Painterly Rendering

Painting Strategies for Strokes

Technical Report: CS-2008-09
Lesley Northam
University of Waterloo

This page describes my implementation of Hertzmanns' painterly rendering algorithm. It also shows my results, and my personal additions to the algorithm. akita

Implementation Details

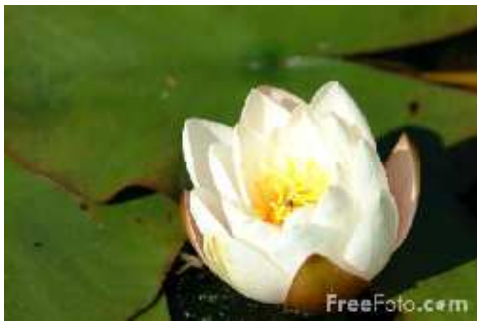
To implement this algorithm, I will make small improvements to a basic painterly rendering algorithm. This progression of algorithms will help illustrate the gradual improvement to the result image.

The implementation language is C++. To aid in image processing, Magick++ (C++ version of ImageMagick) is used. In addition, "algebra.cpp" and "algebra.h" from CS488 are used for the Point2D object.

Blurring
Blurring is done using the Magick++ gaussian blur function. The kernel size is chosen to be 3, to match the Sobel kernel used later.

Gradient
To compute the gradient, Hertzmann suggests using the "Sobel filtered luminance" of the blurred image. A Sobel kernel of width 3 is used after the blurred image is converted to luminance values (using the formula from the paper).

Proof of Concept
The images listed in the eight steps below are "proof of concept". The source image, and the settings used may not produce the best possible results. The source image used for all proof of concept images is:



User Interface
The user interface is shell based, and is not overly user-friendly. To run the program:

paint [options]
[options] = infile outfile minlength maxlength brushfile blur tol grid curve order taper? black? jitter? jitter% maxjitter

- *infile: input image file (can be of any format that ImageMagick supports, JPEG's were used for the purposes of this assignment*
- *outfile: output image file (will be output in JPEG format)*
- *minlength: minimum length of brush stroke in pixels*
- *maxlength: maximum length of brush stroke in pixels*
- *blur: blur factor*
- *tol: tolerance between original image and painted image*
- *grid: grid size*
- *curve: curvature filter for exagerating brush strokes*
- *order: order of painting strokes, 0 = random, 1 = light-to-dark, 2 = dark-to-light*
- *taper?: taper the end of strokes, 0 = no, 1 = yes*
- *black?: black shift dark colours, 0 = no, 1 = yes*
- *jitter?: jitter the colours, 0 = no, 1 = yes*

- *jitter%: percent of pixels to jitter, any integer value between 0 and 100*
- *maxjitter: maximum level of jitter, any positive decimal value*

1. Random Dots

This first step is a very simple approach. Random points are chosen from the blurred source image and a circle(dot) is drawn on the canvas in the corresponding colour at that point (Magick++'s DrawableCircle is used). To give variation in brush choices, this process is repeated for decreasing brush sizes. Brush sizes are input to the system using a ".brs" file, whose structure is defined by:

- The first line of the .brs file is the number of brushes listed.
- Each line after the first contains a single brush size.
- It is assumed that brushes are listed in decreasing order.

Uniform Dot Size

Three Dot Sizes

Three Dot Sizes but Fewer Dots

2. Random Linear Strokes

Two random points are chosen, the second within a given random distance from the first and a line (stroke) is drawn between them. Then as a second variation, a stroke is only drawn if the difference between the canvas and blurred image is greater than some tolerance value at the given start point.

A third variation uses Hertzmann's algorithm, choosing to draw a stroke from the maximal point of error in each grid section (if the average error of the grid block is greater than the tolerance). The grid size is equal to the current brush size times the input grid size. These grid blocks may not fill the image completely. To deal with this, the remaining sections are dealt with in the same way as any other grid section with the exception that the grid size is equal to the remainder of the image.

Randomly Placed Strokes

Random Strokes Only in Areas of Error

Random Strokes Placed at Grid Maximal Error Points

3. Polygonal Linear Strokes

In most forms of painting, strokes can be non-linear (curved). This variation of the algorithm begins the process of dealing with curved strokes. The set of points for each stroke is randomly chosen (using Step 2's final error detection as stroke origin choice). This set of points is stored in a linked list (STL) using the Point2D data structure. The strokes are painted with straight lines connecting each point on the curve.

Single Brush Size, Strokes Painted in Order of Discovery

Multiple Brush Sizes, Strokes Painted in Order of Discovery

Multiple Brush Sizes, Strokes Painted in Random Order

4. Spline Based Curves

To mimic the brush strokes an artist would make, we must turn the randomly generated polygonal linear strokes into curved and flowing strokes.

There are several ways that this could be done. One suggestion, is piecewise cubic b-splines. I chose not to use piecewise cubic b-splines because they do not rest on their start and end points. Thus connecting a multi-piece spline is difficult. In addition, a stroke generated by this method may not follow the points as closely as desired.

However, a b-spline is just an approximation to a Bezier curve. So, I chose to use piecewise cubic Bezier curves.

To generate a piecewise cubic Bezier curve, you need at least four points. But some strokes may have more or less than four points. I felt there were two simple approaches to solving this problem.

Firstly, I could choose the required additional points from the gradient image to "fake" the remainder of the stroke. Then when the stroke is "painted", paint only the portion of the curve corresponding to the "true" points. While this method would produce curves generated completely from piece-wise cubic bezier curve, taking additional points from the gradient field may result in erratic strokes. That is, these "fake" points may cause the curve to go in an undesirable direction -- say into an area of the painting where that stroke is unwanted.

So, I chose to use the points that were given. The algorithm creates as many cubic bezier curves as it can (consuming the involved points as each "section" is rendered). If there are only three points, it is rendered as a quadratic bezier curve. If there are only two points, a linear interpolation method is used. And of course, if there is only one point, a dot is drawn.

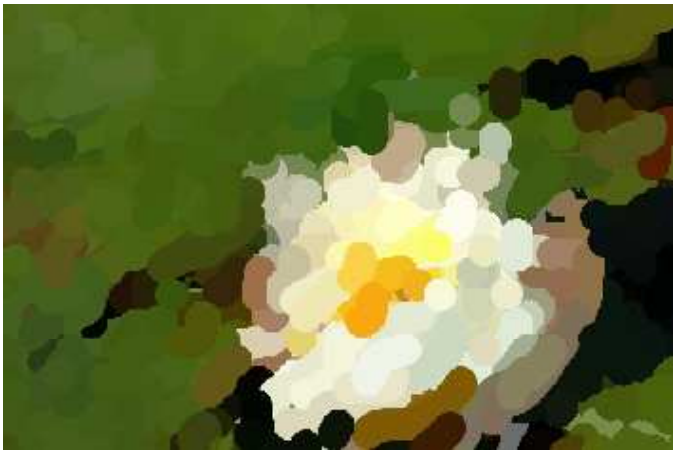
Single Brush Size, Four Point Random Curve

Multiple Brush Sizes, Four Point Random Curve

Multiple Brush Sizes, Five Point Random Curve

5. Following the Gradient Field

To finish off Hertzmanns' main painterly rendering algorithm, the random strokes from before are made to follow the gradient field (as described in the paper). Length in this algorithm is determined by the summation linear length between the points. (Eg. Points={A,B,C,D}, Length = AB + BC + CD).



Single Brush Size, Medium Strokes

Multiple Brush Sizes, Short Strokes

Multiple Brush Sizes, Long Strokes

6. Ordered Stroke Painting

How does an artist paint a picture? First, lets consider the output from step 5, which is Hertzmanns' standard algorithm. Suppose it is a real painting, what kind of paint and what kind of brushes are being used?

Given the strong colours, lack of transparency, and lack of blending exhibited, I would say the results were painted with inexpensive acrylic paint. (Thin to medium viscosity, thick with pigment (dries with a bold colour), fast drying, difficult to blend).

The brush strokes are smooth and solid, with perfectly round endpoints. While not a perfect match, golden sable rounds match the paint strokes best. And the painting surface would be a smooth paper, such as white illustrators paper.

So, if an artist were to use these materials, how would they paint the picture? A common technique, used with nearly every paint type (watercolour, tempra, acrylic, oil, etc) is to paint starting with the lightest colours, and work towards the darkest (the opposite order is not unheard of).

Why is this method used? Objects that disappear into the horizon, are often lighter colours (items at further depth generally painted before items that are near). Lighter paint colours are not usually opaque (especially in the case of the inexpensive acrylics mentioned earlier). If a lighter colour is painted over a dark, more than one coat may be required to get the desired colour. It is easier to darken than it is to lighten.

The first extension to Hertzmanns algorithm, will be to order the strokes. That is, instead of painting each stroke of the same brush size in a random fashion, paint them in order of increasing luminance. The user will be able to choose from random order (Hertzmanns' original algorithm), light-to-dark and dark-to-light.

Light-to-Dark Ordering

Random Ordering

Dark-to-Light Ordering

7. Colour Shifting

Pure impressionism by definition does use black or near black paint. Instead of black, shades of blue are used. Why blue? Impressionist paintings emphasize the flow and play of light over a scene (Monet painted Rouen Cathedral nearly 100 times, each painting at a different time of day). Black is replaced with blue to bring the light of the sky into the dark areas of a scene.

The first colour shift, "Black Shifting", shifts any colour within a certain set range of black towards the blue. The shifting amount is presently fixed, but increases both the blue channel and the total luminance of the colour.

Observing impressionist paintings, another type of colour variation is noted. In an area that should have solid colour, the colour of the strokes is varied such that up close the variance is visible, but at a distance it is not.

This colour jittering is implemented, and the user can control the percentage of pixels jittered, and the level of jittering (how extreme the colour shift is, decimal levels permitted).

Black Shift Colour Jitter (100% Level 1) Black Shift with Colour Jittering (100% Level 0.5)

8. Brush Adjustments

Results

Artistic Movements using Varying Parameters

This brush strokes in the original algorithm (Hertzmann), would represent firm strokes from a brush that does not run out of paint. These firm strokes are almost "angry". To create a more flowing stroke, representing the thinning of paint and the reduction of pressure at the end of a stroke a modification was made to the stroke rendering.

In the last one or two segments of the piecewise bezier stroke, the radius of the stroke is slowly decreased. This will form "pointed" ends to the strokes. The user can select this feature as an option.

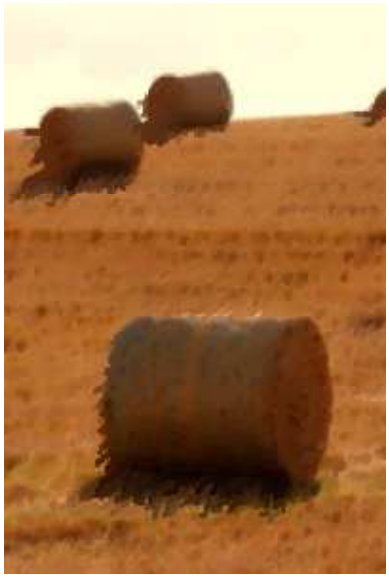
Tapered Strokes (wider brushes) Normal Strokes Tapered Strokes (narrower brushes)

Original Image (Round Bales of Hay)

Impressionism (T=100, R=(8,4,2), f_c=1, f_g=1, f_s=0.5, min=4, max=16)

Expressionism (T=50, R=(8,4,2), f_c=0.25, f_g=1, f_s=0.5, min=10, max=16, j_v=0.5)

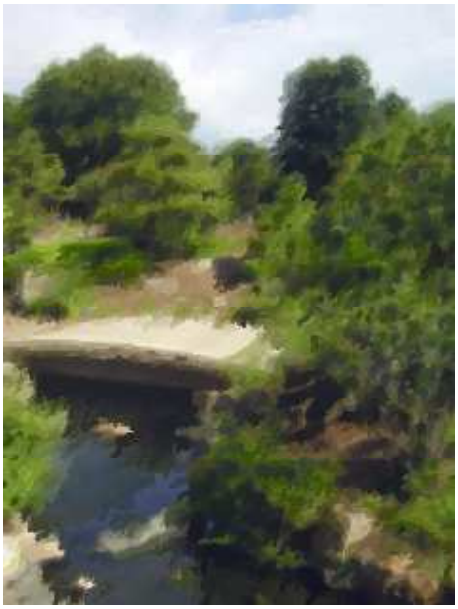
Pointilism (T=100, R=(4,2), f_c=1, f_g=0.5, f_s=0.5, min=1, max=1, j_v=0.3



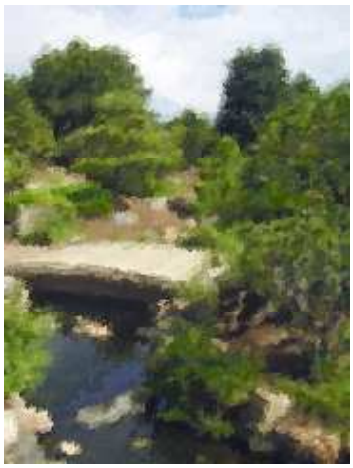
Light-to-Dark (T=25.5, R=(8,4,2), f_c=1, f_g=0.5, f_s=0.5, min=5px, max=50px

Dark-to-Light (T=25.5, R=(8,4,2), f_c=1, f_g=0.5, f_s=0.5, min=5px, max=50px

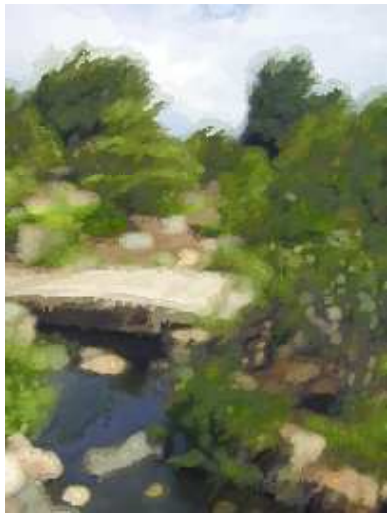
Discussion of Qualitative Differences using the Extention



A) Pure Impressionism with Light-to-Dark Ordering



B) Pure Impressionism with Random Ordering



C) Pure Impressionism with Dark-to-Light Ordering

All three of the above images were generated using the same parameters (Original Image). Tapered strokes, black shifting and slight jittering are used to create a pure impressionist painting. The only difference is the order that the strokes are painted.

Observe that image A has soft edges near skylines and visibly flowing strokes. The smooth colour gradients are preserved. These soft flowing strokes and colour gradients give the image an overall softness. The style of this image is most similar to Monet's impressionist style (based on visual comparison). Image A is my personal preference, and I beleive it has a more natural appearance than image B.

Image B has a certian "digitally produced" appearance. Strokes are randomly cut off, yeilding odd shapes. In the area where the trees meet the sky, the edge is randomized between light and dark. This randomness also gives a "sharper", almost noisy appearance to the image.

Image C is the most unique of the three (although not necessarily the most pleasing). Since the lighter colours are painted over the darker, it has an almost "wet" appearance. While not a simulation of watercolour painting (as there are many details obviously unhandled), it does have a certain degree of visual similarity. The strokes are smooth and flowing, and the colour gradients are preserved. This image has a more natural look than image B.

Images with lighter middle grounds (darker front and backgrounds) may not produce results as pleasing as those images with two colour grounds (front and back). This is because the sorting of the strokes by colour, to some degree extracts and replicates depth information.

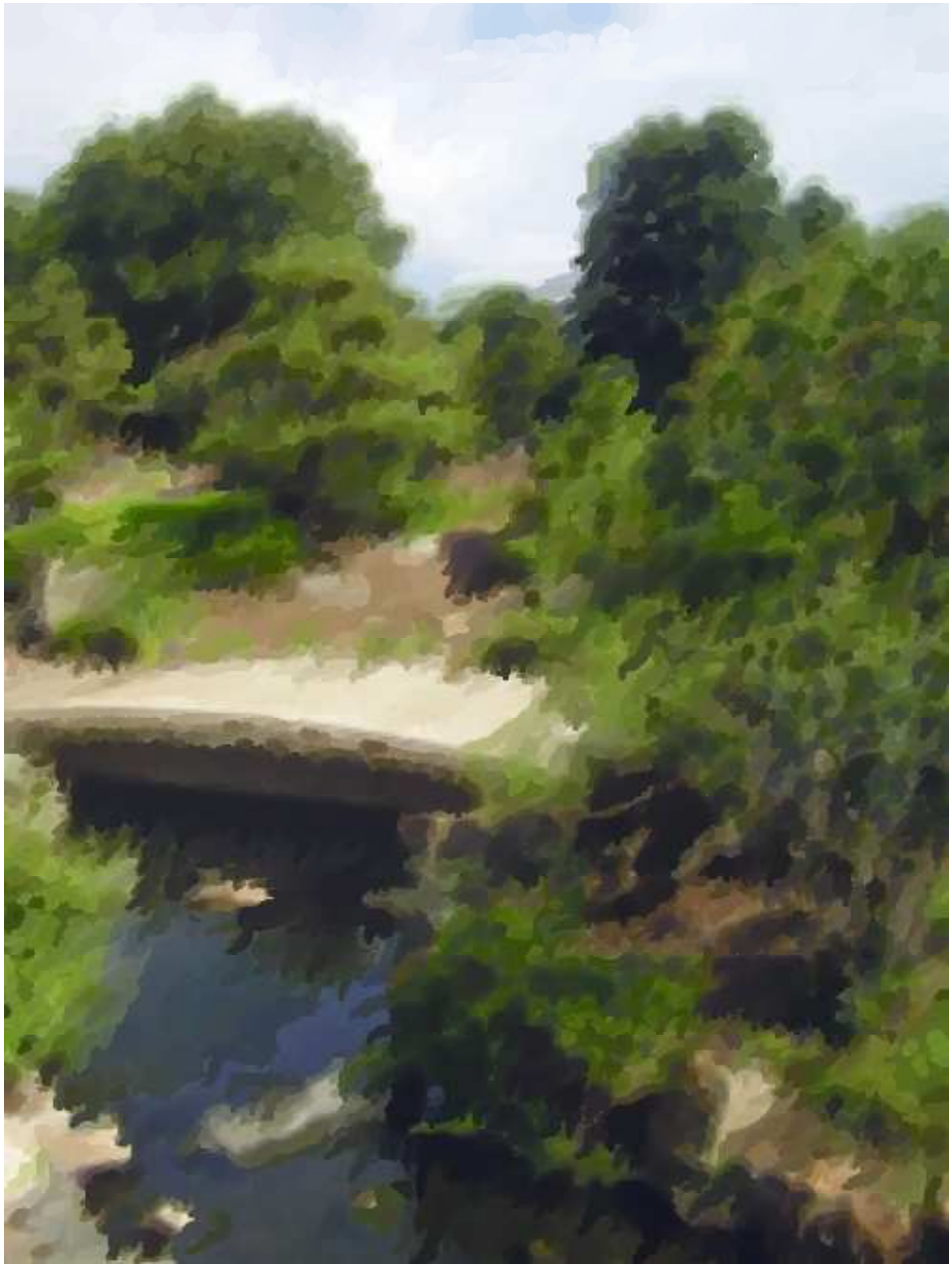
This method of ordering strokes increases the number of art movements that this painterly rendering algorithm can mimic. For example, by preserving colour gradient information Georgia O'Keefe style floral paintings can be created. (Georgia O'Keefe is known for painting close-ups of flowers, her strokes were soft and her images smooth. She is also known for watercolour paintings of similar nature.).

Below is another image rendered using identical parameters, except for the ordering of strokes. The set of strokes for each image is identical, but the difference in results is drastic.

Light-to-Dark Ordering (My main extension)

Random Ordering (Hertzmann's algorithm)

Some Nice Results



Serene Pond
... a Pure Impressionist image
... uses Light-to-Dark ordering and colour shifting
(Original)

Blue Bells
... resembles blended chalk pastels
... Light-to-Dark ordering
(Original)

Pansy
... in the style of O'Keefe
... uses Light-to-Dark ordering
(Original)

Future Work

There are many aspects of the stroke ordering algorithm that could be improved. For example, at the moment the algorithm follows the basic idea of painting strokes in order of luminance. However, if there are multiple sections of a painting with the same colour, the artist will paint them section by section instead of a stroke in one part, then a stroke in another. The sorting algorithm could group strokes of the same colour into regions to mimic this behaviour. The stroke algorithm could also be ordered by depth, if depth information was available.

Piecewise bezier curves are used to implent curved brush strokes. Joins between the strokes can be "non-continuous". A smoother algorithm could be used to give strokes with better flow.

The strokes themselves are very basic here, perfectly loaded brushes with near constant pressure (aside from the "tapering"). There are many different kinds of paint brushes, fans, daggers, rounds, flats, riggers, etc. And there are many different materials for these brushes to be made of, horse hair, sable, nylon, etc. A good improvement would be to model these different brush shapes, and materials.