Noname manuscript No.
(will be inserted by the editor)

Geometry-based Control of Fire Simulation

Yi Hong · Dengming Zhu · Xianjie Qiu · Zhaoqi Wang

Received: date / Accepted: date

Abstract High-level control of fire is very attractive to artists, as it facilitates a detail-free user interface to make desirable flame effects. In this paper, a unified framework is proposed for modeling and animating fire under general geometric constraints and evolving rules. To capture the fire projection on user's model animation, we develop a modified closest-point method (MCPM) to handle dynamic situations while maintaining the robustness of the closest-point method. A control blue core (CBC) is designed and generated automatically from the fire projection at each time step. It translates the geometric constraints and the user-specified evolving rules into implicit control conditions. Our L-Speed function leverages CBC's shape information and conducts the large-scale motion of fire, leaving the basic physically-based model to refine simulation details. The experimental results show the effectiveness of our method for modeling fire propagation along complex curves or surfaces, or forming a flaming shape and following its motion.

Keywords Fire simulation · Modified closest-point method · Control blue core · L-Speed

Y. Hong · D. Zhu · X. Qiu · Z. Wang
Virtual Reality Laboratory, Institute of Computing Technology, Chinese Academy of Sciences, Beijing, China
E-mail: hongyi@ict.ac.cn

Y. Hong
Graduate School of the Chinese Academy of Sciences, Beijing, China

D. Zhu
E-mail: mdzhu@ict.ac.cn

X. Qiu
E-mail: qxj@ict.ac.cn

Z. Wang
E-mail: zqwang@ict.ac.cn

1 Introduction

In fire animation, the high-level motion control is very useful for providing flame effects specified by artist's animations and evolving rules, e.g., a spreading fire along a curve in a shape of butterfly, and a flaming running horse. These effects may be implemented by the fuel-placed method or produced by modern fluid simulation systems. However, it will impose heavy workload on artists, especially when they make fire animation follow the motion of an object. So it is necessary to develop a unified system to free artists from implementation details. While artists only focus on control instructions over the whole procedure of animations, the system follows their instructions and emulates the natural flames.

Our fire simulation model is designed specifically to address the above problem, using an automatical fire control technology to simulate the fire's dynamic behaviors, which are subject to general geometric constraints and flame evolving rules. To achieve this, three key issues deserve our attention: (1) capturing the fire fronts on general geometric constraints, like the complex static or moving curve or surface; (2) translating the geometric constraints and the fire evolving rules into the implicit control conditions; (3) improving the physically-based fire model to work under the implicit control conditions.

Most of previous approaches [1, 14, 23, 32] dealt with fire spreading on static surfaces, and had special codes or instructions to detect and address the problems of evolving fire fronts, such as self-intersections (the swallow tail problem). To solve the above problem more efficiently and expand the fire propagation on general geometric constraints, we use the level-set equations on still or moving curve or surface to track the fire fronts. Although the closest-point method [17]

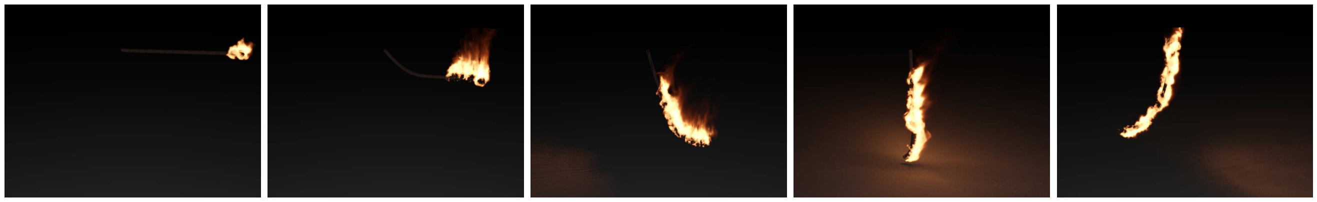

Fig. 1 Fire spreading along a rope

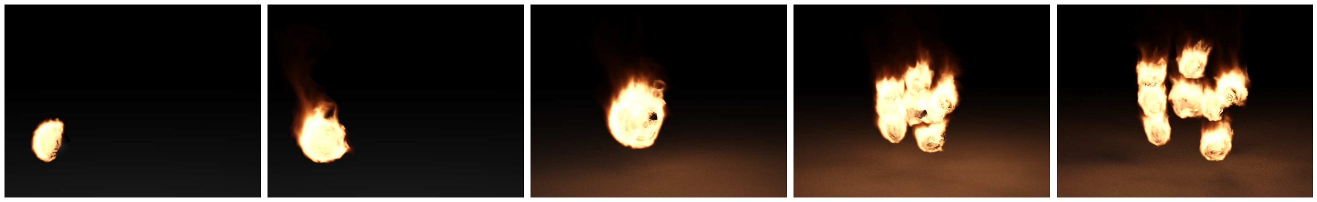

Fig. 2 Moving ball catching fire and then blasting

is a robust solver for level-set equation on general surface, it works only in stationary situations. In order to handle moving scenarios, we extend it to the modified closest-point method (MCPM).

Unlike the existing methods for controlling fluid animation in smoke and water, which obtained the control forces acting on momentum equation [6, 27, 29], our method turns to the 3D shape morphing by exploiting our basic fire model. A virtual control blue core (CBC, Fig. 3) is proposed to get the anticipated state of blue core (the reaction zone of fire). The CBC is auto-generated in each iteration with the information from the geometric constraints and the user-specified fire evolving rules, and works as the high-level control to change the large-scale motion of fire. The fine-scale details are refined by the basic fire model from the physically-based simulation [20].

In order to combine the implicit control condition from CBC with the basic fire model, a L-Speed function is defined based on the theory of level-set metamorphosis [3]. This function carries the CBC's shape information, and serves as a leading component of the blue core's velocity to make the current blue core approximate the corresponding CBC as much as possible. As implicit control conditions generated by CBC feed the physically-based model, our system can automatically and easily generate the expected fire effects.

This paper presents an overall framework for control over fire animation under general geometric constraints and user-given evolving rules. Our contributions are threefold:

- Presenting the MCPM for solving the level-set equation on the moving curve or surface to capture the fire fronts on general geometric constraints.
- Proposing the CBC to automatically control the high-level motion of fire, which is steered by artist-designed geometric topology and fire evolving rules.
- Defining the L-Speed to make blue core approximate its CBC, with the combination of implicit control conditions and the basic fire model.

The remainder of the paper is organized as follows. Sect. 2 gives an overview of the related literature. Our fire simulation model is presented in Sect. 3, and the implementation is discussed in Sect. 4. Finally, we provide experiment results in Sect. 5 and the paper is concluded in Sect. 6 with a summary and future directions.

2 Related work

Simulating fire phenomena is widely used in various areas, such as entertainment, visual simulation and fire prevention. Extensive studies have been conducted, and some have achieved stunning results [9, 10, 20].

The earliest computer graphics fire model, presented by Reeves [25], used a particle system to animate the fire on a planet, and many particles were required to achieve good visual effects. Based on the particle-based simulation, Chiba et al. [5] described a two-dimensional visual simulation method to simulate the spread of fire and the appearance of smoke produced in an incomplete combustion. Stam and Fiume [28] used a finite difference scheme to simulate the creation, spread and extinguishing of fire in 3D space. In order to simulate the unsteady motion of an infinitely thin premixed flame, the surface was tracked in [24] through the explicit representation of connected marker points. In contrast, Nguyen et al. [20] used level-set to track the location of blue core (the combustion reaction area for unburnt fuel undergoing in hot products). In the same year, Melek et al. [19] simulated and controlled the fire phenomenon by describing the motion of a 3-gas system, oxidizing air, fuel gases, and exhaust gases. In addition, the Lattice Boltzmann Model (LBM) was utilized in [30] to simulate physically-based equations describing the fire evolution and its interaction with the environment, and the interactive rendering frame rates was enabled by the texture splat primitives.

To handle the combustion processes like ignition, extinction, and heat generation, Ishikawa et al. [11] introduced physical and chemical rules for simulating fire spread. Burning solid into gases was modeled by Losasso et al. [16], who focused on the forced fuel moving outwards through setting the divergence in the burning region to a positive value when solving for the pressure. Hong et al. [9], based on [20], made use of the detonation shock dynamics (DSD) framework in the fire simulations to obtain interesting features, the flame wrinkling and cellular patterns. Recently, a gradient-based spreading model [15] was proposed to simulate the movement of fire on thin-shell objects with their crumpling and deformation. The novel model in [10] combined the coarse particle grid simulation with a very fine, view-oriented refinement stage based on GPU, and produced gigantic, high-resolution resulting images of fire.

Related to our work of fire simulation with surface constraint, Perry and Picard [23] propagated flames on polygon meshes through a velocity-spread model from combustion science. In order to accelerate computation and get more intuitive control over the simulation without compromising realism, a new method for simulating fire propagation on polygonal meshes was introduced in [1]. Lee et al. [14] evolved the geodesics on polyhedral surfaces, and simultaneously simulated multiple fire fronts. They focused on forwarding the fire fronts on surface triangles or polygons, while Zhao et al. [32] simulated the fire evolution in a different way, using an enhanced distance field on volumetric data sets, and generated the external air velocity field though the LBM approach to affect the movement of the fire front. While these methods rely on complex processes to capture the fire fronts on the static surface, our method overcomes the drawback and naturally tracks the fronts on general interfaces, even the moving curve or surface.

As for the level-set equation on complicated interfaces, an Eulerian formulation for solving partial differential equations (PDE) on a moving interface was studied in [31], in which the interface was passively converted by the fluid flow, whose velocity field was given or computed by combining with the immersed interface method (IIM) in real applications. The recently proposed closest point method [17] was quite attractive for solving level-set equations on stationary surfaces, due to the fact that it was a robust technique for evolving interfaces on very general surfaces, and more importantly it held the advantages of the level-set method itself [22], which not only automatically handled self-intersecting interfaces, but also enabled multiple fire fronts to merge and break apart naturally. However, the closest-point method only handled the cases in static situation. Our method extended it to MCPM, so that we can simulate fire under moving geometric constraints.

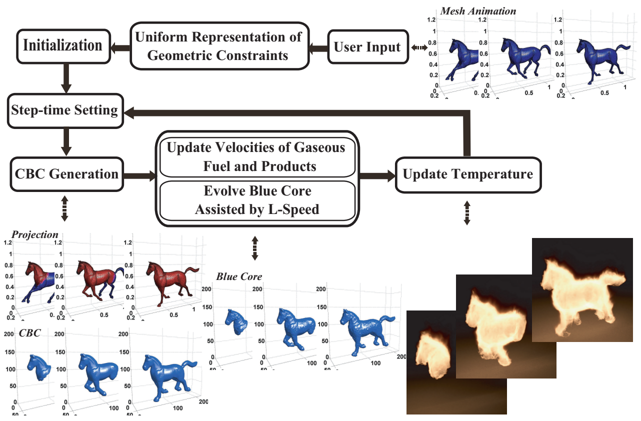
Mesh Animation
Initialization
Uniform Representation of Geometric Constraints
User Input
Step-time Setting
CBC Generation
Update Velocities of Gaseous Fuel and Products
Evolve Blue Core Assisted by L-Speed
Update Temperature
Projection
Blue Core
CBC

Fig. 3 Block diagram of our geometry-based control of fire simulation

Our CBC, working as a guider, is a little similar to the controllable smoke animation [26]. The difference is in that our CBC is auto-generated at every iteration step and translates the given geometric constraints and evolving rules into a sequence of implicit constraint conditions.

3 Fire simulation model

This section describes our geometry-based fire simulation model. The flow chart is given in Fig. 3. In the preprocessing stage, the user-specified geometric constraint on fire is translated into a uniform representation. For example, the curve paths or animated mesh models are represented by the closest-point functions. After the initialization and step-time setting, we capture the fire's projection which evolves on still or moving geometric constraints with the given evolving rules, and generate the corresponding CBC automatically. The shape information of the CBC carried by L-Speed leads the fire's large-scale motion, while the basic physically-based model takes charge of the small-scale details. After updating the blue core, velocities of gaseous fuel and products, and fire temperature, the process is then repeated for the next time step. Additionally, the fire

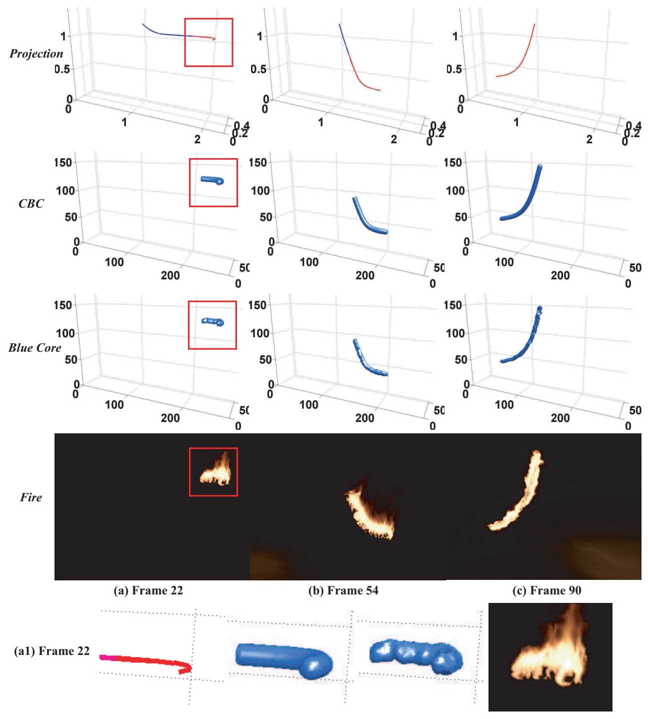

Fig. 4 Projection, blue core, CBC and corresponding fire along a rope

temperature computed by the simulation is used for rendering.

3.1 Uniform representation of geometric constraints

Many fire phenomena interacting with objects can be translated into geometry-based control problems, among which the curve and surface constraints are common. However, these constraint conditions are complex and much different from each other, like the still or moving NURBS curve for a 3D fire path, static or animated mesh models. Thus, a uniform representation for geometric constraints becomes necessary and significant. It makes the computations in subsequent stages free from simulation topology and easy for implementation.

The closest-point function of a curve or surface is an alternative approach to meet the needs of uniform representation. It is widely used in many applications, such as the shape registration [2], the production of volume data from triangular meshes [12], the solution of differential equations on surfaces [17], and so on. Meanwhile, computing the closest-point has been well researched for the NURBS curve [4] and the triangulated surface [12].

For geometric constraint $\mathcal{G}$, the closest-point function, $cp : R^n \rightarrow R^n$, is defined as: given a point $\boldsymbol{x}$, $cp(\boldsymbol{x})$ is a point on $\mathcal{G}$ closest in Euclidean distance to $\boldsymbol{x}$. If the geometric constraint is static, the $cp(\boldsymbol{x})$ keeps the same for every frame, while under moving constraints, we need to calculate and store $cp_0(\boldsymbol{x}) \sim cp_M(\boldsymbol{x})$ from frame 0 to frame M.

In addition, if the artist defines the fuel type $c(\boldsymbol{x})$ on the geometric constraint, we represent it in the uniform way, $c(\boldsymbol{x}) = c(cp(\boldsymbol{x}))$. Moreover, the occupied voxels by the geometric constraint, denoted as $solid(\boldsymbol{x})$, should be calculated through the voxelization technology [13].

Our uniform representation of the geometric constraints eliminates the difference among constraint conditions and simplifies the computation implementation. It enables our model to simulate the fire on distinct curves and surfaces, still or moving, or evolving into a flaming shape while following its motion.

3.2 CBC's generation

The high-level control is desirable in a production environment since animators are interested in modifying the large-scale motion of the fire. We use CBC to undertake this work, which is auto-generated following the instruction of the user-specified evolving rules. This will mitigate the workload of artists especially when they make the fire effects by animating models. We generate CBC in two steps: (1) we capture the fire's projection on general geometric constraints, which forwards under the given rules; (2) from the region surrounding the projection, a surface is extracted as the border of CBC.

3.2.1 Fire's projection

The interface, the border of fire's projection, consists of points (the projection's border on curve) or curves (the projection's border on surface). To locate the projection, we capture the interface on general geometric constraints, and identify which side of the interface a point lies on.

Implicit interface representations hide the information about what makes up the interface. For example, we can designate the $\varphi(\boldsymbol{x}) = 0$ isocontour as the interface, and then simply by looking at the local sign of φ, we determine that $\boldsymbol{x}$ is inside the interface when $\varphi(\boldsymbol{x}) < 0$, outside the interface when $\varphi(\boldsymbol{x}) > 0$. The implicit interface may move in an externally generated velocity field, or in the normal direction, and so on. This depends on the evolving rules specified by users. So a time-dependent implicit interface φ under still or moving geometric constraint $\mathcal{G}(t)$, could be the function of time t, position $\boldsymbol{x}$, and other variables in Eq.(1):

$$\varphi_t = F(t, \boldsymbol{x}, \varphi, \nabla_{\mathcal{G}(t)}\varphi, \nabla_{\mathcal{G}(t)} \cdot \frac{\nabla_{\mathcal{G}(t)}\varphi}{|\nabla_{\mathcal{G}(t)}\ \varphi|}) \tag{1}$$

where $\bigtriangledown_{\mathcal{G}(t)}$ denotes the gradient intrinsic to the static or moving geometric constraint $\mathcal{G}(t)$.

The fire's projection, namely the burning area, is the difference between the current burnt area and the extinct area. If to track the burnt area by $\mathit{\Gamma}_1 = \{\boldsymbol{x}|\varphi_1(\boldsymbol{x}) \leq 0, \boldsymbol{x} \in \mathcal{G}(t)\}$, and the extinct area by $\mathit{\Gamma}_2 = \{\boldsymbol{x}|\varphi_2(\boldsymbol{x}) \leq 0, \boldsymbol{x} \in \mathcal{G}(t)\}$, we get the projection $\mathit{\Gamma} = \mathit{\Gamma}_1 \setminus \mathit{\Gamma}_2$. So two implicit interfaces, φ_1 and φ_2, with respective evolving rules are used. Of course, if there is no extinct area, or to simulate burning into a shape, only one interface is needed.

3.2.2 Projection capturing

To capture the fire's projection, it is natural to leverage the level-set method because of its attractive properties. Besides, the closest-point method [17] is applicable to level-set equations, which serves as a robust technique for evolving interfaces on very general surfaces. However, they only treat the stationary surfaces. So we need to improve the method for dynamic situations.

The level-set equation on the moving curve or surface satisfies an evolution PDE on the kinetic interface (mainly refers to the curve and surface in the following). For example, the projection on a moving interface both shifts with the interface (provided by other simulated or animator-designed animation) and spreads along the interface by the given evolving rules. So the function F in Eq. (1) can be decomposed into two components: the function E, which corresponds to the evolution of the projection along the interface; and the function L, which corresponds to the motion of the interface.

The function E is relevant to the evolving rules, see Sect. 4 for details. And the function L is designed approximately for the moving interface, due to the fact that the level-set morphing [3] smoothly changes the model of one object into the model of another object, which could be regarded as a sort of special mapping function between the two consecutive frames. The following function L depicts the approximate motion from the current frame to the next frame m:

$$L(d_m(\boldsymbol{x})) = -\frac{d_m(\boldsymbol{x})}{D_m(\boldsymbol{x})}\frac{\nabla d_m(\boldsymbol{x})}{|\nabla d_m(\boldsymbol{x})|} \tag{2}$$

Here $d_m(\boldsymbol{x})$ is the signed distance function of the frame m for the moving curve or surface. Infinitely thin curve or surface will be treated as the one with particular thickness. And $d_m(\boldsymbol{x}) > 0$ indicates the inside of curves or surfaces, $D_m(\boldsymbol{x})$ is a characteristic distance function for adjusting the influence of $d_m(\boldsymbol{x})$.

Combining the function L with the closest-point method, the MCPM is to capture the fire's projection on moving curve or surface. The semi-discrete (discrete in time, continuous in space) MCPM with forward Euler time-stepping is as below:

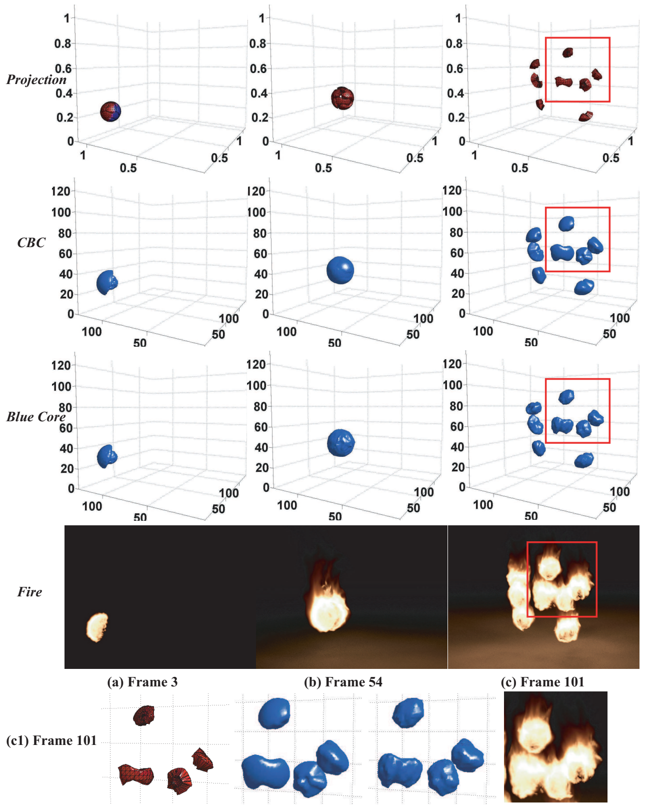

Fig. 5 Projection, blue core, CBC and corresponding fire on a ball

1. Extend the initial conditions of the curve or surface (frame 0) to each point $\boldsymbol{x}$ in the computational domain, i.e., $\varphi^0(\boldsymbol{x}) = \varphi^0(cp_0(\boldsymbol{x}))$.

At time t_n, perform the following steps to advance to time t_{n+1} for the frame m, i.e., $\varphi^{n+1}(\boldsymbol{x}, m) = MCPM$ $(\varphi^n(\boldsymbol{x}), m, ...)$:

2. Perform a forward Euler time-step:

$$\begin{aligned}\tilde{\varphi}^{n+1}(\boldsymbol{x}) = \varphi^n(\boldsymbol{x}) + \triangle t\ (\ &E(t_n, cp_m(\boldsymbol{x}), \varphi^n, \nabla\varphi^n, ...) \\ &+ L(d_m(\boldsymbol{x}))\ )\end{aligned} \tag{3}$$

3. Perform a closest-point extension for each point $\boldsymbol{x}$:

$$\varphi^{n+1}(\boldsymbol{x}) = \tilde{\varphi}^{n+1}(cp_m(\boldsymbol{x})) \tag{4}$$

See the projection in Figs. 3, 4 and 5 for the tracking results of the MCPM.

3.2.3 CBC computation

The projection $\mathit{\Gamma}$ is a set of curves or surfaces, which should be converted into CBC, the surface of a volume surrounding the projection. We set the projection $\mathit{\Gamma}$

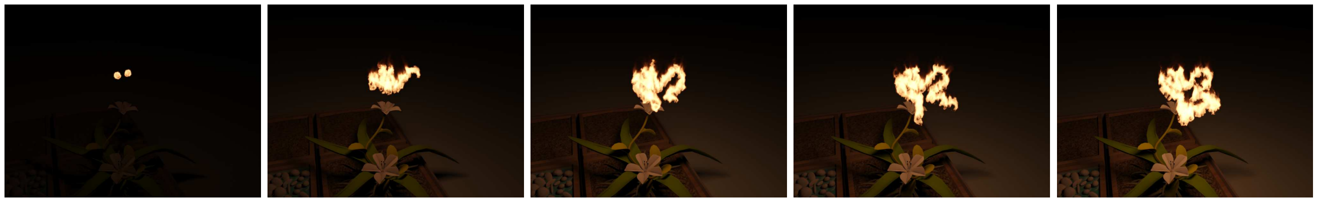

Fig. 6 Fire spreading along a butterfly curve

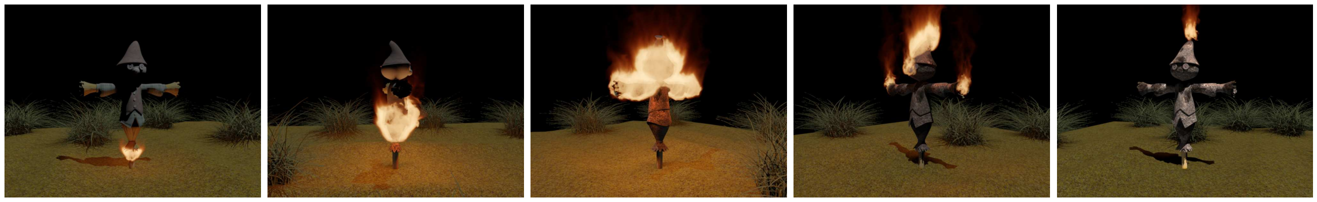

Fig. 7 Fire moving on a scarecrow while the burnt area increasing

as the skeleton of the CBC Ω_{CBC}, and let constant δ represent the minimal distance between Γ and Ω_{CBC}, which can be regarded as the thickness of fuel for fire spreading. Thus, the CBC for curves or surfaces is defined as:

$$\Omega_{CBC} = \{\boldsymbol{x} | \boldsymbol{x} \in surround(\Gamma), ||\boldsymbol{x} - cp(\boldsymbol{x})||_2 = \delta\} \quad (5)$$

where $surround(\Gamma)$ is the volume surrounding the projection Γ, for example, $surround(\Gamma) = \{\boldsymbol{x}|\varphi(\boldsymbol{x}) \leq 0\}$ when using MCPM. Figures 4 and 5 show frames of CBC for the curve and surface.

We adjust the CBC Ω_{CBC} to the signed distance function $d_{CBC}(\boldsymbol{x})$, and $d_{CBC}(\boldsymbol{x}) > 0$ represents the inside of the CBC. In addition, the CBC for fire propagation encircles the curve and surface, while for the shape's formation, the interior zone of the formed shape is also included in the CBC. Figure 3 is a case in this situation.

3.3 Flame generation

Similarly to [20], our fire simulation model uses the incompressible Navier–Stokes equations for modeling the gaseous fuel and products, and exploits level-set equation for tracking the blue core. However, they are not sufficient to control fire's movement under geometric constraints. Our model leads the trend of the fire's blue core through L-Speed, which is based on the level-set metamorphosis and maximizes the similarity between the current blue core and its corresponding CBC at each time step.

3.3.1 Basic model

The flow of fire contains the gaseous fuel and hot gaseous products, which are separated by the blue core. The equations that model the flow, denoted by $\boldsymbol{u} = (u, v, w)$, are given by the incompressible Euler equations respectively:

$$\bigtriangledown \cdot \boldsymbol{u} = 0 \quad (6)$$

$$\boldsymbol{u}_t = -(\boldsymbol{u} \cdot \bigtriangledown)\boldsymbol{u} - \bigtriangledown p/\rho + \boldsymbol{f} \quad (7)$$

So we have the velocity field $\boldsymbol{u}$ including $\boldsymbol{u}_f$ for the gaseous fuel and $\boldsymbol{u}_h$ for the hot gaseous products, one pressure field p, two kinds of gas densities, ρ_f and ρ_h, and the external forces $\boldsymbol{f}$, such as the buoyancy force and the vorticity confinement force with different coefficients for the gaseous fuel and hot products [7].

The level-set method is used to track the location of blue core. ϕ is positive in the space filled with fuel, zero at the blue core, and negative elsewhere. The standard evolving equation is:

$$\phi_t = -\boldsymbol{w} \cdot \bigtriangledown \phi \quad (8)$$

where $\boldsymbol{w} = \boldsymbol{u}_f + S_r\boldsymbol{n}$. The term $\boldsymbol{u}_f$ denotes the velocity of the gaseous fuel, S_r is the reaction speed, and $\boldsymbol{n} = \bigtriangledown\phi/|\bigtriangledown\phi|$ is the local unit normal of the blue core.

When the fuel crosses the blue core and is converted into hot products, the gas expands at a rate derived by solving the jump conditions for conservation of mass and momentum as described in [21]. The fire is described by the temperature and soot density, which can be sculpted by experience curves [20]. This paper focuses on flames, so the smoke is ignored in all images.

3.3.2 L-Speed for blue core

The basic model is to animate fire without the constraint conditions. In order to handle the fire simulation under geometric constraints, we make the current blue core Ω_{BC} match its corresponding CBC Ω_{CBC} at

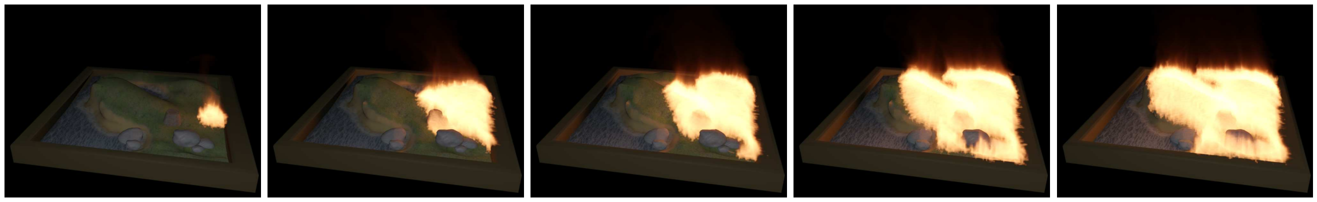

Fig. 8 Fire propagation on a field with the influence of wind, terrain slope, and the fuel type, for which the rock and water area cannot be burnt

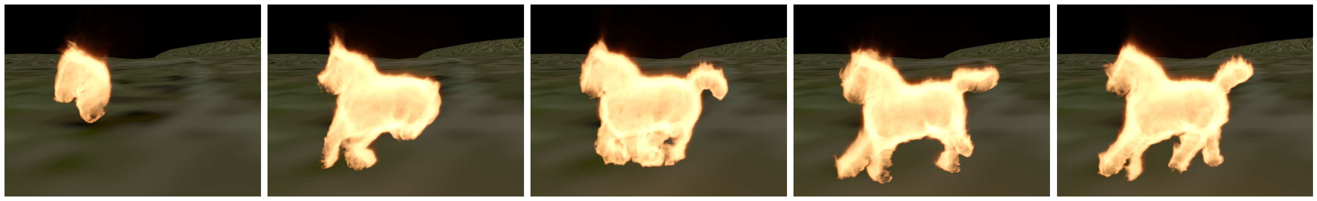

Fig. 9 Fire evolving into a horse running in place

each time step, based on level-set morphing [3]. So an appropriate speed function, $L\_Speed(d_{CBC}(\boldsymbol{x}))$, is designed for leading the source Ω_{BC} to the target Ω_{CBC} as far as possible.

L-Speed for blue core should carry information about the shape of the target as the two get near, and the signed distance transform of the target surface Ω_{CBC} is a natural choice, as it has been computed in the CBC's generation. Besides, the speed function is associated with by the reaction speed S_r. Finally, numerical difficulties and discontinuities need to be avoided in the solution. Thus, the L-Speed is defined as follows:

$$L\_Speed(d_{CBC}(\boldsymbol{x})) = -S_r \frac{d_{CBC}(\boldsymbol{x})}{D_{CBC}(\boldsymbol{x})}\boldsymbol{n} \tag{9}$$

where $d_{CBC}(\boldsymbol{x})$ is the signed distance function of the current CBC Ω_{CBC}, and $d_{CBC}(\boldsymbol{x}) > 0$ represents the inside of CBC; $D_{CBC}(\boldsymbol{x})$ is a characteristic distance function for adjusting the influence of $d_{CBC}(\boldsymbol{x})$, and $\boldsymbol{n} = \bigtriangledown\phi/|\bigtriangledown\phi|$ is the local unit normal of Ω_{BC}.

The velocity of blue core in the basic model is then modified by adding the L-Speed function. The blue core's moving velocity function is defined as:

$$\boldsymbol{w} = \boldsymbol{u}_f + S_r\boldsymbol{n} + L\_Speed(d_{CBC}(\boldsymbol{x})) \tag{10}$$

The "Blue Cores" in Figs. 3, 4 and 5 show the evolving blue cores under L-Speed's macro-level and physical model's micro-level regulations.

Blue core and CBC must overlap, so that it is possible to perform deformation. The blue core shrinks outside CBC, and expands inside CBC. In the beginning, we initiate both of them as the same. For each following time step, in consideration of more overlapping areas, an appropriate step-time is set with meeting the Courant–Friedrichs–Levy condition (CFL condition).

Furthermore, to obtain a good morphing for blue core, some regions need to be set as solid during the computations. When the fire spreads on static or moving curve or surface, the regions occupied by object should be solid. While burning into a shape is different, those solid regions increase along with the blue core, until the whole shape comes into being.

4 Implementation

The method described in last section provides a unified framework for fire simulation under geometric constraints. The data structures used in our system are shown in Table 1.

In this section, we concentrate on the fire evolving rules and take the following case as an example. In addition, the pseudocode of the simulate pipeline and each main step is presented in Algorithm 1–4. More related numerical calculation details are discussed in [20].

We use level-set equation under geometric constraint to track the projection of CBC, and take the fire's projection moving in an externally generated velocity field for example:

$$\varphi_t + \boldsymbol{u}_{evolve} \cdot \bigtriangledown_{\mathcal{G}(t)}\varphi = 0 \tag{11}$$

where $\boldsymbol{u}_{evolve}$ depends on the evolving rules. For example, in *CBC_Generate(...)*, it comes from the influences of wind, terrain slope and fuel type, which are described as follows:

Wind. $\boldsymbol{u}_{wind\_g}$ is the wind field's contribution to the velocity for forwarding the projection of CBC. It is based on Mallet et al. [18], which simplified the wind-aided model of Fendell and Wolff [8] without

Computed Field	Volume Size / Grid Precision	$N_{x,y,z}$ / h	
Voxel Center Data	Closest Point	$cp_{i,j,k}$	$i = 1, ..., N_x$ $j = 1, ..., N_y$ $k = 1, ..., N_z$
	Flammable Zone	$c_{i,j,k}$	
	Solid Zone	$solid_{i,j,k}$	
	Projection	$\varphi_{i,j,k}$	
	CBC	$d_{CBC\,i,j,k}$	
	Blue Core	$\phi_{i,j,k}$	
	Temperature	$T_{i,j,k}$	
	Soot Density	$\rho_{s\,i,j,k}$	
	Pressure	$p_{i,j,k}$	
Voxel Face Data	Velocities	$u_{i+1/2,j,k}$	$i = 0, ..., N_x$ $j = 1, ..., N_y$ $k = 1, ..., N_z$
		$v_{i,j+1/2,k}$	$i = 1, ..., N_x$ $j = 0, ..., N_y$ $k = 1, ..., N_z$
		$w_{i,j,k+1/2}$	$i = 1, ..., N_x$ $j = 1, ..., N_y$ $k = 0, ..., N_z$

Table 1 Data structures

losing its main features, primarily the overall shape of the fire front.

$$\boldsymbol{u}_{wind\_g} = \begin{cases} (\varepsilon_0 + c_1\sqrt{U}\cos^{3/2}\theta)\boldsymbol{n}_{\mathcal{G}} & \theta \le \pi/2 \\ \varepsilon_0(\alpha + (1-\alpha)|\sin\theta|)\boldsymbol{n}_{\mathcal{G}} & \theta > \pi/2 \end{cases}$$

here ε_0, c_1 and α are parameters, U is the wind speed, $\boldsymbol{n}_{\mathcal{G}} = \nabla\varphi/|\nabla\varphi|$, and $\theta = (\widehat{\boldsymbol{v}_{wind}, \boldsymbol{n}_{\mathcal{G}}})$.

Terrain Slope. $\boldsymbol{u}_{slope\_g}$ is the velocity contribution from terrain slope. Observe the truth that propagating up a vertical wall occurs more quickly than across a horizontal floor.

$$\boldsymbol{u}_{slope\_g} = (\tan(\pi/4 - \beta/2) + 1)V\boldsymbol{n}_{\mathcal{G}} \tag{12}$$

here β is the angle between the $z = (0, 0, 1)$ vector and $\boldsymbol{n}_{\mathcal{G}}$, and V is the fire speed when across a horizontal floor.

Fuel Type. $c(\boldsymbol{x})$ is the function of local properties such as the fuel type. For example, if there is a noncombustible part of curve or some water area on surface, then $c(\boldsymbol{x}) = 0$ for it, and $c(\boldsymbol{x}) = 1$ for the rest.

5 Results and discussion

All of our simulation was taken on an Intel Core 2 Duo E6600 2.4GHz processor with 2 GB RAM, and the images were created by using our unpublished work. Inspired by [20], we created a hierarchical partition of the volume light to approximate its energy distribution, and developed an efficient solution to simulate the radiative energy transfer between complex participating medium and the scene.

For the basic fire model in Sect. 3.3.1, we used $S_r =$ 0.18m/s, $\rho_f = 1.0$kg/m^3 (the density of gaseous fuel),

Algorithm 1 $Fire\_Simulate(\mathcal{G}(t), ...)$

```
1: $(cp, ...) \leftarrow Uni\_Represent(\mathcal{G}(t), ...)$ // Sect. 3.1
2: $Initialization(...)$
3: $m \leftarrow 0$ // the frame number counter
4: $cp \leftarrow cp_0$
5: while $m \le$ the needed frame number $M$ do
6:     $\Delta t \leftarrow SetTimeStep(...)$
7:     $d_{CBC} \leftarrow CBC\_Generate(cp, \Delta t, ...)$ // Sect. 3.2
8:     $(T, ...) \leftarrow Flame\_Generate(d_{CBC}, \Delta t, ...)$ // Sect. 3.3
9:     if reach the output condition then
10:        $Rendering(T, ...)$
11:        $m \leftarrow m + 1$
12:        $cp \leftarrow cp_m$ for the moving curve or surface
13:    end if
14: end while
```

Algorithm 2 $(cp, ...) \leftarrow Uni\_Represent(\mathcal{G}(t), ...)$

```
1: // Loop if preprocessing the mesh animation
2: if $Is\_Curve(\mathcal{G}(t))$ then
3:     $cp(\boldsymbol{x}) \leftarrow CalCurvCp(\mathcal{G}(t), N_{x,y,z}, h)$ // [4]
4:     $c(\boldsymbol{x}) \leftarrow CalCurvC(\mathcal{G}(t), N_{x,y,z}, h)$ // defined by artist
5: else if $Is\_Surface(\mathcal{G}(t))$ then
6:     $cp(\boldsymbol{x}) \leftarrow CalSurfCp(\mathcal{G}(t), N_{x,y,z}, h)$ // [12]
7:     $c(\boldsymbol{x}) \leftarrow CalSurfC(\mathcal{G}(t), N_{x,y,z}, h)$ // defined by artist
8:     $solid(\boldsymbol{x}) \leftarrow Voxelization(\mathcal{G}(t), N_{x,y,z}, h)$ // [13]
9: end if
```

Algorithm 3 $d_{CBC} \leftarrow CBC\_Generate(cp_m, \Delta t, ...)$

```
1: // $\varphi^{n+1}(\boldsymbol{x}, m) \leftarrow MCPM(\varphi^n(\boldsymbol{x}), m, ...)$
2: $\boldsymbol{u}_E(\boldsymbol{x}) \leftarrow c(\boldsymbol{x})(\boldsymbol{u}_{wind\_g} + \boldsymbol{u}_{slope\_g})$ // or other functions
3: $\boldsymbol{u}_L(\boldsymbol{x}) \leftarrow L(d_m(\boldsymbol{x}))$ // Eq. (2), and zero for static situation
4: $\widetilde{\boldsymbol{u}}(\boldsymbol{x}) \leftarrow \boldsymbol{u}_E(\boldsymbol{x}) + \boldsymbol{u}_L(\boldsymbol{x})$   $\boldsymbol{u}(\boldsymbol{x}) \leftarrow \widetilde{\boldsymbol{u}}(cp_m(\boldsymbol{x}))$
5: $\widetilde{\varphi}^{n+1}(\boldsymbol{x}) \leftarrow LevelSetSolver(\varphi^n(\boldsymbol{x}), \boldsymbol{u}(\boldsymbol{x}), \Delta t)$ // [22]
6: $\varphi^{n+1}(\boldsymbol{x}) \leftarrow \widetilde{\varphi}^{n+1}(cp_m(\boldsymbol{x}))$
7: for All_Nodes do
8:     if $\varphi^{n+1}(\boldsymbol{x}) \le 0$ then
9:         $d_{CBC}(\boldsymbol{x}) \leftarrow \delta - ||\boldsymbol{x} - cp_m(\boldsymbol{x})||_2$
10:    else
11:        $d_{CBC}(\boldsymbol{x}) \leftarrow -1$
12:    end if
13: end for
14: $ReInit(d_{CBC}(\boldsymbol{x}))$ // adjust $d_{CBC}(\boldsymbol{x})$ to meet $|\nabla d_{CBC}| = 1$
```

Algorithm 4 $(T, ...) \leftarrow Flame\_Generate(d_{CBC}, \Delta t, ..)$

```
1: // add force (wind, buoyancy, vorticity confinement) [7, 20]
2: $\boldsymbol{u}(\boldsymbol{x}) \leftarrow AddWind(\boldsymbol{u}^n(\boldsymbol{x}), \boldsymbol{f}_{wind})$ // wind for example
3: $\boldsymbol{u}(\boldsymbol{x}) \leftarrow AddBuoy(\boldsymbol{u}(\boldsymbol{x}), T^n(\boldsymbol{x}))$
4: $\boldsymbol{u}_f(\boldsymbol{x}), \boldsymbol{u}_h(\boldsymbol{x}) \leftarrow Ghost(\boldsymbol{u}(\boldsymbol{x}), \phi^n(\boldsymbol{x}))$
5: $\boldsymbol{u}_f(\boldsymbol{x}) \leftarrow AddVortF(\boldsymbol{u}_f(\boldsymbol{x}))$
6: $\boldsymbol{u}_h(\boldsymbol{x}) \leftarrow AddVortH(\boldsymbol{u}_h(\boldsymbol{x}))$
7: // advect the gaseous fuel and hot products
8: $\boldsymbol{u}_f(\boldsymbol{x}) \leftarrow Advect(\boldsymbol{u}_f(\boldsymbol{x}))$
9: $\boldsymbol{u}_h(\boldsymbol{x}) \leftarrow Advect(\boldsymbol{u}_h(\boldsymbol{x}))$
10: // update the blue core
11: $\boldsymbol{w}(\boldsymbol{x}) \leftarrow BcVel(\boldsymbol{u}_f(\boldsymbol{x}), Sr, d_{CBC}(\boldsymbol{x}), \phi^n(\boldsymbol{x}))$ // Eq. (10)
12: $\phi^{n+1}(\boldsymbol{x}) \leftarrow LevelSetSolver(\phi^n(\boldsymbol{x}), \boldsymbol{w}(\boldsymbol{x}), \Delta t)$
13: // make free divergence
14: $\boldsymbol{u}(\boldsymbol{x}) \leftarrow JointVel(\boldsymbol{u}_f(\boldsymbol{x}), \boldsymbol{u}_h(\boldsymbol{x}), \phi^{n+1}(\boldsymbol{x}))$
15: $\boldsymbol{u}^{n+1}(\boldsymbol{x}) \leftarrow Project(\boldsymbol{u}(\boldsymbol{x}), \phi^{n+1}(\boldsymbol{x}))$
16: // update the temperature, and soot density if needed
17: $T^{n+1}(\boldsymbol{x}) \leftarrow CalcTemp(T^n(\boldsymbol{x}), \boldsymbol{u}^{n+1}(\boldsymbol{x}), \phi^{n+1}(\boldsymbol{x}))$
18: ${\rho_s}^{n+1}(\boldsymbol{x}) \leftarrow CalcDens({\rho_s}^n(\boldsymbol{x}), \boldsymbol{u}^{n+1}(\boldsymbol{x}), \phi^{n+1}(\boldsymbol{x}))$
```

	Geometry Feature	Evolving Rule	Volume Size	Time(s)		
				Uniform Rep.	CBC	Flame(PCG/1-6e)
Butterfly	Nurbs(155 Nodes)	$V_c = 1.0$	$140 * 130 * 140$	512.55	24.59	64.30(20.14)
Rope	Nurbs(20 Nodes, 97f)	$V_c = 1.4$	$270 * 54 * 168$	53.85×97	18.67	50.14(13.15)
Scarecrow	Mesh(6108v, 2036t)	$V_s = 1.1(0.9)$	$140 * 70 * 224$	260.91	25.84	66.05(15.01)
Field	Mesh(23292v, 7764t)	$\boldsymbol{u}_{evolve}{}^*$	$165 * 163 * 98$	469.72	22.23	65.38(17.82)
Ball	Meshes(101f, 1644v, 3252t)	$V_s = 0.75$	$132 * 132 * 132$	160.0×101	26.21	60.72(17.95)
Horse	Meshes(142f, 8431v, 16843t)	$V_s = 1.64$	$48 * 224 * 224$	500.0×142	28.18	61.11(15.90)

Notes: f(frames), v(vertices), t(triangles); V_c(speed along the curve, m/s), V_s(speed along the surface, m/s).
In the scarecrow's line, V_s =1.1m/s is for the burning speed, and V_s =0.9m/s for the extinguish speed.
$\boldsymbol{u}_{evolve}{}^*$ is calculated according to Sect. 4 with $\varepsilon_0 = 1.0$, $c_1 = 1.0$, $U = 5.0$, $\boldsymbol{v}_{wind} = (-1, -1, 0)$, $\alpha = 0.5$, $V = 1.0$, and $c(\boldsymbol{x})$ is precomputed in the uniform representation stage defined by an artist.

Table 2 Simulation results

$\rho_h = 0.05\text{kg}/m^3$ (the density of hot gaseous products), $\alpha = 0.08\text{m}/(Ks^2)$ (the buoyancy constant), $T_{air} = 300\text{K}$ (the ambient temperature of the air), $T_{ignition} = 500\text{K}$(the ignition temperature of the fuel), $T_{max} = 2300\text{K}$ (the maximum temperature), $c_T = 206\text{K/s}$ (the cooling constant) in most of our experiments. The vorticity confinement parameter was set to $\varepsilon = 10$ for gaseous fuel and $\varepsilon = 70$ for the hot gaseous products. Comprehensive explanations to these variables are given in [20].

For the CBC generation in Sect. 3.2, we set δ as $3h$ for surface constraint, and $4h$ for curve constraint. As for the $D_{CBC}(\boldsymbol{x}, t)$, we reduced it to a constant, 8.182×10^{-4}, then $D_m(\boldsymbol{x}) = D_{CBC}(\boldsymbol{x})/S_r$.

Our method is applied to the fire simulation under still and moving curves, which are represented by NURBS curves. Fig. 6 shows the images for fire propagation along a butterfly curve. It starts to burn from the two antennas of the butterfly. The fires propagate along curves until two fires meet and merge together. In Fig. 1, the fire spreading animation is generated following the motion of a simulated rope.

Figure 7 shows the performance of our model under static surface constraints. The fire is moving on a scarecrow while the extinct area is increasing. In Fig. 2, a moving and burning ball bursts in accordance with the animator-designed animation. To simulate the fire influenced by terrain slope, wind, and the fuel type, Fig. 8 shows the fire propagation on a field in which the rock and water area cannot be burnt, and the fire burns faster on the upgrade than the flat land.

Finally, a fire horse running in place is accomplished as shown in Fig. 9.

The simulation used the serial computation, and the results are given in Table 2. Take the running fire horse for example, we used the volume size with $48 \times 224 \times 224$, Preconditioned Conjugate Gradient (PCG) solving pressure with 1e–6 precision, and the uniform representation of the geometric constraint was done in the preprocessing stage. The simulation used about 1.5 gigabytes of RAM and took about 90 seconds in each iteration, among which the generation of CBC cost about 31.56% and PCG solver 17.81%.

We can also see from the implementation results that after the uniform representation of geometric constraints, the simulation time is basically determined by the volume size of the computed field, regardless of the complexity of geometric constraints.

In our model, the main performance bottleneck lies in CBC generation and PCG solver. Basically, we could accelerate our algorithm by parallel computing. Moreover, the good independence of the uniform representation of the geometric constraints enables high speed-up performance with parallelization.

6 Conclusions

In this paper, we have presented a unified framework to simulate fire under geometric constraints and user-defined evolution rules. It is an easy way to make a variety of fire effects on requests from users, while there are some limitations in our approach. For the moving curve or surface, the difference between the two consecutive frames of the artist's animation cannot be too large for getting a good result. At present, if the two frames change too fast, we need to do the frame interpolation between them at the preprocessing stage.

In our future work, we will focus on improving the shape transformation between CBC and blue core to enable rapid change of frames, even in key-frames. In addition, the algorithm will be parallelized to accelerate the computation, and can be efficiently implemented on current GPUs.

Acknowledgements This work is supported and funded by National Natural Science Foundation of China Grant Nos. 60603082 and 60703019, National Key Technology R&D Program 2008BAI-50B07, and NSFC-Guangdong Joint Fund (U0935003). Also, we would like to thank the anonymous reviewers for their valuable comments and suggestions.

References

1. Beaudoin, P., Paquet, S., Poulin, P.: Realistic and controllable fire simulation. In: Proceedings of Graphics Interface 2001, Ottawa, Ontario, Canada, pp. 159–166 (2001)
2. Besl, P.J., McKay, N.D.: A method for registration of 3-d shapes. IEEE Trans. PAMI. **14**(2), 239–256 (1992)
3. Breen, D.E., Whitaker, R.T.: A level-set approach for the metamorphosis of solid models. IEEE Transactions on Visualization and Computer Graphics **7**(2), 173–192 (2001)
4. Chen, X.D., Yong, J.H., Wang, G., Paul, J.C., Xu, G.: Computing the minimum distance between a point and a nurbs curve. Comput. Aided Des. **40**(10-11), 1051–1054 (2008)
5. Chiba, N., Muraoka, K., Takahashi, H., Miura, M.: Two-dimensional visual simulation of flames, smoke and the spread of fire. The Journal of Visualization and Computer Animation, **5**(1), 37–53 (1994)
6. Fattal, R., Lischinski, D.: Target-driven smoke animation. In: SIGGRAPH '04: ACM SIGGRAPH 2004 Papers, ACM, New York, pp. 441–448 (2004)
7. Fedkiw, R., Stam, J., Jensen, H.W.: Visual simulation of smoke. In: Proceedings of SIGGRAPH 2001, ACM, pp. 15–22 (2001)
8. Fendell, F.E., Wolff, M.F.: Forest fires - behavior and ecological effects. Academic Press (2001)
9. Hong, J.M., Shinar, T., Fedkiw, R.: Wrinkled flames and cellular patterns. In: SIGGRAPH '07: ACM SIGGRAPH 2007 Papers, ACM, New York, pp. 1–6 (2007)
10. Horvath, C., Geiger, W.: Directable, high-resolution simulation of fire on the GPU. In: SIGGRAPH '09: ACM SIGGRAPH 2009 papers, ACM, New York, pp. 1–8 (2009)
11. Ishikawa, T., Miyazaki, R., Dobashi, Y., Nishita, T.: Visual simulation of spreading fire. In: NICOGRAPH Internation '05 pp. 43–48 (2005)
12. Jones, M.W.: The production of volume data from triangular meshes using voxelisation. Computer Graphics Forum **15**, 311–318 (1996)
13. Karabassi, E.-A., Papaioannou, G., Theoharis, T.: A fast depth-buffer-based voxelization algorithm. J. Graph. Tools, **4**(4),5–10 (1999)
14. Lee, H., Kim, L., Meyer, M., Desbrun, M. Meshes on fire. In: Proceedings of the Eurographic workshop on Computer animation and simulation, Springer-Verlag New York, Inc., New York, pp. 75–84 (2001)
15. Liu, S., Liu, Q., An, T., Sun, J., Peng, Q.: Physically based simulation of thin-shell objects burning. Vis. Comput. **25**(5-7), 687–696 (2009)
16. Losasso, F., Irving, G., Guendelman, E.: Melting and burning solids into liquids and gases. IEEE Transactions on Visualization and Computer Graphics **12**(3), 343–352 (2006)
17. Macdonald, C.B., Ruuth, S.J.: Level set equations on surfaces via the closest point method. J. Sci. Comput. **35**(2-3), 219–240 (2008)
18. Mallet, V., Keyes, D.E., Fendell, F.E.: Modeling wildland fire propagation with level set methods. ArXiv e-prints (2007)
19. Melek, Z., Keyser, J.: Interactive Simulation of Fire. In: PG '02: Proceedings of the 10th Pacific Conference on Computer Graphics and Applications, pp. 431 (2002)
20. Nguyen, D.Q., Fedkiw, R.P., Jensen, H.W.: Physically based modeling and animation of fire. In: SIGGRAPH '02: Proceedings of the 29th annual conference on Computer graphics and interactive techniques, ACM, New York, pp. 721–728 (2002)
21. Nguyen, D.Q., Fedkiw, R.P., Kang, M.: A boundary condition capturing method for incompressible flame discontinuities. J. Comput. Phys. **172**(1), 71–98 (2001)
22. Osher, S., Sethian, J.A.: Fronts propagating with curvature-dependent speed: algorithms based on Hamilton-Jacobi formulations. J. Comput. Phys., **79**(1), 12–49 (1988)
23. Perry, C.H., Picard, R.W.: Synthesizing flames and their spreading. In: Proceedings of the Fifth Eurographics Workshop on Animation and Simulation, pp. 1–14 (1994)
24. Qian, J., Tryggvason, G., Law, C.K. A front tracking method for the motion of premixed flames. J. Comput. Phys. **144**(1), 52–69 (1998)
25. Reeves, W.T.: Particle systems - a technique for modeling a class of fuzzy objects. In: SIGGRAPH '83: Proceedings of the 10th annual conference on Computer graphics and interactive techniques, ACM, New York, pp. 359–375 (1983)
26. Shi, L., Yu, Y.: Controllable smoke animation with guiding objects. ACM Trans. Graph. **24**(1), 140–164 (2005)
27. Shi, L., Yu, Y.: Taming liquids for rapidly changing targets. In: SCA '05: Proceedings of the 2005 ACM SIGGRAPH/Eurographics symposium on Computer animation, ACM, New York, pp. 229–236 (2005)
28. Stam, J., Fiume, E.: Depicting fire and other gaseous phenomena using diffusion processes. In: SIGGRAPH '95: Proceedings of the 22nd annual conference on Computer graphics and interactive techniques, ACM, New York, pp. 129–136 (1995)
29. Thürey, N., Keiser, R., Pauly, M., Rüde, U.: Detail-preserving fluid control. In: SCA '06: Proceedings of the 2006 ACM SIGGRAPH/ Eurographics symposium on Computer animation, Eurographics Association, Aire-la-Ville, Switzerland, pp. 7–12 (2006)
30. Wei, X., Li, W., Mueller, K., Kaufman, A.: Simulating fire with texture splats. In: VIS '02: Proceedings of the conference on Visualization '02, Boston, Massachusettsm, pp. 227–235 (2002)
31. Xu, J.J., Zhao, H.K.: An eulerian formulation for solving partial differential equations along a moving interface. J. Sci. Comput. **19**(1-3), 573–594 (2003)
32. Zhao, Y., Wei, X., Fan, Z., Kaufman, A., Qin, H.: Voxels on fire. In: VIS '03: Proceedings of the 14th IEEE Visualization, IEEE Computer Society, Washington, pp. 36 (2003)