A GPU-based Method for Real-time Simulation of Eastern Painting

The Kiet Lu[*]
School of Computing, National University of Singapore

Zhiyong Huang[†]
Institute for Infocomm Research (I^2R), A*STAR, Singapore

Abstract

Different from Western Paintings, one can only appreciate the beauty of Eastern paintings by looking at its “spiritual” aspects: “a liking of simpleness with subtlety” – The Art of Sumi-e. It is the careful brushwork with abstract strokes that embrace chance while leaving nothing up to chance, from any subtle changes in brush’s pressure onto the paper to the fascinating ink diffusion absorbed by painting papers with various microstructures. Hence, the goal of this research work aims to “capture” these expressivenesses of the Eastern Arts, in digitalized form, which will open to new opportunities of exploration of painting techniques on the new digital Media, as well as of the easiness of calligraphical practices, complementary to the traditional way. By combining the flexibility from CPU and the power of GPU parallelism, together with carefully scheduling the simulation tasks between CPU and GPU, we have achieved good results with high visual quality and also are able to fulfill the requirement of real-time simulation.

CCS Categories: I.3.1 [Hardware Architecture]: Graphics processors; I.3.3 [Picture/Image Generation]: Display algorithms; I.3.4 [Graphics Utilities]: Virtual device interfaces; I.3.5 [Computational Geometry and Object Modeling]: Physically based modeling; I.3.6 [Methodology and Techniques]: Interaction techniques

Keywords: Interactive Painting, Brush Modeling, Physically-based Simulation, GPU programming

1 Introduction

Brush simulation is a subject in non-photorealistic rendering. In fact, the first method proposed can be traced back ever since 1986 by Strassmann [12]. In his simulation system, the 2D brush footprint is constructed by “splatting” 1D or 2D bristle map along the drawing line, which is usually sampled from pen-based input. However, the method is only suitable for Western paintings. However, in Eastern painting, the symbolism may take higher priority where “It is a classical art form that emphasizes the skill of using brush and ink and carries deep inner meanings” – the Art of Sumi-e, Shozo Sato. Thus, as illustrated in Figure 1, only consists of a few very abstract strokes of black and white, mastery Eastern paintings can seek to capture the spirit from the physical world and blend it perfectly into the metaphysical world which is difficult to understand in Western terms.

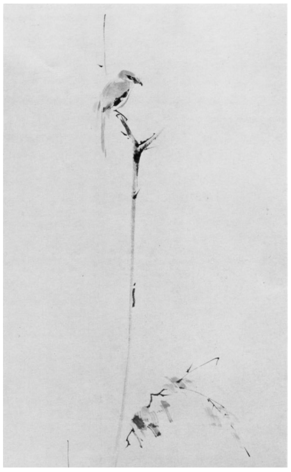
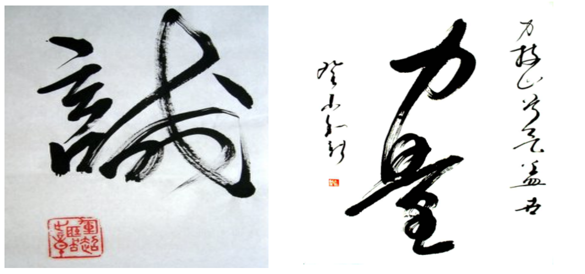


Figure 1. A Sumi-e Masterpiece: Shrike on a Barren Tree

Due to this reason, the restriction from traditional 2D splatting method no doubt becomes the obvious constraint for Eastern paintings, at least in terms of brush footprint expressiveness. In fact, a traditional system can only afford some degrees of freedom, for example, the thickness for each stroke and orientation. However, Chinese brush technique is the great collection of many different stroke styles, especially in calligraphy works illustrated in the Figure 2.



(a) (b)

Figure 2. Two Chinese calligraphy artworks with brush splitting effect

Recently there has risen up a new interest shifting towards simulating the actual 3D model of the painting brush. One of the first efforts is by Xu et al. [14]. Taking into account of many physical conditions like water wetness level, brush pressure and physical energy, a method was proposed of using deformable NURB sweeping surface along the brush main skeleton to generate the 3D brush model by sweeping operation. Due to the use of the 3D brush model, the footprint generated offers much more direct and natural stroke shape compared to the previous work. However, the results are still not persuasive enough because the simplicity of 3D solid volume is not sufficient to capture the flexibilities of the brush. For example, in reality the brush tends to

[*] edwin.kiet@gmail.com
[†] zyhuang@i2r.a-star.edu.sg

branch out into many small groups of bristles called turf when drying out or being pressed by high pressure, which is called "flying-white" calligraphy technique as illustrated in Figure 2.

On the other hand, ink diffusion effect is also a very important factor would contribute into immersive simulation. A wet brush is referred as one that has been coated with so much ink that it drips easily from the bristles. For example, in "chrysanthemum", one of the results produced by our system in Figure 15, the smeared outline is produced by the excessive ink applying on the paper.

In this paper, we propose a GPU-based method for real-time simulation of Chinese painting. There are three main contributions: First, for 3D brush modeling, we introduce a novel 3D brush simulation method. Taking advantage of GPU parallelism, the model can support up to thousands of each individual bristle that altogether form the 3D brush. Our method yields out with more realistic and persuasive results because the process is the natural and direct way for interactive painting application. Second, due to using the individual bristles as the lowest primitives of the virtual brush, the simulation results would be unlimited resolution in theory. Third, post-rendering ink diffusion effect based on physical "Xuan" paper structure is implemented and integrated seamlessly into 3D brush model to achieve more the realistic results. The experimental results showed the effectiveness of the method. For example, the "Orchid" painting can be generated in 1.5 times of A1 resolution 300 dpi with 25 frames per second, see Figure 16.

2 Related Work

In this section, we will summarize the methods of brush painting simulation with their relevance to our method.

2.1 1D brush modeling

Date back to 1986, Strassmann published his Master's thesis "Hairy Brushes" which is regarded as the first generation system for hairy brush painting [12]. It is the pioneer research that defines a painting system with four different components: the brush, the stroke, the dip, and the paper. It would offer a great flexibility for experiment framework, for example, instead of directly rendering the brush footprint to the final results, the system could integrate different ink effects and paper texture maps into the final results.

Strassmann method is able to synthesize many of the interesting effects often seen in sumi-e style paintings. However, there are also some drawbacks that make it far from what painters should experience. Firstly, for each stroke, the system requires user to specify a set of control points and the pressure levels, which is quite different from the actual interactive painting process. Time consumption is another problem for interactive simulation. Due to using CPU to compute the footprint, generating each stroke may take more than one second, which is impossible for interactive application.

2.2 3D brush modeling

With the emerging of powerful CPUs, hairy brush simulation becomes more and more practical [5, 7, 8, 13, 15]. The painting brush model proposed by [8] is the first important improvement of [12], from 1D to actual 3D brush model with three significant advantages: First, it is more natural for painters to directly manipulate the 3D brush rather than abstract the control points. Second, it is more accurate to capture many subtle Eastern painting brush effects as having mentioned before. Third, the system employs the physically-based approach to afford more accessibility for the users to easily adjust many parameters like the stiffness of the bristle, wetness of the brush and the friction level of the paper.

The brush model presented in our work is inspired by [2, 8], where a large amount of bristles, which is represented as parametric curves, is attached to the same root on the brush handle. One problem of Lee's model is that the number of bristles is limited to achieve high realism. Our method addresses this problem by a GPU-based solution which offers more brush dynamics as well as high speed of rendering.

2.3 The skeleton brush

Lee's research [8] suggested that a high level of physical-based simulation needs to be taken into account in order to achieve persuasive results. Following it, Chu and Tai demonstrated a more accurate of physically based system [2]. However, their dynamics model, originally from [1], consists of many nodes which will be too expensive to compute in real-time. Even though it is very physically accurate, the number of bristles that the system can support is limited which affects the stroke quality. The Equality Non-linear Constraint Programming is applied for solving energy minimization problem which is very computational expensive, especially when the number of variables of the optimization equation is fairly high. This inherently limits the levels of detail that their brush model can achieve, especially for high resolution results.

2.4 Ink diffusion model

The additional step to achieve realistic brush painting is the ink diffusion model. A physically-based model to simulate the "Xuan" physical paper structure was proposed in [13] consisting of a fiber network which mimicries the actual "Xuan" paper with cells as fiber piles, linked by fiber threads. However, the problem of this method is the large amount of micro fibers and cells that it has to represent. In reality there are thousands of fibers overlapping onto a small area of paper. As a result, the method becomes too expensive to use for high resolution rendering.

Gou and Kunii later proposed an approach for representing the Nijimi effect [7]. In their work, fiber mesh is subdivided into smaller circular regions. Each region will be assigned with a number of micro fibers. "Xuan" paper mesh data is constructed by traversing each region and generate mesh structure locally. This method is more efficient than [13] because it reduces the complexity of large paper area into small regions and generates the fiber-mesh data by computing the virtual capillary tubes among these regions as fiber-mesh information. The excessive ink is transferred in the diffusion process from the source, along the capillary tubes, to the surrounding paper cells, in a way similar to the real "Xuan" paper.

However, since having to store information of the fibers (such as position, orientation) into fiber data structures, the fiber-mesh construction is still quite expensive for large scale paper. Our work addresses the problem with improvement of this issue. More recently, a physically-based method was proposed for simulating ink dispersion in absorbent paper [3], where the fluid flow model is based on the lattice Boltzmann equation. However, physically-based ink dispersion is not our focus.

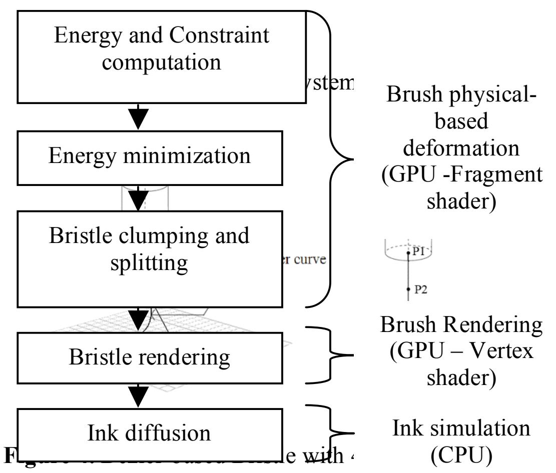
3 Our Method

In this section, we first describe our Bristle-based brush model. Then, we present the use of GPU to achieve real-time simulation.

Finally, we discuss the improvement on the ink diffuse model. For the future work, we are conducting usability study. We have shown a demo to public in the "Window into SoC" on May 19, School of Computing, NUS. Feedbacks from users have been summarized and used for the improvement of our design.

3.1 The bristle-based brush model

In this subsection, we describe Bristle-based Brush (BBB) model design. We use a hybridization approach aiming to achieve both the high quality of realistic brush footprints and real-time simulation. The BBB model follows the real painting brush structure which is made up of a mass group of individual small hair units called bristles, whose size may be up to thousands. The individual deformation is necessary because the brush geometry volume is totally dynamic and more flexible with countless degrees of freedom. Each bristle is presented as a Bezier curve with 4 control points computed based on energy minimization method using equality constraint optimization. The summary of the BBB model is shown in Figure 3.


Energy and Constraint computation
ystem
Energy minimization
Bristle clumping and splitting
r curve
Bristle rendering
Ink diffusion
Brush physical-based deformation (GPU -Fragment shader)
P1
P2
Brush Rendering (GPU – Vertex shader)
Ink simulation (CPU)

Fe with , P3 and P4 (a). Initial non-pressure state of P1, P2, P3 and P4 (b)

The use of Bezier curves requires less computation because of smaller number of the control points. Further more, it can fit well into GPU platform. Depend on the resolution requirements, the number of bristles should be adjusted accordingly. For example, if paper resolution is more than 40 million pixels, more than 1,000 bristles should be used to achieve the realistic level. Finally, BBB can solve the clumping and branching out effect naturally as it works on the lowest level of bristles. The effect can be produced by manipulating the bristles represented by Bezier curves into groups by using K-nearest neighboring algorithm which will be elaborated in subsection 3.3.3.

To use cubic Bezier curves, each bristle is composed of four control points P1, P2, P3 and P4 to define the shape. Note that P1 and P2 are aligned statically to the brush handle main axis as in Figure 4. They are fixed relative to the orientation of the axis, which in charge of the physical stiffness of the brush.

Thus, the actual dynamic factors fall into control points P3 and P4 at the end of the bristle. There are also constraints for them. First, they must always lie right above the paper plane as illustrated in Figure 4. This assures the brush will always touch and do not penetrate the paper when pressed. Second, they must be positioned such that the bristle's arc length keeps constant.

Initially, P3 and P4 are set to be close to each other so that without pressure, the bristle is orthogonal to the paper plane as in Figure 4 (b). During the painting simulation, the length of vector P3P4 changes (stretches or contracts) governed by Physics.

To simulate the bristle deformation accurately in Physics, BBB uses equality constraint optimization to determine the steady state of the bristles (P3 and P4) at each step of the iterative process. Giving the new positions of P1 and P2, due to the brush new orientation, the system will solve the energy minimization problem seeking for the new positions of P3 and P4 to the most stable configuration with least energy stored, similar to that of mass-spring particle system. Naturally, the sequential quadratic programming (SQP) with equality constraints is employed for solving the problem. The detail of solving the minimization energy with equality constraint is firstly described in 3.1.1 followed by the energy function formulation in 3.1.2.

3.1.1 Solving energy minimization problem

The optimization problem is solved by sequential quadratic programming (SQP) with the active set of equality constraints [9]. Consider the general problem: *Minimize* $f(X)$ such that $h^j(X) = 0, j = 1, 2, \dots, r$. To find optimal solution of $f(X)$, according to [11], it can be shown that given estimates X^k, $k = 0, 1, \dots, n$ with respective to multipliers values λ^k, the iteration step $\boldsymbol{s}$ of iteration $k+1$, such that $X^{k+1}=X^k+\boldsymbol{s}$ is given by the solution of the following *k-th SQP* problem:

SQP-k *(Xk, λk)* // Minimize with respect to $\boldsymbol{s}$
$F(s) = f(X^k) + \nabla^T f(X^k)\boldsymbol{s} + \frac{1}{2}\,\boldsymbol{s}^T H^L(X^k)\,\boldsymbol{s}$ *(1)*
Such that
$h(X^k) + h'(X^k)^T\boldsymbol{s} = 0,$ *(2)*
and the Hessian of the classical Lagrangian with respect to x is $H^L(X^k) = \nabla 2f(X^k) + \sum\nabla^2 h^j(X^k)$.

Table 1. SQP-k problem

Given the estimates X^K, the solution of *SQP-k* yields $\boldsymbol{s}$, thus we can construct the next *SQP* problem: *SQP-k+1* until $\boldsymbol{s}=0$.

For an active set of equality constraints, according to [11], the problem of finding solution ***s*** of *SQP* with only equality constraints presented above can be much simplified in *SQP* with active set of equality constraints. First, consider a *SQP* problem with equality constraint:

$$Minimize\ f(X) = \tfrac{1}{2} X^T AX + b^T x + z \qquad (3)$$

Subject to

$$C^T X = D \qquad (4)$$

Suppose that the active set at X^* is known or it can represent the active set in matrix form by $C^T X = D$ as in equation (4). Then, the solution $X^\#$ is obtained by minimizing $f(X)$ over the set $\{X \mid C^T X=D\}$. The solution is obtained by solving the linear system:

$$\begin{bmatrix} A & C^T \\ C & 0 \end{bmatrix}\begin{bmatrix} X \\ \lambda \end{bmatrix}=\begin{bmatrix} -b \\ D \end{bmatrix} \qquad (5)$$

To sum up, given a set equality constraints of Equation (4), we can solve the *SQP* problem with linear equation system of Equation (5).

Note that *SQP-k* problem of Equation (1) with the equality constraint from Equation (2) is exactly the same with Equations (3) and (4) where $X=\mathbf{s}$, $z=f(X^k)$, $b=\nabla f(X^k)$, $A = H^L(X^k)$, $C = h'(X^k)$ and $D = -h(X^k)$. In other words, we can find ***s*** the Newton step of X in Equation (1) by solve this linear equation:

$$\begin{bmatrix} H^L(X^K) & h'(X^K)^T \\ h'(X^K) & 0 \end{bmatrix}\begin{bmatrix} s \\ \lambda \end{bmatrix}=\begin{bmatrix} -\nabla f(X^K) \\ -h(X^K) \end{bmatrix} \qquad (6)$$

Thus, with $X^{k+1}=X^k+\mathbf{s}$ we may construct the next *SQP* problem *SQP-k+1* until getting convergent to optimal solution $X^\#$ with $\mathbf{s}=0$. At that point, $F(X^\#)$ is absolutely minimal and the constraint $h^j(X^\#)=0$ will also be satisfied.

It is now straight forward to apply it into bristle deformation by minimization of bristle energy and its physical constraints. Here is the algorithm:

For each bristle represented by a cubic Bezier curve, let the previous state of the four control points be $O_i=(P1_i, P2_i, P3_i, P4_i)$, we need to determine the new configuration of new $P3_{i+1}$ and $P4_{i+1}$ of O_{i+1} as bristle's new state: $O_{i+1}=(P1_{i+1}, P2_{i+1}, P3_{i+1}, P4_{i+1})$ in the following steps:
1) Set initial values of $O_{i+1}=O_i$.
2) Update O_{i+1} with new brush transformation (includes translate and rotation). O_{i+1} state now becomes non-optimized (or less stable) and we need to determine new positions for $P3_{i+1}$ and $P4_{i+1}$ to have O_{i+1} with least energy state.
3) Based on O_i and new O_{i+1}, the system calculates the energy function as well as constraint equation for SQP to find new $P3_{i+1}$ and $P4_{i+1}$ (subsection 3.1.2).
4) Using Equality constraint optimization to find $P3_{i+1}$ and $P4_{i+1}$.
5) Update the new positions and repeat the loop.

Table 2. Algorithm of bristle deformation

3.1.2 Constraint and energy formulation

The most important thing for solving *SQP* with equality constraint as presented in 3.1.1 is to setup the proper constraint and its energy measurement. The purpose is to make sure the system stable and able to reach the optimal solution in numerical solution.

First, it is necessary to ensure that the arc length of Bezier curve to remain unchanged. Second, the bristle must not penetrate the paper. So, there are two bristle-paper contacting constraints to be satisfied: the *P3P4* vector must fall completely right above the paper plane, forcing the bristles to lie along and the arc length must be preserved.

For the non-penetrating constraint, we assume *y* component of *P3* and *P4* are *0* so that they always lie on paper plane as in Figure 4 (a). For the arc length constraint, one straight forward approach is to sample the Bezier curve with fixed intervals and get the approximate of the arc length by summing the segments: $ArcLength(O_i)=\sum \mid Pt^j - Pt^{j+1} \mid$ with Pt^j the sample points on Bezier curve at interval jth and $O_i=(P1_i, P2_i, P3_i, P4_i)$ as current state of the Bristle curve. Thus, the equality constraint for *SQP* can be formulated as: $H(O_i)$: $ArcLength(O_i - BezierLength = 0$ with *BezierLength* the desirable length of the bristle.

For energy function, due to light weight characteristic of brush bristle, it is reasonable to ignore the potential energy from gravity as well as kinetic energy that actually contributes very minimum to the overall energy function. As the result, each bristle has two major components, bend potential energy and friction lost energy. The bend potential energy *EBend* is straight forward to compute from the angles *O1* between *P1P2* and *P3P2, O2* between *P2P3* and *P4P3* (Figure 5). $EBend(O_i)=Kbend\ (w_1|O_1|+w_2|O_2|)$, where *Kbend* is the bend coefficient, and w_1 and w_2 the weights for each angle. The weights can be tuned experimentally.

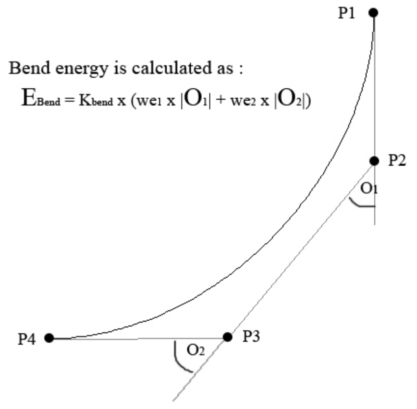
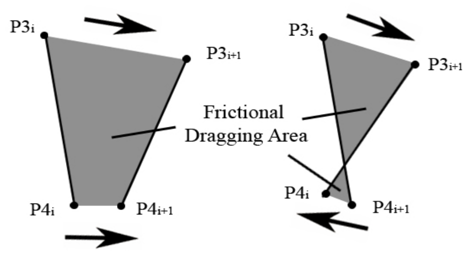

P1
Bend energy is calculated as :
$E_{Bend} = K_{bend} \times (we_1 \times |O_1| + we_2 \times |O_2|)$
P2
O1
P4
P3
O2

Figure 5. Bend energy computation

Friction lost energy is more complicated to compute because it involves in the dragging area sweeping from $(P3_i, P4_i)$ to $(P3_{i+1}, P4_{i+1})$ as shown in Figure 6.



(a) (b)

Figure 6. Different cases for frictional lost energy based on dragging area

As two vectors $P3_iP4_i$ and $P3_{i+1}P4_{i+1}$ can cross each other as shown in Figure 6(a), forming non-convex area is hard to handle. Therefore, to have a robust solution, we subdivide the two vectors into *N* segments and calculate the sweeping area for each pairs of them as in Figure 7.

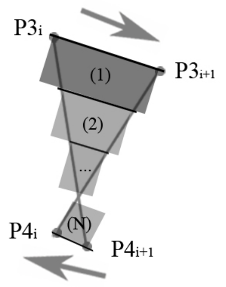


Figure 7. Discrete solution for dragging area approximation

The Friction energy is formulated by $E_{frict}(O_i)=K_{frict}\ \Sigma(w_i\ A_i)$ where K_{frict} is the frictional coefficient, A_i is the approximated areas for the sweeping operation of the *ith* segment, w_i is the weight coefficient. *P3* are put with higher weight as it contributes more frictional lost energy.

3.2 The use of GPU

The use of GPU is a common practice in real-time graphics system nowadays. However, one limitation of GPU programming is that for each rendering kernel, the fragment shader can output maximum only a set of four 32-bit floating point channels at a time, or *128-bit* of information in total, even though it can read much more texture inputs from the main memory. In CG language, though it is possible to pack *8* or *16* values in to *4* channels [10], the information will be lost proportionally which is undesirable. Control point positions must be precisely computed; otherwise it will not converge in solving the equality constraint optimization problem. Thus, in order to map the algorithm onto GPU platform, a special way of data structure use needs to be considered.

3.2.1 The data structure

First, because of the non-penetrating constraint on control points *P3* and *P4*, their *Y* components can be removed from the equality constraint optimization process (which always zero). Further more, the control points *P1* and *P2* are only related to the brush handle axis and can be computed globally, based on each new transformation. So, for individual bristle deformation, this leaves us exactly only *4* parameters to consider: X_3, Z_3, X_4 and Z_4 which can be arranged nicely in *4* separate channels of one texture pixel. Consequently, bristle information can be well-compressed into only one texture unit with four *32-bit* components. This has shown a great advantage in the viewpoint of GPU pipeline as it takes only one rendering pass in GPU to get all the bristle deformation results [4]. The process of the GPU fragment program is listed as follows:

1) Initialize bristle information *(X3, Z3, X4, Z4)* and transfer it into texture with *(R, G, B, A)* format. 2) For each pixel, the fragment shader will • Generate *P1* and *P2* based on the global brush handle axis. • Read input texture *(R,G,B, A)* and construct *P3* and *P4* as *(X3, 0, Z3)* and *(X4, 0, Z4)* with *X3=R, Z3=G, X4=B* and *Z4=A*. • Compute new *P1', P2', P3'* and *P4'* by translation and rotation of *P1, P2, P3* and *P4*, as first raw deformation. • Setting up constraint and energy function based on *P1, P2, P3, P4, P1', P2', P3',* and *P4'* as described in previous section. • Solve the energy minimization problem to find new *(X3', Z3', X4', Z4')* of *P3'* and *P4'*. • Output to as *(R, G, B, A)* format.

Table 3. The process of GPU fragment program

3.2.2 Footprint generation on GPU

It takes the second rendering pass of GPU to generate the footprint. After the new positions of *P3* and *P4* are determined from GPU fragment shader, another vertex shader is used to construct and render the bristles with four new Bezier control points.

Note that only bottom parts of the Bezier curves actually contribute to the footprint as suggested in Figure 8. Those upper parts are irrelevant and should be discarded. This is done easily by setting the far plan of the viewing frustum with proper distance in order to cull off the unnecessary parts.

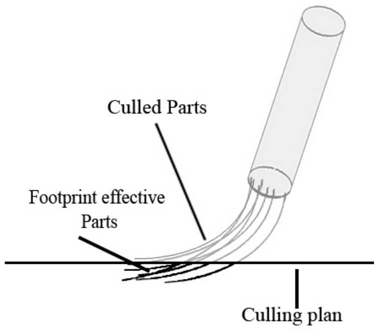


Figure 8. Illustration of Bristles culling for footprint synthesizing

3.2.3 Bristle clumping and splitting on GPU

The brush model can clump and split the bristles dynamically based on the wetness and pressure level of the brush physically, independent of the resolution.

Thus, we can cluster the bristles into smaller groups with same representational control points *P3* and *P4*. First, the system needs to derive the wetness level of the virtual brush by defining *Wet(t)* as the function of time *t*. In our simulation system, it is defined as linear function of time: $Wet(t) = K_{wet}\, t.$, where K_{wet} is the wetness coefficient decided by the paper type.

Next, we need a scheme to cluster bristles together and assign those groups with their unique representational control points *P3* and *P4*. This can be done dynamically without interfering the GPU pipeline by using a look-up coordinate texture table to replace the original coordinates automatically. As in Figure 9, the vertex shader will access only those presentational texture coordinates from the lookup table, forming new representational *P3* and *P4*. As the result, with the look-up table, bristles are can be clustered together as in real brush.

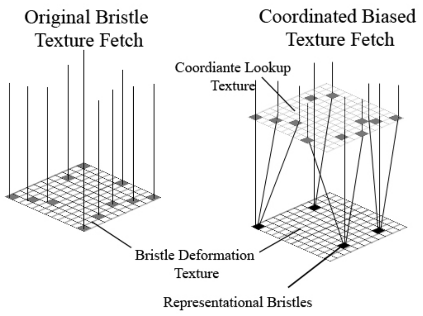

Original Bristle Texture Fetch
Coordinated Biased Texture Fetch
Coordiante Lookup Texture
Bristle Deformation Texture
Representational Bristles

Figure 9. Coordinate look-up Texture fetch

In order to construct the look-up table, K-nearest neighboring algorithm is used as following:

1) Generate initialized Coordinate values, ranging from *(0, 0)* to *(1, 1)*, same as the original.
2) Based on the wetness level, determine number of bristle groups that the Brush should have. The higher *Wet(t)* is, the lesser number of groups the system should have.
3) For each group, decide its representative coordinate by using randomize function.
4) For each pixel in the Bristle deformation Texture, find the Cartesian nearest neighbor among the representative coordinates defined at step 3 and replace it accordingly.

Table 4. The process to construct a look-up table

3.3 Physical-based ink diffusion with "Xuan" paper structure

This subsection will focus on the ink diffusion effect on "Xuan" paper structure as the post-processing stage of the simulation. The physically-based "Xuan" paper structure is discussed followed by the elaboration of the ink diffusion scheme.

From the microscopic pictures of painting papers, its structure is a fibrous mesh consisting of irregularity distributed fibers [7]. This means that even though the fibers are rather distributed randomly, due to its geometry characteristic, the fiber density at local level still varies non-uniformed. This non-uniformed structure of "Xuan" paper is the source of "Nijimi" diffusion effects. Note that the fiber mesh structure is impossible to generate by using simple probability distribution functions and needs to work on the fiber geometry level.

3.3.1 Paper mesh construction

We build up paper mesh data from actual individual micro fibers by considering the position, orientation and geometry information of individual fibers. Improving the method in [7], we present a method for generating "Xuan" paper mesh data with less computational cost and practical to high resolution requirement.

The major idea of "Xuan" paper mesh data is to identify capillary tubes forming among the paper cells. Capillary tubes are defined as many micro interlacing fibers staying closed together, can transfer or diffuse the liquid to the other ends of the tube. Hence, capillary tubes mesh information is very crucial for diffusion effect because the ink liquid flows very strongly along the capillary tubes and this active flowing process decides "Nijimi" image.

The micro fibers structure is stored in texture format. As shown in Figure 11, a set of different fibers geometry and orientation are presented by a texture map with *alpha* value to provide fiber shape information. Each texture has a unique color ID *IDColor* which is assigned based on the fibers main axis orientation as in Figure 10. Those fiber textures are then cloned, randomly distributed as in "Xuan" paper. Fiber density Texture *(D)* is generated naturally by accumulating the number of fibers overlapped at each point in the texture. For efficiency, the whole fiber "splatting" process is GPU accelerated.

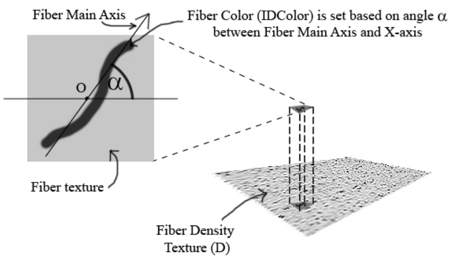

Fiber Main Axis
Fiber Color (IDColor) is set based on angle α between Fiber Main Axis and X-axis
o
α
Fiber texture
Fiber Density Texture (D)

Figure 10. "Xuan" fiber structure

Next, the capillary structure data *(C)* is determined in CPU, with the assumption that all the capillary tubes formed by connected fibers having the same radius. It is further assumed that fibers will form the diffusion tube whenever there are *N* fibers with similar orientation connected together at any points on the paper. In our implementation, it is assumed that a tube is formed with $N \geq 3$. Note that the orientation has already been coded by *IDColor* as in Figure 11. Hence, for each position in the paper, the fiber *IDColor* is compared with its neighbors. If the color difference falls into the threshold, which means that they are oriented pretty much in same direction, it is considered that that tube has been formed among them and recorded into the output texture. As the result, the capillary texture data *(C)* will record the capillary tube mesh information at each cell to other neighboring cells.

3.3.2 Nijimi diffusion effect

Nijimi diffusion takes place only when the brush deposits an excessive amount of liquid onto the paper. This is so-called the source of ink diffusion that will transfer its liquid to the surrounding areas with lower ink level, starting from its nearest neighboring points. In this simulation, it is assumed that the speed of ink diffusion is constant. Therefore, at each steps of the diffusion, only *8* adjacent neighboring points will receive the liquid transferring as in Figure 11.

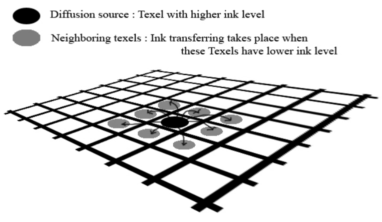

● Diffusion source : Texel with higher ink level
● Neighboring texels : Ink transferring takes place when these Texels have lower ink level

Figure 11. Ink transferring scheme

For each of liquid transferring, there are three main characteristics of "Nijimi" diffusion that has to be strictly followed:

1) Ink pigment and water will flow along the capillary tubes formed by parallel micro fibers to its neighboring areas.

2) The flow only flows from higher ink density to lower with decreasing diffusion.

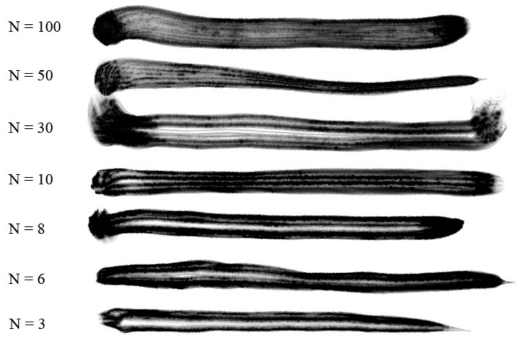
3) The absorbency at each point based on the local fiber density.

Therefore, in our implementation, the amount of ink exchanged is proposed as a function based on the paper absorbency which involves in the fiber density *(D)* and the capillary tube *(C)* information. The amount can be computed as the multiplication of *(D)* and *(C)* as: *InkTransfer= inktrans WDC*, where *W* is the weighting constant. *W=1/4* or *1/8* if *4* or *8* neighboring points are transferred, *inktrans* is the coefficient to control the amount of ink transferring.

4 Results and Limitation

4.1 Results

We have implemented the proposed method and it yields several interesting results. Figure 12 shows the brush strokes simulation with different wetness levels.



N: number of bristle groups

Figure 12. Brush clumping and splitting with different groups

Figure 13 shows the simulated results with bamboo painting techniques. This is described as "the sections between knots near the ends of the stems should be short, those forming the middle of the stem should be long, and while at the base of the plant they again are short".

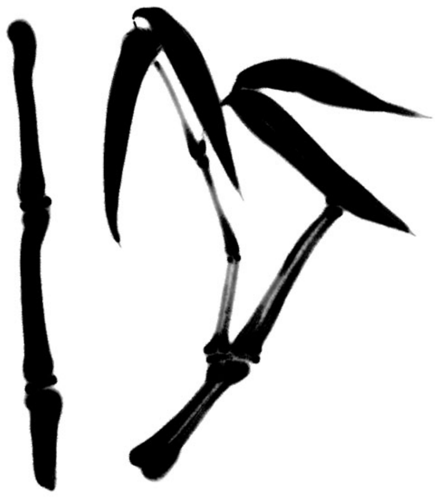


Figure 13. Bamboo illustration of 3D brush dynamic deformation

Figures 14 and 15 show painting and calligraphy brushworks as illustration for the combination effects between 3D brush modeling and ink diffusion simulation with different water levels, demonstrating from "Nijimi" diffusion effects to "flying white" bristle splitting out phenomenon.

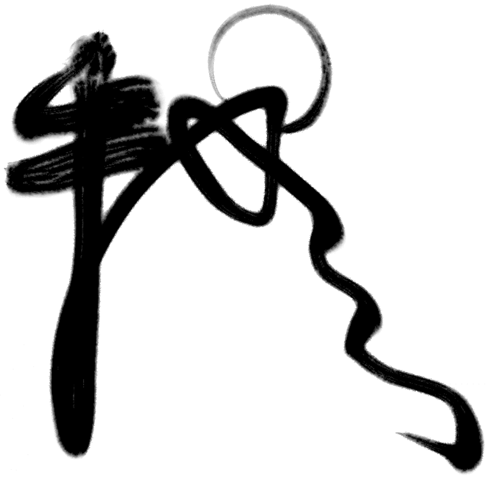


Figure 14. Vietnamese calligraphy of "Soul"

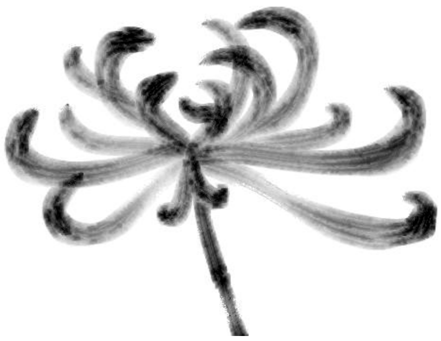


Figure 15. Chrysanthemum with Nijimi diffusion and "flying white"

Figure 16 shows part of the virtual brush result inspirited from actual brushwork.

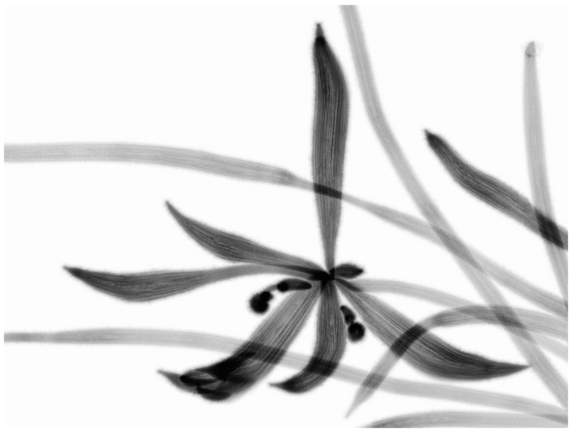


Figure 16. A "snapshot" of A1 – 300 dpi resolution result

All the results are generated by real-time rendered on a Pentium *4 3.2 GHz, 1GB* Ram with *NVIDIA 7800 GT* graphic card. For example, the result in Figure 16 was painted in approximate *25* frames per seconds. The input devices are the Intuous A6 Wide Tablet and Art Pen.

4.2 Limitation

The first limitation of this Bezier-based simulation is the assumption on P3 and P4 controlling points that they both have to lie on paper plan. The purpose is to reduce the complexity of the energy minimization problem from 6 parameters to 4 parameters (Y value of P3 and P4 is assumed to be Zero). This results with unrealistic result for some cases of extremely fast stroke in Chinese calligraphy, which is currently out of our research range.

Another limitation is that the global behaviors for each bristle have not been touch, namely self-collision between neighboring bristles as well as attraction force from the ink liquid that holds the bristles together.

5 Conclusion and Future Work

In this paper, we propose a GPU-based method for real-time simulation of Chinese painting. It includes physically-based brush deformation and seamless integration with ink diffusion rendering on "Xuan" paper structure. 3D Brush is modeled as a large number of small bristles. Each bristle is represented by a piece of cubic parametric Bezier curve. The deformation is physically based that takes into account of determining the balance state by minimization the actual physical bend potential energy and frictional energy of the bristles. Bristles splitting and clumping effect is simulated by accessing to the wetness level as in real brush painting. The footprint is generated by protecting the Bezier curves onto the paper plan which allows higher resolution output compared to previous methods. Ink diffusion is simulated based on "Xuan" paper structure. The contributions of this work are as follows:

1) A novel 3D brush simulation method can model up to thousands of each individual bristles that altogether form the 3D brush.
2) Mapping from the algorithm onto GPU parallel exploits programmable graphics hardware (GPU) to achieve interactive simulation speed.
3) Post-rendering ink diffusion effect based on physical "Xuan" paper structure is implemented and integrated seamlessly into 3D brush simulation to achieve more the realistic results.

The self-collision among individual bristles can be considered in future work. We also can work on simulating other Eastern techniques, for example "ink washing" with various colors, instead of using gray-scale intensity. Finally, we would like to thank the comments from Graphite 2007 reviewers.

6. References

[1] W. Baxter, V. Scheib, M. Lin, and D. Manocha, Dab: Interactive haptic painting with 3d virtual brushes. SIGGRAPH 2001, pp. 461–468.

[2] N. S.-H. Chu and C.-L. Tai, Real-time Painting with an Expressive Virtual Chinese Brush. IEEE Computer Graphics and Applications, September/October, 2004, 24(5). pp. 76-85.

[3] N. S.-H. Chu and C.-L. Tai, MoXi: Real-time ink dispersion in absorbent paper, ACM Transactions on Graphics (SIGGRAPH 2005 issue), Vol. 24, No. 3, August 2005, pp. 504-511.

[4] D. Goddeke, Playing Ping Pong with render-to-texture, University of Dortmund, Germany, 1999.

[5] C. J. Curtis, S. E. Anderson, J. E. Seins, K. W. Fleischer, D. H. Salesin, Computer–generated watercolor, SIGGRAPH 1997, pp. 421-430.

[6] R. B. Girshick, Simulating Chinese brush painting: The Parametric Hairy Brush, SIGGRAPH 2004 poster, pp. 22.

[7] Q. Guo, T. Kunii, "Nijimi" rendering algorithm for creating quality black ink paintings, CGI 2003, pp. 152-161.

[8] J. Lee, Diffusion rendering of black ink paintings using new paper and ink models. Comput. & Graphics, 25:295-308, 2001.

[9] J. Nocedal, S. Wright, Numerical optimization. New York: Springer, 1999, ISBN-13: 978-0387987934.

[10] M. Pharr and R. Fernando (series editor), GPU gems 2: programming techniques for high-performance graphics and general-purpose computation, edited by Addison-Wesley, 2005, ISBN-13: 978-0321335593.

[11] J. A. Snyman, Practical mathematical optimization: An introduction to basic optimization theory and classical and new gradient-based algorithms, Springer; 1 edition (November 29, 2005), ISBN-13: 978-0387298245.

[12] S. Strassmann, Hairy brushes, SIGGRAPH 1986, pp. 225–232.

[13] M.-J. Sun, J.-Z. Sun, B. Yun, Physical modeling of "Xuan" paper in the simulation of Chinese ink-wash drawing, CGIV 2005, pp. 317-322.

[14] S. Xu, M. Tang, F. Lau, and Y. Pan, Non-photorealistic rendering styles: A solid model based virtual hairy brush, Computer Graphics Forum 21 (3), 2002, 299-308.

[15] T. Nishita, S. Takita, E. Nakamae, A display algorithm of brush strokes using Bezier functions, Computer Graphics International 93, pp.244-257, 1993.