Technical Report on Live Paint:
Painting with Procedural Multiscale Textures

Tarun Jain
Student
Master of Technology
International Institute of Information Technology
Bengaluru, Karnataka 560100
Email: Tarun.Jain@iiitb.org

***Abstract*—Live Paint [1] focuses on actively procedural multiresolution paint textures. Here I explain the mechanism explained in Live Paint multiresolution system for generating images with optimal details at varied resolution levels. Procedural Textures can scale up to infinite depth and can be used to generate greater details on magnification of an image. Thus procedural textures can be used to build an effective tool that will help sharpen images at higher resolution levels. I explain basic elements of the system and then put them altogether to build entire multiresolution paint system, specifying examples wherever needed.**

***Keywords*—Multiresolution Paint System, Level, Lazy Evaluation, Procedural Textures, Procedural Bandpass Pyramids, Procedural Ink.**

I. Introduction

The present day paint tools generally operate at fixed resolution. This restrict the paint tools to generate a fixed amount of details. Even if the image is represented in multiresolution layers, it contains information in finite depth. Thus, on magnifying the image after a certain level, the image becomes blurry.

Multiresolution images could serve the purpose of giving a fine detailed image at any resolution, if tool have the capability to exploit underlying multiresolution image representation. Live Paint [1] gives a framework for the design and implementation of multiresolution painting tools. This framework is based on general procedural textures. In this system, user can view and magnify an image at any desired resolution.

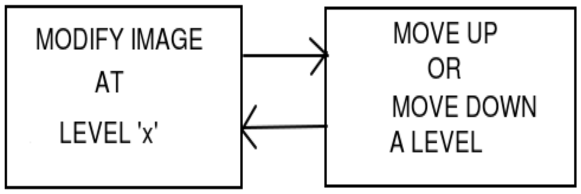
The structure of the paper is as follows: Section 2 will explain all the basic building blocks needed to understand the multiresolution paint system for Live Paint and reviews the previously done work utilised to generate the design of the tool;; Section 3 combines all the basic blocks and previously done work to construct a design of multiresolution paint system; Section 4 discuss the related works and examples; Section 5 concludes with final remarks and a discussion of future work.

II. Basic Building Blocks or Background Work

Before diving deep into the design and implementation of the multiresolution tool system, it will be better to get familiar with the terminologies to represent an image in the system, how to generate textures, what changes should be made when user paints, what illusions will be given to the user to save time of executing again and again, last but definitely not the least how to generate infinite information and represent our image in finite space in memory.

- Terminologies
    1) Multiresolution Image Representations
        - ○ Lowpass Pyramid
        - ○ Bandpass Pyramid
    2) Image Operations - Multiresolution Painting and Compositing
        - ○ Re-execution
        - ○ Lazy Evaluation
    3) Multiresolution Textures
        - ○ Texture Instance
        - ○ Procedural Bandpass Pyramid
        - ○ Procedural Ink

Painting Process is a cyclic process defined as:



Lowpass Pyramid: A Lowpass pyramid consists of multiple copies of the image at different resolution, [2]. It contains a redundant representation since the same information is present at all levels of the pyramid.

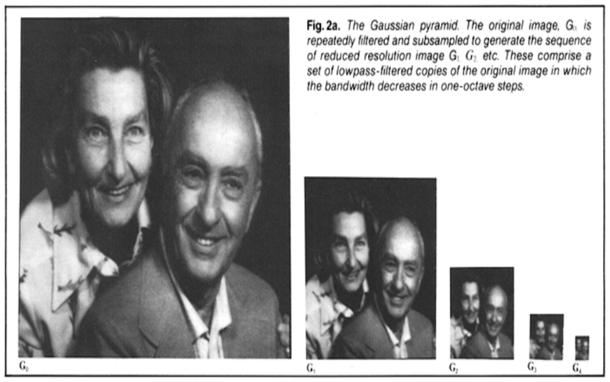


Fig. 2a. *The Gaussian pyramid. The original image, G_0 is repeatedly filtered and subsampled to generate the sequence of reduced resolution image G_1, G_2 etc. These comprise a set of lowpass-filtered copies of the original image in which the bandwidth decreases in one-octave steps.*

Figure 1: Lowpass Pyramid

Bandpass Pyramid: A Bandpass pyramid consists of a coarse resolution version of the image, together with a sequence of image details that are required to produce finer resolution versions of the image from coarse versions, [3]. The Bandpass Pyramid can be constructed by taking differences between consecutive levels of the lowpass pyramid.

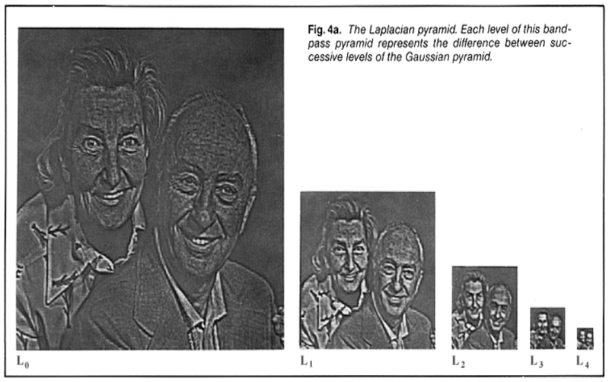


Fig. 4a. The Laplacian pyramid. Each level of this bandpass pyramid represents the difference between successive levels of the Gaussian pyramid.
L0
L1
L2
L3
L4

Figure 2: Bandpass Pyramid

Suppose a bandpass pyramid is composed of following resolution $2^nX2^n, 2^{n-1}X2^{n-1}, ..., 2X2$. The pixels of 2^kX2^k image of a bandpass pyramid are called "level-k bandpass coefficients". To reconstruct an image at level 'm', successively add bandpass coefficints starting from 2X2 to 2^mX2^m level. This gives a finite depth.

Image Operations: Multiresolution system has to deal with many levels. It is not possible to make changes to all the levels wherever there is any change made at level 'x'. Thus, paint tool gives a illusion to the user by updating changes at currently visible level. Changes to the image are cached and propagated to other resolution levels later.

Re-Execution: In this strategy, when user make changes the entire level is re-drawn whenever the level is visited by user on magnifying.

Lazy Evaluation: Modifications to the image are iteratively propagated through the image pyramid when the user magnifies the view of successively more detailed levels. Changes are updated when user first magnifies the view.

Two Types of Multiresolution Paint Systems:

1) Lowpass pyramid + Re-execution strategy
2) Bandpass pyramid + Lazy evaluation strategy

Procedural Bandpass Pyramids: It can be of infinite depth unlike image's bandpass pyramid which could be of finite depth. We can define a bandpass pyramid for various types of textures by given type, values for x ,y and level 'l', which returns the difference in a textures's appearance when viewed at successive magnifications. Example of Bandpass pyramid for 'rock' texture is defined as:

$$add\_texture(type, x, y, l) = \frac{pseudorandom(x, y, l)}{2^{l/2}}$$

where pseudorandom function is implemented by the same permutaion method used to choose pseudorandom gradients from Z^3 for the Noise Function [4]. Procedural bandpass filter generate noise by above formula, thus it can give sharper image for any magnification level. *Problem:* If user paints a texture then zooms in and out a few times, we add bandpass coefficients to the texture again and again which finally results into incorrect image because of the addition of random noise. Texture propagation to other level is controlled by 'procedural ink'.

Texture Instance: Texture Instances refers to a primitive texture component. For example: checkerboard of a certain size. All textures are build as a combination of texture instances.

Procedural Ink: Procedural Ink is texture in latent state. It is represented by vector of amplitudes; each element of the ink vector contains the information about one entire texture instance. The first ink channel (ink.aplha) is reserved for storing the attenuation of ink vector itself. All elements of the ink vector will be attenuated by this ink.aplha. Ink always flows downhill - towards levels of fine details.

III. Discussions

Multiresolution paint system adds texture in two different ways:

- Whenever the user paints a texture, the system must display that texture's initial appearance.
- As the user successively magnifies the view, the system must continue to add detail to the painted texture.

Texture procedure function performs the above operation with the help of two tools: Procedural Bandpass Pyramids and Procedural Ink. This tool mix the texture by layering the ink vectors. The resulting composite ink vector is then used to generate texture. Ink vector is used to scale the bandpass coefficients that must be added in order to sharpen one texture instance. Ink is only used for texture refinement at the moment that it first enters a view level. Texture created by the user is called at two points:

Texturing during Painting
When the user paints at a sample using drawing ink, and with opacity drawing alpha, then the color, alpha and ink at the sample are modified as follows:

$$color := texture(drawin\_ink)OVER_{drawing\_alpha}color$$

$$alpha := 1.0OVER_{drawing\_alpha}alpha$$

$$ink := drawing\_inkOVER_{drawing\_alpha}ink$$

where "$bOVER_ta$" denotes linear interpolation a + t(b-a). Note that this is the first time ink is injected into the system.

Texture refinement during Magnification

To propagate ink and color to new level, this tool does the following operations:

$$color := coarse + texture(new\_ink)$$

$$+(1 - new\_ink.alpha) * detail$$

$$ink := new\_inkOVER_{new\_ink.alpha}ink$$

- detail: the portion of a samples color at the more detailed level that was too finely detailed to be visible at the coarser level. This quantity is generated by the multiresolution paint system at the moment that magnification was reduced. If this is the first time that we have ever visited the more detailed level, then this quantity will be zero. In our implementation, this quantity is computed from B-spline wavelets.

- coarse: a magnified view of what is currently visible at the coarser level.
- new ink: a magnified view of the new ink from all coarser levels which now needs to flow down to this more detailed level.

Here the new level can be updated if we find the color and ink for this level. The color is found by the texture generated by new_ink, by the updated details multiplied by the opacity of the new_ink and the old texture of the level 'coarse'. Similarly new_ink can be generated as shown in above equation.

From Ink to Texture
Textures comes in differen types. Each type requires a different refinement method. **Thus, a procedural texture is built up by adding bandpass coefficients to each texture instance modulated by the ink vector, at successive levels of detail.** All textures instances are combined linearly. The texture procedure simply loops through the ink vector and sums the contribution from each instance i:

$$\sum_i amplitude_i * add\_texture(type_i, x, y, level - base\_level_i)$$

Note that this arrangement allows us to linearly combine different ink vectors.

IV. Related Works

Similar to the texture type 'rock' there can be procedural bandpass pyramids for various types such as White, Stripes, Sawtooth, Squares. The procedural bandpass pyramids for these are given in Live Paint [1], Section 5.3.

Instead of using textures for painting, we can use Images as Procedural Brushes. The data structure needed for implementing this requires atleast three things - size of the base image (n x m), number of levels of the bandpass pyramid, bandpass pyramid. Thus, textures can be synthesized from Image Samples.

V. Conclusions

In this paper, I have explained the LIVE PAINT [1], the concept of actively procedural multiresolution paint textures. These are live picture elements that continue to refine themselves when viewed at magnifications greater than the one at which they were originally painted.

References

[1] Ken Perlin, Luiz Velho. Live Paint: Painting with Procedural Multiscale Textures.

[2] Lance Williams, Pyramidal parametics. *Computer Graphics (SIGGRAPH '83 Proceedings)*, 17(3):1-11, July 1983.

[3] Peter J. Burt. The laplacian pyramid as a compact image code.*IEEE Transactions on Communications*, 31:532540, April 1983.

[4] Ken Perlin. An image synthesizer. *Computer Graphics (SIG- GRAPH 85 Proceedings)*, 19(3):287293, 1985.