Implicit painting of CSG solids

E. Akleman

Abstract

Implicit painting is a non-photorealistic rendering method for painting implicitly represented CSG solids. The method is based on the fact that when a difference equation is applied to a set of particles these particles will move in 3D space. We view the motion of the particles as the motion of the hands of several painters and the trajectories of the particles as long unbroken brush strokes, over the boundaries of solids obtained by set-operations. These boundary surfaces are used as if they are the canvases of painters. The difference equations provide the trajectories of the brushes, brush and paint types. We consider this painting as a creative or 'artistic' process in which the resulting artwork can be an image, a stereo image or even an animation that shows the painting process.

Introduction

There is much recent interest in both computer graphics research community and the animation industry in the development of new techniques for creating artistic [4] or natural looking images [3]. We believe the process of oil painting gives us a good paradigm to obtain both artistic or natural worn and torn images. The richness of painted images comes from the application of many layers of different paints over the canvas by the painter. Our goal is to obtain the same kind of richness that comes from the hard work of the artist by creating continuous motion of brushes over the boundaries of solids obtained by set-operations.

Meier obtained a painting effect by attaching brush images to static particles [4]. In this work, we attach objects, instead of images, to brushes (particles) relying on the continuous motion of brushes to produce the desired look. Although, this painting idea is based on oil painting, implicit paintings are different from oil paints in two ways.

- When painters do not paint a portion of canvas we can still see the canvas. However, in implicit surface paint, if a painter does not paint a certain portion of the surface, it leaves a hole which gives an interesting effect.
- Implicit paints can be considered as sculptures in addition to paintings. Therefore, these sculptures can be visualized in 3D by using stereo views. Such stereo views reveals the potential of implicit painting as a new painting paradigm. These paintings can also be transformed into stereo photographic prints.

In this paper, we focus on painting CSG solids since they provide simple algorithms for painting. The paper is organized as follows: In the next section, we will derive the set of difference equations which we use for implicit painting starting from Witkin and Heckbert equations. In the same section, we also explain speed, trajectory and paint controls. The following section introduces a simple algorithm for CSG solids and scenes. We give examples of of implicit painting of CSG solids in the last section.

Implicit painting

In oil painting, there is no objective goal. Painters do not stop painting untill they are satisfied; they use different brushes, different paints. These decisions of painters are completely subjective. Some painters even leave most of the canvas unpainted by stopping after a few brush strokes. On the other hand, some painters continuously applies new layers by covering canvas with several layers of paints.

Our goal is to paint the boundaries of set-theoretic solids exactly like oil painters paint their canvases. Painter will control a set of virtual brushes that continuously move in 3D space unless painters want them to stop. Characteristics of implicit representations are extremely suitable for development of difference equations for painting. We must first introduce the implicit solids $\mathcal{V}(f)$, and boundaries of these solids (implicit surfaces) $\mathcal{S}(f)$, by relating them to their implicit equations as

$$\mathcal{V}(f) = \{\ p \ \mid\ f(p) \leq 0\ \},$$

and

$$\mathcal{S}(f) = \{\ p \ \mid\ f(p) = 0\ \}.$$

In other words, the surface $\mathcal{S}(f)$ consists of all points $p = (x, y, z)$ that are zeros of the function F from $\Re^3$ to $\Re$.

In order to move the brushes on the surface of $\mathcal{S}(f)$, we need a set of difference equations. These *implicit painting* equations will be in the form of

$$p(n+1) = p(n) + v_0(f, X),$$

where $p(n+1)$ and $p(n)$ are the 3D positions of the brush at the times $n+1$ and n respectively. The velocity of the brush is given by the vector $v_0(f, X)$, where X denotes all the variables that effects the direction and length of the velocity vector. The painter will control the trajectory of the brush by changing X.

Like an actual brush, these *virtual* brushes, when they move, have to leave some paint behind them. This paint will be created by attaching *paint* objects, such as cylinders or quadrilaterals, to each moving brush. The attributes of the paint objects, such as sizes and material properties, will be functions of position and time. Painters will also be able to control these attributes. By changing these attributes, it is possible to obtain several *paint* effects. For instance, depending on the radius and connectivity, cylinders create effects such as wire, toothpaste, rusted wire or tree branches; on the other hand, quadrilaterals can create the appearance of ribbon or paper; and several cylinders together create classical oil paint effect.

In order to compute the orientations of the paints, it is necessary to have a coordinate system that is attached to each brush. The direction of velocity vector

$$n_0 = \frac{p(n+1) - p(n)}{|p(n+1) - p(n)|}$$

gives us the direction of one of the axes of this coordinate system. The gradient vector, ∇f, is enough to get the other axis directions:

$$n_1 = \frac{\nabla f \times n_0}{|\nabla f \times n_0|}$$

$$n_2 = n_1 \times n_0.$$

Examples of attachment of different paint objects to the brushes are shown in Figure 1.

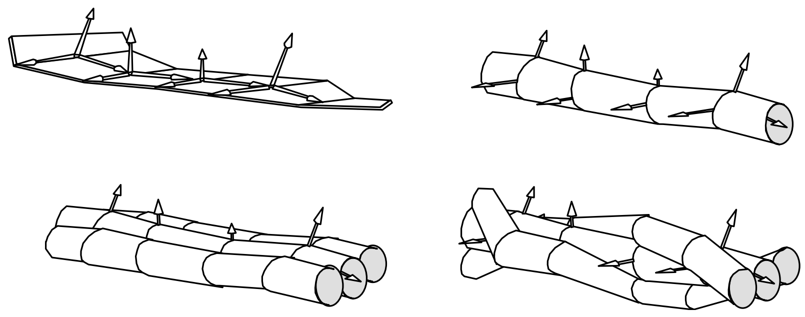

Figure 1. Examples of attachment of paint objects to the brushes.

Implicit surface painting equations

There are two characteristics of implicit representations that make them extremely suitable for development of difference equations for painting.

- ∇f is a vector that gives a normal vector to the surfaces $\mathcal{S}(f)$. We can use this normal vector to move in 3D space to reach $\mathcal{S}(f)$. This idea is used by Bloomenthal [2] in order to sample implicit surface, $\mathcal{S}(f)$.
- Any vector perpendicular to ∇f will be on the surface of $\mathcal{S}(f)$. By using this vector, it is possible to make the particles float over surfaces. Witkin and Heckbert [7] used this fact in order to achieve a good sampling of the surface. By including local repulsion between the *floating* particles; and by letting particles be born and die, they were able to spread the particles evenly over the surface.

In order to develop an implicit painting equations which will give control to the painter, we revised Witkin and Heckbert's differential equations. Since the motion of one brush does not have to be dependent on the positions and motions of other brushes, the new equations do not have to have all the terms of Witkin and

Heckbert's general equations. We exclude the local repulsion term from these equations. This exclusion provides two advantages for painting. First, computation of brush positions greatly simplifies. Second, with local repulsion the particles will stop moving when a good sampling of the surface has been achieved; that is not desirable for painting, in which the brushes have to continue to move unless painters want them to stop.

Let p denote the position of a particle, let ∇f be the derivative of F with respect to a vector and let r denote the desired value of $\dot{p}$, By using Witkin and Heckbert's approach we obtain following equation which is simpler than Witkin and Heckbert's differential equations since we do not have to change shape parameters, derivative of the shape parameters according to time.

$$\dot{p} = -a_1 v_1(f, b) + a_2 v_2(f, r),$$

where

$$v_1(f) = \frac{f\nabla f}{\nabla f \bullet \nabla f},$$

$$v_2(f, r) = r - \frac{\nabla f \bullet r}{\nabla f \bullet \nabla f}\nabla f.$$

The two terms of this equation are fundamentally different. The first term, $-a_1 v_1(f)$, is the feedback term to make the particle attract and stay on the surface $\mathcal{S}(f)$. The second term, $a_2 v_2(f, r)$, makes the particle move on the surface since it is the projection of desired vector r on to the surface $\mathcal{S}(f)$.

By using an Euler approximation, the desired difference equation $p(n+1) = p(n) + v_0(f, X)$ can be obtained with $v_0(f, b, r) = \delta(-a_1 v_1(f) + a_2 n_2(f, r))$, where δ is the speed constant. Since a_1, a_2 also control the speed, we can eliminate δ and choose

$$v_0(f, b, r) = -a_1 v_1(f) + a_2 n_2(f, r).$$

In this equation v_0 is the function of f, a_1, a_2 and r only . Thus, $X = \{a_1, a_2, \mathbf{r}\}$, where a_1 and a_2 control the speed, and r controls trajectory.

Speed control

The value of a_1 determines the speed at which particles approach the surface. The value of a_2 itself is not critical. However, the relationship between a_1 and a_2 is important. For a better sampling of the surface the value of a_1 should be greater than the value of a_2. If not, the particles move in a distant orbit around the surface, as illustrated in Figure 2. In implicit painting, better sampling is not essential. Sometimes we can choose $a_2 >> a_1$ to make the particles move in a distant orbit around the surface for more volumetric-looking results.

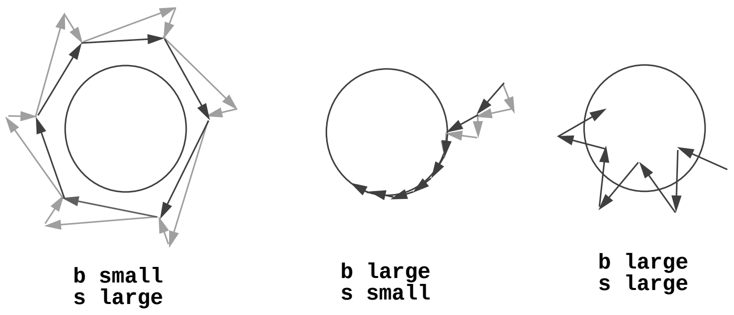

Figure 2. The effect of the relationship between a_1 and a_2.

Trajectory control

If the control vector r is constant, the trajectory equation will eventually move the particle to the points where $|\nabla f \times r| = 0$. We call these points attraction points. When a particle reaches any attraction point, it will stop moving, as shown in Figure 3.

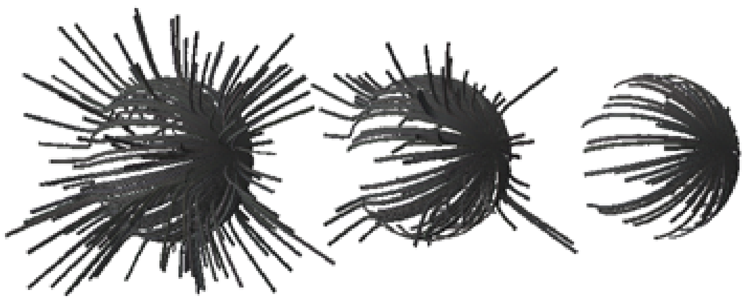

Figure 3. The particles moving towards an attraction point.

It is possible to make brushes perpetually move on the surface by simply replacing r with $\nabla f \times r$. As a result, the differential equation becomes:

$$\dot{p}_k = -a_1 v_1(f) + a_2 v_2(f, \nabla f \times r).$$

Under the influence of this differential equation the particles will perpetually rotate around attraction points. Attraction of the particles to a sphere, torus and double-torus is shown in Figure 4. In this case, if r is not changed, the brushes will perpetually draw the same curve as shown in in Figure 5.

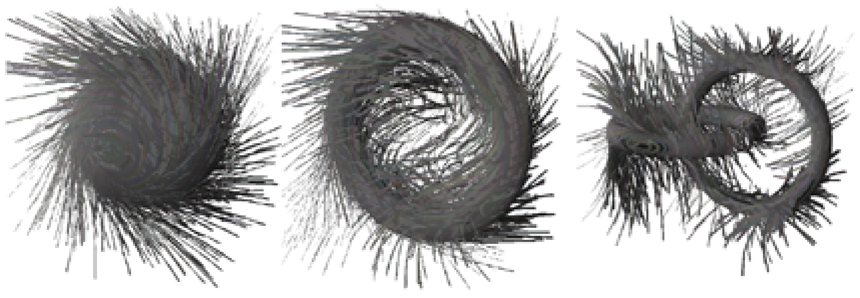

Figure 4. Trajectories of difference equations by using $v_2(f, \nabla f \times r)$.

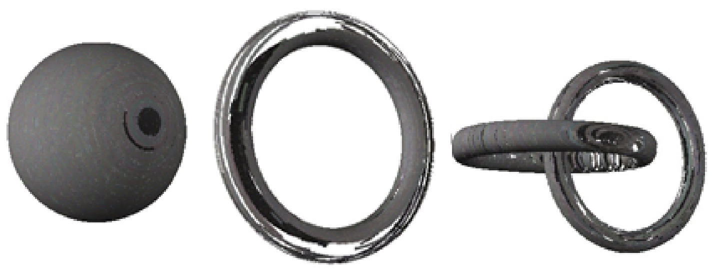

Figure 5. Perpetually rotating particles.

The quality of sampling can easily be improved by using different r vectors for each brush. Since brushes rotate around different attraction points, they cover the surface evenly as shown in Figure 6. Note that, in this example, a different equation is applied to each particle. These equations are randomly generated by creating a random vector r_k for each brush $k = 0, 1, \ldots, N-1$.

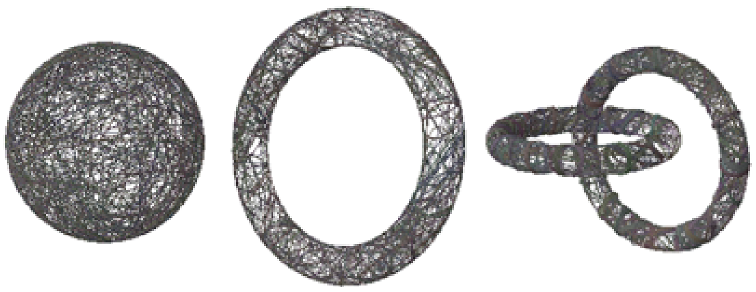

Figure 6. Effect of the use of a different r_k vector for each particle: trajectories of 200 particles for a short period of time. Note the uniformity of the distribution of the trajectories.

The quality of sampling can also be improved by changing the control vector r_k with time. The simplest change is the random walk by using the function

$$r_k(n+1) = \frac{r_k(n) + c\ n_{rnd}}{|r_k(n) + c\ n_{rnd}|},$$

where n_{rnd} is any randomly generated unit vector and c is a positive real-numbered coefficient. The value of c determines how quickly a particle changes direction in its random walk. Examples of trajectories of implicit painting equations for different c values are shown in Figure 7. The lower values of c do not change the positions of attraction points, therefore the brushes will rotate almost in a cycle. While c is increasing the particles will start to cover the surface more uniformly. However, an interesting phenomenon occurs for higher values of c. In this case, the motion becomes completely random. Because of this highly random motion, brushes stay in an area for longer time and leave most parts of the surface uncovered. We consider each of these as a different painting effect. Low c values create a look of careful painting. On the other hand, higher c value gives an impression of corroded material. Figure 7 shows how c affects the uniformity of trajectory distribution and related painting effects.

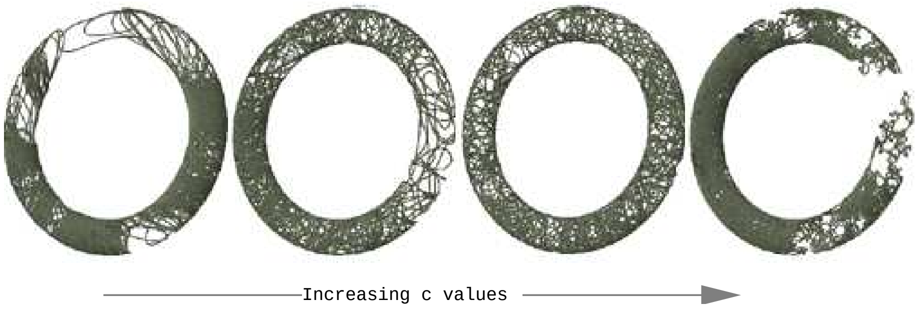

Figure 7. Trajectory of only one particle over a torus for changing c values.

Paint control

Additional paint effects can be achieved by changing the sizes, the material properties and the types of the primitive objects with time. The sizes of the primitive objects can be changed simply by an additional equation. Material properties require more involved equations. There is a need for four types of material property. These are vectors that include a diffuse colour and the shader coefficients such as those for specular reflection k_s, diffuse reflection k_d, ambient k_a and transmission k_t. The four properties are:

1. *Material property of simple solids.* The scenes are constructed with set-operations over simple solids. Each one of these simple solids will have its own material property. Let $\mathcal{S}(f_j)$ denote a simple solid j in a scene with J solids where $j = 0, \ldots, J-1$. Then $m_{\mathcal{S}(f_j)}$ will denote the material property of the solid j, defined by $\mathcal{S}(f_j)$. These material properties should be defined by the user and must be included in the scene description.

2. *Original material property of paints.* Each paint object l attached to brush k will have a material property denoted by $m_{k,l}$. These properties are arbitrarily chosen when the brushes are first created. They will be updated according to the position of the brush in space. In order to make this update, we describe material property for every point in space.

3. *Material property of a given point.* Since a brush can be anywhere in the space, for every point p in the space we need to provide a material property. Material property of a point, m_p, will be computed as a function of material properties of simple solids.

4. *Material property of the paint in a given point.* This is the material property of paint l when its brush k is in the position p. It will be denoted by $m_{k,l,p}$. A linear interpolation will be used to compute its value

$$m_{k,l,p} = (1 - k_{t,k,l})m_{k,l} + k_{t,k,l}m_p,$$

   where $k_{t,k,l}$ is the transmission coefficient of the paint object l of particle k. The colour of the paints is computed by using $m_{k,l,p}$. The gradient vector at the point p_k is used as a normal vector in the computation of colour.

Although the original material properties of the paints are different, when they pass thorough the neighbourhood of one point, they all share some amount of the material property of that point. The layers of paints which share the some amount of the same material property create a mixture that gives approximately the right colour around that given point.

Painting algorithm for CSG solids

It is possible to use extremely simple difference equations and algorithms if the scenes to be painted are constructed by set-operations. One of the important properties of implicit solids is that they can easily be constructed by functional operations, as shown by Rvachev and Ricci [6, 5]. Let two solid shapes be given by the inequalities

$$\mathcal{V}(f_1) = \{p | f_1 \leq 0\},$$
$$\mathcal{V}(f_2) = \{p | f_2 \leq 0\}.$$

Then, by using maximum and minimum operators, we can obtain set-operations:

$$\begin{array}{rcl} \mathcal{V}(f_1) \cap \mathcal{V}(f_2) & = & \mathcal{V}(max(f_1, f_2)), \\ \mathcal{V}(f_1) \cup \mathcal{V}(f_2) & = & \mathcal{V}(min(f_1, f_2)), \\ \mathcal{V}(f_1) - \mathcal{V}(f_2) & = & \mathcal{V}(min(f_1, -f_2)). \end{array}$$

These maximum and minimum operators directly provide algorithms for painting solids that are obtained by set-operations. For example, suppose a solid is constructed by an intersection operation $\mathcal{V}(max(f_1, f_2))$. Then the algorithm for painting will be as follows:

- If $f_1 \geq f_2$ then apply the implicit painting equations for $\mathcal{S}(f_1)$ and choose $m_{\mathcal{S}(f_1)}$ as the material property of point p,
- Otherwise, apply the implicit painting equations for $\mathcal{S}(f_2)$ and choose $m_{\mathcal{S}(f_2)}$ as the material property of point p.

For union and set-difference, similar algorithms are used. Complicated scenes can be created by using very simple solids such as half-spaces, ellipsoids, cones, cylinders, and toroids. Since all of these solids can be described low-degree equations, the related difference equations for painting also become extremely simple. Examples of some simple painted surfaces generated using the algorithm described in the paper are shown in Figure 8.

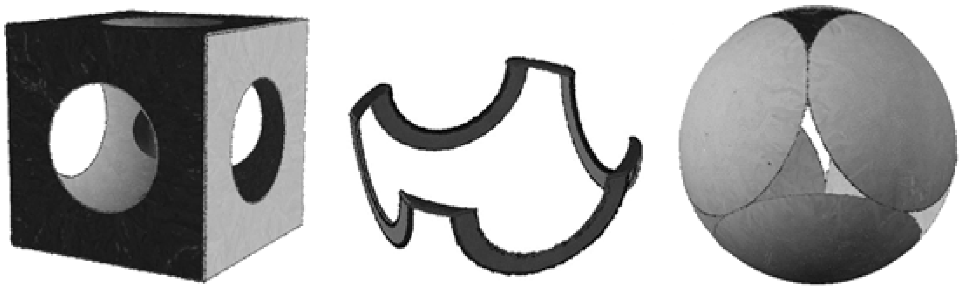

Figure 8. Some simple objects constructed by set-operations and painted by the implicit surface painter.

Figure 8 also illustrates an additional effect that can be obtained by implicit painting. When paints move around sharp edges that result from set-operations, they create double edges as shown in Figure 9. Since the motion of the brushes random, this double edge has a random quality. This randomness along with double edge creates an effect resembling hand-drawn lines.

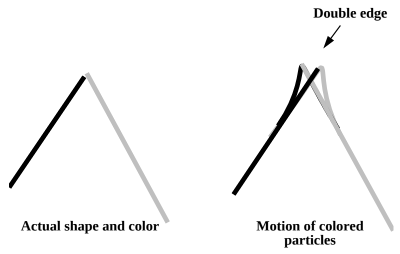

Figure 9. The double edge resulting from brush motion.

Examples and discussion

By using set-operations over simple objects, it is possible to construct interesting scenes. We constructed the two scenes that are illustrated in Figure 10, and painted these scenes by using the implicit surface painter. The results are shown in Figures 11 and 12. In order to obtain different painting styles we played with the values of a_2 and c. In addition we changed the sizes and types of paints. We also changed viewing positions. Finishing one painting depend on the chosen values. As shown earlier, small a_2 value provide a better approximation of the surface but it takes longer time to cover the whole surface. Likewise, choice of c values effect the time to cover the surface. In addition, the size of paint objects effects the coverage time. Bigger objects cover the surface faster. It is always possible to play with a_2 and c values to make coverage faster.

Another factor that effect the time is the complication of the scene. For instance, in the house scene there is a huge hollow sphere that represents the sky. Although only a small portion of this sky can be seen by the viewer, brushes try to cover all of it. If there are many objects outside the viewing area, the time spent in rendering unseen objects can be considerable. However, it is always possible to force the brushes stay in a certain portion of the space.

In general, by using an SGI O2 workstation, it takes around one minute to create an image which we find satisfactory.

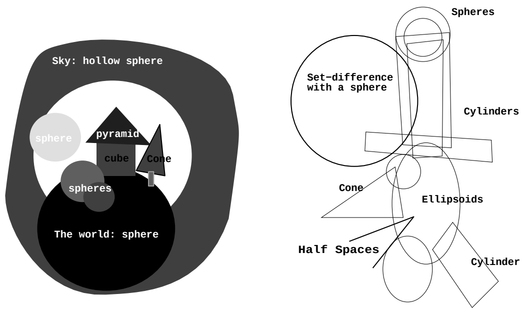

Figure 10. Descriptions of scenes that are painted.

Recently, we made experiments with different functions of $r_k(t)$ that require normalization of vector v_2. By using these functions we were able to provide additional controls over the brush motion. These new functions of $r_k(t)$ give good results on smooth surfaces such as spheres or toroids. However, the random walk still works better for CSG solids. We have not yet investigated of space-filling curves which can be useful in implicit painting of CSG solids.

Acknowledgments

This work is supported by *Texas A&M University Program to Enhance Scholarly and Creative Activities.* We are also grateful to Don House, John Woodwark, John Ferguson and the referees for their helpful suggestions.

References

[1] J. I. Blinn, "A Generalization of algebraic surface drawing", *ACM Transactions on Graphics,* 1, 3, (235-256), 1982.

[2] J. Bloomenthal, "Techniques for implicit modeling", *ACM SIGGRAPH Tutorial 23: Modeling and Animating with Implicit Surfaces,* eds. B. Wyvill and J. Bloomenthal, (13.1-13.18), 1990.

[3] J. Dorsey and P. Hanrahan, "Modeling and rendering of metallic patinas", *Computer Graphics,* 30, 3, (387-396), 1996.

[4] B. J. Meier, “Painterly rendering for animation”, *Computer Graphics,* 30, 3, (477-486), 1996.

[5] A. Pasko, V. Adzhiev, A. Sourin and V. Savchenko, “Function representation in geometric modeling: Concepts, Implementations and Applications”, *Visual Computer,* 11, (429-446), 1995.

[6] A. Ricci, “A constructive geometry for computer graphics”, *The Computer Journal,* 16, 2, (157-160), May 1973.

[7] A. P. Witkin and P. S. Heckbert, “Using particles to sample and control implicit surfaces”, *Computer Graphics,* 27, 3, (269-277), 1994.

[8] G. Wyvill and A. Trotman, “Ray tracing soft objects”, *ACM SIGGRAPH Tutorial 23: Modeling and Animating with Implicit Surfaces,* eds. B. Wyvill and J. Bloomenthal, (13.1-13.18), 1990.

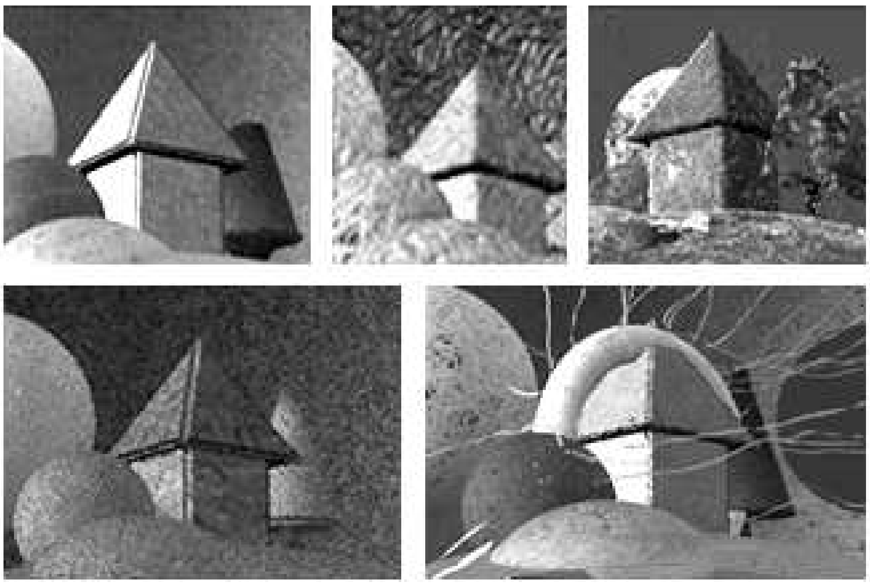

Figure 11. Different implicit paintings of a house scene.

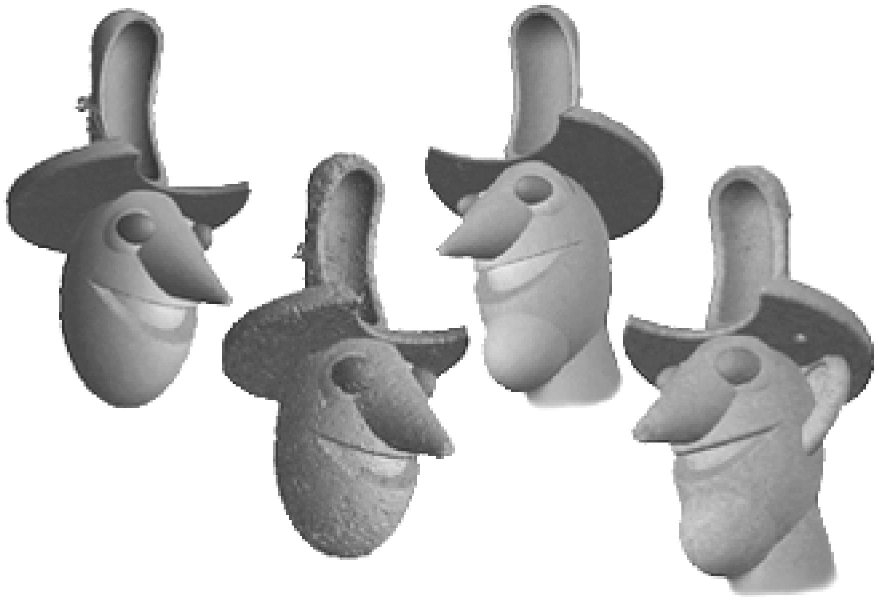

Figure 11. Implicit paintings of cowboys.