i perpendicular to the view vector:

$$n_i \cdot (x_i - C) = 0$$

where C is the camera center (see figure 4). In this definition we do not take into account occlusion of the lines.

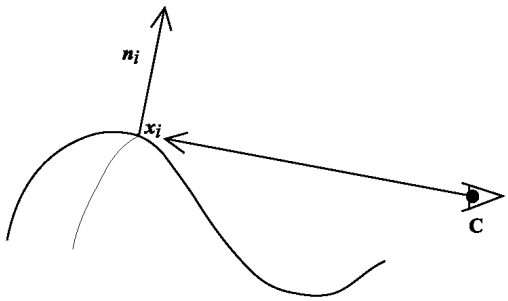

Figure 4: Silhouette of smooth surfaces.

To detect the silhouette in polygonal meshes an iteration through all the mesh edges is done. Each edge that connect back-facing polygons with front-facing polygons is defined as a silhouette edge. This loop must be performed for every camera change so there is need of speed up techniques.

Smooth surfaces, between others, are NURBS and subdivision surfaces. Both of this representations are based in polyhedral meshes; the actual smooth surface approximates or interpolates the mesh (see figure 5a and b). We will discuss just surfaces approximated by triangular mesh.

First for every vertex we compute:

$$d_i = \frac{n_i \cdot (x_i - C)}{||n_i|| ||(x_i - C)||} \qquad s_i = \begin{cases} + & d_i \geq 0 \\ - & d_i < 0 \end{cases}$$

We want to localize the points where d_i=0. Given a mesh edge between points x_i and x_j, if $s_i \neq s_j$ then there is a silhouette point on the edge (Note that the surface normal and the view vector are both continuous. Then a 0 crossing must exist). The position is approximated by linear interpolation of the vertices:

$$x' = \frac{|d_i|}{|d_i| + |d_j|} x_i + \frac{|d_j|}{|d_i| + |d_j|} x_j$$

If we have triangles there is just one unique case where a triangle will contain a silhouette. The points for every triangle are then connected (see figure 5c and d) and produce a piecewise-linear approximation to the silhouettes of the smooth surface. If the mesh have large triangles then the approximation will be very coarse. The solution to this is to refine the mesh.

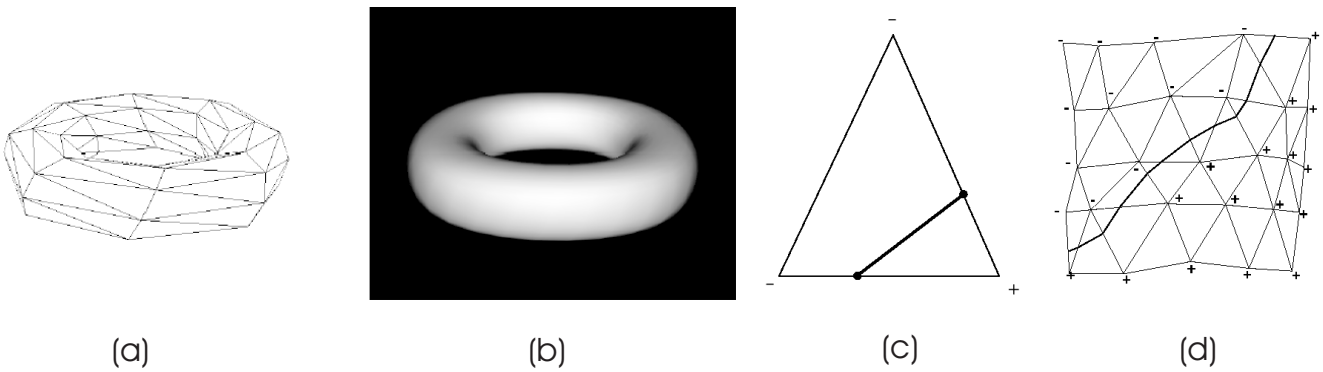

Figure 5: Silhouette extraction in object space for smooth object.

Once we have curves of interest from a surface we would like to determine which portions of these curves are visible. Hidden visibility is one of the oldest problems in computer graphics. The basic algorithm for computing visibility is to break all curves at potential changes in visibility, producing a set of curves where each curve is entirely visible or entirely invisible. There are three situations where the visibility of a surface curve can change: it passes under a silhouette, boundary or crease in the image plane; it passes intersects a silhouette, crease, or self-intersection curve on the surface; it is a silhouette or boundary and has a cusp.

Once the curves have been divided we can then test the visibility of each curve by performing a ray tests. A ray is cast in the direction of the view vector. If the ray intersects any surfaces before the curve then the curve is invisible; otherwise it is visible.

3 Pen-and-Ink Illustration

There are several advantage of pen-and-ink illustration over the shaded information. It is easier to copy, easier to reproduce in a book. It blends nicely with text, it has a linear quality since they use the same ink and paper. Its simplicity provides an appealing crispness and directness. There is also the possibility to give more emphasis in the important features.

Several elements are involved in a pen-and-ink illustration: strokes, tone and textures, and outlines.

The strokes are obtained by placing the nib of a pen in contact with a paper and allowing the nib to trace a path (thickness vary by varying the pressure on the nib). The strokes must not be too thin (washed-out appearance), and they must look natural not mechanical(thickness should vary along its lenght). Wavy lines give an impression of schematic and not yet finish.

Tone and value refer to the amount of visible light reflected towards the observer from a point in the surface. Combination of strokes must give the overall impression of the desired tone. The tone is achieved by the ratio of black to white ink in a region of the illustration. Tones are drawn with strokes of roughly equal spacing and weight. If the character of the strokes are varied the same strokes that are used to achieve a tone, are used to simulate texture. For example, crisp straight lines are good for glass, sketchy line is good for old materials.

Color and shading must be suggested by the combination of individual strokes. Manually is very difficult and time consuming to render the strokes for a large area, and it is nearly impossible to lighten once it has been rendered.

In realistic rendering there are no outlines they use the variation of tones and texture. However outlines are a natural mean of portraying objects. The character of the outline can also be a very good indicator of texture. Outlines are used for the contour of an object and the essentials of its interior.

Combination of tone and outlines is used. For example, straight lines are good for hard objects and soft lines are good for soft objects. Thick outlines suggest shadows, when there is no tone the outline is used to show the shape.

There are different methods how to produce pen-and-ink illustration in computer graphics. Some are based just in using as a base 2D pictures and human interaction. We will present a system where 3D models are used to automatically obtain a pen-and-ink illustration.

The 3D graphics pipeline must be modified to be used in a pen-and-ink illustration. Although this modification, some elements of the 3D graphics pipeline are kept. We also have a 3D model where textures are assigned to every surface. In pen-and-ink illustration this textures are stroke textures. A lighting model is used, for example a Phong model. Also the visibility of the surfaces must be calculated and it can be done with visible surface algorithms like the BSP-tree. Shadows need to be calculated and it can be done using algorithm used for realistic rendering which render the shadows by computation of shadow polygons.

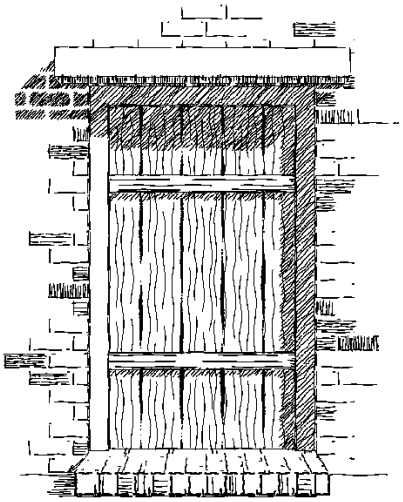

Figure 6: A door illustration where the lines of wood grain are drawn with an even pressure while the lines between the planks use varying pressure

However for pen-and-ink illustration some other requisites must be fulfilled.

We need to consider the 2D adjacency information which influence the rendering of the strokes. This is achieved using a 2D spatial subdivision, a 2D BSP-tree.

The polygons are not scan converted anymore for texturing them. The texture and tone must be conveyed with some form of hatching.

The strokes must be clipped to the regions they are texturing. The clipping must be fast and should not be pixel-based. We should not remove the pixels of the stroke that are outside the clipping region since this would give unnatural appearance. Wavy strokes should be able to go outside of the clipping region.

Boundary outlines must be drawn in a way that takes into account texture of the surrounded region and the adjacency information stored in the planar map.

To render the scene we render visible surfaces and shadow polygons. The visible surfaces are rendered using procedural textures attached to each surface to generate the strokes that convey the correct tone and texture for the surface. Then the strokes are clipped to the visible portion of surface using the planar map. Finally the outlines are drawn following.

3.1 Strokes

A stroke consist of: a path $P(u)$that gives the the overall sweep of the stroke as function of the parameter u; a nib, $N(p)$ which represents the the cross-sectional footprint of the stroke(e.g. circle), the nib is given in function of the pressure; a character function which describes the waviness of the curve $C_w(u)$ (how the curve departs from the path) and a function $C_p(u)$ that gives the pressure of the nib. A stroke is defined as all pixels in the region that:

$$S = (P(u) + C_w(u)) * N(C_p(u))$$

where * denotes the convolution of two parameterized point sets ($A(u) * B(u) = \bigcup\{a+b|a \in A(u) \wedge b \in B(u)\}$).

A stroke is rendered by scan converting the path after waviness is added and stamp a copy of the nib scaled by the pressure value in place of drawing each pixel. Different shapes could be achieved from the combination of C_w and C_p. C_w can be a sine-wave with randomly perturbed amplitude and wavelenght. C_p can be a random function that lifts the pen off the paper or a random sine wave that creates different thickness. In the figure 6 the lines of wood grain are drawn with an even pressure while the lines between the planks use varying pressure.

3.2 Tone and textures

In pen-and-ink illustration, tone and textures are produced at the same time. Using a Phong model, the tone v is computed. The tone is a value between 0 and 1 where 0 determine white and 1 define black. This tone reflects the luminance of the scene in this point.

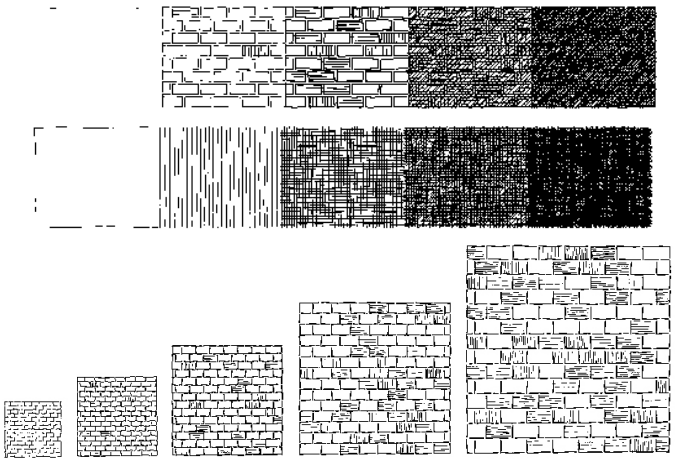

Figure 7: Priority strokes textures.

Then the procedural texture is used to generate the tone v. As a simple example if A is the area and a nib of w is used the number of non overlapping strokes needed is $v \cdot \frac{A}{w}$. But to be precise it is also important to take into account the overlapping between textures. For example, in the cross-hatching it can be calculated as in any direction x lines. It means we have $2x$ strokes. If we agree that the overlapping will be x^2. Then $2x - x^2$ should be the tone v. then $x = 1 - sqrt\sqrt{1-v}$.

For complicated stroke textures prioritized stroke textures are used. The strokes have a priority assigned. Then first we render the highest priority strokes if the tone is not achieved then the next higher prioritized strokes are added until the proper tone is achieved(see figure 7). For example the brick elements have higher priority, then the shaded bricks, the hatching strokes have along the entire surface has the lowest priority. The width of the nib also change depending of the tone.

A common problem with the figures created with computer is that they do not scale well. Enlargement is usually done in pixel bases and it produces ugly aliasing artifacts, or drawing the strokes at the same resolution which yields to thinner strokes and an overall lighter illustration. Reduction usually leads to large black mass of overlapping strokes. The prioritized stroke textures take into account the tone and the resolution, as can be seen in figure 7.

3.3 Outlining

Traversing the edges stored in the planar map, the outlines are extracted. The outlines are rendered taking into account tone and surrounded regions (adjacency information stored in the planar map). Interior outlines are used within a polygon to suggest shadow or to give view-dependent accents to the stroke texture. Every texture has its own boundary outline. An edge shared for two faces is just rendered if the tone between the two faces are not different enough.If not the boundary of the closer face is chosen.

3.4 Indication

It is important to convey the impression of texture without drawing every last stroke. This lends economy to an illustration and makes it more powerful engaging the imagination of the viewer rather than revealing everything. This pen-and-ink illustration technique is called indication.

It is not easy to automatize the process of indication in order that enough detail is in the right place and also fading the detail out in the anornamented parts of the surface in a subtle way. A semi-automatic method is used. The user interactively places detail segments on the image to indicate where the detail should appear. Each segment is projected and attached to the texture of the 3D surface for which indications is being designed. A field is defined by the detail segment in any point. Then the field is evaluated in the center of any element of the texture (e.g. brick) and the element is rendered just in case the indication value is over a preset value (see figure 8).

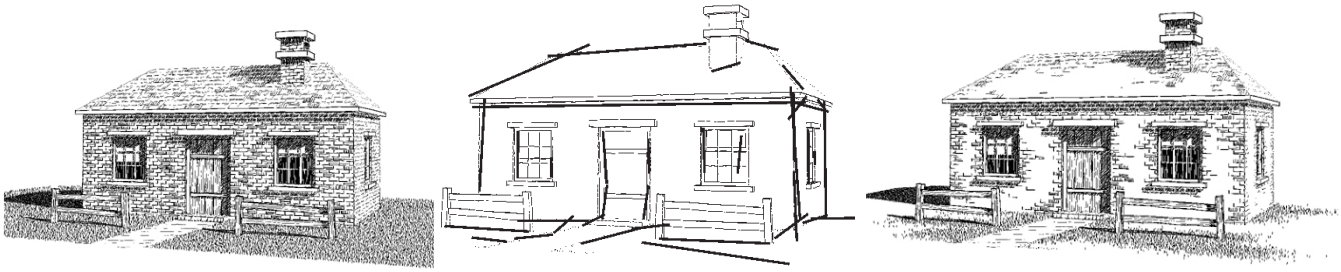

Figure 8: From left to right: (a)illustration without using indication technique,(b) detail segments,(c) final result using the indication technique

4 Painterly Rendering

In this section we will present techniques to generate painting like pictures having as input a 2D image. It is pretend to simulate the brush strokes that a painter would generate to paint the scene represented in the picture. In this work we will basically present techniques to automatically generate renderings that resemble expressionism or impressionism paintings. There are techniques which do it interactively. However the interaction is costly and it is impossible for generating video animations if it has to be done from frame to frame.

4.1 Brush Strokes

A brush stroke is represented by a line with a line center (c_x, c_y), with a length L, with a brush thickness radius R, and with a given orientation theta θ. (c_x, c_y) are positioned every 2 pixels in X and Y direction (in practice the user sets the radii and length and the initial spacing distance). (c_x, c_y) are floating points for subpixel positioning.In figure 9b, θ is a constant of 45 degrees; the stroke has a constant color assigned by bilinear interpolation of the original image in the point (c_x, c_y). The rendering order of the strokes is randomized.

To create a hand touched look, a random increment is assigned to the length and the radius, ΔL and ΔR. The color is perturbed by a random amount for any component,$(\Delta r, \Delta g, \Delta b)$, and it is scaled,$\Delta intensity$ in a random amount. The angle is also perturbed by an random increment angle of θ, $\Delta\theta$.

In order to preserve the edges the strokes are clipped to edges that they encounter in the original image (see figure 9c). The original image is first blurred with a Gaussian filter (noise reduction). The intensity of the image is calculated using $(30 * r + 59 * g + 11 * b)/100$. Then a edges detection is applied. Given a stroke center (c_x, c_y) the line is growing till the length L is reached, or the edge is found. An edge is found if the magnitude of the gradient decreases in the direction the stroke is going.

The strokes are rendered with a fall-off that decreases the alpha for composing from 1.0 to 0.0. to avoid aliasing. Usually the strokes pass the clipped endpoints of the line (this fact adds also a kind of hand touch).

Usually the painters generate strokes following the shapes of the figures and not in a unique direction. To simulate that effect the strokes are drawn normal to the gradient direction, which is the direction of 0 changes (see figure 9d). The equation

$$\theta = \arctan\left(\frac{G_x}{G_y}\right) + 90$$

where (G_x, G_u) is the gradient vector which gives the new direction. A perturbation $\Delta\theta$ is also added.

Using the previous method it comes the problem that in homogeneous regions we get noisy orientations. In order to draw more coherent strokes, when the gradient is near to 0, the gradient is smoothly interpolated using the direction defined in the region boundaries (see figure 9e). The good gradients not necessarily lie in a uniform grid, therefore the interpolation used does not assume uniformly spaced data in both directions.

Textured brushes can also be used to add character to the strokes. A rectangle is defined for the line stroke and a texture (see figure 9f). The texture is multiplied by the stroke color in each stroke to obtain the final color.

Figure 9: The images from left to right top to bottom order: (a) original image, (b) strokes with 45 degrees orientation, (c) edge clipped strokes, (d) normal gradient oriented strokes, (e) the orientation of the strokes in homogeneous regions is interpolated from the neighboring gradients, (f) textured strokes

4.2 Video animations

If a video wants to be produced using the previous technique, we have to take into account frame to frame coherence. The frames cannot be rendered independely, otherwise the images will be flickering since random values will be different every

time.

The first frame is rendered as specified. The strokes are kept in a database and modified from frame to frame.

Then the optical flow between two consecutive frames is calculated. The optical flow gives information about the coherent movement of the image pixels in the image. It generates a vector field which is used to move the brush strokes following the flow.

After displacement, some strokes may be outside the edges of the image. These ones will be discarted. We also have to eliminate strokes where there are too much of them and add strokes where they are too sparse.

A Delenuary triangulation using the strokes centers is performed. Afterwards the triangles are subdivided in order that there are no triangles with an area larger than the maximum supplied area. New vertices and brush strokes are introduced in this areas (see figure 10).

On the other hand, if the distance between two points of an edge is less than a user specified length the corresponding brush stroke that is drawn closer to the back is discarded.

We have now a list of old strokes and new strokes. The old strokes order is kept for temporal coherence. The new strokes are randomized respect themselves and uniformly distributed between the old strokes.

Using this technique we can generate impressionist paintings videos having as an input a video sequence.

4.3 Brush strokes by layers

In the previous sections all the strokes in the painting had similar thickness which does not really corresponds to reality. Often the artist starts painting a rough sketch and go back over the painting with a smaller brush to add detail. The thickness of the strokes is used as a painting technique to draw the attention to the parts of the image that have more detail or the parts the painter wants.

In the technique we will present the fine brushes are just used where they are necessary to refine the painting and the rest is left. We have different brush thickness $(R_1,\ldots,R_n)$. The algorithm processes the painting adding new layers to a current image. For any layer a reference image is calculated which is a blurred image of the original one using a Gaussian filter with a standard deviation depending on the brush thickness $\theta_i = f_\theta R_i$ (f_θ is a constant factor). Initially the canvas has a constant color.

The reference image represent the image we want to approximate by painting with the current brush size R_i. We want to capture details that are at least as large as the brush size. The procedure locates the parts of the current painted image that differ from the reference image (i.e., blurred image) and paints them with brush strokes. A threshold T is defined, the areas where the current image differ less than this threshold from the blurred image are left unchanged. Large values of T produce rough paintings. T is decreased to produce paintings that closely match the source image.

The brush strokes are rendered randomized using the Z-buffer.

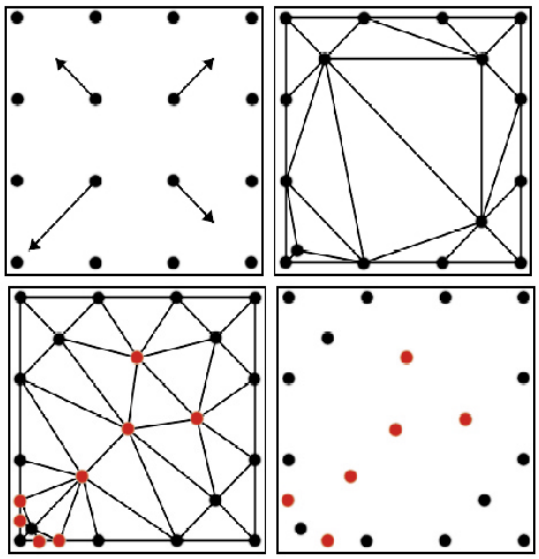

Figure 10: From left to right and top to bottom the steps showing and example of how the strokes are moved, added and eliminated

4.4 Curved Brush Strokes

Long curved brush strokes expresses a gesture, a curve or a play of light in on a surface.

The strokes we will present are cubic B-spline with constant thickness and color. The strokes rendering is done by dragging a circular brush mask along the sweep of the spline.

The control points of the spline are placed that follow the normal of the image gradient. The spline control points placement algorithm begins in a point in the image (x_0,y_0), with a given radius brush R. The assigned color is the color of the reference image in (x_0,y_0). The next point is placed in the direction normal to the gradient at a distance R from (x_0,y_0). As we mentioned R represents the level of detail we will capture with the brush.

We actually have 2 direction that are normal to the gradient. The sign of the direction is chosen in order to minimize the curvature. We chose the one with an angle smaller or equal to $\pi/2$ respect to the previous direction. The stroke control points are generated till the color of the stroke deviates from the color under the current control point of the curve more than a specified threshold. The stroke is also terminated when the predetermined maximum stroke length is reached.

Using the different parameters described in the previous algorithm (i.e., the brush sizes R_i, the approximation threshold T, the blur factor f_θ, the minimum and maximum stroke length, the opacity and the color) different painterly styles (e.g. impressionism, expressionism, puntillism, watercolor) can be performed (see figure 12).

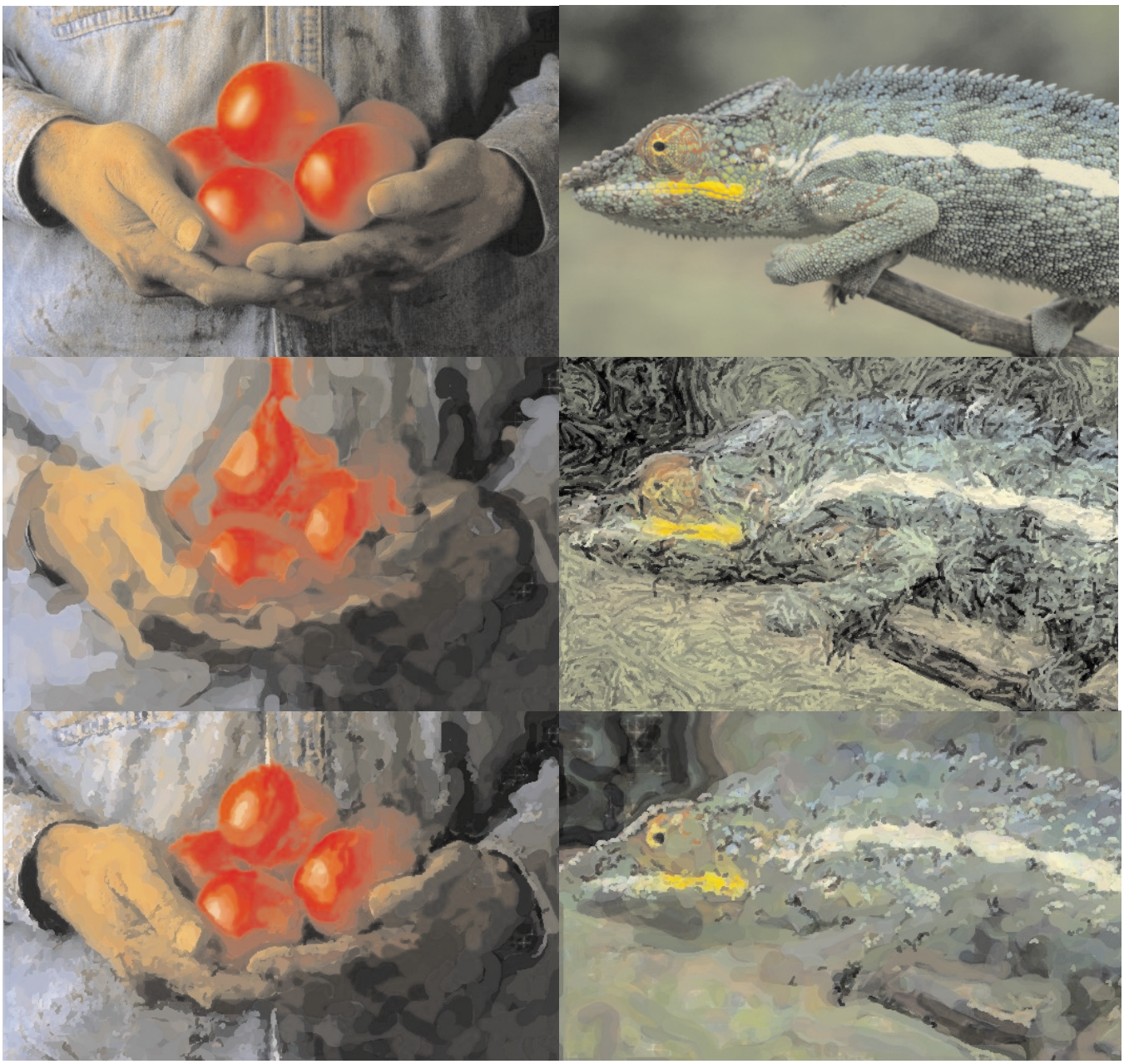

Figure 12: The left images from top to bottom show the results of using brush strokes with different thickness and layers to generate an impressionist alike image: (a) original image, (b) first layer with a high value of R, (c) result of applying 4 layers with different stroke radius. The right images from top to bottom show different painterly styles that can be generated with the curved strokes: (a) original image (b) expressionim (c) watercolor.

5 Technical illustration

The illustrations in manuals and textbooks or encyclopedias reveals shading and line conventions that are different than the ones presented till now. The use of this conventions produces technical illustration as a subset of non-photorealistic rendering

techniques. Lines provide better shape information since humans are good in inferring shape from line drawings. These techniques can draw the attention to details that would be lost in shading.

When in addition to edge lines shading is performed, shape information can be maximazed if the shading uses colors and intensities that are distinct from both the black edge lights and the white highlights. The shading is done to add subtle shape attributes and adds information about the material properties. Shadows are used only when they do not obscure any detail.

5.1 Outlines

In technical illustration different line thickness have different meanings. A normal thickness is used throughout the image. A bold line is used with the outer edges and parts with open space behind them. Variation of line thickness along a single line, emphasizing the perspective of the drawing, with heavy lines in the foreground, tapering towards the farthest part of the object.

Lines, in almost all technical illustrations, are drawn in black. Occasionally, if the illustration incorporates shading, another convention may apply in which some interior lines are drawn in white. By using this convention, lines drawn in black and white suggest a light source, and denote the models orientation. For example, figure 13 shows how an artist may use white for interior lines, producing a highlight.

5.2 Diffuse Shading

In diffuse shading, a hue shift is added to the shading model. This allows reduction in the dynamic range of the shading to ensure that the highlights and edges will be visible.

The luminance of traditional shading sets as the cosine of the angle between light direction and surface normal:

$$I = k_d * k_a + k_d * \max(0, \vec{l} \cdot \vec{n})$$

where I is the color to be displayed, k_d is the RGB diffuse color and k_a is the ambient illumination.

In figure 14a a diffuse shaded image using the previous equation is done. It can be observed that it hides shape and material information in the dark regions. Adding edge lines and highlights would add information. Edge lines and highlights cannot be effectively added to figure 14a because the highlights would be lost in the light regions and the edge lines would be lost in the dark regions.

To add edge lines to the diffuse shaded image k_a could be raised until it is large enough that the dim shading is visually distinct from the black edge lines, but this would result in loss of fine details.To make the highlights visible on top of the shading, k_d could be lowered until it is visually distinct from white. An image with hand-tuned k_a and k_d is shown in figure 14b. This is the best achromatic image using one light source and traditional shading. This image is poor at communicating shape information, such as details in the claw nearest the bottom of the image, where it is colored the constant shade $k_d k_a$ regardless of the surface orientation.

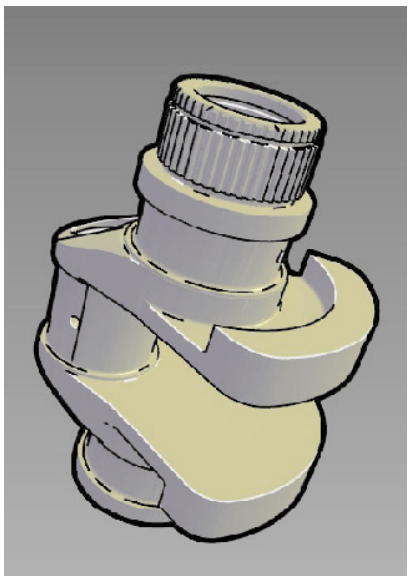

Figure 13: Technical illustration showing how interior lines are drawn in white to show highlights and thick black lines are used for the outlines

In a colored medium such as air-brush and pen, artists often use both hue and luminance (gray scale intensity) shifts. Adding black and white to a given color results in what artists call shades in the case of black and tints in the case of white. When color scales are created by adding gray to a certain color they are called tones. Such tones vary in hue but do not typically vary much in luminance. Tones are considered a crucial concept to illustrators and are especially useful when the illustrator is restricted to a small luminance range. Another quality of color used by artists is the temperature of the color. The temperature of a color is defined as being warm (red, orange, and yellow), cool (blue, violet, and green), or temperate (red-violets and yellow-greens). The depth cue comes from the perception that cool colors recede whereas warm colors advance. Not only is the temperature of a hue dependent upon the hue itself, but this advancing and receding relationship is effected by proximity.

The classic computer graphics shading model can be generalized to experiment with tones by using the cosine term $(\vec{l} \cdot \vec{n})$ to blend between two RGB colors, k_{cool} and k_{warm}:

$$I = \left(\frac{1 + (\vec{l} \cdot \vec{n})}{2} \right) k_{cool} + \left(1 - \frac{1 + (\vec{l} \cdot \vec{n})}{2} \right) k_{warm} \quad (1)$$

The result of applying the previous equation is shown in figure 14c. It uses color scale but with little luminance. It can be seen that the sence of depth can be comunicated with hue shifts. However the lack of strong cool-to-warm luminance hue shifts makes the shape information subtle. The colors chosen for the hue shift must be taken with care. Red-green undesirable due to red-green color blindness. So blue and yellow are the most common used. Yellow (sunlight) and blue (blue sky) have a large intensity shift that make it useful for our commitment.

Some luminanace variation is also desired in the change of tones. Then we have a blue to yellow tones shift and a scaled object color shade. The final used model is a linear combination of these two techniques. It is used a combination of tone scaled object-color and a cool-to-warm undertone. The undertones are simulated by a linear blend between blue-yellow and black/object-color tones:

$$k_{cool} = k_{blue} + \alpha k_d \quad k_{warm} = k_{yellow} + \beta k_d$$

This equation is combined with equation 1 to obtain the lighting model. Alpha and beta will determine the prominence of the object color and the strength of the luminance shift. In this model subtles of the claws are visible (see figure 14d).

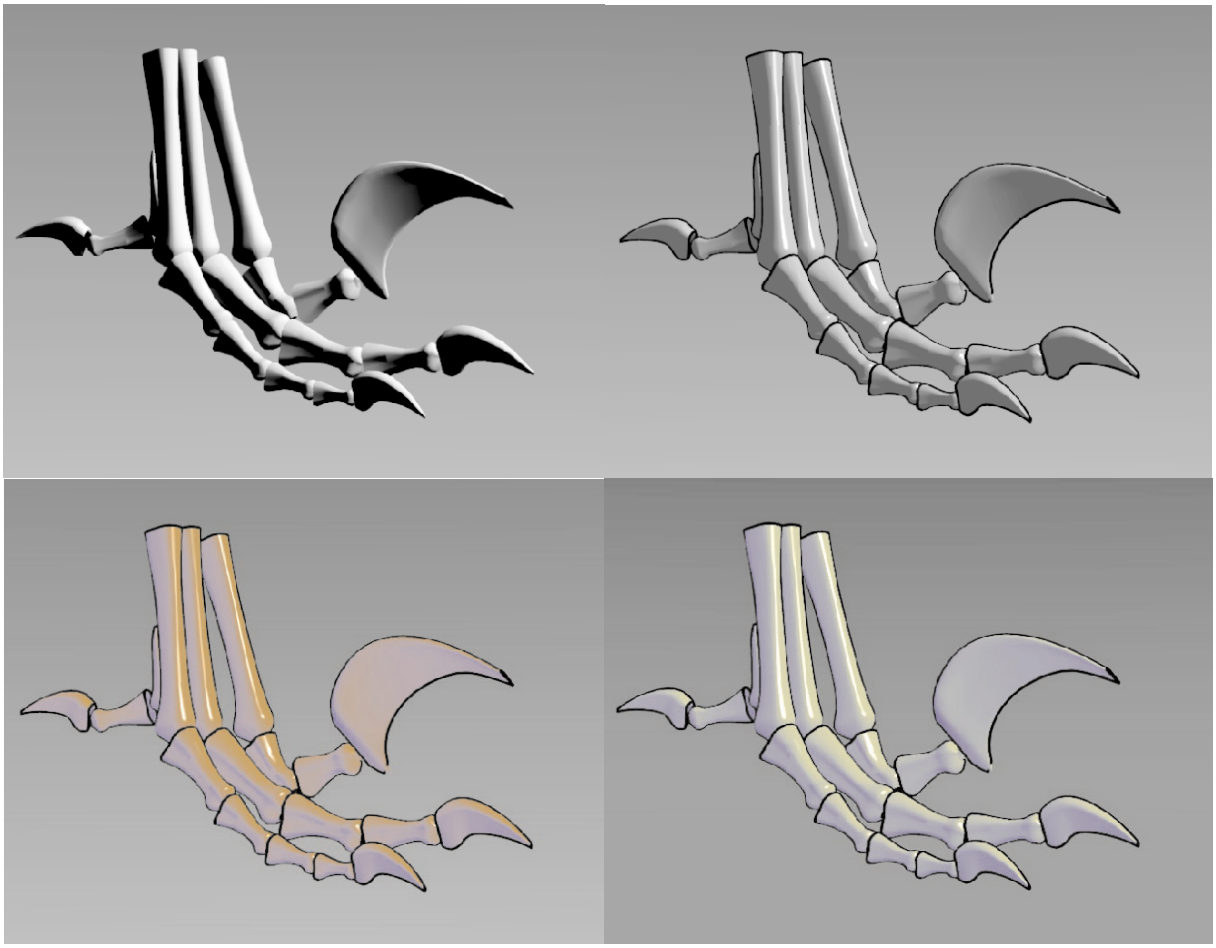

Figure 14: Images from left to right and top to bottom : (a) Usual diffuse shading, (b) diffuse shading with outlines and highlights, (c) Hue shift shading (d) Hue and luminace shading

5.3 Shading metal objects

The shading is used to distinguish between different materials. A metallic surface is presented by alternating light and dark bands. Milling creates anisotropic reflection lines which are streaked in the direction of the axis of minimum curvature, parallel to the milling axis. This is generated by maping a set of 20 stripes varying intensity along a parametric axis of maximum curvature. The stripes are random intensities between 0.0 and 0.5 with stripes closest to the light source direction is over written with white. Between stripe centers the colors are interpolated. The metal object using cool-warm hue is not as visual convincing as the achromatic image but it is more concisten when matte and metallic objects are shown together (see figure 15).

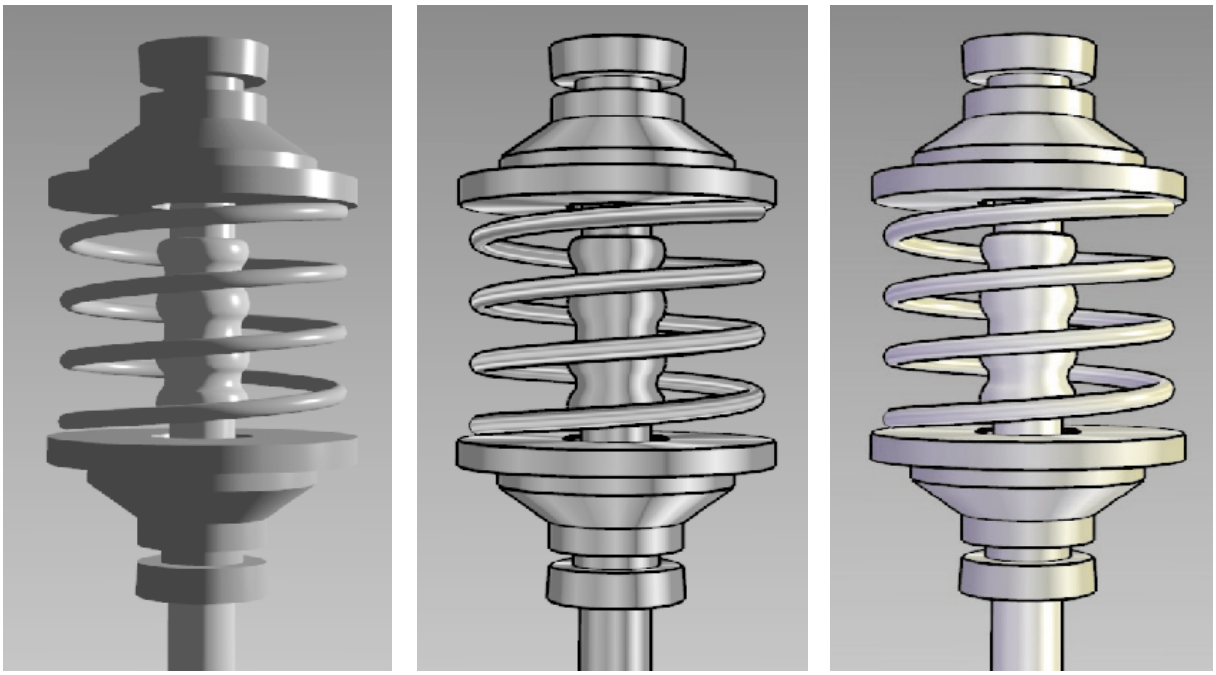

Figure 15: From left to right: traditional shading, achromatic metallic shading, cool-warm hue shift metallic shading.

References

[1] Non-photorealistic rendering, 1999. SIGGRAPH Lecture Notes 17.

[2] Cassidy J. Curtis, Sean E. Anderson, Joshua E. Seims, Kurt W. Fleischer, and David H. Salesin. Computer-generated watercolor. In Turner Whitted, editor, *SIGGRAPH 97 Conference Proceedings*, Annual Conference Series, pages 421–430. ACM SIGGRAPH, Addison Wesley, August 1997. ISBN 0-89791-896-7.

[3] Amy Gooch, Bruce Gooch, Peter Shirley, and Elaine Cohen. A non-photorealistic lighting model for automatic technical illustration. In Michael Cohen, editor, *SIGGRAPH 98 Conference Proceedings*, Annual Conference Series, pages 447–452. ACM SIGGRAPH, Addison Wesley, July 1998. ISBN 0-89791-999-8.

[4] Aaron Hertzmann. Painterly rendering with curved brush strokes of multiple sizes. In Michael Cohen, editor, *SIGGRAPH 98 Conference Proceedings*, Annual Conference Series, pages 453–460. ACM SIGGRAPH, Addison Wesley, July 1998. ISBN 0-89791-999-8.

[5] Peter Litwinowicz. Processing images and video for an impressionist effect. In Turner Whitted, editor, *SIGGRAPH 97 Conference Proceedings*, Annual Conference Series, pages 407–414. ACM SIGGRAPH, Addison Wesley, August 1997. ISBN 0-89791-896-7.

[6] Lee Markosian, Michael A. Kowalski, Samuel J. Trychin, Lubomir D. Bourdev, Daniel Goldstein, and John F. Hughes. Real-time nonphotorealistic rendering. In Turner Whitted, editor, *SIGGRAPH 97 Conference Proceedings*, Annual Conference Series, pages 415–420. ACM SIGGRAPH, Addison Wesley, August 1997. ISBN 0-89791-896-7.

[7] Takafumi Saito and Tokiichiro Takahashi. Comprehensible rendering of 3-D shapes. volume 24, pages 197–206, August 1990.

[8] Michael P. Salisbury, Sean E. Anderson, Ronen Barzel, and David H. Salesin. Interactive pen–and–ink illustration. In Andrew Glassner, editor, *Proceedings of SIGGRAPH '94 (Orlando, Florida, July 24–29, 1994)*, Computer Graphics Proceedings, Annual Conference Series, pages 101–108. ACM SIGGRAPH, ACM Press, July 1994. ISBN 0-89791-667-0.

[9] Michael P. Salisbury, Michael T. Wong, John F. Hughes, and David H. Salesin. Orientable textures for image-based pen-and-ink illustration. In Turner Whitted, editor, *SIGGRAPH 97 Conference Proceedings*, Annual Conference Series, pages 401–406. ACM SIGGRAPH, Addison Wesley, August 1997. ISBN 0-89791-896-7.

[10] Mike Salisbury, Corin Anderson, Dani Lischinski, and David H. Salesin. Scale-dependent reproduction of pen-and-ink illustrations. In Holly Rushmeier, editor, *SIGGRAPH 96 Conference Proceedings*, Annual Conference Series, pages 461–468. ACM SIGGRAPH, Addison Wesley, August 1996. held in New Orleans, Louisiana, 04-09 August 1996.

[11] Georges Winkenbach and David H. Salesin. Computer–generated pen–and–ink illustration. In Andrew Glassner, editor, *Proceedings of SIGGRAPH '94 (Orlando, Florida, July 24–29, 1994)*, Computer Graphics Proceedings, Annual Conference Series, pages 91–100. ACM SIGGRAPH, ACM Press, July 1994. ISBN 0-89791-667-0.

[12] Georges Winkenbach and David H. Salesin. Rendering parametric surfaces in pen and ink. In Holly Rushmeier, editor, *SIGGRAPH 96 Conference Proceedings*, Annual Conference Series, pages 469–476. ACM SIGGRAPH, Addison Wesley, August 1996. held in New Orleans, Louisiana, 04-09 August 1996.