Vis Comput (2014) 30:81–92
DOI 10.1007/s00371-013-0783-7

ORIGINAL ARTICLE

Real-time simulation of lightweight rigid bodies

Haoran Xie · Kazunori Miyata

Published online: 21 February 2013

Abstract Unlike common rigid bodies, lightweight rigid bodies have special and spectacular motions that are known as free fall, such as fluttering (oscillation from side to side) and tumbling (rotation and sideways drifting). However, computer graphics applications cannot simulate the dynamics of lightweight rigid bodies in various environments realistically and efficiently. In this study, we first analyze the physical characteristics of free-fall motions in quiescent flow and propose a new procedural motion-synthesis method for modeling free-fall motions in interactive environments. Six primitive motions of lightweight rigid bodies are defined in a phase diagram and analyzed separately using a trajectory-search tree and precomputed trajectory database. The global paths of free-fall motions are synthesized on the basis of these primitive motions by using a free-fall motion graph whose edges are connected in the Markov-chain model. Then, our approach integrates external forces (e.g., a wind field) by using an improved noise-based algorithm under different force magnitudes and object release heights. This approach exhibits not only realistic simulation results in various environments but also fast computation to meet real-time requirements.

Keywords Natural phenomenon · Motion synthesis · Lightweight rigid body · Real-time simulation

H. Xie (✉) · K. Miyata
School of Knowledge Science, Japan Advanced Institute of Science and Technology, Ishikawa, Japan
e-mail: xiehr@jaist.ac.jp

K. Miyata
e-mail: miyata@jaist.ac.jp

1 Introduction

Rigid-body simulations are widely used in applications ranging from movies to engineering and games. All lightweight objects do not fall straight down; for example, a piece of paper or a leaf wavers and flutters down in a seemingly unpredictable motion. The main challenge in the real-time simulation of lightweight rigid bodies is the requirement to effectively deal with chaotic principles, which have not been completely resolved in physics. The research on these principles has a rich history, beginning with James Maxwell.

The complexity of the motions of lightweight rigid bodies lies in the coupling of the forward motion of the object with lateral oscillations due to the surrounding fluid as well as the production and influence of the vortices around the object. The dynamics of the body of lightweight objects involves multiple hydrodynamic effects (e.g., lift force, drag force, and vortex shedding), which exhibit both regular and chaotic behaviors. Simulating these dynamics is challenging and promising for the visual simulation of many phenomena, and it is related to the research on unsteady dynamics such as flight aerodynamics, bubble-rising and boiling, meteorology, and hydrodynamics.

A common method to simulate the motions of a lightweight rigid body is to use a key-frame control, which requires the animator to exert considerable effort and possess a high level of expertise. Reliable motion can be simulated physically by modeling the dynamics of the surrounding flow. Although this model involves inertial forces and vortex effects, the approach is not suitable for real-time applications due to its high computational cost. In this study, a procedural motion-synthesis approach, which includes the effects of lift and drag forces, is proposed to efficiently simulate the realistic motions of lightweight rigid bodies. Moreover, this approach provides accurate simulation results un-

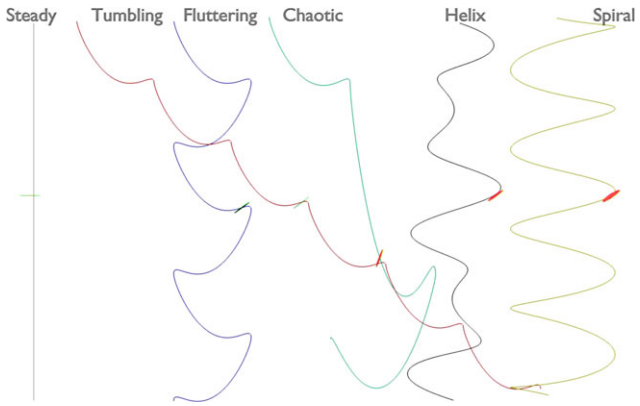


Fig. 1 Six primitive motions are summarized from experimental studies [1, 27]: steady descent, tumbling, fluttering, chaotic, helical, and spiral motions

der external forces (e.g., wind). Our major contributions are as follows:

- We propose a data-driven approach of motion synthesis, which uses a precomputed trajectory database and a free-fall motion graph.
- We apply a separate synthesis method, which uses six primitive motions (Fig. 1) of free-fall behavior, defined in a phase diagram that charts the dimensionless moment of inertia and the Reynolds number.
- To connect each primitive motion, we use a Markov chain model, based on the motion groups of these primitive motions, which allows an accurate estimation because of the apparent features of each primitive motion. The estimation is made in terms of a hypothesis about the global motion paths of lightweight rigid bodies, which has been verified experimentally.

The rest of this paper is organized as follows. Section 2 presents studies and technologies related to the proposed method. An overview of the proposed methodology is described in Sect. 3. In Sect. 4, we describe the modeling of primitive motions on a phase diagram. In Sect. 5, we discuss how to synthesize the global motion paths of free fall in a still fluid. The implementation of the wind field in the noise-based method is discussed in Sect. 6. The simulation results of free-fall motions in real time with and without wind conditions are presented in Sect. 7. The conclusion and possibilities for future studies are described in the last section.

2 Related work

Scientific interest in the phenomena of lightweight rigid bodies dates back one and a half centuries, when James Maxwell noticed the torque due to gravity and lift force in cases where forces do not act at the same point [11]. After Maxwell's research, a few achievements were made, but the problem remains unsolved.

A phase diagram using the Reynolds number and the dimensionless moment of inertia can differentiate the motion patterns observed in experiments [4]. Recently, experiments have found three additional typical trajectories in a three-dimensional (3D) environment: zigzag, transitional helical, and spiral motions [27]. Tanabe et al. [20] developed a simple phenomenological model of a falling paper by solving ordinary differential equations (ODEs), which are based on the Kutta–Joukowski theorem. In addition, another study [14] investigated the relationship among different parameters that can affect the path of a leaf's motion by performing more than 6,000 3D experiments. A few studies have focused on the numerical simulations of free fall in two dimensions. Andersen et al. [1] discussed the direct numerical simulation of the two-dimensional (2D) Navier–Stokes equation and presented a fluid force model based on ODEs derived from experiments and simulations. However, no convincing numerical simulation could successfully explain chaotic motion in 3D space.

In the field of computer graphics (CG), there is insufficient research that focuses on the simulations of lightweight rigid bodies in various environments. However, there are several simulations depicting the flow of a falling motion. Wei et al. [23] provided the lattice Boltzmann method for simulating soap bubbles and feathers in fluid simulation; however, this approach cannot produce a desired motion trajectory and has difficulty in simulating multiple objects in real time because it requires a high computational cost. For example, a single bubble requires 2.8 fps from the central processing unit (CPU) and 11.5 fps from the graphics processing unit (GPU); a single feather requires 0.76 fps from CPU and 6.1 fps from GPU. Related example-based approaches based on the Markov model [15], captured videos [2], segments from fluid simulations [16], trajectories animated by Maya [22], and sketches by a designer [6] were proposed. All these simulations ignore the natural motion of lightweight rigid bodies and consider it a completely complex and unpredictable dynamic motion, which is modeled by stochastic processes or a simple particle representation. Recently, a study was conducted on underwater rigid-body dynamics [24], in which the added-mass tensor was computed in a precomputation step, resulting in better simulation results of lightweight rigid bodies; but this approach is only suitable for simulating regular motions in a low Reynolds number flow.

Some commercial CG tools including Lightwave and Maya cannot animate a lightweight rigid body. Instead, they use particle simulation to model a falling object by adjusting the drag and lift parameters in a wind field (e.g., nCloth effect in Maya). In all these studies, the motion paths are unpredictable, and it is difficult to achieve a realistic motion.

Data-driven motion synthesis combines the controllability of procedural and physically based animation with the realistic appearance of a pre-recorded motion stream (e.g., motion capture). The first study about automatically organizing sample motion clips into graphs for effective motion synthesis proposed the use of a motion graph [3]. Later, Kovar et al. built an extended motion graph by using a local search with a branch-and-bound algorithm [9]. Besides their use in character animation, motion graphs are also used in other physical simulations such as tree animation [5, 26]. Our approach includes the motion graph technique for simulating the motion of a lightweight rigid body.

To obtain a wind field, a direct method is to simulate the turbulent flow under a boundary condition by solving differential equations using a Fourier filter [17, 18]. Another method is to simulate the motion in a wind field using noise functions (fractional Brownian motion) [8, 13]. The flow-based method provides physically accurate and realistic results but requires a high computational cost. On the other hand, the noise-based method is much simpler and more suitable for real-time simulation but at the cost of physical accuracy. To overcome the inaccuracy of the noise-based method, a physically based analysis of wind characteristics is essential.

3 Overview

Our simulation method is illustrated in Fig. 2. The method has two important steps: motion modeling and motion synthesis of free-fall motions. In the motion-modeling phase, the input parameters are introduced, including the physical characteristics of the object (such as release height and mass) and the fluid in which it is released. We transform these parameters into two key dimensionless numbers: the Reynolds number Re and the dimensionless moment of inertia I^*. Next we use a phase diagram to obtain the main primitive motions in which the motion of the object is stable.

To synthesize the primitive motions of a free fall, we use a trajectory-search tree to represent chaotic, fluttering, and tumbling motions, and develop a precomputed trajectory database to provide featured motion segments. The global trajectory is achieved in the motion synthesis phase with an assumed hypothesis of motion classification, which has been verified by numerous experiments. In the motion graph, primitive motion sequences are treated as nodes, and edges are connected with probabilities from the discrete-time Markov chain model. In addition, wind-field interactions with the falling object are calculated effectively. Finally, the final simulation of the lightweight rigid body is achieved in real time.

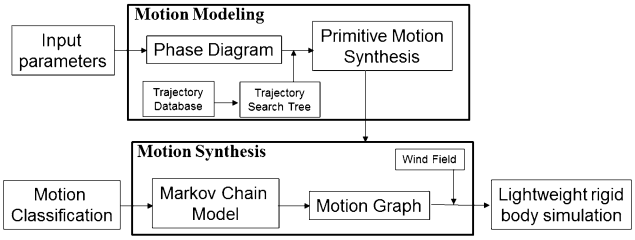


Fig. 2 Flow of our simulation method

4 Motion modeling

4.1 Input parameters

There are two important dimensionless quantities for representing the motion of a falling object: the Reynolds number Re and the dimensionless moment of inertia I^*, which are calculated using the following equations:

$$Re = \frac{UL}{\nu}, \tag{1}$$

$$I^* = \int_V \frac{\rho(x,y,z)}{\rho_f a^5} \times \begin{bmatrix} y^2+z^2 & -xy & -xz \\ -xy & z^2+x^2 & -yz \\ -xz & -yz & x^2+y^2 \end{bmatrix} dx\,dy\,dz, \tag{2}$$

where U is the velocity scale of the flow, L is the length of the object, and ν is the kinematic viscosity of the fluid; $\rho(x,y,z)$ describes the density function of the object. In special cases, $I^* = \frac{\pi\rho_s b}{64\rho_f a}$ (disk) and $I^* = \frac{8\rho_s(a^2+b^2)b}{3\pi\rho_f a^3}$ (rectangle), where a and b are the length and width of the cross-section of the object, respectively, and ρ_s and ρ_f are the density of the object and the surrounding flow, respectively.

Considering the object fixed in a uniform flow with velocity U, U is estimated by assuming that the buoyancy-corrected gravitational force $(\rho_s - \rho_f)Vg$ on the object is in balance with the drag force $S\rho_f U^2$ on the object, V and S are the volume and the area of the cross-section of the object, respectively [7]:

$$U = \sqrt{\frac{V}{S}\left(\frac{\rho_s}{\rho_f} - 1\right)g}. \tag{3}$$

The Re–I^* phase diagram (Fig. 3) was introduced in previous studies [4, 19, 25, 27]. The regimes in the diagram represent different primitive motions: steady descent (SD), periodic tumbling (PT), transitional chaotic (TC), periodic fluttering (PF), transitional helical (TH) and periodic spiral (PS) motions. The motion trajectories of these motions are illustrated in Fig. 1.

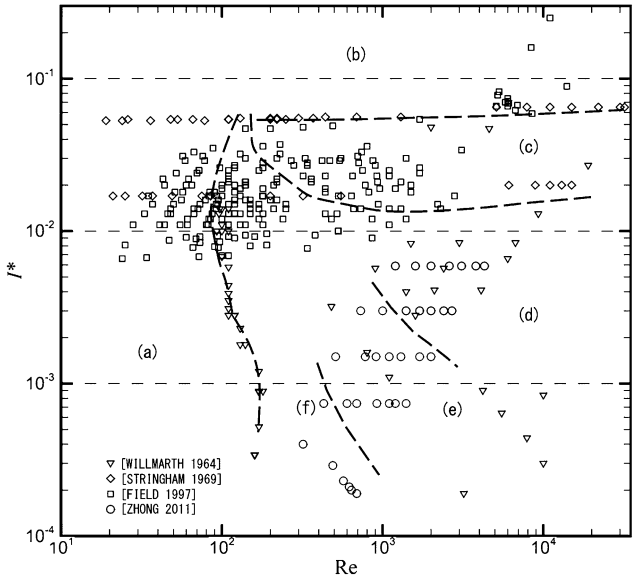


Fig. 3 Re–I^* phase diagram of free-fall motions including six regimes: (*a*) steady descent, (*b*) tumbling, (*c*) chaotic, (*d*) fluttering, (*e*) helical, and (*f*) spiral motions. The symbols in the diagram represent experimental results from previous studies

4.2 Precomputed trajectory database

It is difficult to build a trajectory database of free-fall motions because capturing the accurate trajectories of a falling lightweight object in the real world is apparently infeasible because of chaotic motions and a short time interval. By using fluid simulations to track vortex particles frame-by-frame by following the velocity vectors is also not suitable for the following reasons: (i) they cannot detect all primitive motions, (ii) they have difficulty in capturing realistic motion trajectories, and (iii) it is difficult to control such simulations because of various parameter adjustments.

Another approach is to use the Kutta–Joukowski theorem [20], which accounts for drag and lift forces, to solve the following 2D ODEs:

$$\begin{cases} \ddot{x} = -(A_\perp \sin^2\theta + A_\parallel \cos^2\theta)\dot{x} \\ \quad + (A_\perp - A_\parallel)\sin\theta\cos\theta\,\dot{y} \\ \quad - kL\pi\rho_f V^2 \cos\beta\cos\alpha/m, \\ \ddot{y} = -(A_\perp \cos^2\theta + A_\parallel \sin^2\theta)\dot{y} \\ \quad + (A_\perp - A_\parallel)\sin\theta\cos\theta\,\dot{x} \\ \quad + kL\pi\rho_f V^2 \cos\beta\sin\alpha/m, \\ \ddot{\theta} = -A_\perp\dot{\theta} - 3\pi\rho_f V^2 \cos\beta\sin\beta, \end{cases} \tag{4}$$

where (x, y) and θ are the position and angle of the center of mass of the object, respectively. In addition, (u, v) and ω are the linear and angular velocities of the object, respectively. It is important to note that $u = \dot{x}, v = \dot{y}, \omega = \dot{\theta}$, and $V^2 = \dot{x}^2 + \dot{y}^2$. Moreover, m is the mass of the object, which is calculated from the object's density and volume.

Fig. 4 (**a**) Fluttering trajectory determined by solving ODEs in Eq. (4) with $A_\perp = 4.1$, and $A_\parallel = 0.9$. (**b**) Impractical trajectory computed by solving ODEs in Eq. (4) with $A_\perp = 4.6$, and $A_\parallel = 0.15$

The parameters $A_\perp$ and $A_\parallel$ are the drag coefficients in the directions perpendicular and parallel to the object, respectively. The angles α and β are defined as $\alpha = \arctan(u/v)$, and $\beta = \alpha + \theta$. The parameter k is defined as follows:

$$k = \begin{cases} 1, & \text{sign}(v)\sin\beta \geq 0, \\ -1, & \text{sign}(v)\sin\beta < 0. \end{cases} \tag{5}$$

We applied the standard fourth-order Runge–Kutta algorithm to solve the second-order ODEs in Eq. (4), as shown in Fig. 4(a). Similar to a fluid-simulation approach, it is difficult to control the ODE model with the parameters $A_\perp$ and $A_\parallel$. For example, the calculated motion trajectory in Fig. 4(b) is impractical because it is unnatural for an object to fall vertically after a fluttering motion. Nevertheless, because Eq. (4) accounted for the effects of drag and lift forces, the object-orientation results by solving Eq. (4) are more accurate than other approaches.

There are two essential steps before building a trajectory database: the motion segmentation of free-fall trajectories and segment clustering. To obtain various motion segments, we use harmonic functions for describing general fluttering motions:

$$\begin{cases} x_t = x_0 - \frac{A_x}{\Omega}\sin(\Omega t), \\ y_t = y_0 - Ut - \frac{A_y}{2\Omega}\cos(2\Omega t), \end{cases} \tag{6}$$

where A_x and A_y are the amplitudes of vertical and horizontal velocities of the falling object generated by oscillations due to the surrounding viscous flow; Ω describes the angular frequency of the falling motion; and U is the average decent velocity. The segment breakpoints are selected as the turning points of the trajectory given at time steps $t_i = \frac{2k+1}{2\Omega}\pi$, and $k \in Z \geq 0$.

After segmentation, numerous motion segments are obtained by modifying the parameters A_x, A_y and U in Eq. (6) as shown in Fig. 5. A motion-segment set $(S_i | i = 1, 2, \ldots, N)$, where N is the number of segments, is classified on the basis of the value of the feature vector of each segment from the starting point P_i^0 to the end point P_i^1. Feature vector sets $V\{V_i = P_i^1 - P_i^0, i = 1, 2, \ldots, N\}$ are assigned into classes by using the K-means algorithm.

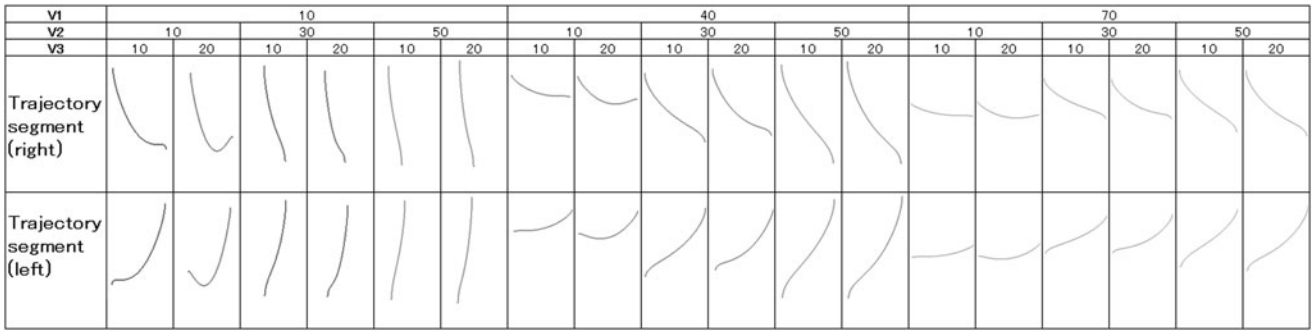


Fig. 5 Trajectory segments are obtained by the control of parameters in Eq. (6). V1, V2, and V3 represent A_x, A_y, and U, respectively. Ω is set at a constant value 9.8

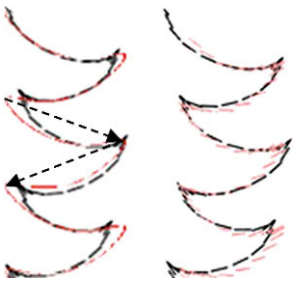


Fig. 6 Comparing the synthesized trajectory (*red*) and measured data (*black*) of fluttering motion for no orientation (*left*) and interpolated orientations (*right*). The *arrows* represent feature vectors

The orientation of the object in each frame of S_i is linearly interpolated by the angles calculated from Eq. (4) as shown in Fig. 6. Finally, the position and orientation data of segments are stored in a trajectory database.

4.3 Primitive motion synthesis

4.3.1 Trajectory-search tree

We compared the trajectories of chaotic, periodic fluttering, and tumbling motions from experimental data. Because the airflow behind a falling object reveals vortex shedding and turbulence, the object reaches turning points at which the angular velocity becomes zero and the velocity in the oscillation direction is also zero, but the velocity in the vertical direction is maximized. The object encounters the two alternatives of gliding left or gliding right (fluttering or tumbling). The simple structures of fluttering, chaotic, and tumbling motions are illustrated in Fig. 7. Every child of this tree structure represents a motion segment derived using the feature vector as the search key from the precomputed trajectory database.

4.3.2 Unified trajectory functions

While projecting the motion paths of primitive motions onto an X–Y plane, the curves of the six primitive motions have characteristic shapes as shown in Fig. 8: the steady-descent-motion trajectory appears as one point; the

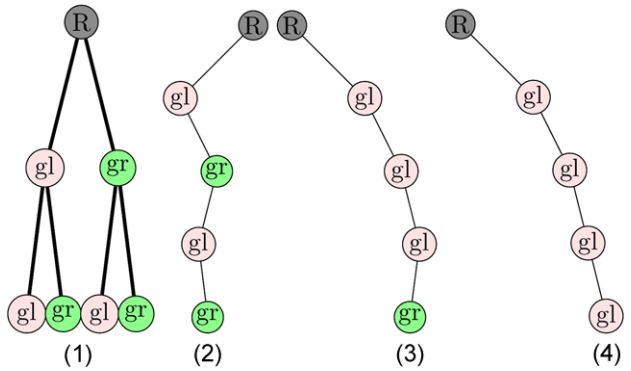


Fig. 7 (**1**) First two levels of a small trajectory-search tree, and the tree structures of (**2**) fluttering, (**3**) chaotic, and (**4**) tumbling motions created by the traversal of the four levels of the search tree (*gl*: glide left; *gr*: glide right)

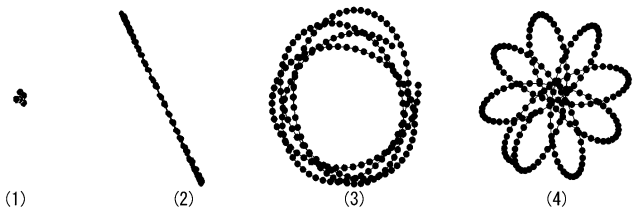


Fig. 8 Measured curves projected onto an X–Y plane: (**1**) steady descent motion; (**2**) fluttering, chaotic, and tumbling motions; (**3**) spiral motion; and (**4**) helical motion

fluttering-, tumbling-, and chaotic- motion trajectories are in a straight line; the spiral-motion trajectory is a circle; and the helical-motion trajectory is similar to an eight-petal rose curve. All these curves are represented by the following equations:

$$\begin{cases} x_t = A_e \cos(\Omega t)(1 + \epsilon_e \sin(k\Omega t)), \\ y_t = A_e \sin(\Omega t)(1 + \epsilon_e \sin(k\Omega t)), \\ z_t = h - Ut, \end{cases} \tag{7}$$

where A_e is the amplitude of the elliptical oscillation generated in the X–Y plane; ϵ_e is the aspect ratio of the minor

axis to the major axis of the oscillation ellipse; k is the ratio of the period of elliptical oscillation to that of rotation of the falling object; h is the height from where the object is released; and Ω is the angular frequency of the falling motion. As deduced from Eq. (7), the simple form of trajectories is as follows:

$$\begin{cases} \epsilon_e \to 0, & k = 1 : PS; \\ A_e \to 0, & k \to 0 : SD; \\ \epsilon_e \neq 0, & k = 4 : TH. \end{cases} \tag{8}$$

4.3.3 Initialization of primitive motions

Because the nature of chaotic motions is more complex than that of periodic motions (PF and PT), we synthesized a chaotic motion with a feature vector, $V = rV_0$, and an amplitude of oscillation, $A_e = rA_0$, where r is a random number between 0.1 and 10 calculated by the Box–Muller algorithm; V_0 is a feature vector that has an orientation equivalent to the release angle of the falling object; and A_0 is the initial amplitude of object oscillation.

In the experimental data mentioned in [14], the deviations of primitive motions D_i from the release position are given in a normal distribution having a Gaussian form $A_i e^{-(\frac{r-B_i}{C_i})^2}$, and they are linearly related to the release height h as determined by the following equation:

$$D_i = \frac{kB_iL}{a}, \tag{9}$$

where k is the deviation coefficient. Because the frequency of the tumbling motion is given as $\Omega \sim \sqrt{b}/a$ [10], the initial amplitude of oscillation, A_0, is determined as follows:

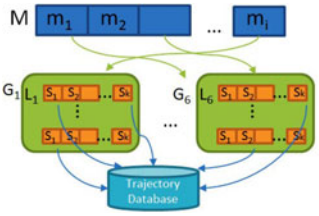
$$A_0 = \frac{D_iU}{h\Omega}, \tag{10}$$

where U is the average falling velocity determined from Eq. (3). The final synthesized trajectories of primitive motions are shown in Fig. 1.

5 Motion synthesis

5.1 Motion classification

The primitive motions $\{L_i|S_1, S_2, \ldots, S_k\}$ (where S_k is the kth segment in primitive motion L_i) are constituted by motion segments, S_n, from a precomputed trajectory database. Because primitive motions are the basic motions of lightweight rigid bodies observed in various experimental studies [4, 19, 25, 27], the motion groups $\{G_i|1 \le i \le 6\}$ are used to represent these similar motion sets of primitive motions.



(a)

Class 1	L_4
Class 2	$L_3 \to L_6, L_3, L_4 \to L_6$
Class 3	$L_1 \to L_2, L_1 \to L_6$
Class 4	$L_4 \to L_6, L_3$
Class 5	$L_2 \to L_6$
Class 6	L_4, L_6
Class 7	L_1

(b)

Fig. 9 (**a**) Motion classifications (*blue*: motion classes; *green*: motion groups; and *orange*: motion segments). (**b**) Motion classes observed in experiments

For a free-fall motion M, $M\{M = m_1 \parallel m_2 \parallel \cdots \parallel m_i\}$ is annotated by a label, which is set to L_i where L_i is a primitive motion. We define L_i as follows: L_1, L_2, L_3, L_4, L_5, and L_6 represent SD, PT, TC, PF, TH, and PS motions, respectively, as shown in Fig. 9(a). On the basis of thousands of experiments [14], the trajectories are classified into seven motion classes, as shown in Fig. 9(b). Note that L_3 and L_5 are not observed in study [14]; however, in our study, we consider the combination of (L_4, L_2) as L_3 and rotational motion as a motion accompanying primitive motions. Then, we make the following hypothesis:

Hypothesis *If* $M\{M = m_1 \parallel m_2 \parallel \cdots \parallel m_i\}$ *represents the motion of a lightweight rigid body in three dimensions, then* $m_i = L_{j_i} \in \{L_j|1 \le j \le 6\}$, *and the subscript sequence* $\{j_1, j_2, \ldots, j_i\}$ *should be an increasing sequence.*

We analyzed this hypothesis qualitatively. When an object starts falling from a release point, the vortexes are gradually generated behind the object because of the vorticity of the surrounding flow. Then, the motion of a lightweight rigid body becomes increasingly sensitive to internal forces including drag and lift forces.

In terms of this hypothesis, the number of potential motion classes of all primitive motions is determined as follows:

$$N = \sum_{i=1}^{i \le k} C_k^i, \quad k \in Z,\ k \in [1, 6], \tag{11}$$

where k is the level of the main primitive motion determined from the phase diagram by the calculated Re and I^* in Sect. 4.1.

5.2 Markov chain model

We focused on how to determine m_i for motion M. A first-order discrete-time Markov chain model is proposed for solving this issue.

We considered a discrete-time stochastic process $\{X_n\}$ with $N_0 \in Z \equiv i \in [1, 6]$ as the state space, which corresponds to motion groups $\{G_i|1 \le i \le 6\}$ (Fig. 9). The

Markov property asserts that the distribution of the random variable X_{n+1} in the process $\{X_n\}$ depends only on the current state $\{X_n = i_n\}$, instead of depending on the whole history $\{X_0 = i_0, \ldots, X_n = i_n\}$:

$$P[X_{n+1} = j | X_n = i_n] = P[X_{n+1} = j | X_0 = i_0, \ldots, X_n = i_n], \quad (12)$$

where $j, i_0, \ldots, i_n \in N_0$. The stochastic process $\{X_n\}$ is a Markov chain.

If we suppose that the state space N_0 is the motion group G and the process $\{X_n\}$ occurs for a discrete time set $\{X_t\}$, then the transition probability $p_{ij} = P[X_{t+1} = L_j | X_t = L_i]$ is the conditional probability for transition from primitive motion L_i to primitive motion L_j. The transition matrix is given as $P = (p_{ij})$ (Fig. 10). Because the process $\{X_t\}$ is stochastic, the following conditions should be satisfied:

$$p_{ij} \geq 0 \quad \text{and} \quad \sum_j p_{ij} = 1, \quad i, j \in [1, 6].$$

According to the hypothesis in Sect. 5.1, P has the following form:

$$P = \begin{pmatrix} Q & R \\ 0 & T \end{pmatrix}_{i \times j}. \quad (13)$$

Next, we discuss the realization of Markov chain $\{X_t\}$ and the calculation of the transition matrix P.

To obtain process $\{X_t\}$, the model starts with an initial state at time $t_0 = 0$. Then, an iteration step is executed for a transition from the state L_i at time t to state L_j at time $t + 1$, and the calculation depends on the probabilities at the ith row of the transition matrix P (i.e., $P_i = (p_{ij} | j = 1, 2, \ldots, 6)$).

The state-transition probabilities for the transition matrix P are found by counting the state transitions that occurred in the experimental data. If we set N_i as the number of all transitions from state L_i in the experimental data, and N_{ij} as the number of transitions from state L_i to state L_j, then the probability is given by $p_{ij} = \frac{N_{ij}}{N_i}$, as shown in Table 1.

The advantages of using this first-order discrete-time Markov chain model are as follows:

- The next motion is only related to the current state in the case of primitive motions.
- The features of each primitive motion are so apparent that a valid transition matrix can be successfully obtained from the experimental data.
- The computational cost of the model is low.

5.3 Graph construction

The special free-fall motion graph, shown in Fig. 11, is based on the motion graph described by Kovar et al. [9]. This graph is a complete directed graph: each node of the graph is connected to other nodes in the same graph. We used $G = (V, E)$ to represent a graph, where V is the node set, and E is the edge set. Every frame in a motion sequence of primitive motion appeared as a node in the motion graph; a transition splice in the motion sequence appeared as an edge between nodes. We searched this graph in one direction (from the top down) in the order of m_i according to the hypothesis presented in Sect. 5.1. Therefore, it is impossible for a motion to become a tumbling motion after a spiral motion. The transition probability was attached to the edge be-

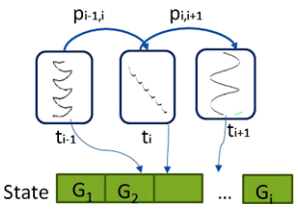


Fig. 10 Markov chain model states and transition probabilities

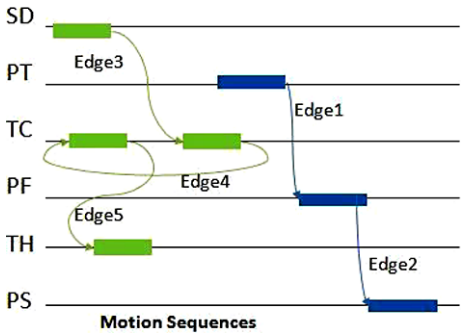


Fig. 11 Free-fall motion graph. A motion path represents a collection of splices between sequences. In this case, two example motions are shown

Table 1 State-transition probabilities used in this study

P	L_1	L_2	L_3	L_4	L_5	L_6
L_1	0.1538	0.3462	0.0385	0.0385	0.3077	0.1154
L_2	0	0.3261	0.0217	0.0870	0.3043	0.2609
L_3	0	0	0.4444	0.0833	0.2500	0.2222
L_4	0	0	0	0.6957	0.0870	0.2174
L_5	0	0	0	0	0.9231	0.0769
L_6	0	0	0	0	0	1.0000

tween the nodes using a discrete-time Markov chain model (Sect. 5.2).

6 Motion in wind

6.1 Wind field

Suppose the velocity of wind is $V = (V_u, V_v, V_w)$, where V_u, V_v, and V_w denote wind velocity components along the x-, y-, and z-axes of the coordinate system in the simulation, respectively, and let $U(h)$ be the mean wind velocity at height h, then according to the logarithmic wind law [21], $U(h)$ is given as follows:

$$U(h) = \frac{u_*}{k} \ln\left(\frac{h}{z_0}\right), \tag{14}$$

where u_* is the friction velocity (m/s); k is the von Karman's constant ($k = 0.40$); and z_0 is the roughness parameter (conceptually, it is the height at which V reduces to 0). The value of z_0 depends on the types of ground terrain (we selected $z_0 = 0.3$).

The fBm method can suitably represent the wind [12]. The spectral-density function of the wind field, based on Kolmogoroff's law, is given as follows:

$$S_u(n) = u_*^2 \left(\frac{U(h)\phi}{h}\right)^{2/3} \frac{C}{n^{5/3}}, \tag{15}$$

where $\phi = \epsilon k h / u_*^3$, ϵ is the dissipation rate according to Kolmogoroff's law, and C is a constant, such that $C = \alpha(2\pi k)^{-2/3}$, where α is determined experimentally as 0.5. Therefore, $C = 0.3$ for the u wind direction, and $C = 0.4$ for the v and w wind directions. To obtain the representation of fBm in the form of $S(f) = A/f^{\beta}$, where A is the amplitude in wind direction (u, v, w), we adopted the same approximations of A_u, A_v, and A_w as those by Khorloo et al. [8] as follows:

$$\begin{cases} A_u = u_* (\frac{U(h)}{h})^{2/3}, \quad \beta = 5/3, \\ A_v = 0.88 A_u, \\ A_w = 0.55 A_u, \end{cases} \tag{16}$$

where u_* is calculated from Eq. (14). To obtain the wind velocities, we applied inverse Fourier transform to the following equation:

$$S_p(f_1, \ldots, f_n) = \frac{A_p}{(\sqrt{\sum_{i=1}^{n} f_i^2})^{\beta+n-1}}, \tag{17}$$

where n is the dimension number, and p is the wind direction (u, v, w).

Considering the computation costs of 2D and 3D wind fields (the consumed time per timestep: a 2D grid, 100×100, required 8 ms; a 3D grid, $100 \times 100 \times 100$, required 1,363 ms), we utilized a 2D wind field to approximate a 3D wind field by using the release height of the falling object. The wind field $u(p, t)$ is represented as follows:

$$u(p, t) \equiv u(p', h, t) = \frac{\ln(h) - \ln(z_0)}{\ln(h_0) - \ln(z_0)} u(p', h_0, t), \tag{18}$$

where p' is the 2D position, and h_0 is the release height of the falling object.

A 2D wind field for two different heights is illustrated in Fig. 12 (the mean wind velocity U is 4.0 m/s, and the grid size is 100×100).

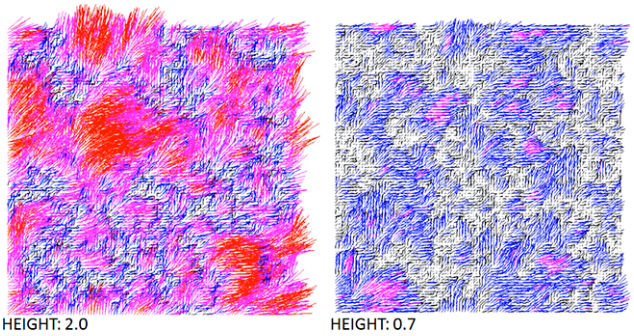


Fig. 12 Two-dimensional wind field, HEIGHT represents the distance (m) above the ground. The color represents the velocity length compared to the mean wind velocity U (*red*: $V > 2U$; *pink*: $U < V \leq 2U$; *blue*: $U/2 < V \leq U$; *black*: $V \leq U/2$). The wind direction is along the x-axis

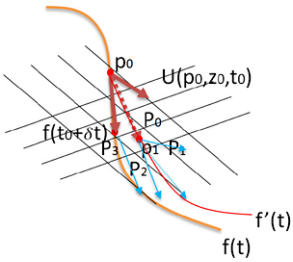


Fig. 13 Trajectory of free-falling behavior in a wind field

6.2 Wind-object interaction

Let $u(p, h, t)$ be a 2D wind field at position p and height h, and let the wind velocity at point $p_0 = (x_0, y_0, z_0)$ be $u(p_0, z_0, t_0) = (u_x, u_y, u_z)$, where u_x, u_y, and u_z correspond to the u, v, and w components, respectively, in Eq. (16).

To represent the computed trajectory of a falling object in a still fluid, we set the trajectory as a function $f(t)$, where $f(t)$ is a set of points per frame in the time domain. At time t_0, $f(t_0)$ is a quaternion (p_0, θ_0), which includes the position and orientation of the object. After time step δt, $f(t_0 + \delta t)$ comes to point p'.

The next point after p_0 is set to be p_1, $\widehat{p_0 p_1} = \widehat{u(p_0, z_0, t_0)} + \widehat{p_0 p'}$ (Fig. 13). If p_1 does not coincide with

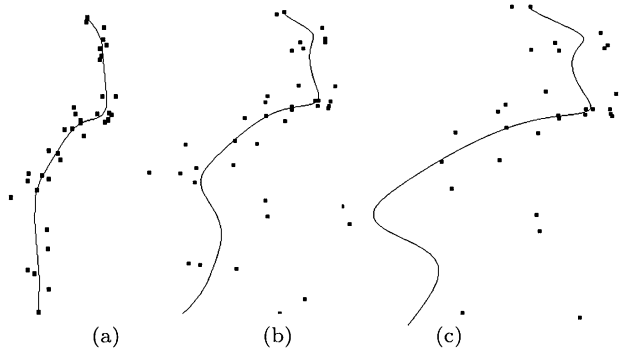


Fig. 14 Motion paths for different values of the mean wind velocity U: (**a**) $U = 1.0$ m/s, (**b**) $U = 3.0$ m/s, and (**c**) $U = 5.0$ m/s. The wind direction is from right to left

any grid node of the wind field, assuming the neighboring 2D grid nodes around p_1 are $Pi(0 \leq i \leq 3)$, then the wind velocity at p_1 is calculated from the linear interpolation of u_i at P_i. After the iterations, the new trajectory $f'(t)$ of the falling object in a wind field is formed using a Bezier curve to produce a smooth path with control points $p_i(i = 0, 1, \ldots)$ (Fig. 14).

Next, we considered the rotation of an object under the influence of wind. A falling object such as a leaf or a piece of paper can change its orientation in a wind field. To achieve a realistic effect of wind, we applied a noise function to the orientation calculation of the falling object:

$$\theta(t) = WN(t), \tag{19}$$

where $N(t) = \sqrt{u_x^2 + u_y^2 + u_z^2}/|U|$, in which U is the mean velocity of wind field; u_x, u_y, and u_z are the velocity components of wind at time t; and W is the maximum motion angle, which is provided by the animator.

7 Results

From the input parameters in Sect. 4.1, the main primitive motion, L_i, was determined from the calculations of I^* and Re in the phase diagram. According to the hypothesis in Sect. 5.1 and the Markov chain model, the global path synthesis started from a random primitive motion $L_j(i < j)$, and the next primitive motion was estimated by the transition matrix, mentioned in Sect. 5.2. The primitive motion segments were determined from the precomputed trajectory database in Sect. 4.2. To effectively evaluate our simulations, we compared them with the experimental videos of the falling lightweight rigid bodies.

The simulation results presented in Figs. 15–18 suggest that our simulations were realistic and our approaches were applicable in various flows such as water and air flows; the reader is also encouraged to watch the video accompanying

Fig. 15 Comparison between the simulation and the ground truth of a one-yen Japanese coin falling in water

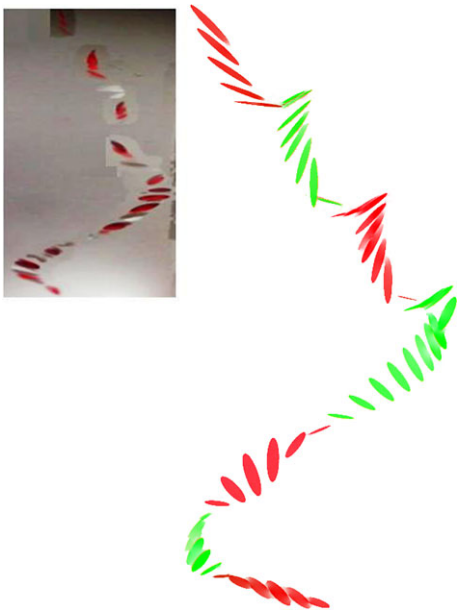


Fig. 16 Comparison between the simulation and the ground truth of a piece of paper falling in air

this paper. Figure 15 shows an aluminum circular disk (e.g., a one-yen Japanese coin with a radius of 1.0 cm and a thickness of 0.15 cm) as it fell into still water from a height of 50 cm. A regular fluttering motion was observed. We applied the Re–I^* phase diagram to determine that the main primitive motion was the fluttering motion for $I^* = 10^{-2}$ and $Re = 3.55 \times 10^3$. Both the simulation and the video show the fluttering motion as the global falling path.

Figure 16 shows a comparison between our simulation and a video of an elliptical piece of paper (with a major axis of 8.0 cm, a minor axis of 2.0 cm, and a thickness of

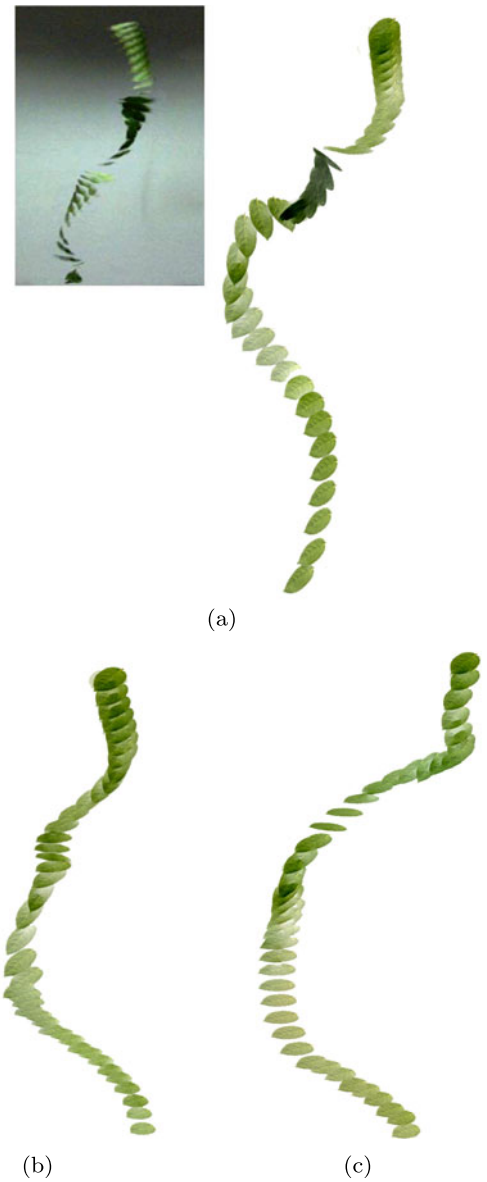


Fig. 17 (**a**) Comparison between the simulation and the ground truth of a leaf falling in air. (**b**) The simulation of the motion in a wind field of mean wind velocity $U = 3.0$ m/s and (**c**) $U = 5.0$ m/s. The wind direction is from right to left in a screen space

0.01 cm) as it fell in still air from a height of 3.1 m. Because $I^* = 2.2 \times 10^{-3}$ and $Re = 6.8 \times 10^3$, the Re–I^* phase diagram indicates that the main primitive motion is the spiral motion. The falling motion consists of tumbling and spiral motions, which is consistent with the hypothesis presented in Sect. 5.1.

Figure 17(a) shows a comparison between our simulation and a video of a leaf (with a major axis of 7.3 cm, a minor axis of 4.2 cm, and a thickness of 0.03 cm) as it fell in still air from a height of 2.0 m. Because $I^* = 6.3 \times 10^{-3}$ and $Re = 1.2 \times 10^4$, the Re–I^* phase diagram indicates that the main primitive motion is the transitional helical motion.

Fig. 18 Simulation of multiple leaves falling from a tree

The simulated result consists of steady descent, tumbling, and helical motions. Figures 17(b)–(c) show the final simulated motions in two strong wind fields that correspond to the conditions in Fig. 14(b)–(c), respectively. We found that in a strong wind field, the tumbling motion disappears, and the object travels far.

Figure 18 shows an integrated scene of multiple leaves falling from a tree. Although Re and I^* are the same for all leaves, they have different motion trajectories according to the Markov chain model. The implementation of the simulation does not include collision detections among objects.

All simulations were implemented on an Intel Core i7 CPU with 3.20 GHz and 12.0 GB RAM in real time (less than 3 ms per frame for one body). Because most of the computation was executed offline, the runtime motion synthesis and optimization process rarely consumed much memory; therefore, our simulation is not only realistic but also feasible for interactive applications.

8 Conclusion

This study presented a framework for simulating the realistic motions of lightweight rigid bodies in both still fluids and

wind fields. In addition, it presented the frontier research on the physical details of the dynamics of a lightweight rigid body. Furthermore, it proposed an effective motion synthesis method to achieve realistic simulations in real time.

This study is limited to rigid bodies with regular geometries (such as rectangular, circular, and elliptical objects with constant densities). For a rigid body with irregular geometry and uneven density distribution, it is difficult to determine how the geometry and density-distribution modifications influence the motion. When a paper or plastic object falls freely, the shape can be deformed, a factor that was omitted in this study. Furthermore, we obtained the orientation of the rigid body from a precomputed trajectory database, but the rotational axis and angle of the rigid body were difficult to determine in the cases of spiral and helical motions because they are related to its geometry and appear irregular, as observed by Razavi [14].

In our future research, we would like to integrate this simulation method with collision detections in multibody systems. This will require the interactive motion graphs of rigid bodies and a proper collision response implementation that relies on the approximations of velocities from motion sequences. Another interesting topic is to apply this method to articulated bodies with high degrees of freedom, which would require us to investigate the constraint solver of each joint. This could be used to simulate creatures' swimming or flying.

References

1. Andersen, A., Pesavento, U., Wang, Z.J.: Unsteady aerodynamics of fluttering and tumbling plates. J. Fluid Mech. **541**, 65–90 (2005)
2. Aoki, K., Hasegawa, O., Nagahashi, H.: Behavior learning and animation synthesis of falling flat objects. J. Adv. Comput. Intell. Intell. Inform. **8**(2), 223–230 (2004)
3. Arikan, O., Forsyth, D.A.: Interactive motion generation from examples. ACM Trans. Graph. **21**, 483–490 (2002)
4. Field, S.B., Klaus, M., Moore, M.G., Nori, F.: Chaotic dynamics of falling disks. Nature **388**, 252–254 (1997)
5. Haevre, W.V., Fiore, F.D., Reeth, F.V.: Physically-based driven tree animations. In: Chiba, N., Galin, E. (eds.) Eurographics Workshop on Natural Phenomena, pp. 75–82. Eurographics Association, Goslar (2006). doi:10.2312/NPH/NPH06/075-082
6. Haiyan, L., Qingjie, S., Hui, Z., Qin, Z.: Example-based motion generation of falling leaf. In: 2010 International Conference on Computer Design and Applications (ICCDA), vol. 5, pp. V5-217–V5-221 (2010)
7. Jones, M.A., Shelley, M.J.: Falling cards. J. Fluid Mech. **540**, 393–425 (2005)
8. Khorloo, O., Gunjee, Z., Chiba, N.: Wind field synthesis for animating wind-induced vibration. Int. J. Virtual Real. **10**, 53–60 (2011)
9. Kovar, L., Gleicher, M., Pighin, F.: Motion graphs. ACM Trans. Graph. **21**, 473–482 (2002)
10. Mahadevan, L., Ryu, W.S., Samuel, A.D.T.: Tumbling cards. Phys. Fluids **11**, 1–3 (1999)
11. Maxwell, J.C.: On a particular case of the descent of a heavy body in a resisting medium. Camb. Dublin Math. J. **9**, 145–148 (1854)
12. Olesen, H.R., Larsen, S.E., Hojstrup, J.: Modelling velocity spectra in the lower part of the planetary boundary layer. Bound.-Layer Meteorol. **29**, 285–312 (1984)
13. Ota, S., Tamura, M., Fujimoto, T., Muraoka, K., Chiba, N.: A hybrid method for real-time animation of trees swaying in wind fields. Vis. Comput. **20**, 613–623 (2004)
14. Razavi, P.: On the motion of falling leaves. ArXiv e-prints (2010)
15. Reissell, L.M., Pai, D.K.: Modeling stochastic dynamical systems for interactive simulation. Comput. Graph. Forum **20**, 339–348 (2001)
16. Shi, L., Yu, Y., Wojtan, C., Chenney, S.: Controllable motion synthesis in a gaseous medium. Vis. Comput. **21**(7), 474–487 (2005)
17. Shinya, M., Fournier, A.: Stochastic motion–motion under the influence of wind. Comput. Graph. Forum **11**(3), 119–128 (1992)
18. Stam, J., Eugene, F.: Turbulent wind fields for gaseous phenomena. In: SIGGRAPH'93, pp. 369–376. ACM Press, New York (1993)
19. Stringham, G.E., Daryl, S., Guy, H.: The behavior of large particles falling in quiescent liquids. Geol. Surv. Prof. Pap. **562-C**, C1–C36 (1969)
20. Tanabe, Y., Kaneko, K.: Behavior of a falling paper. Phys. Rev. Lett. **73**, 1372–1375 (1994)
21. Thuillier, R.H., Lappe, U.O.: Wind and temperature profile characteristics from observations on a 1400 ft tower. J. Appl. Meteorol. **3**, 299–306 (1964)
22. Vázquez, P.-P., Balsa, M.: Rendering falling leaves on graphics hardware. J. Virtual Real. Broadcast. **5**(2) (2008). http://hdl.handle.net/2117/13041
23. Wei, X., Zhao, Y., Fan, Z., Li, W., Qiu, F., Yoakum-Stover, S., Kaufman, A.E.: Lattice-based flow field modeling. IEEE Trans. Vis. Comput. Graph. **10**, 719–729 (2004)
24. Weismann, S., Pinkall, U.: Underwater rigid body dynamics. ACM Trans. Graph. **31**(4), 104:1–104:7 (2012)
25. Willmarth, W.W., Hawk, N.E., Harvey, R.L.: Steady and unsteady motions and wakes of freely falling disks. Phys. Fluids **7**, 197–208 (1964)
26. Zhang, L., Zhang, Y., Jiang, Z., Li, L., Chen, W., Peng, Q.: Precomputing data-driven tree animation. J. Vis. Comput. Animat. **18**(4–5), 371–382 (2007)
27. Zhong, H., Chen, S., Lee, C.: Experimental study of freely falling thin disks: transition from planar zigzag to spiral. Phys. Fluids **23**(1) (2011)

Haoran Xie is a Ph.D. candidate in the School of Knowledge Science, Japan Advanced Institute of Science and Technology (JAIST), Japan. He received his B.S. degree in the Department of Applied Mathematics, Anhui University, China in 2006. His current research interests include computer graphics and computer animation.

Kazunori Miyata has been a Professor in the School of Knowledge Science at the Japan Advanced Institute of Science and Technology (JAIST) since 2002. Prior to joining JAIST, he was an Associate Professor in the Department of Imaging Art at the Tokyo Institute of Polytechnics. He received his B.S. degree from Tohoku University in 1984, and his M.S. and Ph.D. from the Tokyo Institute of Technology in 1986 and 1997. His research mainly focuses on rendering & modeling natural objects, texture generation, and multimedia applications. He is a member of ACM and IEEE.