Brush Strokes for Video: Just as Boring as Brush Strokes for Still Images

Michael Hines
mhines3@illinois.edu
April 21, 2009

Hertzman extends on his previous work of painterly rendering of still images by extending it to video, this time with help from Perlin. Not much new this time around. I think I feel slightly duped with this paper, because I feel like the only thing that changed was a shiny new format for the pseudo-code. Let’s compare:

```
procedure paintLayer(canvas, referenceImage, R)
    S := a new set of strokes, initially empty
    // create a pointwise difference image
    D := difference(canvas, referenceImage)
    grid := \mathbb{R}
    for \(x = 0\) to imageWidth steps size grid do
        for \(y = 0\) to imageHeight steps size grid do
            // sum the error near \((x, y)\)
            M := the region \([x - \text{grid}/2, x + \text{grid}/2, y - \text{grid}/2, y + \text{grid}/2]\)
            areaError := \(\sum_{(i, j) \in M} D_{i,j} / \text{grid}^2\)
            if areaError > \(T\) then
                // find the largest error point
                \((x, y) := \text{arg max}_{(i, j) \in M} D_{i,j}\)
                \(s := \text{makeStroke}(\mathbb{R}, x, y, \text{referenceImage})\)
                add \(s\) to \(S\)
            end if
        end for
    end for
    paint all strokes in \(S\) on the canvas, in random order
```

The only interesting thing is the optical flow idea that’s mentioned but not explained. The concept is almost self explanatory. If points move, the flow of the area of points is the average deformation of the rectangle determined by the motion of points in that area. How this is computed is the interesting part. The only thing we get is this: \(F(x, y, t) = (x' - x, y' - y)\). Which is just plain confusing. What is \(t\), and why isn’t it part of the equation? Additionally, there’s

```
function paint(L, // source image
    I_s, // canvas, initially blank for still images
    \(R_1, R_2, // brush sizes
    \text{firstFrame}) // boolean true for still images

Create a summed-area table \(A\) from \(I_s\)

\text{refresh} := \text{firstFrame}

\text{foreach} brush size \(R_i\), from largest to smallest, do

Use \(A\) to compute a blurred reference image \(I_{R_i}\)

grid := \(f \cdot R_i\)

Clear depth buffer

\text{foreach} position \((x, y)\) on a grid with spacing grid

\(M := \text{the region} [x - \text{grid}/2, x + \text{grid}/2, y - \text{grid}/2, y + \text{grid}/2]\)

areaError := \(\sum_{(i, j) \in M} |I_{R_i}(i, j) - I_{R_2}(i, j)|\)

if \text{refresh} or areaError > \(T\) then

\((x, y) := \text{arg max}_{(i, j) \in M} |I_{R_i}(i, j) - I_{R_2}(i, j)|\)

\text{paintStroke}(x, y, I_s, \text{referenceImage})

\text{refresh} := \text{false}
```

Wait, wait. I see the difference. The second one says “refresh or”. Glad we cleared that up.

Use this at your own risk. This is not how you write code.

1
no way to know in captured or post-rendered video which pixels correspond to
which points, so it must be inferred somehow. That’s the interesting part, but
it’s just not there.

The good news is that this research was done under the same grant as his
previous work. So there was no additionally wasted money on a continued
research grant.

References

You Aren’t Gonna Need It

Michael Hines
mhines3@illinois.edu
April 21, 2009

As part of the research for her dissertation, Karahalios was faced with the challenge of creating an anonymous, but compellingly interactive remote video environment. The project benefited from its strict scope. I often find that the most well-done solutions to a problem are the ones that are themselves solutions to a bigger problem. When people try to solve generic problems, like how to make a photo look like a painting using curved brush strokes, they often end up creating something that has no use.

On the other hand people solving larger problems seem to have the smaller problems very well-defined, this often helps keep the focus on the question “what do I need to do?” rather than “what else can I throw in?” My friend pointed out to me the other day a concept known as YagNi in the Extreme Programming paradigm. YagNi is an acronym for “You aren’t gonna need it.” When researchers implement things without a specific purpose, it tends to just clutter the work up with features that people just plain don’t use.

Telemural was one of the well-defined projects. It had three main requirements:

1. Both remote and local spaces had to be distinctly recognizable on a single screen
2. The space had to grow as users interacted with it
3. The space had to preserve users’ anonymity

Each feature of Telemural fit one of these requirements, and the result was elegantly simple. Users who interacted with the screen more, got to see more of themselves appear.

I think the interface of Telemural would also work well as a persistent graffiti wall. If users interacting with the wall could leave their silhouettes on the wall instead of having the images disappear with every frame, there might be potential for Telemural to also tell a story of a social space.

References

The Interactive Artistic Rendering is a paper about using geograftals to render textures onto images. As far as I can tell geograftal is a term invented in this paper. I looked up what a graftal is, and found out that it’s a grammar based representation of a fractal. Well that’s not what I wanted. My understanding is that a geograftal is just a 2d image that is attached at a point onto 3d image and rotated to face the camera.

These geograftals are duplicated all over a 3d model and then rendered with specific properties different from the surface properties. Then the resulting composite model is rendered. If it sounds like the model is simply being textured, I think that’s probably because that’s exactly what’s going on.

I don’t know what else to say about this piece, it doesn’t really seem new or different at all.

References