Real-time 3D Fire Simulation Using a Spring-Mass Model

Murat Balci* Hassan Foroosh
Computational Imaging Lab., Computer Science, University of Central Florida
{balci,foroosh}@cs.ucf.edu

Abstract

We present a method for real-time simulation of 3-dimensional fire inspired by an old mechanical trick known as the "silk torch". Motivated by the proven illusive effect of silk torch, we model the kinematics of a flames by a mass-spring system, and its turbulant visual dynamics by texture sequencing with variable speeds and transparencies that depend on the speed of the vaporized fuel. The approach allows for incorporating external forces such as the gravity, and the wind force for added realism. Also, a specific characteristic of our method is that any object inserted in a flame can be modeled simply as an external force in the mass-spring system, making real-time interactions with fire a simple addition to the overall system. While the approach maintains an extremely low computational cost, these flexibilities increase the realism of our 3D fire by allowing for real-time control and the interactivity, which is a highly desirable requirement in applications such as augmented reality.

1 Introduction

Fire can be defined as a rapid, self-sustaining oxidation process of combustible gases ejected from a fuel. It is one of the most important natural phenomena, and still a challenging one to simulate with computer graphics in real-time and in a realistic fashion. Its simulation can be used in a wide variety of important applications such as special effects for movies, scientific visualization, augmented reality, games, etc.

Fire simulation techniques in computer vision/graphics can be broadly classified into two categories. In the first category are the 2-dimensional methods that generate a fire effect using a sequence of textures. The sequence, often rendered on a billboard, is an animation of either real or synthesized procedural textures [6, 17, 19]. In the most sophisticated versions the dynamics of fire are simulated using Perlin noise or other similar approaches [16]. The advantage of these methods is the simplicity of implementation in 3D environments, which has made them widely available. Despite the fact that these texture effects can be associated with the statistical model of fire turbulence [20], they often do not blend realistically in a 3D environment. On the other hand, they require to be always rendered either orthogonal to the viewing direction, or alternatively mapped directly to the rendered image plane. Furthermore, they do not provide the possibility of interacting with fire, e.g. in a virtual/augmented reality environment.

In the second category are the 3-dimensional methods [2,7,8,10,13–15,21]. The primary advantage of these methods is the added realism in terms of (i) their convincing visual appearance and dynamics, (ii) modeling of propagation and expansion, and (iii) the possibility of interacting with other virtual or real objects. However, different algorithms provide a trade-off between these three aspects. Central issues in this category are the representation and modeling of the burning gases in the flames, the model of their thermodynamics, and the choice of the rendering approach. Combination of various representations and dynamics have been proposed in the literature. Some examples include the earlier work of Reeves [18] using the animation of particle systems, the use of the physical model of the emission and transmission [10], evolution of fire front particles on polygonal meshes [14], tomographic modeling and simulation [8] the use of chain of particles [2], the evolution of blobs based on a diffusion model [21], and the physics-based modeling of the vaporized fuel using the Navier-Stokes equations, implicit surfaces, and the level set method [15].

In practice, the model used for simulating a 3-dimensional fire and its dynamics imposes also a trade-off between the realism and the rendering speed. For rendering, most successful methods rely on either surface crawling, implicit surfaces, volume rendering, or voxelization with ray-tracing. Although, the visual appeal can be improved by increased accuracy of simulating the actual physics of the fire, the computational cost can make the simulation prohibitively inaccessible to real-time applications. For instance, the seminal work of Nguyen et al. [15] is known to provide some of the most compelling visual effects based on the actual physics of the fire in both modeling and rendering, but the approach is reported to provide a prohibitively slow rendering of one frame per 5 minutes [15].

*The support of the I2Lab in the form of research fellowship is gratefully acknowledged.

The approach proposed in this paper provides an optimal trade-off between realistic visual appearance and real-time rendering. The idea has its root in an old physical/mechanical arrangement known as the silk torch, which is proven to have strong illusive effects on visual perception of fire.

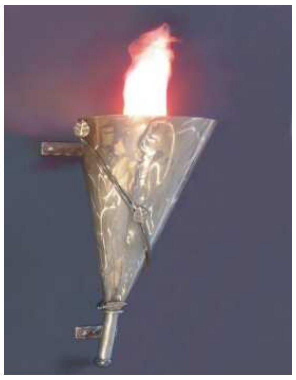

Figure 1: A mechanical silk torch.

2 Silk Torch

Silk torch is an old trick used originally by magicians in stage performances to produce the illusive effects even at close proximity for fooling the spectators. Its origin has also been traced back to the ancient Greek theatre. Essentially, silk torch is made of mechanically moving pieces of silk fabric that together with a source of wind, the gravity, and a source of light produce astonishingly realistic flickering effects of a true fire.

In recent years, there has been many applications of this old trick in interactive theme parks. For instance, Disney used orange lights on a fan-blown sheet of plastic to simulate leaping flames in some of their rides. Various forms of silk torch are now also available as commercial products for decorative purposes. Figure 1 shows an example of such product, and illustrates the effectiveness of this mechanical fire.

Our work is inspired by this ornamental equipment, but since we are designing a computer silk torch, the actual kinematics and the visual effects of our silk fabric is based on the physics of a flame. We call this a “silk flame”. For instance a large fire may comprise of several silk flames, each with its associated internal and external forces. In the next section, we describe how these internal and external forces are set according to the physics of the combustion, and how the kinematics of these silk flames varies in time, based on the physics of fire. This would give a silk flame its astonishingly realistic look, when rendered with texture sequencing and transparencies that reflect the characteristics of a burning fuel.

3 Modeling the Kinematics

We simulate the our silk flame using a mass-spring model (see Figure 2). A mass-spring model comprises of a set of nodes that are connected to adjacent ones by springs [12]. The nodes and the springs undergo displacements and deformations based on the forces applied to them. A silk flame has the additional property that its kinematics are controlled by the physics of combustion process. Combustion researchers have studied fire’s physical properties in details. Fuel type (solid, liquid, gaseous), and the type of oxidizers are some of the main parameters that determine the kinematics of the flame [3]. On the other hand, heat is the measure of the molecular activity within the flames, which affects the speed of molecules. These factors define the combustion process, which in turn defines the internal forces applied to the flame particles. External forces may include the wind, the gravity, and interactions with external objects (whether flammable or not). Our model of the silk flame incorporates all these factors.

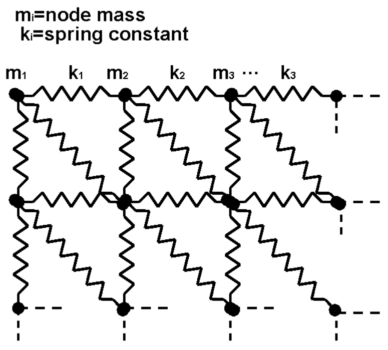

Figure 2: The mass-spring model.

However, it is important to note that in our model, we are not simulating the exact physics of the flame particles, but rather our goal is to emulate the motion and the kinematics of the flame, when it is sustained by a fuel. In particular, in our case the flame is not truly subject to buoyancy since the nodes of the mass-spring model do not break away from the flame. However, they undergo displacements due to the heat energy E released by combustion. Assuming the conservation of energy, E also represents the internal energy of our mass-spring model, which depends only on the current state of the silk flame, i.e. inter-node distances and velocities. The gradient of E with respect to the state of a node (i.e. its position) defines the net internal force applied to that node, which is internally represented as the sum of damping, shearing and bending forces associated with the springs.

The net force $\mathbf{f}_i^t$ applied at any time t to the node i is the sum of all internal and external forces, and defines the acceleration of the node, and hence its instantaneous velocity

$\mathbf{v}_i^t$ and position $\mathbf{x}_i^t$. The state of the silk flame at time t can then be formally represented as a single vector

$$\mathbf{S}^t = [\mathbf{x}^t, \mathbf{v}^t] \tag{1}$$

where $\mathbf{x}^t = [\mathbf{x}_1^t, ..., \mathbf{x}_n^t]$ and $\mathbf{v}^t = [\mathbf{v}_1^t, ..., \mathbf{v}_n^t]$, where n is the number of nodes in a silk flame. Differentiating this state vector with repect to time yields

$$\begin{aligned} \dot{\mathbf{S}}^t &= [\dot{\mathbf{x}}^t, \dot{\mathbf{v}}^t] && (2) \\ &= [\mathbf{v}^t, \mathbf{a}^t] && (3) \end{aligned}$$

where $\mathbf{a}^t = [\mathbf{a}_1^t, ..., \mathbf{a}_n^t]$ is the acceleration vector due to the forces.

Given the net force applied to this system, we have from Newton's second law

$$\dot{\mathbf{S}}^t = [\mathbf{v}^t, \mathbf{M}^{-1}\mathbf{f}^t] \tag{4}$$

where $\mathbf{f}^t = [\mathbf{f}_1^t, ..., \mathbf{f}_n^t]$ is the net force vector, and

$$\mathbf{M} = \begin{bmatrix} \mathbf{M}_1 & \cdots & 0 \\ \vdots & \ddots & \vdots \\ 0 & \cdots & \mathbf{M}_n \end{bmatrix} \tag{5}$$

where the 3×3 sub-matrix $\mathbf{M}_i = \text{diag}(m_i, m_i, m_i)$, with m_i representing the mass of the node i.

Equation (4) is an ordinary differential equation. Given the state of the mass-spring system at time t, our goal is to formally determine $\mathbf{S}^{t+\delta t}$, where δt is a positive short time interval. Using Euler's method

$$\dot{\mathbf{S}}^{t+\delta t} = \mathbf{S}^t + \delta t \dot{\mathbf{S}}^t \tag{6}$$

Leading to

$$\begin{aligned} \mathbf{x}^{t+\delta t} &= \mathbf{x}^t + \delta t \mathbf{v}^t && (7) \\ \mathbf{v}^{t+\delta t} &= \mathbf{v}^t + \delta t \mathbf{a}^t && (8) \end{aligned}$$

In practice these equations can be solved sequentially, as follows

$$\begin{aligned} \mathbf{v}^{t+\delta t} &= \mathbf{v}^t + \delta t \mathbf{a}^t && (9) \\ \mathbf{x}^{t+\delta t} &= \mathbf{x}^t + \delta t \mathbf{v}^{t+\delta t} && (10) \end{aligned}$$

Although easy to implement, Euler's method introduces some stability issues [1]. In particular, in practice the solution is stable if δt is smaller than the natural period of the system. In a mass-spring model, the latter is given by

$$T \simeq \pi \sqrt{\frac{m_{\text{min}}}{k_{\text{max}}}} \tag{11}$$

where m_{min} is the smallest mass in the system, and k_{max} is the largest stiffness coefficient among all springs.

A proper simulation typically requires high stiffness (i.e. high values of spring stiffness coefficients). As a result, the updating time intervals need to be very small in order to retain the stability of the system. To alleviate this problem often higher order integration such as fourth-order Rung-Kutta is used, or implicit integration [1, 4, 11] and hybrid implicit-explicit methods [5] have been proposed. However, these solutions are rather very costly, and in the case of the implicit methods may also lead to solving non-linear problems.

Our goal, however, is to obtain a real-time simulation. A good trade-off between stability and the computational cost can be obtained if we update the velocities indirectly, rather than using a coupled set of differential equations as in the Euler's method. A solution to this problem can be found in molecular dynamics literature [9], which naturally applies to our problem. For this purpose, note that by applying the Taylor series twice as follows:

$$\mathbf{x}^{t+\delta t} = \mathbf{x}^t + \delta t \mathbf{v}^t + \frac{1}{2}\delta t^2 \mathbf{a}^t + \mathcal{O}(\delta t^3) \tag{12}$$

$$\mathbf{x}^{t-\delta t} = \mathbf{x}^t - \delta t \mathbf{v}^t + \frac{1}{2}\delta t^2 \mathbf{a}^t + \mathcal{O}(\delta t^3) \tag{13}$$

and by adding both sides, we get

$$\mathbf{x}^{t+\delta t} = 2\mathbf{x}^t - \mathbf{x}^{t-\delta t} + \delta t^2 \mathbf{a}^t \tag{14}$$

This latter equation allows to update the positions of the nodes in a spring-mass system, without requiring an explicit computation of their velocities. Of course, if needed the velocities can still be approximated using

$$\mathbf{v}^{t+\delta t} = \frac{\mathbf{x}^{t+\delta t} - \mathbf{x}^t}{\delta t} \tag{15}$$

In molecular dynamics, this approach is referred to as the Verlet integration method [9]. It provides two advantages over other existing methods. First and foremost, since the velocities are computed only implicitly, they are always consistent with node displacements and the process is more stable. Secondly, it provides a substantial gain in the computational cost, allowing for real time processing and simulation.

The above scheme provides a very simple and efficient way of updating the state of our spring-mass system. In the next section, we describe how it can be exploited to provide the natural flickering effect of a silk torch or a real fire.

4 Modeling Visual Dynamics

Fire may be viewed as a dynamic exchange of matter and energy sustained by the chemical reactions of a fuel source with oxidizers. This dynamic behavior is what makes its simulation challenging from computer graphics point of view. In the previous section, we suggested that the kinematics of flames can be simulated with a mass-spring model. However, since this model is only emulating the true kinematics, very much similar to a mechanical silk torch, we need to augment its visual appearance with texture, and transparency. Furthermore the texture and the transparency must vary dynamically in time and space according to the characteristics of the combustion process.

Fire is a blackbody radiator. The gaseous medium ejected from the burning fuel modifies the intensity field of light by emission, absorption, and scattering. Under equilibrium, the emission of the gas is proportional to the blackbody emission [3].

$$Q_\lambda = E_\lambda \frac{2h}{\lambda^5 c}\left(\exp\left(\frac{hc}{\lambda k T}\right) - 1\right)^{-1} \tag{16}$$

where E_λ reflects the contribution of the wavelength λ, h is the Planck's constant, c is the speed of light, k is the Boltzmann constant, and T is the temperature.

Although, a direct use of such physics-based model for rendering the texture dynamics has been suggested in the literature [15, 21], we prefer not to adopt this line of approach due to its high computational cost. In particular, we suggest that important key effects of this model can be closely emulated by using an approach similar to the mechanical silk torch.

Essentially, if we observe a real fire, we notice that under equilibrium, the wavelength and intensity of light vary locally within each flame, and also change dynamically in time. The variations in wavelength result in various colors from the blue core to yellow, and red in different parts of the flame. This can be readily captured by either textures from a real fire, or by procedural textures synthesized by an algorithm.

The dynamic variations of the intensity field, however, cannot be done realistically by simple texture mapping. For this purpose, we need to investigate how the variations in the intensity field depicted by (16) can be emulated by a silk flame. The silk flame, of course, undergoes motion and deformation according to the internal and external forces as described in the previous section. This, effectively, introduces some level of realism to the texture. However, a fixed texture on a silk flame would look more like a waving fabric. To give a compelling fire effect, we therefore need to let the texture evolve. As is shown in Appendix A, the black body emission in (16) has a first order Taylor series approximation of the form

$$Q_\lambda \simeq E_\lambda \frac{2}{3}\frac{kM}{c^2\lambda^4 R} v_{rms}^2 - E_\lambda \frac{h}{\lambda^5 c} \tag{17}$$

In other words, to a first order approximation the blackbody emissions are proportional to the temperature, or equivalently proportional to the square of the rms molecular speed. It is therefore reasonable to assume that to a first order approximation the fire texture should also evolve proportional to the square of the rms molecular speed. This corresponds to our intuition that a faster moving fire has also a faster evolving texture and light. Of course in our simulation the rms molecular speed is computed simply by taking the rms of nodal speeds in our mass-spring model.

To emulate this effect, we used texture sequencing with variable transparencies: after initializing the texture of each silk flame using a randomly chosen image from a set of N frames of a fire sequence, we let the texture of each flame cycle through the sequence at the speed defined by the overall kinetic energy of the silk flame, which in turn is determined by the internal and the external forces in the mass-spring system. We found that to maintain realism only a small number of texture images, e.g. $N = 8$ would be sufficient, independently of the number of silk flames in the simulated fire. Note that the texture sequence can either be obtained from a real fire, or alternatively generated in real-time using a procedural fire texture synthesis that evolves in time either linearly with the temperature profile of the flame or quadratically with its speed (advection).

5 Rendering

The proposed method can be considered in two phases: initialization and the rendering cycle. During initialization, basic initial settings such as lighting parameters, texture-mapping parameters, loading the textures, enabling the vertex array functionalities, and the initialization of silk flames are performed. To obtain a realistic waving effect for the silk flames, a wind field is associated with each flame. Each flame wind field includes procedural turbulent effects to obtain realistic results. Other than that, there are no restrictions on the other external forces in the environment such as additional external winds, the gravity, etc.

Rendering Cycle is composed of three steps:

- Updating flame vertices and other flame parameters: First, update the silk flame meshes, according to the total forces effecting on each vertices. Secondly, update the texture of each flame. The texture update speed should be proportional to the actual speed of the flame. Third, update the flame wind fields to cause the waving effect of the flames. Note that unlike a billboard the turbulent motion of the flame plays an important role in the realism of the results. This is done by randomely modifying the vector components, which are orthogonal to the flame sheets.

- Draw opaque environment before rendering the flames or other transparent objects.

- Render the flames. We used vertex arrays, which made the implementation faster, and the rendering process easier as mass-spring meshes. For rendering we disabled the depth mask calculations, before rendering these vertex arrays due to the transparent nature of flames. The basic algorithm is quite simple, and easy to implement. Finally, we generated alpha values based on the original RGB color values of the textures, leading to more control over transparency.

6 Results and Performance

The computer silk torch approach that we described in the previous sections does not only generate astonishingly realistic fire simulations and behaviors, but also presents various advantages over other existing 3-dimensional methods. In particular, the speed of its real-time operation is superior (to our best knowledge) to all existing methods. We obtained 80 frames per second without using any GPU programming, or assembly code, and by relying only on standard OpenGL capabilities. No ray tracing is required either for visual quality or 3D effects. The silk flames are readily rendered by standard OpenGL functions, simply by dumping the coordinates of the nodes in the mass-spring model and their corresponding texture map coordinates at a given time step.

Of course, like most 3-dimensional techniques one can interact with the fire (in our case in real-time). For instance, it is possible to put an object in the flame, and let it catch fire. For this, since we are using a mass-spring model, we do not need to perform collision detection, which is typically highly costly. Instead, we represent the intruding object as a force field, which of course can modify the state of the mass-spring models in silk flames.

The user can set the wind field, the gravity (if he/she wishes so, e.g. for special effects), and also the parameters for internal forces to control the strength of connectivity of the mass-spring nodes, e.g. pressure, oxygen level, fuel, etc. All parameters can be modulated in real-time, and all simulations run in real-time. The rendering speed is about 80 frames per second at full screen display of 1024×768, on a PC with Intel processor of 2.3 GHz. This rendering time includes all other objects in the fire's environment, and the illumination caused by the fire.

We simulated fires with different fuel types, and experimented with various combustion speeds, wind effects, and external object interactions. Figure 3 shows different frames of a burning candle with occasional wind action. Figure 4 shows an example of a burning solid fuel, with average rms molecular speed, using 8 real fire textures. An interesting feature of our approach is that very much similar to physics based methods [15, 21], external objects such as a log can be immersed in the flames with realistic results of fire rolling up from the base of the log. To achieve this effect, we simply turn off collision detection with silk flames. This also allows different silk flames to penetrate into each other, for instance in the case of multiple sources of fuel (e.g. several burning logs) piled on top of each other. Note, however, that we have not simulated the burning and consumption of the fuel (i.e. logs would not get consumed and converted to ashes).

Figure 6 shows an example of different transparency levels and change in flame edges. And Finally, Figure 7 shows an example of interaction with an external object, where a match is approached to the flame and caught fire, and then moved back. Again realistic wind effect are seen in the sequence. As described earlier the introduction of the match is modeled as an external force field that can penetrate in the silk flame of the candle, and affect the state of the mass-spring system.

7 Conclusion

We have developed a 3-dimensional fire simulation technique inspired by the old magicians' trick of silk torch. In addition to its compelling visual realism, a computer torch, as we proposed herein, provides the possibility of incorporating some of the physics of fire. The physics of combustion is in fact only emulated with our method to provide photo-realism and interactivity. The technique can be used in real-time applications while providing high quality visual effects for applications such as mixed- or augmented-reality, where visual-fidelity plays an important role in users belief of immersion in the environment. Real-time realistic rendering is achieved without any GPU or assembly language programming at about 80 frames per second for a full screen of 1024×768 on a 2.3GHz PC.

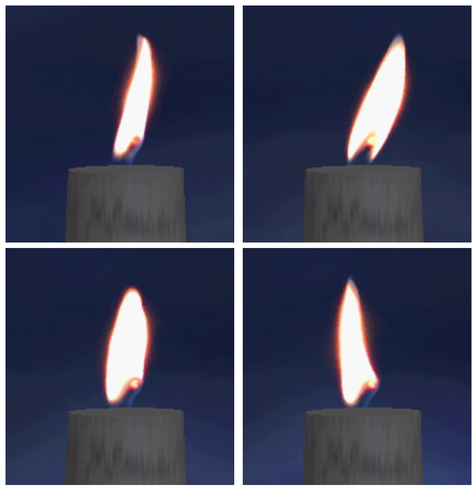

Figure 3: Illustration of candle fire and wind.

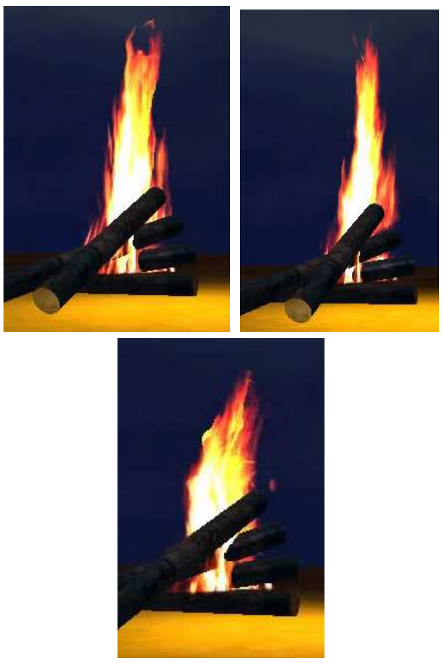

Figure 4: Illustration of a camp fire using 8 real fire textures.

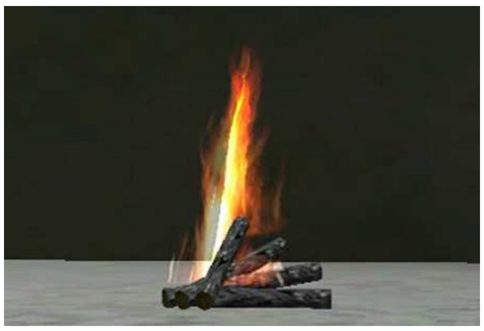

Figure 5: Different transparency level and change in flame edges.

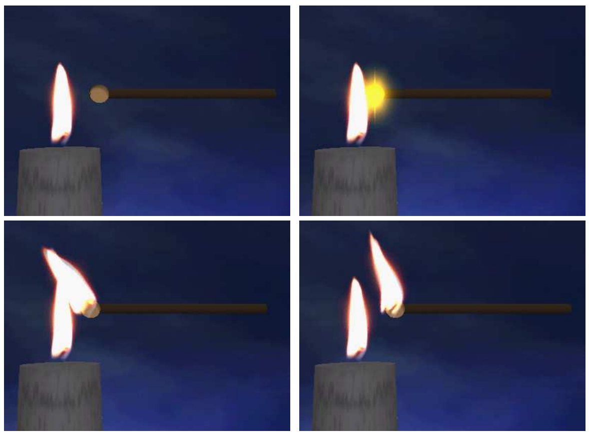

Figure 6: Candle fire lighting a match.

Appendix A: Taylor Approximation of Balck Body emmission

The combustion process generates and ejects hot gases into air that get accelerated under internal and external forces. The kinetic theory relates the pressure P and the volume V to the average molecular kinetic energy. Using the ideal gas law, we get

$$PV = nRT = \frac{2}{3}N\left(\frac{1}{2}mv^2\right) \tag{18}$$

where n is the number of moles, R is the universal gas constant, N is the number of atoms, m is the molecular mass, and $v = |\mathbf{v}|$ is the speed.

The average translational kinetic energy of the molecules is then deduced from above using the Boltzman distribution, which leads to the following familiar equation in thermodynamics for the kinetic energy

$$E_{avg} = \frac{1}{2}mv^2 = \frac{3}{2}kT \tag{19}$$

From this, one can then get the root mean square (rms) molecular speed

$$v_{rms} = \sqrt{\frac{3kT}{m}} = \sqrt{\frac{3RT}{M}} \tag{20}$$

where M is the molar mass.

A careful inspection of (16) shows that for a given wavelength, the black-body emission is only a function of the temperature. Using a Taylor series expansion of (16) with respect to $\frac{1}{T}$, we get

$$\begin{aligned} Q_\lambda &\simeq E_\lambda \frac{2h}{\lambda^5 c}\left(\frac{\lambda k}{hc}T - \frac{1}{2} + \mathcal{O}(T)\right) && (21) \\ &= E_\lambda \frac{2k}{\lambda^4 c}T - E_\lambda \frac{h}{\lambda^5 c} && (22) \end{aligned}$$

which upon substituting from (20), yields

$$Q_\lambda \simeq E_\lambda \frac{2}{3}\frac{kM}{c^2\lambda^4 R}v_{rms}^2 - E_\lambda \frac{h}{\lambda^5 c} \tag{23}$$

Therefore to a first order approximation the black-body emissions are proportional to the temperature, or equivalently proportional to the square of the rms molecular speed. It is therefore reasonable to assume that to a first order approximation the fire texture should also evolve proportional to the square of the rms molecular speed. This corresponds to our intuition that a faster moving fire has also a faster evolving texture and light.

Appendix B: Prodedural Texture Synthesis

The approach that we used is as follows. Using two arbitrary seed images from a real fire sequence a synthetic sequence of arbitrary length is generated. For this purpose, we model the motion of flame betwen the seed images using the following probabilistic approach. Let I^1 and I^2 denote the two seed images. At any pixel (x, y) we define the following probabilities for the synthesized image I_t:

$$\mathrm{pr}(I^t_{xy}|I^1_{x'y'}, I^2_{xy}) = \frac{1}{\sqrt{2\pi}\sigma} \exp\left(-\frac{(I^1_{x'y'} - I^2_{xy})^2}{2\sigma^2}\right) \tag{24}$$

where pixel (x', y') is in some neighborhood of (x, y). Since the three color channels are orthogonal, we may assign to each channel-color of pixel (x', y') a probability according to (24). Call these $p^r_{x'y'}$, $p^g_{x'y'}$, $p^b_{x'y'}$ for the RGB channels. Then $p_{x'y'} = p^r_{x'y'}p^g_{x'y'}p^b_{x'y'}$. This probability models our belief of which of the neighboring pixels (x', y') is more likely to move to position (x, y) based on observing the second image I^2. In this way the most probable motion field is estimated based on two observed images, and used for evolving one of the seed images. For real-time texture synthesis, the magnitude of motion vector can b modulated using the internal energy (i.e. the r.m.s. speed of the mass-spring system). Once a new image is generated, it replaces one of the earlier seeds and the whole process can be repeated indefinitly to generate arbitrary fire textures. Figure 7 shows an example of the output of the algorithm.

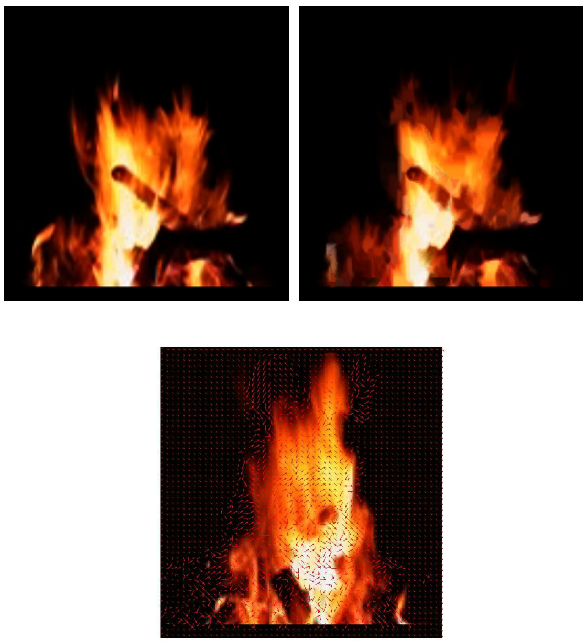
Figure 7: Left: a seed image, right: a synthesized image, bottom another synthesized imaged with superimposed estimated motion field.

References

[1] D. Baraff and A. Witkin. Large steps in cloth simulation. In *SIGGRAPH '98: Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, pages 43–54. ACM Press, 1998.

[2] P. Beaudoin, S. Paquet, and P. Poulin. Realistic and controllable fire simulation. In *GRIN'01: No description on Graphics interface 2001*, pages 159–166. Canadian Information Processing Society, 2001.

[3] L. de Goey and D. Roekaerts. *Lecture Notes of the J. M. Burgerscentrum Course on Combustion*. Eindhoven University of Technology, Department of Mechanical Engineering, 2003.

[4] M. Desbrun, P. Schroder, and A. Barr. Interactive animation of structured deformable objects. In *Proc. of the 1999 conference on Graphics interface '99*, pages 1–8, 1999.

[5] B. Eberhardt, O. Etzmu, and M. Hauth. Implicit-explicit schemes for fast anima-tion with particles systems. In *Proc. of Eurographics Workshop on Computer Animation and Simulation*, pages 137–151, 2000.

[6] D. S. Ebert, F.K. Musgrave, D.P., K. Perlin, and S. Worley. *Texturing and Modeling: A Procedural Approach*. Morgan Kaufmann Publishers Inc., 2002.

[7] R. Fedkiw, J. Stam, and H.W. Jensen. Visual simulation of smoke. In *SIGGRAPH '01: Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, pages 15–22. ACM Press, 2001.

[8] S.W. Hasinoff and K.N. Kutulakos. Photo-consistent 3d fire by flame-sheet decomposition. In *Proc. ICCV*, pages 1184–1191, 2003.

[9] W. Huang and B. Leimkuhler. The adaptive verlet method. *SIAM J. Sci. Comput.*, 18(1):239–256, 1997.

[10] M. Inakage. A simple model of flames. In *CG International '90: Proceedings of the eighth international conference of the Computer Graphics Society on CG International '90: computer graphics around the world*, pages 71–81. Springer-Verlag New York, Inc., 1990.

[11] Y.-M. Kang, J.-H. Choi, H.-G. Cho, and D.-H. Lee. An efficient animation of wrinkled cloth with approximate implicit integration. *The Visual Computer*, 17(3):147–157, 2001.

[12] C.K. Koh and Z. Huang. A simple physics model to animate human hair modeled in 2d strips in real time. In *Eurographics*, pages pp. 127–138, 2001.

[13] A. Lamorlette and N. Foster. Structural modeling of flames for a production environment. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 729–735. ACM Press, 2002.

[14] H. Lee, L. Kim, M. Meyer, and M. Desbrun. Meshes on fire. In *Proceedings of the Eurographic workshop on Computer animation and simulation*, pages 75–84. Springer-Verlag New York, Inc., 2001.

[15] D.Q. Nguyen, R. Fedkiw, and H.W. Jensen. Physically based modeling and animation of fire. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 721–728. ACM Press, 2002.

[16] K. Perlin. Improving noise. In *SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques*, pages 681–682. ACM Press, 2002.

[17] J Portilla and E P Simoncelli. A parametric texture model based on joint statistics of complex wavelet coefficients. *Int. J. of Computer Vision*, 40(1):49–71, 2000.

[18] W. T. Reeves. Particle systems: a technique for modeling a class of fuzzy objects. *ACM Trans. Graph.*, 2(2):91–108, 1983.

[19] J. Rhoades, G. Turk, A. Bell, A. State, U. Neumann, and A. Varshney. Real-time procedural textures. In *SI3D '92: Proceedings of the 1992 symposium on Interactive 3D graphics*, pages 95–100. ACM Press, 1992.

[20] G. Sakas. Modeling and animating turbulent gaseous phenomena using spectral synthesis. *Vis. Comput.*, 9(4):200–212, 1993.

[21] J. Stam. Interacting with smoke and fire in real time. *Commun. ACM*, 43(7):76–83, 2000.