# **Chapter 2: Literature Review**

Three sections make up the watercolor painting simulation [7]. First, several methods of applying water and pigments to the paper are employed. Second, a calculation is made to determine how the water and pigment move in response to different forces during discrete time intervals. Finally, the image can be produced in a variety of ways given a specific discrete-time simulation state. These three elements can be summarized as fluid simulation, pigment motion, and pigment composition. Several ways for simulating watercolor pigment will be covered in this chapter.

## **2.1 Fluid Simulation**

Both the direction of water flow and the hues that it carries are determined by fluid modeling. It serves as the foundation for the watercolor simulation. Using the 2.2 pigment component techniques, we can ascertain the distribution of the pigments based on the motion of the water and then render the finished product from it. The Navier-Stokes equations, Stam's stable fluid, and Lattice-Boltzmann equations are the three main methods used to simulate the movement of water applied to the paper surface.

1. Navier-Stokes equation
2. Stam’s stable fluid
3. Lattice-Boltzmann equation

## **2.2 Pigment Composition**

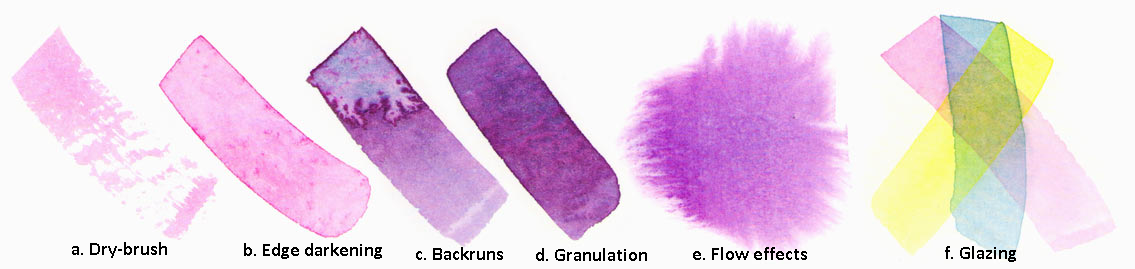
Color pigments are synthetic materials employed in real life. Any pigmented object's color must be rendered while taking into account the physical interactions between its pigmented surfaces. The standard red, green, and blue (RGB) triplet is a simple way to specify color in a graphical setting. The RGB method is only appropriate for additive colorants, like colored light, so this approach is far from realistic. The CMY color model and Kubelka-Munk color model are two alternate methods for rendering colored pigment.

1. CMY Color system
2. Kubelka-Munk Color Model

## **2.3 Watercolor Simulation**

Small *et al.* published the first paper related to the simulation of watercolor painting in 1991. They proposed a way to simulate watercolor effects using a system of cellular automata on the Connecting Machine [7]. In this model, each cell needs to know only itself and its immediate environment. Small *et al.* suggest that future work could be to define new kinds of paper that have extremely specific properties not available in nature. The work of Small *et al.* lays the foundation for watercolor simulations for following investigations, even though the outcomes generated by this model remain short of realism.

Based on Small’s watercolor simulation model, Curtis *et al.* improved the cellar automation to simulate the fluid flow and pigment dispersion of watercolor [3]. This improved method adopts a more sophisticated three-layer paper mode, a more complex shallow water simulation, and a more faithful rendering and optical compositing of pigmented layers based on the Kubelka-Munk model [3]. Figure 3 presents the simulated watercolor effects created by Curtis *et al.*. Because their solution is resolution-dependent, generalizing to a resolution-independent model for watercolor simulation is an important goal for future work.They achieved impressions of real watercolor behavior, but due to the complexity of the simulation model, they were unable to achieve a fully interactive system.

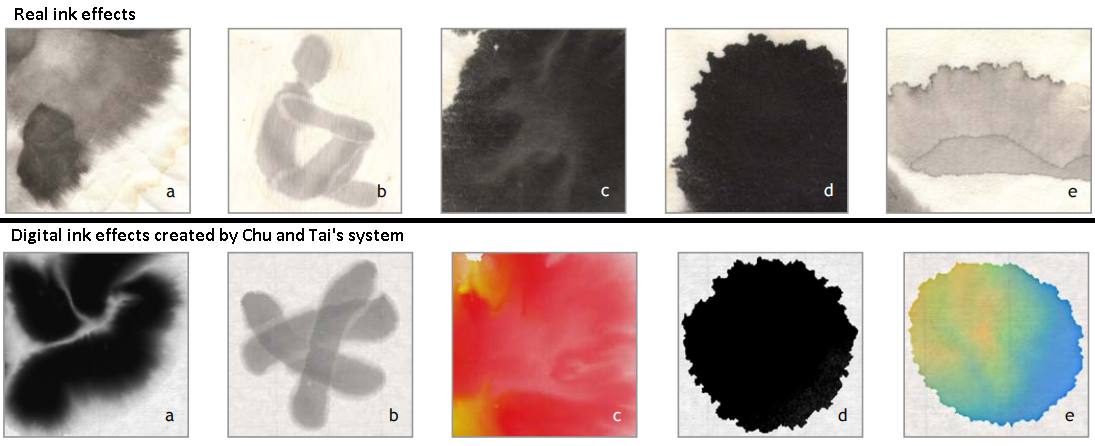


*Figure 3. Simulated watercolor effects created by Curtis et al. system.*

All the above methods to simulate fluid dynamics used the Navier-Stokes equation, which describes the behavior of fluid flow in the continuum approximation. Van Laerhoven and Van Reeth [6], however, utilized the work of Stam to describe a number of fast and stable procedures to simulate fluid flow using implicit solvers [11], [12]. In addition to the different fluid simulation solver, they further improved the three-layer canvas model for their real-time watercolor simulation. Their canvas consists of three active layers, similar to Curtis *et al.* model of canvas, and an unlimited number of passive layers, which contain previously drawn strokes that have dried and no longer participate in the simulation, except when the canvas is rendered. Although their simulation model is able to produce 22 frames per second, it has not been demonstrated that the algorithm can handle high resolutions on commonly available hardware since the system requires high computational loads and resources.

Another way to simulate the pigment-water solution fluidity is the Lattice Boltzmann equation, which is a more detailed description of the behavior of a gas. The watercolor simulation works done by Chu and Tai and Oh *et al.* use the Lattice Boltzmann equation to simulate the fluid movement.

Chu and Tai present a physically-based method for simulating ink dispersion in absorbent paper for art creation purposes [8]. In addition to the usage of the Lattice Boltzmann equation for ink flow simulation, they utilized both CPU and GPU for these computational-intensive simulations, with the overall system frame rate of 44 frames per second. Figure 4 depicts a comparison of real ink effects (top) versus digital ink effects created by Chu and Tai’s system. Since they limited their dispersion simulation to three pigments at a time with RGBA texture, one of the improvements suggested is to employ the Kubelka-Munk model for simulating optical blending.



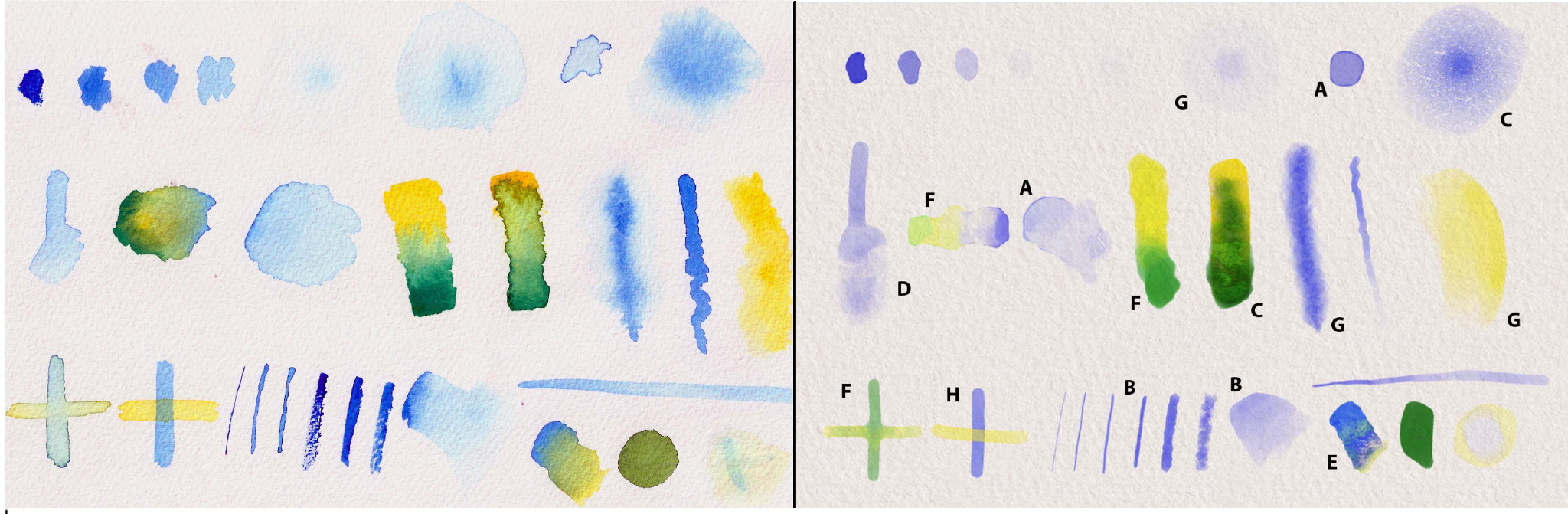
*Figure 4. A comparison of real ink effects (top) versus digital ink effects created by Chu and Tai’s system. (a) Feathery pattern. (b) Light fringes. (c) Branching pattern. (d) Boundary roughening. (e) Boundary darkening.*

Oh *et al.* presents a novel watercolor painting drawing system which can work even on low powered computing machines, such as tablet PCs [9]. Similar to Chu and Tai’s system, their system solves the Navier-Stokes equations for fluid flow in the Lattice Boltzmann method (LBM). In addition, their system deals only with pigmented regions and not the entire drawing canvas, which accelerates the simulation calculation. Figure 5 exhibits two watercolor paintings generated by Oh *et al.* system. However, due to the usage of LBM for fluid flow simulation, detailed fluid behavior is not simulated in the system, which makes the simulation not realistic as real watercolor painting.



*Figure 5. Watercolor painting by Oh et al. system*

DiVerdi *et al.* [10] presents a procedural vector based algorithm for generating watercolor-like dynamic paint behavior in a lightweight manner. Their formulation uses a particle-based model of pigment flow, rather than grid based, which allows for rendering at arbitrary resolutions and is fast to calculate. As the authors mentioned in the paper, the method is not sufficient to describe realistic watercolor painting. Figure 6 displays the comparison of similar strokes made with real watercolor paints (left) versus their algorithm (right). Moreover, it is hard to guarantee that the method can also show interactive performance on recent slate devices with high resolution.



*Figure 6. A comparison of similar strokes made with real watercolor paint (left) versus DiVerdi et al. algorithm right). Paper texture is added to our results for comparison purposes. The strokes are chosen to showcase a variety of characteristic watercolor behaviors, including edge darkening (A), nonuniform pigment density (B), granulation (C), rewetting (D), back runs (E), color blending (F), feathering (G), and glazing (H). Strokes exemplifying particular effects have been labeled with the corresponding letter.*