# A Real-Time Chinese Calligraphy Creation System

Yanzhou Gong, Ziqiang Ni,Weixing Huang,Jian Wang*,Guigang Zhang
Institute of Automation, Chinese Academy of Science, Beijing, China
{yanzhou.gong, ziqiang.ni, weixing.huang, jian.wang, guigang.zhang}@ia.ac.cn
Corresponding author: Jian Wang (jian.wang@ia.ac.cn)

***Abstract*—In this paper, we design a system that can make calligraphy in real-time by using real Chinese brush. The system consists of two parts: calligraphy stroke generation and ink dispersion. In the calligraphy stroke generation, we use Kalman filter algorithm to deal with the unsmooth points at low speed, and combine arc and linear interpolation method to interpolate points with high speed, then we fill the interpolated points with brush strokes based on experience. In the ink dispersion, modified LBE algorithm was adopted to render initial brush stroke. Finally, our experiments test this system's performance and functionality, it shows that this system works well.**

***Keywords- calligraphy stroke generation; ink dispersion; brush strokes; real Chinese brush***

## I. INTRODUCTION

Nowadays, great progress has been made in computer graphics, due to the efforts of computer science researchers and more powerful computer hardware. Traditional art works which are generally created on real paper with brush has achieved new development by combining with computer science. Western painting art, such as oil painting and watercolor painting, has made remarkable achievements in computer graphics. Also, Chinese calligraphy, one of representative of oriental art, achieves new development owe to the effort of computer researchers, but lots of researchers focus on the result of simulation rather than the process of calligraphy creation. Hard pan with no flexible material, such as digital pen and electronic pen, has been used to make electronic calligraphy creation, experiencers with these tools may feel very difficult to create comparing to real calligraphy brush, which may affect the creation of calligraphy. In our opinion, simulating the real result of Chinese calligraphy is important, but guaranteeing that creators obtain approximately real experience of creating in simulating system is also essential. In this paper, a system focus on the process of Chinese calligraphy is proposed, in this system, creators can use a real Chinese calligraphy brush to make calligraphy work. Moreover, we discussed several key problems about this system. In order to obtain better creating experience, we simplify some details of simulating aim to make system faster.

The rest of the paper is organized as follows. Section 2 summarizes the related research work regarding simulation of Chinese calligraphy. Section 3 introduces the whole system flow. Section 4 shows the generation of footprint by real calligraphy brush. Section 5 presents ink dispersion simulation, implementation and results and future work is presented in Section 6 and 7.

## II. RELATED WORK

In simulation of Chinese calligraphy, a lot of works have been made, mainly concentrating on the generation of footprints by deformation of brush and ink dispersion simulation.

### *A. Generation of Footprints*

In 1986, Strassmann et al.[2], inspired by the traditional Japanese art known as 'sumi-e', proposed a simple modular system, which contains four units: Brush, Stroke, Dip and Paper. Model of brush in this system is described as one-dimensional array of bristles. Footprints generated by this model are not spontaneous due to that system cannot simulate the variety of brush deformation. Then Hsu et al. [3] use skeletal strokes based on 2D-deformation model to make footprints of brush, but footprints image must be sampled in advance and stored in system. At that time, researchers focused on the render of strokes [4].

With the deep development of study, researchers gradually find that the real brush effect is difficult to achieve unless they utilize the principle of the real brush deformation.

In 1997, Lee et al. [5] establish a 3D virtual brush model based on elastic mechanics, some new strokes are obtained comparing to the method of render of strokes. Chu et al. [6] established a 3D virtual brush model based on the brush Dynamics, and solved the motion equation with method of energy minimization. This Model has been applied in a system named 'MoXi'.

### *B. Ink Dispersion Simulation*

Guo et al. [07] proposed a model to simulate the effect of 'Nijimi', in which the flow of ink is a one-dimensional process. Cellular automation are firstly used to simulate Watercolor [08], then Zhang et al. [09] based on 2D cellular automaton establish a simple behavioural model of ink, and it works well when applying this model to a Suibokuga-like rendering of 3D trees, the advantage of this model is simple and easy to control, but spontaneity of ink dispersion cannot simulate well due to the limitation of this model. In 2003, Shi et al.[10] tried to use particle system to simulate Ink dispersion, In this system pick-up edges from input strokes are used to initialize the particle movement, Pseudo-Brown movement is the driving force, this system provided a new idea, but it can only simulate part of ink dispersion and is difficult to commonly use. Di et al. [11] proposed to use genetic algorithm to simulate Chinese ink wash drawing. Cassidy et al. [12] used a Kubelka-Munk compositing model to simulate the optical effect of the superimposed glazes.

978-1-5386-2937-6/17 $31.00 © 2017 IEEE
DOI 10.1109/ISM.2017.105

Van et al. [13] create physically-based system for creating images with watery paint. In 2005, Chu et al. [15] simulated ink dispersion by using Lattice Boltzmann Method.

## III. System Introduction

In our system, users can create calligraphy works on touch screen with real Chinese brush. Firstly, system collects original points generated by moving brush, then it determines how to deal with these points according to the speed of brush, if current speed is below a threshold, Kalman filter algorithm is applied, if speed is high, method contains both linear and circular interpolation is selected, then we make final footprints by applying brush strokes to the interpolated points, after that, we execute the operation of simulation ink dispersion. The flow of whole system shows in Figure 1.

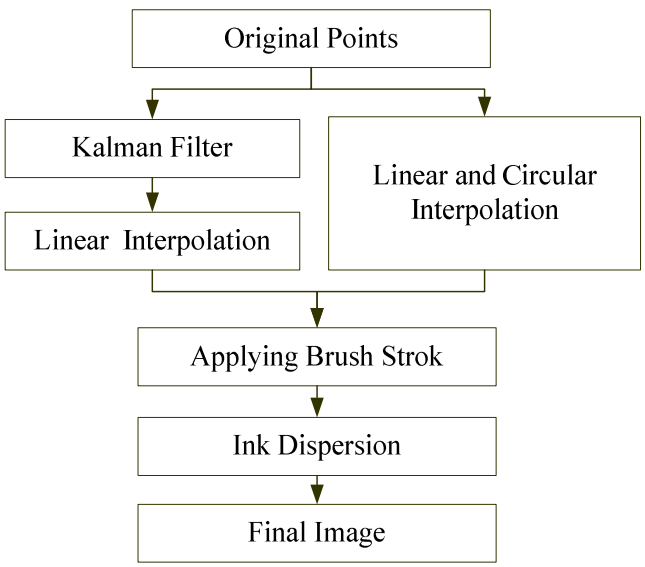



Figure 1. Data process of whole system

## IV. Generation of Footprints

### A. Original Points Preprocessing

There is a prominent problem when we experience Chinese calligraphy in electronic system with real brush: the focus of brush on screen changes continuously, which make the experiencers feel that they cannot control the focus of brush freely, this problem is obvious when brush moves slowly. One of the most possible explanation is unstable deformation of brush, which result in the touching area between screen and brush vary all the time, thus no matter which type of touch screen we select, the original points we collected will contain noise all the time. In Figure 2, we show a contrast result which we separately moving real brush(Figure 2b) and new pencil(Figure 2a) slowly along a straight steel ruler, it is obvious that the points generated by real brush has more noise, In our system, we eliminate this effect by Kalman filter.

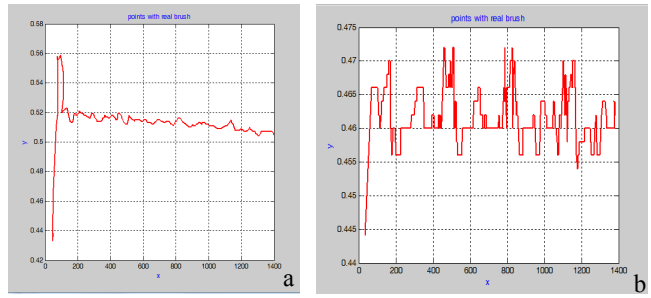



Figure 2. Original points generated by real brush and new pencil

Kalman filter algorithm was firstly introduced by Rudolf Emil Kalman[1] in 1960, it is an optimal recursive data processing algorithm and has been proved effective in many areas such as missile tracking , face recognition and fault diagnosis. Classical iterative equation of Kalman filter shows in (1-5):

$$\hat{F}_k^- = A\hat{F}_{k-1} + Bu_k \tag{1}$$

$$P_k^- = AP_{k-1}A^T + Q \tag{2}$$

$$K_k = P_k^- H^T (HP_k^- H^T + R)^{-1} \tag{3}$$

$$\hat{F}_k = \hat{F}_k^- + K_k(z_k - H\hat{F}_k^-) \tag{4}$$

$$P_k = (I - K_k H)P_k^- \tag{5}$$

Equations (1-2) are prediction equation. A is state transition matrix. B is a system control input matrix. Q is the process noise covariance matrix associated with noisy control [16]. $\hat{F}_{k-1}$ is an update vector of input at time k-1, $\hat{F}_k^-$ is an estimate vector of input at time k, $P_{k-1}$ is an update covariance matrix at time k-1. $P_k^-$ is an estimate covariance matrix at time k. $u_k$ is an input matrix of system.

Equations (3-5) are update equation. $K_k$ is a temporary matrix denote Kalman Gain. H is a transformation matrix that maps the state vector parameters into the measurement domain[16], R is covariance of measurement noise. $z_k$ is the vector of measurements. $I$ is an identity matrix.

We use Kalman filter to process our original point collected at low speed stage. The parameter details of Kalman filter in our system is as follows:

- A is a 4×4 identity matrix.
- We set B to be a 4×4 identity matrix for there is not excitation in our system .
- Q is a noise covariance matrix related about speed of brush moving. We set matrix $Q = w_i * I$, $w_i$ is defined as

$$w_i = \min(M_{max},\ m_0 e^{m_1 * v^2 + m_2 * v + m_3}), \tag{6}$$

- where $M_{max}$ is a upper limit of $w_i$, it can ensure that outputs of Kalman filter change within a certain range. $m_0$, $m_1$, $m_2$ and $m_3$ are fixed system parameter. v indicates speed of brush moving.

When users move the real brush on the touch screen, $w_i$ varies continuously, so does matrix of Q. Q denotes the effect of speed as noise.

- $\hat{F}_{k-1}$ is an update vector of location of points at time k-1.
- $\hat{F}_k^-$ is an estimate vector of location of points at time k.
- $P_{k-1}$ is an $4\times4$ update covariance matrix at time k-1, $P_0 = iniP * I$. The range $100 \leq iniP \leq 1000$ work well in our system.
- $u_k$ is an zero matrix for there is no excitation in our system.
- System matrix H=[[1,0,0,0],[0,1,0,0]].
- R is set to be $4\times4$ identity matrix, because we convince the error of collecting points is so small that it can be ignored.
- We set location of original points as the value of input vector: $z_i = (x_i, y_i, 0,0)^T$, where $x_i$ and $y_i$ is the coordinates of original points .

In the system, we deal with original points at low speed by equation (1-5). The result with and without Kalman filter is presented in Figure 3, where the lines in b and d are generated by linear interpolation, a and c are processed by Kalman filter and linear interpolation, it is obvious that the lines in a and c are more smoother.

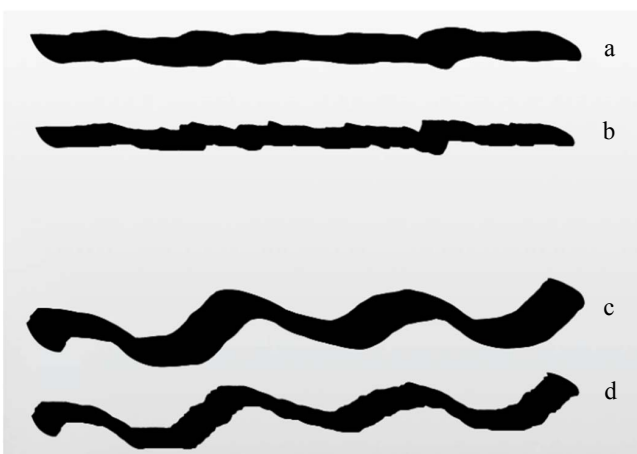



Figure 3. The contrast result of with and without Kalman filter

Another question maybe asked by someone is that why we do not use Kalman filter to process the original points with high speed. Let's look at what will happen if we apply Kalman filter on the points with high speed. In Figure 4, heavy line is predicted by Kalman filter, and thin line is generated by measured points. It's obvious that the distance between predicted points and measured points is far. Explanation for this result is that we set the speed of moving brush as the system process noise. In Kalman filter algorithm, process noise should be as close as possible to obey zero mean multivariate normal distribution. We assume that our system follow this rule at low speed. But it is not suitable for all speed, actually, speed of brush obey no distribution for we can move brush freely.

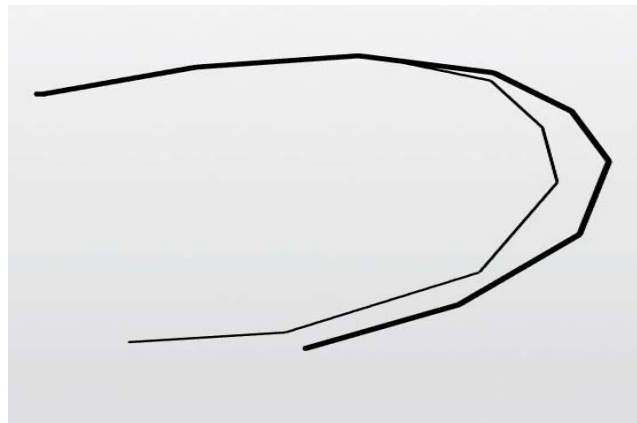


Figure 4. Moving brush quickly with Kalman filter

### B. Smoothing trajectory of brush

We solve the problem of the unsmooth points at low speed by Kalman filter, the other question is that how we handle the rest of points with high speed. It is not acceptable that we simply apply linear interpolation on these points which may affect creation of calligraphy. In Figure 5(a), we obtain broken line at high speed by using linear interpolation. Generally speaking, there is four methods commonly used to interpolate. The first one is NURBS (Non-uniform rational B-splines) curve which is widely used in CNC system [17]. With so many parameters, it is not easy to control its shape accurately. Also, it's time-consuming when calculating its control points. Second curve is Cubic B - spline, which is popular and ideal method to smooth points. In Figure 5(b), we use this method interpolates p0-p3. The dotted line is the result of interpolating p0-p2, solid line is acquired by adding point of p3. We can see that the interpolating curve changes in order to make the curve smooth when new point is added. It can be allowed in some applications that curve changes a bit little, especially the situation of real-time smoothing mouse points for users are not sensitive to the minor adjustment of interpolated curve. But it is unacceptable in our system for ink dispersion simulation will execute twice in both solid and dotted line. The rest of interpolating method is circular and linear interpolation. In our system we use both of them. The strategy is as follows:

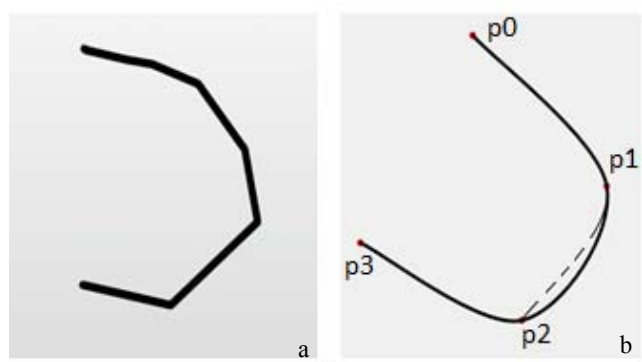



Figure 5. linear and cubic B-spline interpolation

We obtained three sets of continuous points: $T_{i-2}$ , $T_{i-1}$ , $T_i$ . where $T_i$ is the current point, set θ is a included angle to measure the vector $\overrightarrow{T_{i-2}T_{i-1}}$ and $\overrightarrow{T_{i-1}T_i}$. θ ranges from 0 to π . if θ is below a threshold value, we think that users make it intentionally, so interpolates $T_{i-1}$ and $T_i$ with linear interpolation. Circular interpolation will be applied when θ is higher than threshold value. Result using this strategy is presented in Figure 6(a-d).

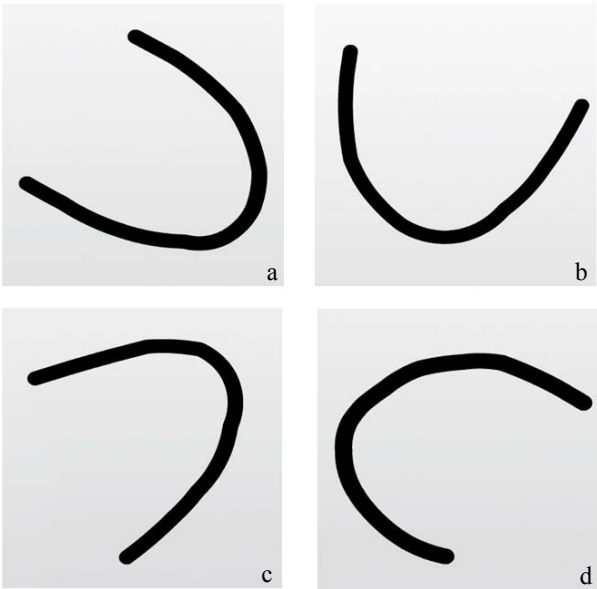



Figure 6. linear and circular interpolation

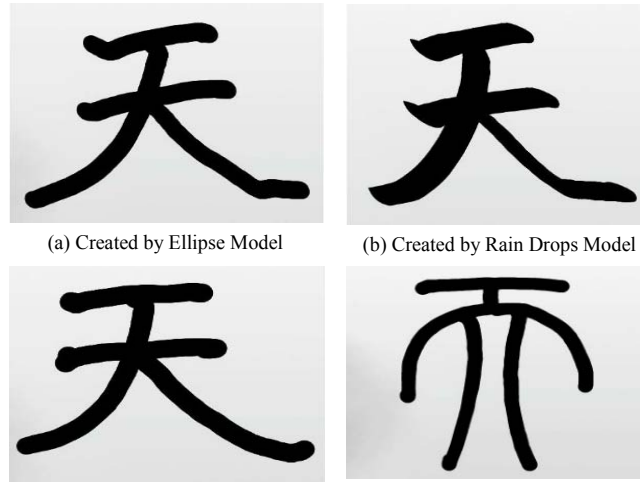
### C. Applying brush stoke

After smoothing and interpolating the collected points, brush stokes should be applied on each interpolated points. In our system, instead of establishing physical model, a virtual brush model based on experience [18] is being used for decreasing the calculation time of generating strokes on CPU. There are three commonly used brush models, which show in Figure 7.

(a) Ellipse Model (b) Rain Drops Model (c) Circle Model

Figure 7. Three experience model

The Rain Drops Model is generally used for its shape is quite similar to the calligraphy stroke produced by real brush on the paper at the beginning, but this mode appears to make inconsistent stokes when moving the brush in different direction (see in Figure 8b). Circle Model can produce even stoke and it fits to make strokes of seal calligraphy (see in Figure 8d). Effect of Ellipse Model is between Rain Drops Model and Circle Model. Works with these three models are presented in Figure 8.



(a) Created by Ellipse Model (b) Created by Rain Drops Model

(c) Created by Circle Model (d) Created by Circle Model

Figure 8. Calligraphy works with three models.

## V. INK DISPERSION SIMULATION

### A. LBE Method Introduction

In our system, we use method based on fluid mechanics to simulate ink dispersion, typically, there are two methods, one is Navier-Stokes(N-S)[12], the other is Lattice Boltzmann equation (LBE) [19]. We prefer to use LBE than N-S because of two considerations [14]: (1) N-S equations are second-order partial differential equations, while LBE contains a set of first-order partial differential equations. (2) N-S solvers with global data communication and nonlinear convective term is difficult to utilize GPU, while LBE method fits to utilize GPU and can well run with computer hardware upgrading. So our system uses LBE Method to simulate ink dispersion.

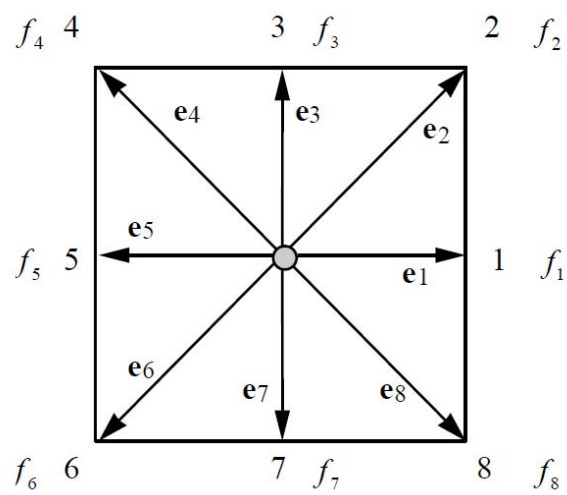



Figure 9. D2Q9

Boltzmann equation based on molecular motion theory and statistical mechanics is a numerical method of simulating flow field. It derives from micro-dynamic, which regarded the macroscopic movement of fluid as average statistical result of massive micro-particles. Lattice Boltzmann model greatly simplifies the Boltzmann original equation for it converts original continuous velocity to discrete structure and facilitates the calculation of collision equation. Method of LBE is high-accurate, stable, simple and easy to be programed.

LBE not only can simulate the two-dimensional fluid motion, but also three-dimensional fluid motion, In our system, simulation of ink only needs two-dimensional fluid motion, so D2Q9, classical model of LBE, is selected, where 'D2' represents that the model is used in two-dimensional space , 'Q9' denotes the number of discrete velocity, the model is presented in Figure 9[14], where $\mathbf{e_1}$-$\mathbf{e_8}$ denotes direction vector of particle moving, $\mathbf{e_0}$ is a zero vector corresponding to the stationary particles[13]. $f_i$ is defined as particle distribution function, which can be simply regarded as the density moving along the direction of $\mathbf{e_i}$, particle distribution function[13] is

$$f_i(\mathbf{x} + \mathbf{e_i}\Delta t) = (1 - \omega)f_i(x, t) + \omega f_i^{(eq)}(x, t) , \qquad (6)$$

Equilibrium distributions is

$$f_i^{(eq)} = w_i \left\{\rho + \rho_0 \left[\frac{3}{c^2}\boldsymbol{e_i} \cdot \mathbf{u} + \frac{9}{c^4}(\boldsymbol{e_i} \cdot \mathbf{u})^2 - \frac{3}{2c^2}\mathbf{u} \cdot \mathbf{u}\right]\right\} , \qquad (7)$$

Detailed description of parameters in (6-7) can be found in paper [13],

*B.* Place of Modifying

In our ink simulation, we modify the LBE method which is introduced by Chu [13]. There have three main differences between our system and Chu:

*1) Removing the movement of pigments*

In the method of Chu, ink dispersion simulation consists of both water and pigments, and the simulating time of pigments movement almost same to water flow. In order to accelerate the algorithm iteration rate of ink flow, we just use water movement to simulate whole ink dispersion in this system.

*2) Paper normalization*

In the method of Chu, a variety of papers is added to the simulation of ink dispersion in order to obtain the spontaneous result. In our system, we use one texture paper to simulate the real paper, which can greatly simplify algorithm.

*3) Boundary trimming.*

In general, we do not use the screen resolution (output image) as the size of ink simulation due to the limitation of GPU performance. Actually, the size of iteration region is smaller than the output image. A problem called contour jaggies (or blocky named in [13]) in image processing is proposed, which shows in Figure 10, it is obvious that the boundary is unsmooth. Chu enhances the boundary by overwriting pixel value of edge. In our system, we select an efficient method by comparing several filters.

Figure 10. Output image with contour jaggies

*C.* Flow of LBE in our system

We select a frame of LBE algorithm to introduce the simulation of ink dispersion in our system, the flow is as follows:

- Step1 Collision

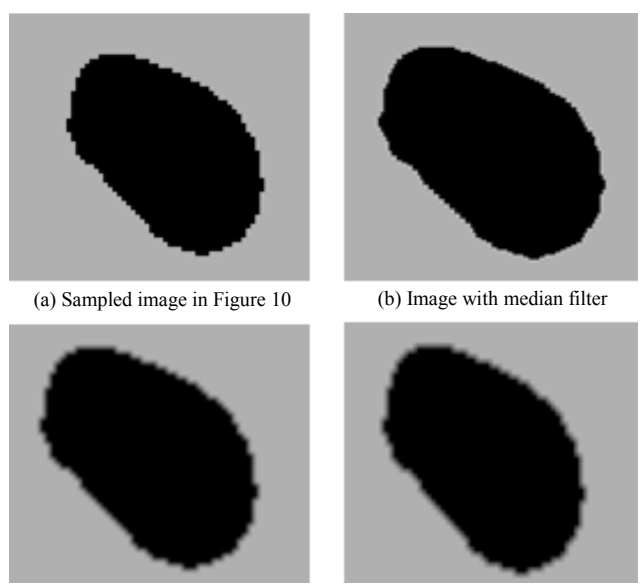
  We mainly use (6) to simulate the collision.

- Step2 Streaming

  We use (7) to simulate the streaming.

- Step3 Ink Deposition

  The amount of ink coming from brush is updated to the flow layers.



(a) Sampled image in Figure 10 (b) Image with median filter

(c) Image with gauss filter (d) Image with mean filter

Figure 11. Images with filtering algorithm

- Step4 Image Mapping

  Small iteration image is enlarged to a bigger one.

- Step5 Boundary Trimming

  In this step, we devote to suppressing the contour jaggies. Since our system is real-time, we use some simple filters such as median filter, mean filter and gauss filter. According our experiments, median filter is more efficient than others. A example showing in Figure 11 can prove this conclusion, a is a sampled image in Figure 10, we apply three filters on image b, c and d separately. It is obvious that the process of mean filter and gauss filter make boundary blur, and median filter works well, thus we use median filter to trim boundary in our system.

## VI. IMPLEMENTATION AND RESULTS

We conduct all test on a machine with an i7-4790 CPU and AMD Radeon R5 240 GPU with 1G video memory, and program our system with C Language, OpenGL and Cg, the performance of ink simulation is shown in TABLE 1 (column names of table is explained in [15]), according to our observation frame rate above 50 can achieve a real-time effects. So our system with the highest rate beyond 100 can reach it in most cases. Works created by this system with soft brush is presented in Figure 12, it can be observed that our system can create smooth calligraphy strokes with real Chinese brush.

TABLE I. PERFORMANCE OF INK SIMULATION

| *Sim.Resol.* | *d-scale* | *Data Type* | *Fr./sec.* |
|---|---|---|---|
| $512^2$ | 1 | 16 | 130 |
| $512^2$ | 1 | 32 | 54 |
| $512^2$ | 2 | 16 | 88 |
| $512^2$ | 2 | 32 | 40 |

Notes: Measured without Kalman filter

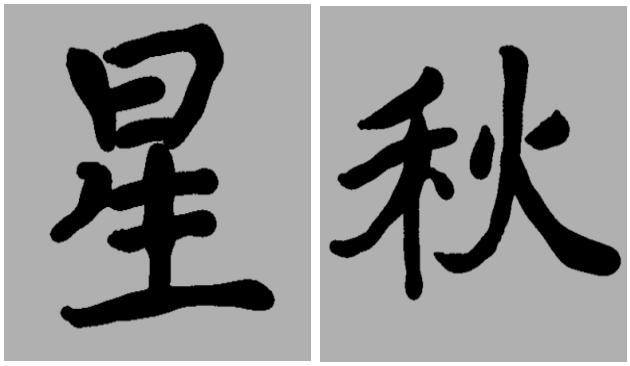


Figure 12. Calligraphy works

## VII. FUTURE WORK

In this paper, we propose a system to create Chinese calligraphy with real brush, and solve several critical problems. And there still has some improving space in our system:

- Touch screen which offers the touching length of x and y direction needs to be filtering too.
- In current system, we simplify some steps in order to improve system efficiency, but the quality of output image is declining, and it should be improved in the future.


## ACKNOWLEDGMENT

We would like to thank all colleagues and students who helped for our work and thank the support: the Support Program of the National '12th Five-Year-Plan of China' under Grant No.2015BAK25B03.


## REFERENCES


[1] Kalman, Rudolph Emil. "A new approach to linear filtering and prediction problems." Journal of basic Engineering 82.1 (1960): 35-45.

[2] Strassmann, Steve. "Hairy brushes." ACM Siggraph Computer Graphics. Vol. 20. No. 4. ACM, 1986.

[3] Hsu, Siu Chi, and Irene HH Lee. "Drawing and animation using skeletal strokes." Proceedings of the 21st annual conference on Computer graphics and interactive techniques. ACM, 1994.

[4] Chu, Siu Hang. Making digital painting organic. Hong Kong University of Science and Technology (Hong Kong), 2007.

[5] Lee, Jintae. "Physically-based modeling of brush painting." Computer Networks and ISDN systems 29.14 (1997): 1571-1576.

[6] Chu, NS-H., and Chiew-Lan Tai. "An efficient brush model for physically-based 3D painting." Computer Graphics and Applications, 2002. Proceedings. 10th Pacific Conference on. IEEE, 2002.

[7] Guo, Qinglian, and Tosiyasu L. Kunii. "Modeling the diffuse paintings of 'Sumie'." Modeling in Computer Graphics. Springer Japan, 1991. 329-338.

[8] Small, David. "Modeling watercolor by simulating diffusion, pigment, and paper fibers." Proceedings of SPIE. Vol. 1460. 1991.

[9] Zhang, Qing, et al. "Simple cellular automaton-based simulation of ink behaviour and its application to suibokuga-like 3d rendering of trees." The Journal of Visualization and Computer Animation 10.1 (1999): 27-37.

[10] Yongxin, Shi. "Graphical simulation algorithm for Chinese ink wash drawing by particle system." Journal of Computer Aided Design and Computer Graphics 15.6 (2003): 667-672.

[11] Di, Zang Chao Tang. "GENETIC ALGORITHMS-BASED SIMULATION OF DIFFUSION BORDER OF CHINESE INK DRAWING [J]." Computer Applications and Software 2 (2009): 087.

[12] Curtis, Cassidy J., et al. "Computer-generated watercolor." Proceedings of the 24th annual conference on Computer graphics and interactive techniques. ACM Press/Addison-Wesley Publishing Co., 1997.

[13] Van Laerhoven, Tom, and Frank Van Reeth. "Real‐time simulation of watery paint." Computer Animation and Virtual Worlds 16.3‐4 (2005): 429-439.

[14] Yu, Dazhi, et al. "Viscous flow computations with the method of lattice Boltzmann equation." Progress in Aerospace Sciences 39.5 (2003): 329-367.

[15] Chu, Nelson S-H., and Chiew-Lan Tai. "Moxi: real-time ink dispersion in absorbent paper." ACM Transactions on Graphics (TOG). Vol. 24. No. 3. ACM, 2005.

[16] Faragher, Ramsey. "Understanding the basis of the kalman filter via a simple and intuitive derivation [lecture notes]." IEEE Signal processing magazine29.5 (2012): 128-132.

[17] Luo, Fuyuan, Youpeng You, and Juan Yin. "Research on the algorithm of NURBS curve bidirectional optimization interpolation with S-type acceleration and deceleration control." Jixie Gongcheng Xuebao(Chinese Journal of Mechanical Engineering) 48.5 (2012): 147-156.

[18] MI, Xiao-Feng, et al. "An Experience Based Virtual Brush Model [J]." Journal of Computer Research and Development 8 (2003): 013.

[19] Wei, Xiaoming, et al. "Lattice-based flow field modeling." IEEE Transactions on Visualization and Computer Graphics 10.6 (2004): 719-729.