Visual Comput (2005) 21: 474–487
DOI 10.1007/s00371-005-0296-0

ORIGINAL ARTICLE

Lin Shi
Yizhou Yu
Christopher Wojtan
Stephen Chenney

Controllable motion synthesis in a gaseous medium

Published online: 19 July 2005

L. Shi (✉) · Y. Yu · C. Wojtan
Department of Computer Science,
University of Illinois at
Urbana-Champaign, 201 N. Goodwin Ave.,
Urbana, IL 61801, USA
linshi@uiuc.edu, yyz@cs.uiuc.edu,
wojtan@cc.gatech.edu

S. Chenney
Department of Computer Science,
University of Wisconsin, 1210 W. Dayton
St., Madison, WI 53706, USA
schenney@cs.wisc.edu

Abstract The generation of realistic motion satisfying user-defined requirements is one of the most important goals of computer animation. Our aim in this paper is the synthesis of realistic, controllable motion for lightweight natural objects in a gaseous medium. We formulate this problem as a large-scale spacetime optimization with user controls and fluid motion equations as constraints. We have devised novel and effective methods to make this large optimization tractable. Initial trajectories are generated with data-driven synthesis based on stylistic motion planning. Smoothed particle hydrodynamics (SPH) is used during optimization to produce fluid simulations at a reasonable computational cost, while interesting vortex-based fluid motion is generated by recording the presence of vortices in the initial trajectories and maintaining them through optimization. Object rotations are refined as a postprocess to enhance the visual quality of the results. We demonstrate our techniques on a number of animations involving single or multiple objects.

Keywords Spacetime constraints/optimization · Stylistic motion planning · Smoothed particle hydrodynamics · Vortices · Trajectory library

1 Introduction

The generation of natural motion that satisfies user-defined requirements is an important goal of computer animation. This article presents a technique to synthesize realistic, controllable motion for lightweight objects, alone or in a group, being acted on by a gaseous medium. The techniques we present are applicable in synthetic animation and video production because lightweight objects, such as leaves, feathers, and bubbles, are ubiquitous in our environment, and their motion suggests grace and fluidity.

Real-world moving objects receive influences (external forces) from the surrounding air, the most apparent being air drag. At the same time, air has its own dynamics and receives influences from the objects. We define "lightweight" objects to be those that do not strongly influence the large-scale behavior of the air, so that the motion of one object does not significantly impact the motion of its neighbors. Furthermore, lightweight objects largely follow the velocity field of the air, and hence their motion provides a visualization of most characteristics of the underlying flow. In particular, most gaseous phenomena should be almost divergence free, and hence the lightweight objects should rarely, if ever, collide with each other as they approximately follow streamlines.

We focus in on the motion of the objects themselves, because they are directly visible. We are concerned with the fluid only to the extent that it can be interpreted from the objects' motion. Hence we frame the problem as one of object motion synthesis in the presence of constraints imposed by the fluid. Inspired by the seminal work on spacetime constraints [50], we formulate this problem as a large-scale optimization. Directly solving this large-scale problem using an off-the-shelf optimization toolbox would be infeasible since the number of parameters ne-

cessary for realistically describing fluid motion is prohibitively large, especially when the fluid is simulated using a voxel grid. Therefore, we have devised a number of novel and effective methods to make this problem tractable.

- *Smoothed particle hydrodynamics* (SPH) are used [20, 28, 32] to describe fluid motion and provide optimization constraints. Although SPH is numerically less accurate than a grid-based simulation, it can generate reasonable fluid behavior using a relatively small number of particles, which significantly reduces the number of constraints required in the optimization. The reduction in accuracy is acceptable because we do not directly visualize the fluid, but rather the objects immersed in the fluid.
- *Data-driven synthesis* is exploited to generate an initial trajectory for each object. The trajectory may have certain desirable features required by the user. The optimization then makes the initial trajectories more physically plausible. The user is thus able to guide the optimization toward a specific local minimum by biasing the starting point. Such control is necessary because the problem may have numerous equally plausible local minima, while a user almost certainly has a specific one in mind.
- *Vortices* are important in producing interesting fluid motion. During optimization, vortices are identified and recovered in a separate pass to improve efficiency.
- *Rotational components* of the objects are largely independent of their overall trajectory. We ignore them in the optimization and refine them in a separate postprocessing stage to produce visually pleasing results.

1.1 Background and related work

The lightweight objects that we consider are particles for the purposes of control, with rigid or deformable body characteristics added as a postprocess. Several techniques have been proposed for animating lightweight objects in a wind field, where the objects do not influence the air. Early work by Wejchert and Haumann [49] modeled the wind field as a linear combination of basis fields and advected leaves in the flow. Chenney [7] used a tile-based approach to define the flow and also advected leaves. Both of these approaches offer indirect control of the object motion through control of the flow. Wei et al. [48] simulated the motion and deformation of lightweight objects in a wind generated by the Lattice–Boltzmann Model but did not consider control. These algorithms are best suited for synthesizing and editing velocity fields instead of providing direct control over individual trajectories traced from specific locations, and the resulting fields may have visual artifacts because they are not physically motivated. In this paper, we adopt the SPH formulation because it is physically motivated, gases are slightly compressible, and it is more convenient to incorporate SPH into an optimization framework.

We now review other related work on solving constraint problems in graphics and discuss data-driven motion synthesis, which is the basis of our method for creating initial trajectories.

Constrained rigid bodies. Several approaches have been proposed for generating multibody constrained motion. Each employs a specific optimization technique, including genetic algorithms [43], gradient-based methods [8], plausible simulation with randomness [3], and stochastic sampling [9]. They all consider full rigid body motion (including rotation), but, more importantly, the only nonconstant external forces on the bodies arise due to collisions. On the other hand, we consider the continuous influence of an external fluid medium, and the larger state space that results prevents the application of existing rigid body methods.

Particle-based simulation. When generating trajectories, we model lightweight objects as particles, and the SPH fluid model we use is also particle based. The use of particles to simulate natural phenomena has a long history in computer graphics. It was the introduction of particle systems with pairwise interactions that made it possible to realistically simulate fluids, deformable objects, and even flocks and herds. Miller and Pierce [31] simulate deformable objects with particle interactions based on the Lennard–Jones potential force. Terzopoulos et al. [44] paired particles to better simulate deformable objects. Tonnesen [45] improved particle motion by adding additional particle interactions based on heat transfer among particles. Reynolds [39] modeled the group behavior of birds by considering flocks as particle systems. Particle systems are typically controlled by modifying the interaction rules. In the work most closely related to ours, Anderson et al. [1] generated constrained behaviors of flocks with a two-stage process: in the first pass, path editing is used to create an initial solution that meets the constraints but largely ignores the underlying model, while the second, stochastic optimization, pass improves the result with respect to the model. A similar two-stage process is taken in this paper. However, we deal with a different domain and underlying physical model and use data-driven synthesis for the first pass and gradient-based optimization for the second stage.

Lucy [28] and Gingold and Monaghan [20] introduced a flexible gridless particle method called *smoothed particle hydrodynamics* (SPH) to simulate astrophysical problems. We provide an overview of the method in Appendix B. SPH has recently been adapted to compressible and incompressible fluid simulation [32, 33]. For graphics applications, Desbrun and Cani [11] and Müller et al. [34] improved SPH to compute the dynamics of deformable substances and liquids. Hadap and Magnenat–Thalmann [21] modified the formulation of SPH to simulate hair–hair interactions by considering hair as a fluidlike

continuum. Recently, another particle method, Moving Particle Semi-Implicit (MPS), was developed [54] along with its application to incompressible fluid simulation in graphics [36].

Grid-based fluid simulation and control. Recent years have seen rapid progress on 3D grid-based fluid simulation in the graphics community. Foster and Metaxas [17] were the first to bring grid-based methods to graphics. Stam [41] introduced a combination of a semi-Lagrangian advection scheme and implicit solvers to achieve unconditional stability. Fedkiw et al. [14] introduced vorticity confinement and a higher-order interpolation technique, while Foster and Fedkiw [15] described a hybrid liquid volume model combining implicit surfaces and massless marker particles. The work was further improved in terms of accuracy by Enright et al. [12] by using particles on both sides of the interface between air and water. Yngve et al. [53] provided one method for simulating compressible fluids for explosion generation. Several techniques have been proposed to control the fluid itself, typically the density or velocity field. Foster and Metaxas [16] introduced embedded controllers that allow animators to intuitively specify and control a 3D fluid animation. Foster and Fedkiw [15] suggest animator-designed "fake" space curves and surfaces to control the motion of liquids. Rasmussen et al. [38] use dense particles to control not only the velocity of the fluid in their local neighborhoods but also viscosity, the level set of the liquid surface, and the velocity divergence. Treuille et al. [46] proposed a gradient-based method for controlling both smoke and liquid simulations through user-specified keyframes. This approach dynamically simulates the derivatives of the velocity field in the same framework for simulating the velocity field itself. A novel multiple shooting scheme was designed for matching multiple keyframes, and the adjoint method was adopted by McNamara et al. [29] to significantly improve the efficiency of these derivative evaluations. At the same time, an efficient and novel technique to match smoke density against user-specified distributions was reported by Fattal et al. [13], which involves a custom-designed driving-forces term and a smoke-gathering term. Meanwhile, another efficient method based on the level sets of the smoke was introduced by Shi and Yu [40]. It can effectively control the smoke shape without the forcing and gathering terms in [13]. While these control schemes are designed for grid-based fluid simulations themselves, our particle-based approach is focused on objects immersed in a fluid and does not attempt to determine forces acting on the fluid itself.

An alternative to constrained simulation is fluid editing, or constructive methods for defining flows. Pidgin et al. [35] extract an Advected Radial Basis Function (ARBF) model from Eulerian simulations, which a user can control to a certain extent by editing the path lines of the particles and maintaining coherence by enforcing spatiotemporal constraints. To simulate and control breaking waves, Mihalef et al. [30] introduced the Slice Method, in which a library of 2D breaking waves is used to integrate 3D shapes while controlling the 3D geometry. The subsequent dynamics is then computed with the aid of a 3D Navier–Stokes solver. Their algorithm is efficient and physical, but the tradeoff is a lack of precise control over the final output. However, their approach is similar to ours in its use of a library of precomputed solutions as the basis for a final answer.

Data-driven motion editing and synthesis. Data-driven motion editing and synthesis typically addresses human motion. Captured motion data usually have a high temporal resolution and can be considered as a temporal signal. Therefore, warping and signal processing operations can be applied to modify the motion to achieve certain goals. Early work should thus be classified as motion editing rather than synthesis [5, 47, 51]. Recent work on data-driven motion synthesis breaks long motion sequences into segments and builds a graph with connections among the segments where a smooth transition is possible [2, 4, 23, 25, 26, 37, 42, 52]. Novel sequences are generated by random or controlled walk on the graph. Choi et al. [10] extended motion synthesis to motion planning using captured data and probabilistic roadmaps. Arikan et al. [2] presented a technique for satisfying both annotation and position constraints in an open field, but the ability to avoid obstacles is not demonstrated. James and Fatahalian [22] applied data-driven synthesis to real-time deformation and rendering by precomputing a motion database and then synthesizing impulse-driven novel motion on the fly. We build upon these techniques in creating initial trajectories for our objects.

2 Overview

A dataflow diagram of our system is given in Fig. 1. The *input* to our problem includes a set of lightweight rigid objects with six DOFs, their initial configurations (mass, positions, velocities, etc.), a set of user-specified constraints (positions, velocities, and even accelerations of certain objects at specific times), and the environmental configuration including a set of obstacles. The *output* includes the complete specifications of the DOFs of the objects at every frame of an animation. In addition, the objects should have plausible motion in a gaseous medium, reflecting characteristics of the underlying fluid motion. In particular, the object motion should appear to be divergence free and continuous. The user controls the final result via the set of constraints. There are two convenient ways to specify the constraints, keyframes and partial trajectories. A keyframe specifies the positions of all the ob-

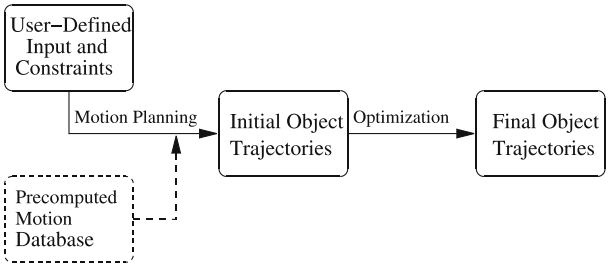

Fig. 1. System overview

jects at a particular time simultaneously, while a partial trajectory specifies a subsequence of the positions of a single object.

In this problem, the objects have passive motion largely controlled by the gas. To reflect interesting motion of the fluid, we need a reasonably accurate CFD algorithm, while the objects can be modeled in a simple manner. We use particles as the dynamic model for both the gaseous medium and the objects. The gas consists of a number of particles coupled together using SPH (Appendix B). Each object is approximated with a small number of particles whose relative positions are fixed throughout the animation. Gas particles and object particles are labeled differently and have different masses. Gas particles also interact with object particles following SPH, while different objects do not directly interact with each other.

We cast this problem as a spacetime constrained optimization, which is the last stage in Fig. 1. Since the gas and object particles have interactions, the unknowns include the positions and velocities of all the particles at all frames. The SPH equations of motion as well as user-specified controls are set up as optimization constraints. Directly optimizing such a large-scale problem without a reasonable initialization would be infeasible. We actually devise a data-driven motion synthesis algorithm to generate initial trajectories of the objects before nonlinear optimization. The initial trajectories are synthesized using a method similar to those in Lee et al. [25], Kovar et al. [23], and Arikan et al. [2]. However, our synthesis method is based on robotic motion planning [6, 24] and therefore can avoid obstacles as well as satisfy position constraints. Since such a synthesis requires a large motion database to start with, we synthetically run gaseous simulations using the method of Fedkiw et al. [14] with a number of combinations of force fields and obtain a spacetime dataset from each simulation. We trace motion trajectories from all simulations to form the database. This database is precomputed only once and used to generate initial trajectories for all examples in this paper.

We also model forces explicitly since they are responsible for interesting motion. All the forces together are modeled as a force field that is further decomposed into two components: a vortex force field with zero divergence but nonzero curl and a laminar field with smooth force vectors. Every trajectory from the motion database is annotated with the type of force field that generated it during gas simulations. During optimization, if a segment of an initial trajectory is annotated with a vortex field, a parametric vortex model is fit to the forces along the trajectory first. We carry out vortex fitting and the aforementioned spacetime optimization in two alternating passes.

3 Precomputing the trajectory database

In this section, we discuss the steps in building the motion database for subsequent motion planning. Since this part is not the focus of this paper, we only present the outline of the steps and refer the reader to the relevant papers for further details.

3.1 Fluid simulation

Building the database requires a large number of sample trajectories. Since capturing trajectories from the real world seems infeasible and fluid simulation has become quite mature recently, we collect the sample trajectories from grid-based gaseous simulations using the algorithm of Fedkiw et al. [14], which can produce high-quality results. This overall approach is similar to the precomputation stage in James and Fatahalian [22], which focuses on deformable objects.

We use a $64 \times 64 \times 64$ grid with the Neumann boundary condition. We run a number of simulations each of which has a distinct force field and an optional set of simple obstacles, such as boxes and cylinders. Each force field is superposed from multiple vortex fields and wind fields. The vortex fields are defined parametrically as in Appendix C, and the wind fields have parallel force vectors with a deteriorating strength in both space and time. We start tracing a trajectory from the center of a voxel in the first frame of a simulation and keep tracking the trajectory from frame to frame by following the velocity vectors until the last frame or the boundary of the work space has been reached. Each point on the trajectory is annotated with the type of force field present in its neighborhood. Since the simulation grid has an unnecessarily large number of voxels, we only trace a trajectory from a subset of the voxels uniformly distributed in the simulation grid. According to our experiments, the degree of freedom in the shape of the trajectories is not overly large. We stop observing new trajectory shapes once the number of simulations exceeds a certain threshold or the length of the simulations exceeds a certain duration. Thus, we empirically decide the number of simulations we need and the duration of the simulations.

3.2 Compression and clustering

The amount of raw trajectory data is too large to be conveniently used for any subsequent steps. Thus we need to perform segmentation, compression, and clustering. Segmentation partitions each complete trajectory into shorter segments, which are considered as motion units that can be rejoined during motion synthesis. We compute a curvature at every point on the trajectory and classify high curvature portions of the trajectory as features. We maintain the integrity of the features by partitioning the trajectory at points with locally minimal curvature. The segments thus obtained have various lengths. Segments with approximately the same length are grouped together and truncated to have exactly the same length. Further compression and clustering are performed separately on each group. We use singular value decomposition (SVD) to compress the segments as in James and Fatahalian [22]. Note that SVD reduces not the number of segments but the dimensionality required to represent each segment.

In the next step, tree-structured vector quantization (hierarchical *k*-means clustering) [19] is performed to organize the segments from each group into tree structures, which are very efficient for nearest neighbor search. Each group starts from a small number of clusters obtained from an initial *k*-means clustering. Each of the initial clusters is recursively split into smaller ones that have corresponding nodes in an accompanying tree structure (Fig. 2). An internal node of the tree contains a representative of the segments in its subtree.

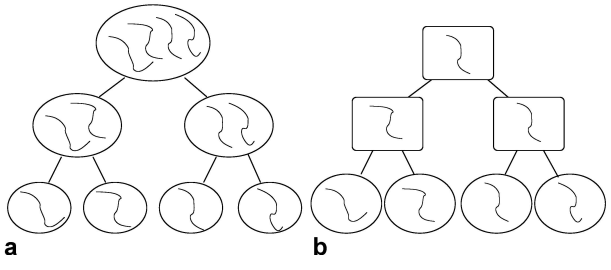

Fig. 2. a An initial cluster of trajectory segments is recursively split into smaller ones until they only have one segment each. **b** There is a tree structure corresponding to this recursive splitting process. Each node in the tree contains a representative of the segments in its subtree. The segments in the original cluster are distributed among the leaf nodes

4 Stylistic motion planning

One of the goals of robotic motion planning is to search for a path between a source and a destination while avoiding obstacles. By *stylistic motion planning* we mean that in addition to path searching, the trajectory and the velocities along the trajectory should mimic the motion style of a class of natural objects. For example, suppose we would like to move a synthetic feather from a source to a destination in the air. Stylistic motion planning should find a motion that not only satisfies the position constraints but also makes the fake feather float and tumble like a real one.

We designed a technique to achieve stylistic motion planning by generalizing a probabilistic robotic motion planning algorithm called Rapidly-Exploring Random Trees (RRTs) [6, 24]. The idea is to adapt this algorithm to assemble a path only using trajectory segments from the precomputed motion database. The trajectory segments represent motion units with a certain desired style.

Motion planning based on RRTs does bidirectional path planning by extending two random trees from the source and destination. During the planning, new path segments avoiding obstacles are appended to the leaves or internal nodes of the two trees in a probabilistic manner so that both trees keep growing until a segment from one tree comes into contact with a segment from the other, which indicates a path has been found. This technique can achieve better performance than other existing path finding algorithms, including probabilistic roadmaps used by Choi et al. [10]. In the original RRTs, all the path segments have the same shape, such as a line segment or a circular arc, and there are no continuity requirements between consecutive segments in a path.

In our stylistic planning, every branch on the two trees has to be a segment sampled from the motion database. Adjacent segments in the trees are likely to be different. In addition, consecutive branches should maintain a certain degree of continuity to make the final motion look natural. In order to be able to evaluate the smoothness of the connection between two consecutive segments as well as facilitate smooth blending between them, we enforce an overlapping portion between them.

To extend a new segment from one tree, we randomly generate a seed point in the search space and identify the closest node from the tree as the *seed node* (Fig. 3). We search the database to sample a segment that satisfies two conditions. (1) the segment can be smoothly connected to the parent branch at the seed node after a rigid body transformation. (Note that the path may become unphysical once a rigid body transformation has been applied. However, the subsequent optimization introduced in the next section can remove the artifacts.) (2) Once one endpoint of the segment is attached to the seed node, the other endpoint is closest to the seed point (Fig. 3a). If the sampled segment hits any obstacle (collision detection) once in position, we simply discard the segment and regenerate a seed point. Otherwise, we compute the minimum connection cost between the new branch and any branch in the other tree and update the minimum connection cost between the two trees. The connection cost between two branches measures the total amount of translation and rotation necessary for a smooth connection and

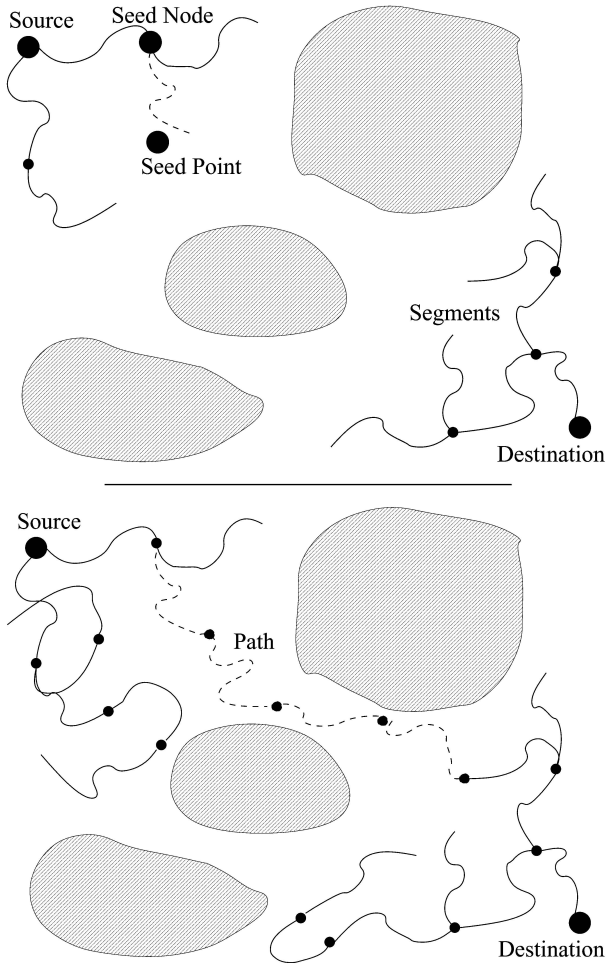

Fig. 3. Path planning by growing two probabilistic trees from the source and destination, respectively. *Top*: A new segment (*dashed branch*) is inserted into the source tree given a seed point. *Bottom*: a path (*dashed*) is found when two branches from the two trees become connected

will be formally defined in Eq. 1. The planning process stops whenever the minimum connection cost drops below a threshold or a maximum number of tree branches have been grown.

In the next stage, we form a path between the source and destination by joining the two tree branches (one from each tree) that actually achieve the minimum connection cost between the two trees (Fig. 3b). Obviously, this last connection introduces the worst continuity in the whole path. Therefore, we uniformly redistribute the discontinuity to all connections in the path. For efficiency, so far we have been using the representative segments at the top level of the tree-structured clusters in the database. Next, we refine the continuity at all connections by descending into lower levels of the clusters. Since every segment s_i in the path except the first and the last has two connections with its preceding and following segments, we traverse the subtree rooted at s_i in the database to potentially find another segment that gives the optimal continuity at these two connections. Finally, we blend every two consecutive segments in their overlapping portion.

To significantly improve performance, we precompute a lookup table that records the connection cost between any pair of representative segments at the top level of the tree-structured clusters. This computation only considers the overlapping portion of the segments (the tail window of the first segment and the head window of the second segment) and finds a translation/rotation matrix $\boldsymbol{M}$ such that those two windows will have best matching. This is equivalent to the following minimization:

$$\text{argmin}_{\boldsymbol{M}} \sum_j \|\boldsymbol{x}_1^j - \boldsymbol{M}\boldsymbol{x}_2^j\|^2 , \tag{1}$$

where $\boldsymbol{x}_i^j$s represent the points in the two windows. Note that the L^2 norm can be replaced by other measures. An approximate solution for the matrix $\boldsymbol{M}$ can be found by solving the principal axes of the two windows using principal component analysis (PCA) and aligning the two local frames defined by the two sets of principal axes. At a smaller computational cost, this approximate solution provides very good results in practice compared to true optimization. If the connection cost estimated using the matrix $\boldsymbol{M}$ is larger than a threshold, we mark the corresponding entry of the lookup table as "not connectible"; otherwise we record the matrix in the table.

5 Optimization

We resort to a large-scale spacetime optimization at the end to improve the initial dynamic appearance produced by motion planning as well as better enforce additional constraints other than position constraints. The motivation is twofold. Certain transitions in motion planning may not be physically realizable since the segments were simply concatenated and blended together. Furthermore, spatial coherence due to fluid motion was not enforced since the trajectories for multiple objects were planned independently. Although we follow SPH (Appendix B) in this section, the basic ideas of optimization can be presented relatively independently of the details in the SPH formulations. In the remainder of this section, we will focus on these basic ideas.

The optimization searches for a plausible fluid dynamic simulation that satisfies all user-defined requirements. Since the objects and gas are coupled together with interactions, we need to define a work space filled with gas particles in addition to the object particles, even though the final solution we seek only concerns the objects. We choose to consider the state variables (positions) of all the particles and their derivatives as the set of unknowns.

Thus, the actual SPH formulations are enforced as additional optimization constraints. The initial trajectories of the object particles are obtained from path planning, while the gas particles are initially positioned on a regular grid with zero velocities.

In addition to the state variables of the particles and their derivatives, the force field also needs optimization. We employ a hybrid representation of the force field, parametric vortex fields plus a nonparametric wind field, which is required to have smooth direction and magnitude throughout the 4D spacetime. A vortex is basically a rotational vector field with a field strength deteriorating with distance and time. A parametric vortex model (Appendix C) is a 4D function of space and time and is denoted as $\boldsymbol{f}_{vtx}(\boldsymbol{x}, t)$.

Let us denote the state of all the particles as $\boldsymbol{q} = \{\boldsymbol{x}_i\}_{i=0}^{n-1}$ and its derivative as $\dot{\boldsymbol{q}} = \{\boldsymbol{v}_i\}_{i=0}^{n-1}$. For the jth frame, we denote the state variables and their derivatives as $\boldsymbol{q}_j = \{\boldsymbol{x}_i^j\}_{i=0}^{n-1}$ and $\dot{\boldsymbol{q}}_j = \{\boldsymbol{v}_i^j\}_{i=0}^{n-1}$, respectively, where $\boldsymbol{x}_i^j$ and $\boldsymbol{v}_i^j$ indicate the position and velocity, respectively, of the ith particle in the jth frame. Thus, the complete set of variables that need optimization is represented as $S = \{\boldsymbol{q}_j, \dot{\boldsymbol{q}}_j, \{\boldsymbol{F}_i^j\}_{i=0}^{n-1} | 0 \le j < f\}$, where $\boldsymbol{F}_i^j$ represents the force vector at $\boldsymbol{x}_i^j$. Note that $\boldsymbol{F}_i^j = \boldsymbol{F}_{vtx4d}(\boldsymbol{x}_i^j, j) + \boldsymbol{h}_i^j + m_i g$, where $\boldsymbol{F}_{vtx4d}(\boldsymbol{x}_i^j, j)$ represents the accumulated vortex force at $\boldsymbol{x}_i^j$, $\boldsymbol{h}_i^j$ represents the wind force at $\boldsymbol{x}_i^j$, and $m_i g$ is the gravity. In what follows, we actually only consider the vortex component of $\boldsymbol{F}_i^j$ as a free variable while the wind component is implicitly constrained.

Our spacetime optimization is a constrained optimization with a cost function and a set of hard constraints. The user-defined requirements are treated as hard constraints. The complete set of such constraints is denoted as $C(S) = \{C_k^j | 0 \le k < m_j, 0 \le j < f\}$, where m_j is the number of constraints at the jth frame and C_k^j is a constraint on either $\boldsymbol{x}_i^j$ or $\boldsymbol{v}_i^j$. For example, a constraint C_k^j on $\boldsymbol{x}_i^j$ has the form $\|\boldsymbol{x}_i^j - \boldsymbol{o}_k\| \le b_k$. A strict positional constraint is imposed when b_k is set to zero. A constraint on $\boldsymbol{v}_i^j$ can be defined similarly.

The cost function for the optimization consists of three terms:

$$E(S) = c_1 E_x(S) + c_2 E_v(S) + c_3 E_c(S)\,, \tag{2}$$

where c_is are coefficients indicating the importance of each term, E_x and E_v represent the penalty terms to enforce the SPH formulation, and E_c represents the spatiotemporal smoothness of the velocity field. To explain the details in these terms, let us consider the jth frame. Given the particle positions and velocities associated with the $(j-1)$th frame, we should be able to run SPH simulation for one frame and predict these quantities for the jth frame. Let us denote the predicted quantities as $\tilde{\boldsymbol{x}}$ and $\tilde{\boldsymbol{v}}$. If the SPH formulation is strictly observed, $\boldsymbol{x}_i^j = \tilde{\boldsymbol{x}}_i^j$ and $\boldsymbol{v}_i^j = \tilde{\boldsymbol{v}}_i^j$. Otherwise, we penalize the discrepancies between them.

$$\begin{aligned} E_x(S) &= \sum_{ij} \|\tilde{\boldsymbol{x}}_i^j - \boldsymbol{x}_i^j\|^2\,, \\ E_v(S) &= \sum_{ij} \|\tilde{\boldsymbol{v}}_i^j - \boldsymbol{v}_i^j\|^2\,. \end{aligned} \tag{3}$$

Note that the SPH formulation is not enforced as hard constraints in the optimization simply to make it tractable because a large number of hard constraints are much more expensive to maintain than soft penalty terms.

To enforce the spatiotemporal smoothness of the velocity field, we require the velocity at $\boldsymbol{x}_i^j$ to be close to the average velocity of nearby particles in the same frame as well as average velocity of the same particle in the previous and subsequent frames; thus E_c has the form:

$$\begin{aligned} E_c(S) = c_{3,1} &\left(\boldsymbol{v}_i^j - \frac{\sum_{k \in N_i^j} \boldsymbol{v}_k^j}{|N_i^j|} \right)^2 \\ + c_{3,2} &\left(\boldsymbol{v}_i^j - \frac{\sum_{k=j-w}^{j+w} \boldsymbol{v}_i^k}{2w} \right)^2\,, \end{aligned} \tag{4}$$

where coefficients $c_{3,1} + c_{3,2} = 1$, N_i^j represents the index set of the neighboring particles of $\boldsymbol{x}_i^j$ in the jth frame, and w represents the size of the temporal window.

Compared to traditional spacetime constraints [50], our formulation introduces particle velocities as additional variables in the optimization while not directly optimizing the wind forces. This is because the SPH equations do not allow the velocities to be easily expressed as finite differences of positions. Therefore, they cannot be treated as derived variables. On the other hand, it is no longer necessary to optimize the wind forces because the smoothness of the velocity field implicitly constrains the wind forces.

We optimize the cost $E(S)$ and vortex force fields in two alternating stages, giving vortex fields higher priority because they introduce interesting fluid motion. Given an estimate of the $\boldsymbol{x}_i^j$ and $\boldsymbol{v}_i^j$, we can estimate $\boldsymbol{F}_i^j$ from Eq. 9. Meanwhile, we also inherit the annotations on the object trajectories obtained from motion planning. When a segment of the trajectories is annotated with a vortex field, we fit the parametric spatiotemporal vortex model to the estimated forces at the points along that segment. The parameters of a vortex model include the spatiotemporal center, orientation, strength, and duration. After the vortex fitting stage, we fix the vortex fields and optimize $E(S)$ subject to constraints to update all the state variables except the user-defined ones. We alternate these two stages a few times to obtain the final solution. The optimization is carried out using sequential quadratic programming [18].

There are two additional details regarding this large-scale optimization. First, to make the initial trajectories from path planning have sufficient influence on the final solution, we allow gradually larger deviations from the initial trajectories as the optimization progresses. At the beginning, every point on the trajectories is enforced as a position constraint with gradually increasing radius. When the radius has become sufficiently large, we then remove the position constraint. Without this strategy, the initial trajectories may be destroyed fairly early in the optimization process without much influence on the final solution. Second, it may not be feasible to optimize all the variables simultaneously on a machine with a limited amount of memory. We only optimize a subset of the variables at a time with the rest of the variables fixed. The optimization is performed repeatedly on disjoint subsets of variables but does not wait until convergence on each subset to avoid local minima.

6 Rotation refinement

Since we only use a small number of particles to represent each object during the spacetime optimization using SPH, we do not expect the rotational components to be very accurate. Therefore, we only obtain the translational component at the center of mass of each object by averaging the translations of all the particles of that object. More accurate rotational components with better visual effects are estimated in a postprocessing step where a polygonal model is used for each object.

The rotation of the objects in an animation is based on a simulated torque from air resistance. The force that creates the torque is a drag force that acts in the opposite direction of the velocity of the moving object. The formula for the magnitude of this drag force is

$$F_d = 0.5\, c_d\, \rho\, v^2\, A\,, \tag{5}$$

where c_d is the drag constant specific to the object, ρ is the density of the gaseous medium, v is the velocity of the center of mass of the object, and A is the foreshortened area over which the force acts.

In order to approximate the torque efficiently, the shape of the object is projected onto a plane perpendicular to the direction of the drag force acting on the object. A new coordinate system within this plane is centered on the object's center of mass. The object is then rasterized, and the torque on the object is compiled by iterating across all the pixels covered by this projected object. The original torque equation simplifies to two separate ones (one for each dimension within the plane in this new coordinate system). After calculating and summing the torques contributed by individual pixels, the resulting accumulated torque is still specified in this new coordinate system. A simple transformation from this local frame to the global coordinate system gives the torque in terms of the global coordinates. The torque and angular velocity of the object can then be integrated in order to animate the object. In addition, damping based on the object's angular velocity can be used to simulate rotational drag forces.

7 Results

To create the models used in the animations, digital photographs of leaves and feathers are used. The shapes are outlined and then triangulated, and these resulting meshes are texture-mapped with the original images. The vertices of the models are displaced in order to curve the leaves and feathers. The models are aligned with one particular axis and centered such that any rotation is about this center of mass.

Multiple examples are shown in Figs. 4–9. These examples either involve a single object or multiple objects. Figures 4 and 5 use one single feather to demonstrate our stylistic path planning technique. In such an example, we always specify a source, where the feather should take off, and a destination, where it is supposed to land. In addition, constraints or obstacles are also placed in the middle of the trajectory. Figure 4 shows two examples with intermediate position constraints and another two examples with partially constrained trajectories. In these examples, the whole trajectory is actually divided into a few nonoverlapping portions each of which has its own starting and ending points, and each nonconstrained portion of the trajectory is planned separately using our algorithm while a desired level of continuity between consecutive portions is maintained. Figure 5 shows obstacle avoidance by placing a maze between the source and destination. Note that we still perform optimization on the planned trajectories along with additional fluid particles, even though there is only one trajectory in each example. For a 300-frame trajectory, the path planning stage took less than 15 min on an AMD 2100+ processor. The optimization stage took approximately 2 h.

Figures 6–9 demonstrate fluidlike group behaviors by using multiple objects such as leaves and feathers. Typically, there is one or more keyframes in each of these examples. Each keyframe specifies the positions of all the objects. The velocities at a keyframe can also be constrained as an option. The objects must satisfy these constraints simultaneously. The examples show two types of keyframes. The first type specifies a static configuration that should be satisfied when the objects fall to the ground, while the second type is inserted in the middle of an animation when the objects are still moving. Because mul-

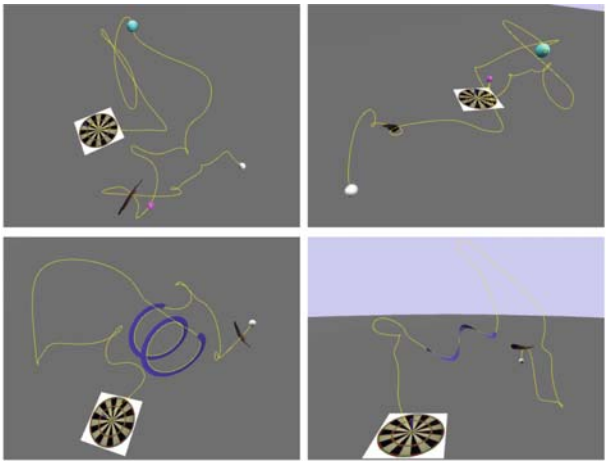

Fig. 4. *Top row*: Two examples of single object motion that satisfies two intermediate position constraints (*red* and *green spheres*). *Bottom row*: Two examples of single object motion with a partially constrained trajectory (*blue spirals*). The *white sphere* is the source and the *cylinder* on the *dart board* is the destination. All trajectories are generated with path planning and optimization

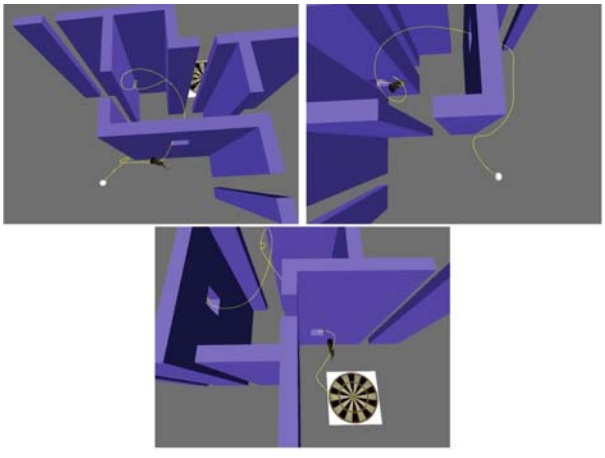

Fig. 5. An example of stylistic path planning with obstacle (*maze*) avoidance. The *white sphere* is the source and the *cylinder* on the *dart board* is the destination

tiple objects indirectly interact with each other, the optimization stage becomes slower to converge and, therefore, more expensive. For a 300-frame animation, it took approximately 5 h for the optimization to converge, while the motion planning stage typically took less than 2 h.

Note that the motion database is precomputed and the same database is used for all the examples. In all the examples, the total number of gas particles is 2500 and the number of object particles is 125. The actual number of variables in the optimization is approximately six times as many as the number of particles since each particle needs six variables to describe its position and velocity. A small number of extra variables are necessary for the parameters in the vortex model.

After optimization, the strength of the vortices and the strength of the wind fields are typically comparable with the same order of magnitude. The wind fields change smoothly with time and location. The average duration of a vortex is about 7 s. The average diameter of a vortex is around one third of the width of the simulation volume.

8 Conclusions and discussions

In this paper, we developed a method to realistically synthesize controllable motion for a single or a group of lightweight rigid objects in a gaseous medium. We adopted an interesting formulation that considers this problem as a large-scale spacetime optimization with user requirements and fluid motion equations as constraints. To make this problem tractable, we designed a novel technique to solve the optimization by taking results from a data-driven motion synthesis approach as initial solutions and extracting vortex fields in a separate stage using both annotations and parametric model fitting. Experiments indicate our method can effectively generate desirable results.

In our experiments the user can usually obtain a satisfactory profile of the trajectories from the motion planning stage. The motion synthesis algorithm is fast (one set of results typically takes about 10 min), so animators can redo the motion planning and/or impose more require-

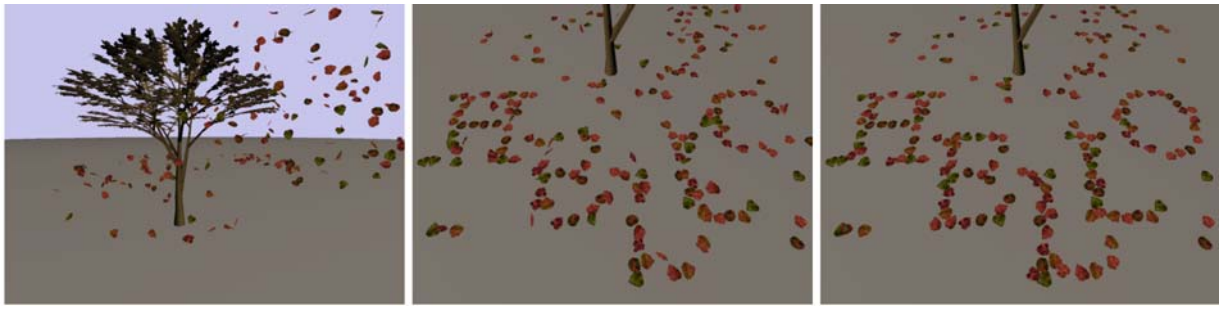

Fig. 6. Leaves fall onto the ground, forming letters

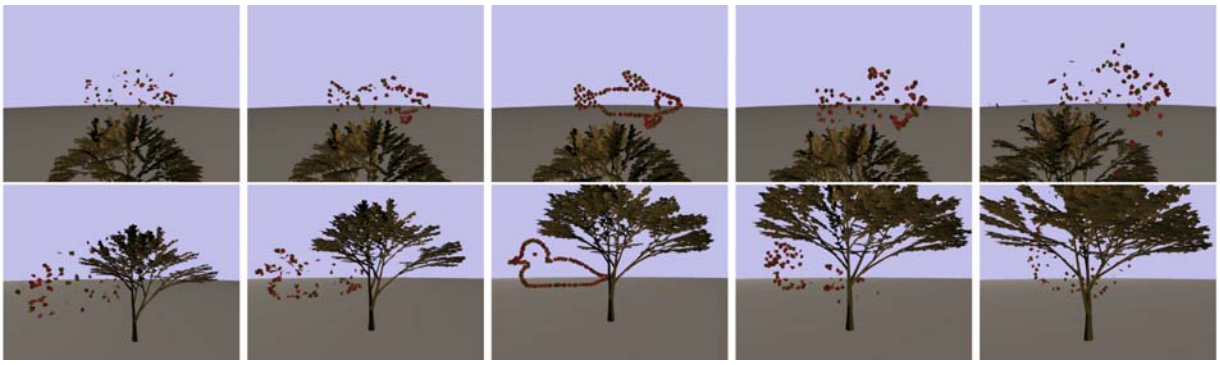

Fig. 7. Leaves form different shapes in the air

Fig. 8. Leaves form shapes on the ground after being disturbed by external forces

ments upon the system if they are not satisfied with the results. Once acceptable initial trajectories are obtained, optimization makes them more coherent and fluidlike. At this stage, usually only relatively small changes will be made to the motion.

Performance of the optimization system is not easy to predict given a set of trajectories and constraints. However, the cost function never increases on an iteration, so the "goodness" of the result is always improving. Animators can stop once acceptable results are generated.

We use incompressible SPH formulas to simulate air because only the objects inside are of interest, not the whole particle set. But this fluid model is only enforced

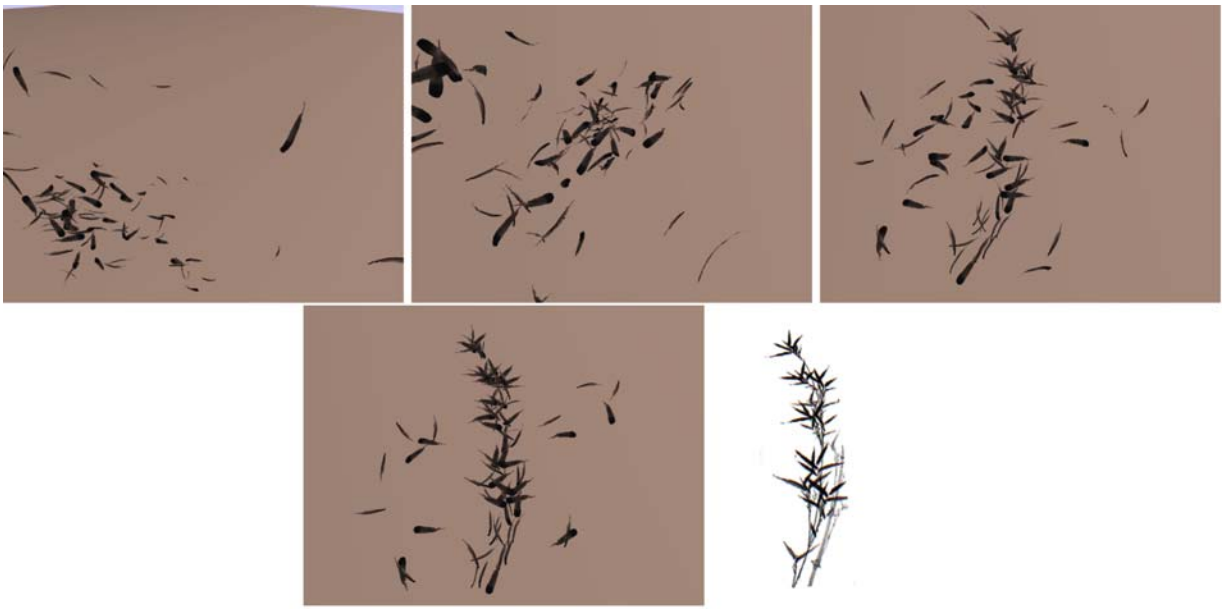

Fig. 9. Feathers fall onto the ground, forming a pattern similar to a painting (*rightmost*)

through soft constraints, so a little compressibility is allowed to make the system less stiff.

A Notation

m	Mass	t	Time
ρ	Density	$\boldsymbol{x}$	Position
P	Pressure	$\boldsymbol{v}$	Velocity
τ	Thermal energy	$\boldsymbol{q}$	State
E	Cost function	$\triangle\boldsymbol{v}$	XSPH variant
		$\boldsymbol{F}$	Force

B Smoothed particle hydrodynamics

Smoothed particle hydrodynamics represents a fluid as a collection of moving elements, particles, with local fluid characteristics. Each particle has a mass, position, and velocity and is influenced by forces such as gravity. In addition, an SPH particle also has local fluid characteristics such as density and pressure. The idea behind SPH is the determination of characteristics of fluid by interpolating from the set of unorganized particles. The interpolation is performed as a weighted sum over particles within a local region defined by a smoothing length h. The weighting scheme is defined by a smoothing kernel $w(r, h)$, which can be a Gaussian or a polynomial with a finite support. The smoothing length defines the scale of the support. As an example, the smoothed estimate of the density at particle i can be formulated as $\rho_i = \sum_j m_j w_{ij}$, where $w_{ij} = w(\|\boldsymbol{x}_i - \boldsymbol{x}_j\|, h)$.

Computing the gradient of an interpolated property is done using the gradient of the smoothing kernel, giving a smoothed estimate of the gradient of the property. Thus, the SPH versions of the Lagrangian equations of motion,

$$\frac{d\rho}{dt} = -\rho\nabla\cdot\boldsymbol{v} \tag{6}$$

$$\frac{d\boldsymbol{v}}{dt} = -\frac{1}{\rho}\nabla P \tag{7}$$

$$\frac{d\tau}{dt} = -\frac{P}{\rho}\nabla\cdot\boldsymbol{v}\,, \tag{8}$$

can be written as:

$$\begin{aligned}
\frac{d\rho_i}{dt} &= \sum_j m_j \boldsymbol{v_{ij}}\cdot\nabla_i w_{ij}\,,\\
\frac{d\boldsymbol{v_i}}{dt} &= -\sum_j m_j\left(\frac{P_i}{\rho_i^2}+\frac{P_j}{\rho_j^2}+\Pi_{ij}\right)\nabla_i w_{ij} + \boldsymbol{F_i}\,,\\
\frac{d\tau_i}{dt} &= \frac{1}{2}\sum_j m_j\left(\frac{P_i}{\rho_i^2}+\frac{P_j}{\rho_j^2}+\Pi_{ij}\right)\boldsymbol{v_{ij}}\cdot\nabla_i w_{ij}\,,
\end{aligned} \tag{9}$$

where $\boldsymbol{v_{ij}} = \boldsymbol{v_j} - \boldsymbol{v_i}$, Π_{ij} is an artificial viscosity added to handle shocks, and $\boldsymbol{F_i}$ is the external force on particle i. The interpretation of other notations is given in Appendix A. These three equations maintain the conservation of mass, momentum, and thermal energy, respectively. Particle interactions are implicitly handled by the smoothing kernel.

In addition to these equations, an *equation of state* must be used to fully describe the behavior of fluid. This equation defines a functional relationship between temperature, density, and pressure. An example of the equation for an ideal gas is $P = (\gamma - 1)\rho\tau$, where γ is a parameter that depends on the gas being simulated.

To put these equations in perspective, we summarize the steps a typical SPH simulation goes through. Particles are initialized to have an initial position, velocity, mass, and energy, and the system is evolved as follows:

```
Update particle densities
Update particle pressures using
           the equation of state
while (time < end_of_time)
 for all particles
  Calculate acceleration due to pressure
                                 gradient
  Calculate rate of change of thermal
                                 energy
 for all particles
  Update position
  Update velocity
  Update thermal energy

 Update particle densities
 Update particle pressures
 Calculate new time step
 time += new_timestep
```

Appropriate numerical schemes for SPH include the leapfrog algorithm and Runge–Kutta methods. Because of the finite smoothing length, the number of pairwise interactions is actually proportional to the number of particles, making SPH simulations efficient if a spatial data structure is used.

In order to simulate incompressible or nearly incompressible (such as a stiff gas) fluids, Monaghan [32, 33] adapted the original SPH. To prevent interpenetrations, a velocity correction, called the XSPH variant, is added when the particle positions are updated. XSPH variant is defined to be

$$\triangle \mathbf{v}_i = \varepsilon \sum_j \frac{m_j \boldsymbol{v}_{ji}}{\bar{\rho}_{ij}} w_{ij} \,, \tag{10}$$

where $\bar{\rho}_{ij} = (\rho_i + \rho_j)/2$. A particle position is updated as follows:

$$\boldsymbol{x}_i' = \boldsymbol{x}_i + \triangle t(\boldsymbol{v}_i + \triangle \boldsymbol{v}_i) \,. \tag{11}$$

In addition, a different equation of state, which keeps compressibility below a few percent, is also adopted:

$$P_i = P_0 \left[\left(\frac{\rho_i}{\rho_0} \right)^{\gamma} - 1 \right] , \tag{12}$$

where P_0 is the reference pressure.

C A parametric vortex model

A vortex in 3D can be defined parametrically using a pair of poles $\boldsymbol{p_1}$ and $\boldsymbol{p_2}$ as follows [27]:

$$\boldsymbol{f}_{vtx3d}(\boldsymbol{x}) = \omega \frac{(\boldsymbol{r_0} \cdot \boldsymbol{r_1}/\|\boldsymbol{r_1}\| - \boldsymbol{r_0} \cdot \boldsymbol{r_2}/\|\boldsymbol{r_2}\|)}{4\pi \|\boldsymbol{r_1} \times \boldsymbol{r_2}\|^2} \boldsymbol{r_1} \times \boldsymbol{r_2} \,, \tag{13}$$

where $\boldsymbol{r_0} = \boldsymbol{p_2} - \boldsymbol{p_1}$, $\boldsymbol{r_1} = \boldsymbol{x} - \boldsymbol{p_1}$, $\boldsymbol{r_2} = \boldsymbol{x} - \boldsymbol{p_2}$, and ω is the strength of the vortex. We define a spacetime vortex as $\boldsymbol{f}_{vtx}(\boldsymbol{x}, t) = G(t)\boldsymbol{f}_{vtx3d}(\boldsymbol{x})$, where $G(t)$ is a Gaussian in the temporal space.

References

1. Anderson M, McDaniel E, Chenney S (2003) Constrained animation of flocks. In: SCA '03: Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on computer animation, pp 286–297. Eurographics Association, Aire-la-Ville, Switzerland
2. Arikan O, Forsyth DA, O'Brien JF (2003) Motion synthesis from annotations. ACM Trans Graph 22(3):402–408
3. Barzel R, Hughes JF, Wood DN (1996) Plausible motion simulation for computer graphics animation. In: Proceedings of the Eurographics workshop on computer animation and simulation '96. Springer, Berlin Heidelberg New York, pp 183–197
4. Brand M, Hertzmann A (2000) Style machines. In: SIGGRAPH '00: Proceedings of the 27th annual conference on computer graphics and interactive techniques. ACM Press/Addison-Wesley, New York, pp 183–192
5. Bruderlin A, Williams L (1995) Motion signal processing. In: SIGGRAPH '95: Proceedings of the 22nd annual conference on computer graphics and interactive techniques. ACM Press, New York, pp 97–104
6. Cheng P, Shen Z, LaValle SM (2001) Rrt-based trajectory design for autonomous automobiles and spacecraft. Arch Control Sci 11(3–4):167–194
7. Chenney S (2004) Flow tiles. In: SCA '04: Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on computer animation. ACM Press, New York, pp 233–242
8. Chenney S, Forsyth DA (2000) Sampling plausible solutions to multi-body constraint problems. In: SIGGRAPH '00: Proceedings of the 27th annual conference on computer graphics and interactive techniques. ACM Press/Addison-Wesley, New York, pp 219–228
9. Chenney S, Forsyth DA (2000) Sampling plausible solutions to multi-body constraint problems. In: SIGGRAPH '00: Proceedings of the 27th annual conference on computer graphics and interactive techniques. ACM Press/Addison-Wesley, New York, pp 219–228
10. Choi MG, Lee J, Shin SY (2003) Planning biped locomotion using motion capture data and probabilistic roadmaps. ACM Trans Graph 22(2):182–203
11. Desbrun M, Cani MP (1999) Space-time adaptive simulation of highly deformable substances. Technical report, INRIA
12. Enright D, Marschner S, Fedkiw R (2002) Animation and rendering of complex water

surfaces. In: SIGGRAPH '02: Proceedings of the 29th annual conference on computer graphics and interactive techniques. ACM Press, New York, pp 736–744
13. Fattal R, Lischinski D (2004) Target-driven smoke animation. ACM Trans Graph 23(3):441–448
14. Fedkiw R, Stam J, Jensen HW (2001) Visual simulation of smoke. In: SIGGRAPH '01: Proceedings of the 28th annual conference on computer graphics and interactive techniques. ACM Press, New York, pp 15–22
15. Foster N, Fedkiw R (2001) Practical animation of liquids. In: SIGGRAPH '01: Proceedings of the 28th annual conference on computer graphics and interactive techniques. ACM Press, New York, pp 23–30
16. Foster N, Metaxas D (1997) Controlling fluid animation. In: CGI '97: Proceedings of the 1997 international conference on computer graphics. IEEE Press, Washington, DC, pp 178–188
17. Foster N, Metaxas D (1997) Modeling the motion of a hot, turbulent gas. In: SIGGRAPH '97: Proceedings of the 24th annual conference on computer graphics and interactive techniques. ACM Press/Addison-Wesley, New York, pp 181–188
18. Fsqp software. http://gachinese.com/aemdesign/FSQPframe.htm. Originally developed at the Institute for Systems Research, University of Maryland
19. Gersho A, Gray RM (1992) Vector quantization and signal compression. Kluwer, Norwell, MA
20. Gingold R, Monaghan J (1977) Smoothed particle hydrodynamics: theory and application to non-spherical stars. Monthly Notices R Astron Soc 181(2):375–389
21. Hadap S, Magnenat–Thalmann N (2001) Modeling dynamic hair as continuum. Eurographics Proceedings. Comput Graph Forum 20(3):329–338
22. James DL, Fatahalian K (2003) Precomputing interactive dynamic deformable scenes. ACM Trans Graph 22(3):879–887
23. Kovar L, Gleicher M, Pighin F (2002) Motion graphs. In: SIGGRAPH '02: Proceedings of the 29th annual conference on computer graphics and interactive techniques. ACM Press, New York, pp 473–482
24. Kuffner J, LaValle S (2000) Rrt-connect: an efficient approach to single-query path planning. In: Proceedings of the IEEE international conference on robotics and automation, pp 995–1001
25. Lee J, Chai J, Reitsma PSA, Hodgins JK, Pollard NS (2002) Interactive control of avatars animated with human motion data. In: SIGGRAPH '02: Proceedings of the 29th annual conference on computer graphics and interactive techniques. ACM Press, New York, pp 491–500
26. Li Y, Wang T, Shum HY (2002) Motion texture: a two-level statistical model for character motion synthesis. In: SIGGRAPH '02: Proceedings of the 29th annual conference on computer graphics and interactive techniques. ACM Press, New York, pp 465–472
27. Ling L, Damodaran M, Gay R (1996) Aerodynamic force models for animating cloth motion in air flow. Visual Comput 12(2):84–104
28. Lucy L (1977) A numerical approach to the testing of the fission hypothesis. Astron J 82(12):1013–1024
29. McNamara A, Treuille A, Popović Z, Stam J (2004) Fluid control using the adjoint method. ACM Trans Graph 23(3):449–456
30. Mihalef V, Metaxas D, Sussman M (2004) Animation and control of breaking waves. In: SCA '04: Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on computer animation. ACM Press, New York, pp 315–324
31. Miller G, Pearce A (1989) Globular dynamics: a connected particle system for animating viscous fluids. Comput Graph 13(3):305–309
32. Monaghan J (1992) Smoothed particle hydrodynamics. Annu Rev Astron Astrophys 30:543–574
33. Monaghan J (1994) Simulating free surface flows with sph. J Comput Phys 110:399–406
34. Müller M, Charypar D, Gross M (2003) Particle-based fluid simulation for interactive applications. In: SCA '03: Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on computer animation. Eurographics Association, Aire-la-Ville, Switzerland, pp 154–159
35. Pighin F, Cohen JM, Shah M (2004) Modeling and editing flows using advected radial basis functions. In: SCA '04: Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on computer animation. ACM Press, New York, pp 223–232
36. Premože S, Tasdizen T, Bigler J, Lefohn A, Whitaker RT (2003) Particle-based simulation of fluids. Eurographics Proceedings. Comput Graph Forum 22(3):401–410
37. Pullen K, Bregler C (2002) Motion capture assisted animation: texturing and synthesis. In: SIGGRAPH '02: Proceedings of the 29th annual conference on computer graphics and interactive techniques. ACM Press, New York, pp 501–508
38. Rasmussen N, Enright D, Nguyen D, Marino S, Sumner N, Geiger W, Hoon S, Fedkiw R (2004) Directable photorealistic liquids. In: SCA '04: Proceedings of the 2004 ACM SIGGRAPH/Eurographics symposium on computer animation. ACM Press, New York, pp 193–202
39. Reynolds CW (1987) Flocks, herds and schools: A distributed behavioral model. In: SIGGRAPH '87: Proceedings of the 14th annual conference on computer graphics and interactive techniques. ACM Press, New York, pp 25–34
40. Shi L, Yu Y (2005) Controllable smoke animation with guiding objects. ACM Trans Graph 24(1):140–164
41. Stam J (1999) Stable fluids. In: SIGGRAPH '99: Proceedings of the 26th annual conference on computer graphics and interactive techniques. ACM Press/Addison-Wesley, New York, pp 121–128
42. Tanco LM, Hilton A (2000) Realistic synthesis of novel human movements from a database of motion capture examples. In: HUMO '00: Proceedings of the workshop on human motion (HUMO'00). IEEE Press, Washington, DC, pp 137–142
43. Tang D, Ngo J, Marks J (1995) N-body spacetime constraints. J Visual Comput Animat 6:143–154
44. Terzopoulos D, Platt J, Fleisher K (1989) Heating and melting deformable models(from goop to glop). In: Proceedings of Graphics Interface, pp 219–226
45. Tonnesen D (1991) Modeling liquids and solids using thermal particles. In: Proceedings of Graphics Interface, pp 255–262
46. Treuille A, McNamara A, Popović Z, Stam J (2003) Keyframe control of smoke simulations. ACM Trans Graph 22(3):716–723
47. Unuma M, Anjyo K, Takeuchi R (1995) Fourier principles for emotion-based human figure animation. In: SIGGRAPH '95: Proceedings of the 22nd annual conference on computer graphics and interactive techniques. ACM Press, New York, pp 91–96
48. Wei X, Zhao Y, Fan Z, Li W, Yoakum–Stover S, Kaufman A (2003) Blowing in the wind. In: SCA '03: Proceedings of the 2003 ACM SIGGRAPH/Eurographics symposium on computer animation. Eurographics Association, Aire-la-Ville, Switzerland, pp 75–85
49. Wejchert J, Haumann D (1991) Animation aerodynamics. In: SIGGRAPH '91: Proceedings of the 18th annual conference on computer graphics and interactive techniques. ACM Press, New York, pp 19–22
50. Witkin A, Kass M (1988) Spacetime constraints. In: SIGGRAPH '88: Proceedings of the 15th annual conference on computer graphics and interactive techniques. ACM Press, New York, pp 159–168
51. Witkin A, Popović Z (1995) Motion warping. In: SIGGRAPH '95: Proceedings of the 22nd annual conference on computer graphics and interactive techniques. ACM Press, New York, pp 105–108
52. Yamane K, Kuffner JJ, Hodgins JK (2004) Synthesizing animations of human manipulation tasks. ACM Trans Graph 23(3):532–539

53. Yngve GD, O'Brien JF, Hodgins JK (2000) Animating explosions. In: SIGGRAPH '00: Proceedings of the 27th annual conference on computer graphics and interactive techniques. ACM Press/Addison-Wesley, New York, pp 29–36

54. Yoon HY, Koshizuka S, Oka Y (1996) A particle gridless hybrid method for incompressible flows. Int J Numer Methods Fluids 30(4):407 – 424




LIN SHI is a doctoral student in the Department of Computer Science at the University of Illinois at Urbana-Champaign. He received his M.S. in physics from New York University in 2001 and his B.S. in physics from Fudan University, China, in 1999. His research interests include physics-based animation and control, motion planning and optimization, texture synthesis, mesh processing, and computational geometry.

YIZHOU YU is currently an assistant professor in the Department of Computer Science at the University of Illinois at Urbana-Champaign. He received his Ph.D. in computer science from the University of California at Berkeley in 2000 and his M.S. in applied mathematics and B.S. in computer science from Zhejiang University, China, in 1994 and 1992, respectively. He has done research in computer graphics and vision including fluid and hair simulation, image-based modeling and rendering, texture analysis and synthesis, visibility and mesh processing, and radiosity and global illumination and has authored or coauthored more than 30 research papers. He is a recipient of the 2002 National Science Foundation Career Award and a 1998 Microsoft Graduate Fellowship. His current interests include data-driven models, physics-based modeling and animation, texture analysis and synthesis, and other computer graphics and vision problems.

CHRIS WOJTAN is a graduate student in the College of Computing at the Georgia Institute of Technology. His advisor is Professor Greg Turk, and he is currently supported by the National Science Foundation Graduate Research Fellowship. Chris received his B.S. in computer science at the University of Illinois at Urbana-Champaign in 2004. During the summer of 2004 he worked on scientific visualization at Lawrence Livermore National Laboratory. His interests include physically based animation and control of physical simulations.

STEPHEN CHENNEY is an assistant professor at the University of Wisconsin at Madison. His interest is primarily in animation research, broadly defined, with results ranging from controlling simulation to nonphotorealistic rendering of animation. He receieved a B.S. from the University of Sydney and a Ph.D. in computer science from the University of California at Berkeley.