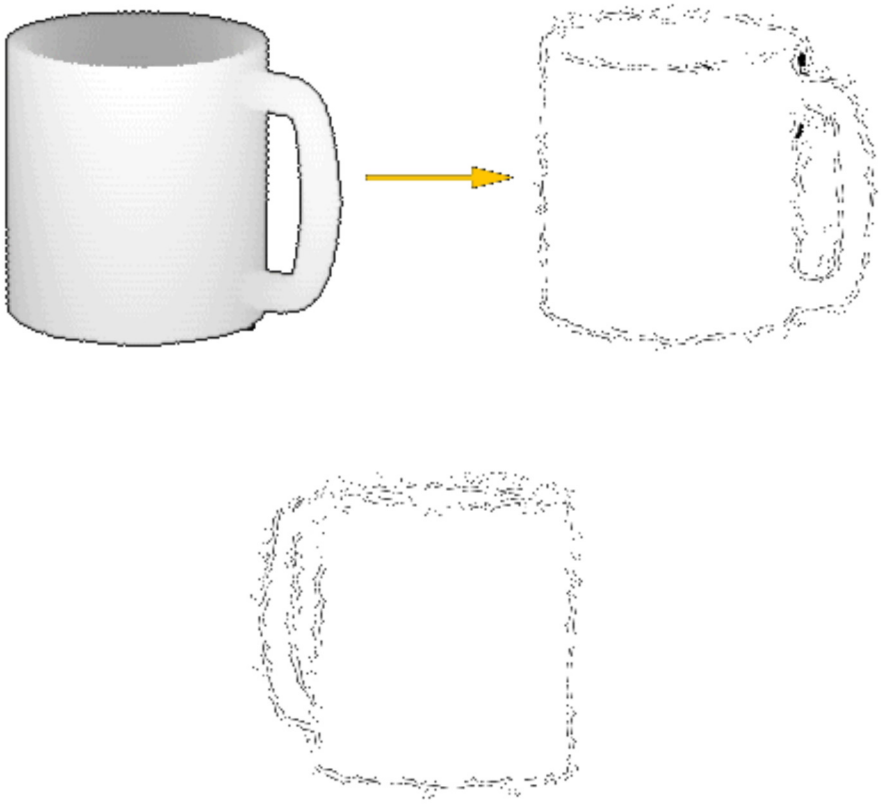
Loose and Sketchy Rendering: Who, What, Why, and How

What is Loose and Sketchy Rendering?

Loose and sketchy rendering was originally developed by Cassidy Curtis of the University of Washington. His technique takes as input the depth map of a 3D modeled scene and creates an image that looks like a sketchy line drawing of the silhouette edges of the scene. When applied to a sequence of images the resulting animation looks like a 3D gesture drawing brought to life.



Why Use Loose and Sketchy Rendering?

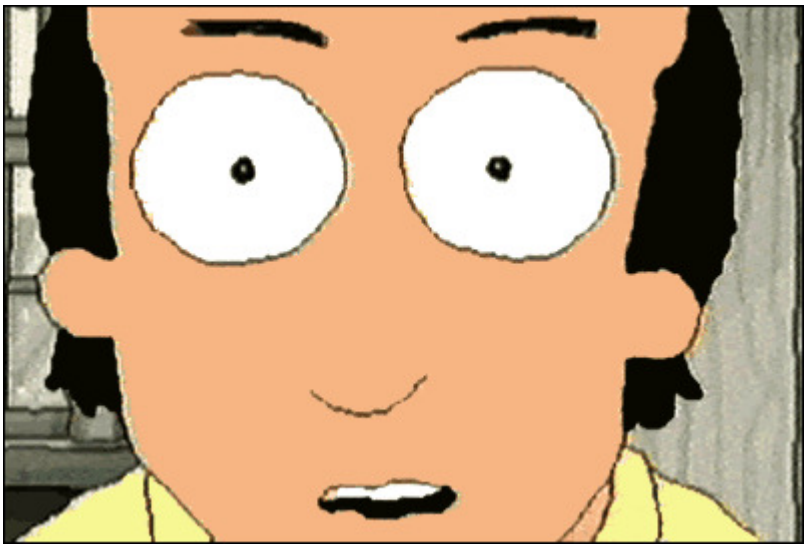
The loose and sketchy rendering technique not only has the advantages that non-photorealistic techniques have over photorealism, but also has many advantages over other non-photorealistic rendering styles.

- Like many other non-photorealistic rendering techniques, loose and sketchy rendering does not require very detailed modeling.
- Unlike several other non-photorealistic techniques, the 3D modeled scenes used for loose and sketchy rendering require no lighting or texturing, since only the depth information of the associated geometry is used as input.
- Also, compared to other computer generated drawing styles, the sketchy, gestural quality of the image created using this technique is considered by some to be the most expressive.
- Unlike many non-photorealistic rendering procedures, loose and sketchy rendering accepts user input to alter the style of the sketch. This gives the user more control over the process.

- Another disadvantage of most non-photorealistic renderers is that when animated, the sequence of frames exhibits the "shower door" effect. This means an animation looks as if it were being seen through textured glass because the brush strokes are attached to the viewplane and not to the animated elements of the scene. The strokes of the loose and sketchy technique, however, are randomly placed and change in every frame. Thus they do not appear to be stuck to the viewplane, but rather seem "alive" as they wiggle and crawl around the scene elements.

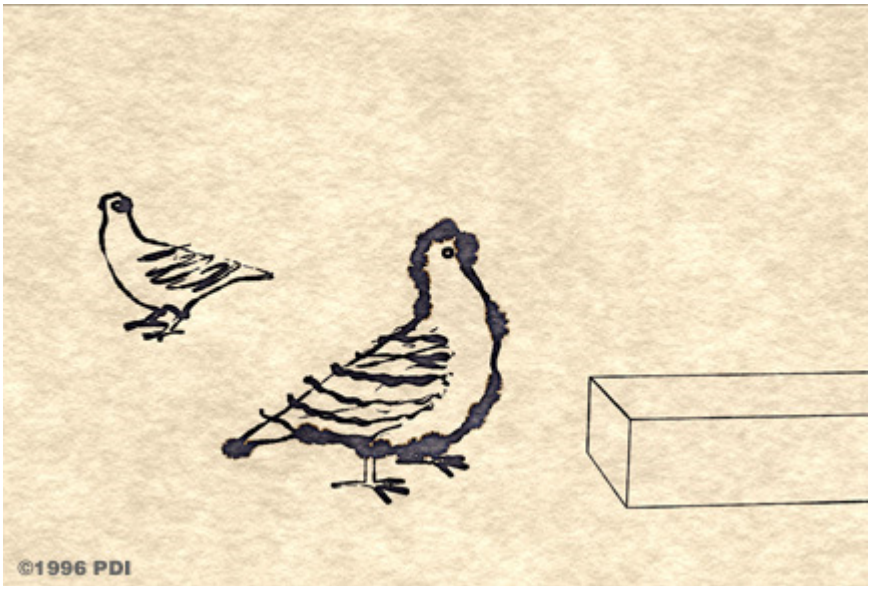
Who Uses Loose and Sketchy Rendering?

Loose and sketchy rendering is similar to a commercially successful 2D animation technique. Tom Snyder Productions has created three television shows, Dr. Katz: Professional Therapist on Comedy Central, Science Court on ABC, and Home Movies on UPN, which are animated using outlines that could easily be described as "loose and sketchy". Using a technique called "Squiggle-Vision", the outlines of the 2D animated elements randomly change from frame to frame, exhibiting the same look and behavior as loose and sketchy rendering. Noting the current success of Squiggle-Vision, development of this technique for the 3D realm has promise.



"Dr. Katz"
(c)Tom Snyder Productions

Cassidy Curtis's earliest attempt at loose and sketchy rendering can be seen in an animation produced at Pacific Data Images, entitled "Bric-a-Brac", which has been screened world-wide. Currently he is working on a new animation "The New Chair", after much development of the loose and sketchy technique to make it more flexible.

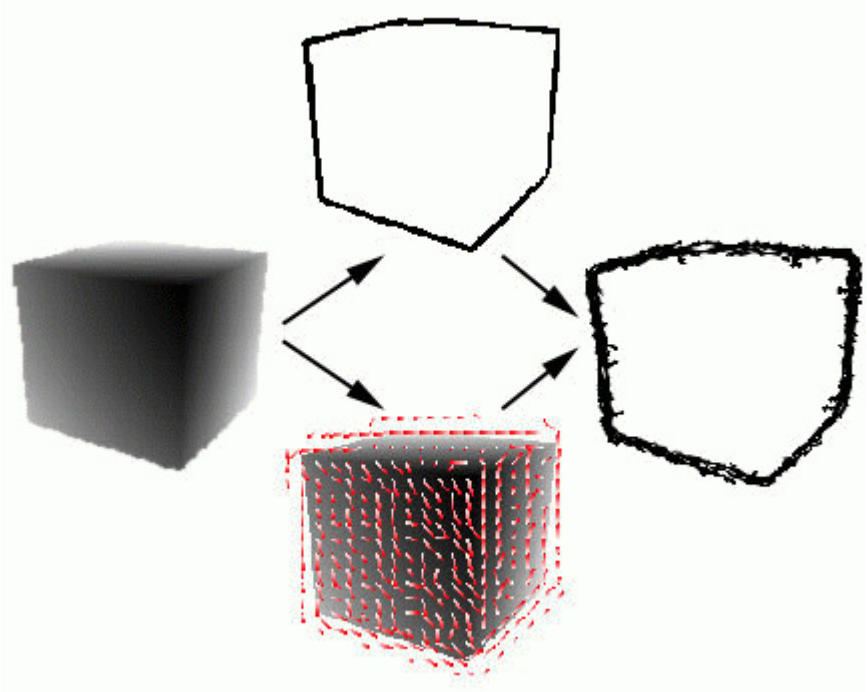


"Bric-A-Brac"
(c)PDI

"The New Chair"
(c)Cassidy Curtis

How is Loose and Sketchy Rendering Done?

Using image processing and a stochastic, physically-based particle system, the loose and sketchy rendering algorithm as developed by Cassidy Curtis draws the visible silhouette edges of a 3D model. It performs this operation using a depth map of the model and a few user-determined parameters.



First the depth map is converted into two images:

- The first image is an edge map, in which the edges of the objects in the image are drawn with solid, continuous lines. The edge map is created by blurring the depth map, calculating the magnitude of the gradient of the depth map, and thresholding to give binary values.
- The second image is the representation of the force field. There are two components of the forcefield.
  - The first component of the forcefield is found by rotating the gradient vectors of the blurred depth map by 90 degrees.
  - The second component of the forcefield is found by taking the magnitude of the gradient of the depth map, storing it as an image, and blurring it. Then the gradient at each pixel of this image is found and rotated by 90 degrees to create another set of forcefield vectors.

  The combined forcefield vector at each pixel is found by subtracting the second component from the first.

Next, to create the loose and sketchy image, a fixed number of particles is generated. Taken one at a time, the initial position of each particle is randomly chosen within the viewplane. There are two tests to see if this is a valid starting position.

- First, the particle must be in an area that needs ink. This is an area that corresponds to an edge of the edge map.
- Second, a bias is given to areas that need more ink. Therefore, the particle should not be positioned in an area in which the strokes of the output image are already dense.

If the particle lands in a valid position, its path is used to draw the loose and sketchy strokes.

Using physically-based modeling techniques, at each timestep the particle's acceleration is affected by the forcefield. The first component of the forcefield serves to push the particles along the edges of the edge

map. The second component forces the particle back onto the edge if it starts to stray.

The acceleration has coefficients for randomness and drag, which affect the type of path the particle will follow. As drag increases, the output image will more closely resemble the edge map. As the randomness increases, the strokes will seem livelier.

As the particle moves, it draws antialiased line segments in the output image, while simultaneously erasing the corresponding edges from the edge map. This keeps edges from being traced multiple times.

If the path of the particle leads to an area that does not need ink, the particle dies and a new one is born.

Pseudocode

```
read in depth map

perform edge detection
    blur the depth map
        for each pixel in the depth map, convolve with a 3x3 tent filter
                1 2 1
                2 4 2
                1 2 1
    find the gradient vectors (gx,gy)
        for each pixel in the depth map, convolve with Sobel filters
            to get gx convolve the depth map with
            -1 -2 -1
            0  0  0
            1  2  1
            to get gy convolve the depth map with
            1 0 -1
            2 0 -2
            1 0 -1
    find the magnitudes of the gradients
        for each pixel's gradient vector
            mag = sqrt(gx*gx + gy*gy)
    threshold to give a binary value
        if (mag < threshold) pixel color = black
        else pixel color = white

create the forcefield vectors
    find the first component of the forcefield
        for each pixel, rotate the gradient vectors by 90 degrees
            ff1x = -1.0 * gy
            ff1y =  1.0 * gx
    find the second component of the forcefield
        store the magnitude of the gradient as an image
            for each pixel
                if (mag > 255) mag = 255
                r = g = b = mag
        blur the magnitude of the gradient image
            convolve each pixel with a 9x9 tent filter
        find the gradient vectors (gx,gy) of this image
```

use same method as on the depth image
for each pixel, rotate the gradient vectors by 90 degrees
ff2x = -1.0 * gy
ff2y = 1.0 * gx
combine the two components to create the forcefield vectors
for each pixel
ffx = ff1x - ff2x
ffy = ff1y - ff2y

begin sketching
for every stroke to be drawn
start the particle
generate a particle at a random position in the loose and sketchy image
test to see if it is a valid position
the particle should be within n pixels of an edge in the edge map
the particle should *not* be within m pixels of a line in the loose and sketchy image
if not valid, keep trying different random positions
using physically based techniques, move the particle
runga-kutta four suggested
acceleration_x = ffx/mass - drag*velocity_x/mass + random_vector_x/mass;
acceleration_y = ffy/mass - drag*velocity_y/mass + random_vector_y/mass;
update the images
draw a line in the loose and sketchy image from the particle's initial position to its new position
erase a line from the edge map from the particle's initial position to its new position
continue moving the particle, drawing, and erasing until the particle's position is no longer valid
the position is not valid if the particle is no longer near an edge in the edge map or if the particle is no longer within the borders of the image

[Loose & Sketchy Rendering] [Disscussion] [User's Manual] [Examples & Tutorial] [Downloads]

[Photorealism vs. Non-Photorealism] [Who, What, Why, & How] [Advancements Made]