CGI2012 manuscript No.
(will be inserted by the editor)

Realistic Motion Simulations of Objects in Free Fall

Haoran Xie · Kazunori Miyata

Abstract The free fall motion of a lightweight object is a familiar and spectacular phenomenon in which the object can flutter (oscillate from side to side) and tumble (rotate and drift sideways). However, in computer graphics, we lack the ability to simulate free fall motion in a still fluid. In this paper, we consider all the physical characteristics of free fall in a still fluid, and propose a new procedural motion synthesis method for modeling free fall motion in interactive environments. Six primitive motions are defined in a phase diagram and analyzed separately using a trajectory search tree and a precomputed trajectory database. The global paths of free fall motion are synthesized on the basis of these primitive motions, using a free fall motion graph whose edges are connected using the Markov chain model. In addition, our approach integrates with wind field methods by using an improved noise-based algorithm under different wind speeds and object release heights. This approach provides not only realistic results in both a still fluid and a wind field but also rapid computation for realtime applications.

Keywords Natural phenomena · Motion synthesis · Free fall motion · Phase diagram · Real-time simulation

1 Introduction

Not all objects fall straight down, for example, a piece of paper or a leaf wavers and flutters down in a seemingly unpredictable motion when released from your hand. To the authors' knowledge, the common and spectacular freely falling principle has not been completely resolved in physics, although research in this area has a rich history beginning with James Maxwell.

The complexity of free fall lies in the coupling of the forward motion of the object with lateral oscillations in the surrounding fluid and the production and influence of vortices around the object. The problem of free fall motion involves multiple hydrodynamic effects (such as lift force, drag force, and vortex shedding), which exhibit both regular and chaotic behaviors. This is a challenging problem in visual simulations in computer graphics of many phenomena related to unsteady dynamics, such as meteorology, flight aerodynamics, bubbles rising and boiling, and seed dispersal.

Simulating free fall motion by using key-frame control requires the animator to exert much effort and expert ability. A physically based method can create reliable results only for a simple model, because a complicated model would involve inertial forces and vortex effects. The heavy computational cost is the fatal disadvantage of applying a physically based method to realtime applications. In this paper, a procedural motion synthesis method that includes the lift and drag forces is proposed to simulate realistic free fall motion in a still fluid. This method also provides proper simulation results in a wind field. Our major contributions are as follows:

- A data-driven motion synthesis method that uses a precomputed trajectory database and a free fall motion graph.
- A separate synthesis method that uses six primitive motions (Fig. 1) of free fall behavior, which are defined in a phase diagram of the dimensionless moment of inertia and the Reynolds number.
- A Markov chain model based on the motion groups of these primitive motions allows an accurate estimation because of the apparent features of each primitive motion.

Haoran Xie · Kazunori Miyata
Japan Advanced Institute of Science and Technology, Ishikawa, Japan.
E-mail: {xiehr, miyata} @jaist.ac.jp

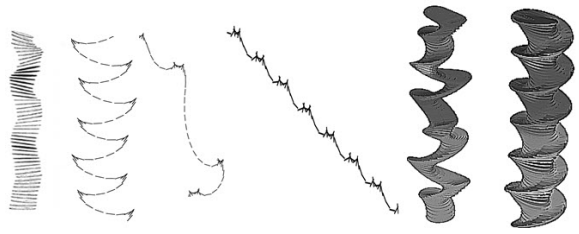


Fig. 1 Experimentally measured trajectories of primitive motions. Left to right: steady decent, tumbling, chaotic, fluttering observed by [1], helix, and spiral observed by [25].

- An improved $1/f^{\beta}$ noise-based wind field implementation.
- A hypothesis about global free fall paths verified by experiments.

1.1 **Related works**

Scientific interest in free fall phenomena dates back to one and half centuries ago, when James Maxwell noticed the torque caused by gravity and lift when forces do not act at the same point [9]. After Maxwell's research, a few achievements have been made, but the problem remains unsolved.

A phase diagram using the Reynolds number and the dimensionless moment of inertia can differentiate free fall motion patterns observed in experiments [4]. Recently, experiments have found an additional three typical trajectories in a three-dimensional environment: zigzag, transitional helix, and spiral motions [25]. Tanabe et al. [19] built a simple phenomenological model of falling paper by solving ordinary differential equations (ODEs) based on the Kutta-Joukowski theorem. In addition, another study [13] investigated the relationship among different parameters that can affect the paths of a freely falling leaf by performing more than 6000 three-dimensional experiments. There are few studies of numerical simulations of free fall in two dimensions. Andersen et al. [1] discussed direct numerical simulations of the two-dimensional Navier-Stokes equation and presented a fluid force model based on ODEs derived from experiments and simulations. No convincing numerical simulation can successfully explain chaotic motion and free fall in three-dimensional space.

Research on free fall simulations is absent in computer graphics. However, there are several simulations of falling motions in a wind field. Wei et al. [22] provided the LBM method for simulating soap bubbles and feathers in a wind field, however, this approach cannot determine the designated motion trajectory, and it has difficulty simulating multiple objects, because it requires a large computational cost (bubble: CPU 2.8 fps and GPU 11.5 fps; feather: CPU 0.76 fps and GPU 6.1 fps). Related example-based approaches have been proposed based on the Markov model [14], captured videos [2], segments from fluid simulations [15], trajectories animated by Maya [21], and sketches made by a designer [6]. All these simulations ignore the nature of free fall and consider it a completely complex and unpredictable dynamics motion, which is modeled by stochastic processes or a simple particle representation. Some commercial computer graphics (CG) tools, including Lightwave and Maya, do not have the function of free fall animation; instead, they provide particle simulation to model a falling object by adjusting the drag and lift parameters in a wind field. In all of these works, the motion paths are unpredictable, and it is not easy to achieve realistic motion.

Example-based data-driven motion synthesis combines the controllability of procedural and physically based animation with the realistic appearance of a recorded motion stream (like motion capture). The first paper about automatically organizing example motion clips into graphs for efficient motion synthesis proposed a motion graph technique [3]. Later, Kovar et al. built an extended motion graph using local search with a branch and bound algorithm [7]. Besides being used in character animation, motion graphs are also used in other physical simulations, such as tree animation [5] [24]. Our method includes the motion graph technique for synthesizing free fall motion.

To obtain a wind field, a direct and straight method is to simulate the turbulent flow under a boundary condition by solving differential equations using a Fourier filter [16] [17]. Another method is to simulate the motion in a wind field using noise functions (fractional Brownian motion) [12] [10]. Comparing the flow-based and noise-based methods, the flow-based method provides physically accurate and realistic results but requires a high computational cost, whereas, the noise-based method is much simpler and suitable for real-time simulations but at the cost of physical accuracy. To overcome the inaccuracy of the noise-based method, a physically based analysis of wind characteristics is necessary.

1.2 **The content of the paper**

Our simulation method is illustrated in Fig. 2. This method has two important steps: motion modeling and motion synthesis of free fall motion. In the motion-modeling phase, the input parameters are introduced, including the physical characteristics of the object and the fluid in which it is released (release height, mass,

etc.). We transform these parameters into two key non-dimensional numbers: the Reynolds number (Re) and the dimensionless moment of inertia ($I*$). Next, we use the free fall phase diagram to obtain the main primitive motion in which the motion of the object is stable. To synthesize the primitive motions of free fall, we use a trajectory search tree to represent chaotic, fluttering and tumbling motions, and we use 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 determined 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,the wind field interactions with the falling object are calculated efficiently. In the end, the final free fall simulation is achieved in real time.

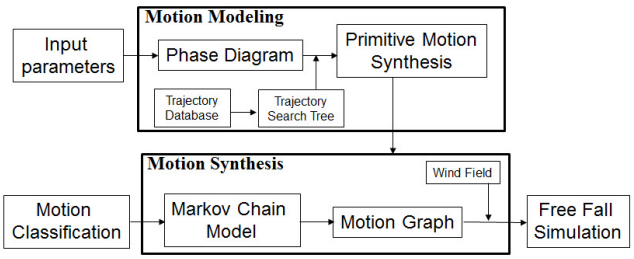


Fig. 2 Flow of our simulation method

The rest of this paper is organized as follows. In the next section, we describe modeling primitive motions based on a phase diagram. In Sect. 3, 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. 4. The simulation results of free fall motion in realtime under both wind and no wind conditions are presented in Sect. 5. The conclusion and possibilities for future work are described in the last section.

2 Motion modeling

2.1 Input parameters

A falling object is characterized by the following quantities:

- h: height of release
- L: length of the object
- a: length of the cross section of the object
- b: width of the cross section of the object
- ρ_s: density of the object
- ρ_f: density of the fluid
- ν: kinematic viscosity of the fluid
- g: gravity acceleration

From these parameters, three dimensionless quantities are derived: the Reynolds number (Re), the aspect ratio of the object ($\epsilon = \frac{b}{a}$), and the dimensionless moment of inertia (I^*). Re and I^* are the two key quantities for building a phase diagram of free fall motion (Fig. 3). Here,

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

where U is the velocity scale of flow. In addition,

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

where $\rho(x,y,z)$ is 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).

Commonly, the velocity scale U is approximated by the average descent velocity of the falling object.

$$U \sim \sqrt{(\frac{\rho_s}{\rho_f} - 1)gb} \tag{3}$$

For a lightweight(thin) object, the aspect ratio ϵ is so small that we omit its effect ($\epsilon \ll 1$).

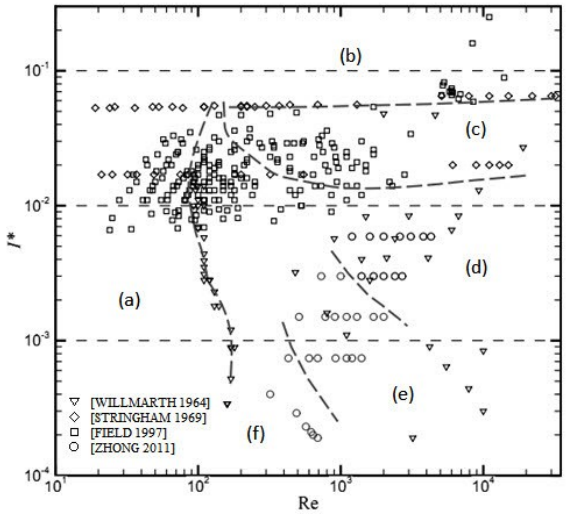


Fig. 3 The Re-I^* phase diagram of free fall motions, including six regimes:(a) steady descent, (b) tumbling, (c) chaotic, (d) fluttering, (e) helix, and (f) spiral motions. The symbols in the diagram represent experimental results from previous works.

The *Re-I** phase diagram (Fig. 3) was introduced in previous works ([23], [18], [4], [25]). The regimes in the diagram represent different primitive motions. The tumbling, fluttering and spiral motions almost appear periodic; the chaotic motion appears to be the transitional motion between the tumbling and fluttering motions, and the helix motion appears to be the transitional motion between the spiral and fluttering motions.

The definitions of the primitive motions are as follows:

- (a) steady descent (SD): the object drops straight down in the vertical direction
- (b) periodic tumbling (PT): the object turns continuously end-over-end and drifts in one direction
- (c) transitional chaotic (TC): the object begins to oscillate with increasing amplitude, the fluttering motion finally turns in to a tumbling motion, and chaotic motion is observed
- (d) periodic fluttering (PF): the object oscillates from side to side with a well-defined period
- (e) transitional helix (TH): the object moves in a helical path at a constant speed
- (f) periodic spiral (PS): the object falls downward circularly in three-dimensional space.

The motion trajectories of these primitive motions are illustrated in Fig. 1.

2.2 **Pre-computed trajectory database**

It is not easy to build a trajectory database of free fall motion, because capturing accurate trajectories of a falling lightweight object in the real world seems to be infeasible because of chaotic motions and the short time interval. Using fluid simulation to track vortex particles from frame to frame by following velocity vectors is also not suitable for the following reasons: (1) they cannot detect all primitive motions, (2) they have difficulty capturing realistic motion trajectories, and (3) various parameters adjustments make such simulations difficult to control.

Another approach is to use the Kutta-Joukowski theorem [19], accounting for the drag and lift forces, to solve the following ODEs:

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

where (x, y) and θ are the position and angle of the center of mass of the falling object. In addition, (u, v) and ω are the linear and angular velocities of the object. Notice 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 aspect parameters. The parameters $A_\perp$ and $A_\parallel$ are the drag coefficients in the directions perpendicular and parallel to the falling object, respectively. The angles α and β are defined as $\alpha = arctan(u/v)$, and $\beta = \alpha + \theta$, parameter k is defined as follows:
if $sign(v)sin\beta \geq 0, k = 1$;
if $sign(v)sin\beta < 0, k = -1$.

We apply the standard fourth-order Runge-Kutta algorithm to solve the second-order ordinary differential equations presented in Eq. (4), as shown in Fig. 4 (a). Similar to the fluid simulation approach, it is difficult to control the ODEs model with the parameters $A_\perp$ and $A_\parallel$. For example, the calculated motion trajectory in Fig.4 (b) is meaningless, because it is not natural for an object to fall vertically after a fluttering motion. Nevertheless, the object orientation results obtained by solving Eq. (4) are more accurate than other approaches.

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

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

$$\begin{aligned}
x_t &= x_0 - \frac{A_x}{\Omega} sin(\Omega t) \\
y_t &= y_0 - Ut - \frac{A_y}{2\Omega} cos(2\Omega t)
\end{aligned} \tag{5}$$

where A_x and A_y are the amplitudes of the vertical and horizontal velocities of the falling object generated

by oscillations due to the surrounding viscous flow, Ω describes the angular frequency of falling motion, and U is calculated from Eq. (3). The segment breakpoints are chosen as the turning points of the trajectory given at time steps $t_i = \frac{2k+1}{2\Omega}\pi, k \in Z \geq 0$.

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

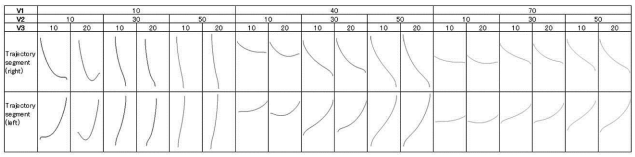


Fig. 5 Trajectory segments obtained from Eq. (5).

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

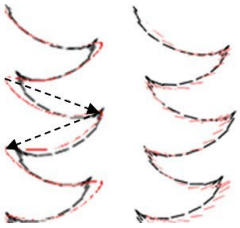


Fig. 6 Comparing synthesized trajectory (red) and measured data (black) of fluttering motion, for no orientation (left) and interpolated orientation (right). Arrow lines represent feature vectors.

2.3 Primitive motion synthesis

2.3.1 ***Trajectory search tree***

We compare the trajectories of periodic fluttering, chaotic motion and tumbling motion data from experiments. Because the airflow behind a free falling object reveals vortex shedding, turbulence and other complex motions, the object comes to turning points, where the angular velocity is zero, and the velocity in the oscillation direction is also zero, but the velocity in the vertical direction is maximized. The object faces two alternatives of sliding left or sliding right (fluttering or tumbling). Simple structures of fluttering, chaotic motion, and tumbling motion are illustrated in Fig. 7.

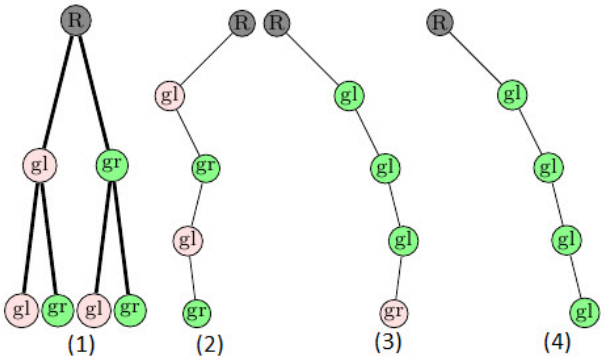


Fig. 7 The first two levels of a small trajectory search tree (1) and the tree structures of fluttering (2), chaotic motion(3), and tumbling motion(4) created by traversal of four levels of the search tree. (gl: glide left; gr: glide right)

In the tree structure, every child represents a motion segment derived from the precomputed trajectory database using the feature vector as the search key.

2.3.2 ***Unified trajectory functions***

When projecting the motion paths of primitive motions onto the XY plane, we notice that the curves of six primitive motions have characteristic shapes: the steady descent motion trajectory is one point; the fluttering, tumbling motion and chaotic motion trajectories are in a straight line; the spiral motion trajectory is a circle; and the helix motion trajectory is similar to an eight-petal rose curve, as shown in Fig. 8. These curves are

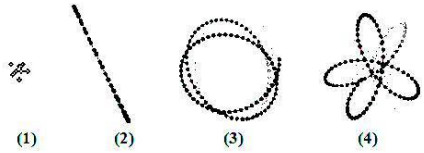


Fig. 8 Measured curves projected onto the XY plane: (1) steady descent; (2) fluttering, chaotic motion and tumbling motion; (3) spiral motion; and (4) helix motion.

all represented in the following equations:

$$\begin{aligned} x_t &= A_e cos(\Omega t)(1 + \epsilon_e sin(k\Omega t)) \\ y_t &= A_e sin(\Omega t)(1 + \epsilon_e sin(k\Omega t)) \\ z_t &= h - Ut \end{aligned} \tag{6}$$

where A_e is the amplitude of the elliptical oscillation generated in the XY plane, ϵ_e is the aspect ratio of the minor axis and the major axis of the oscillation ellipse, k is the ratio of the period of elliptical oscillation to the period of rotation of the falling object, and Ω is the angular frequency of the falling motion. Ad deduced from Eq. (6), the simple form of the free fall trajectory is as follows:

- $\epsilon_e \to 0, k = 1 \mapsto$Spiral motion
- $A_e \to 0, k \to 0 \mapsto$Steady descent motion
- $\epsilon_e \neq 0, k = 4 \mapsto$Helix motion

Because the nature of chaotic motion is more complex than that of periodic motion (PF and PT), we synthesize 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 1/10 and 10 calculated by the Box-Muller method, V_0 is a feature vector whose orientation is the release angle of the falling object, and A_0 is the initial amplitude of object oscillation.

From experimental data [13], we know that the deviations of primitive motions D_i from the release position are distributed in a normal distribution of Gaussian functions $A_i e^{-(\frac{r-B_i}{C_i})^2}$ and linearly with the release height h:

$$D_i = \frac{kB_iL}{a} \tag{7}$$

where k is the deviation coefficient. Because the frequency of tumbling motion is given as $\Omega \sim \sqrt{b}/a$ [8], the initial amplitude of oscillation is

$$A_0 = \frac{D_iU}{h\Omega} \tag{8}$$

in which U is the average falling velocity [Eq. (3)].

The final synthesized trajectories of primitive motions are shown in Fig. 9.

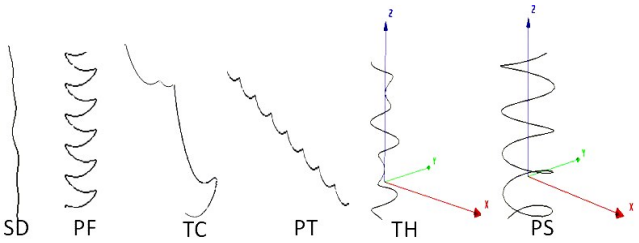


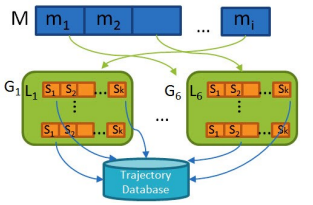
Fig. 9 Six synthesized trajectories determined in this study; they correspond to the measured trajectories in Fig. 1.

3 Motion synthesis

3.1 Motion classification

The primitive motions $\{L_i|S_1, S_2, ...S_k\}$ (S_k is the k-th segment in primitive motion L_i) are synthesized from motion segments S_n from a precomputed trajectory database. Because primitive motions are the basic free fall motions observed in various experimental works, the motion groups $\{G_i|1 \leq i \leq 6\}$ are used to represent free fall motion.

For a free fall motion M, $M\{M = m_1 \parallel m_2 \parallel ...m_i\}$ is annotated by a label L_i, where L_i is a primitive motion. We define L_i as follows: L_1: SD; L_2: PF; L_3: TC; L_4: PT; L_5: TH; and L_6: PS, as shown in Fig. 10 (a). Based on thousands of experiments [13], all free fall trajectories are classified into seven motion classes [Fig. 10 (b)]. Based on the experimental data, we make the following hypothesis:



(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. 10 (a) Motion classifications (blue: motion classes; green: motion groups; yellow: motion segments), (b) Motion classes in experiments.

Hypothesis *If* $M\{M = m_1 \parallel m_2 \parallel ... \parallel m_i\}$ *represents free fall motion in three dimensions, then* $m_i \in \{L_j|1 \leq j \leq 6\}$*, and the subscript sequence* $\{j_1, j_2, ..j_i\}$ *should be an increasing sequence.*

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

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

$$N = \sum_{i=1}^{i \leq k} C_k^i, k \in Z, k \in [1, 6] \tag{9}$$

where k is the level of the main primitive motion determined by using the calculated Re and I^* discussed in Sect. 2.1.

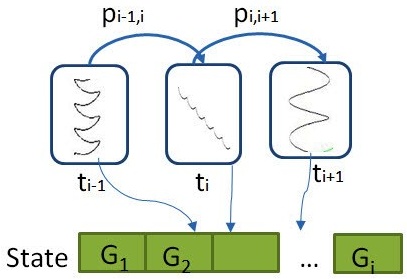


Fig. 11 Markov chains model states and transition probabilities

3.2 Markov chain model

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

Let us consider 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 $\{G_i | 1 \leq i \leq 6\}$ (Fig. 10). 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:
$P[X_{n+1} = j | X_n = i_n] = P[X_{n+1} | X_0 = i_0 ... X_n = i_n]$
where $j, i_0, ..., i_n \in N_0$. The stochastic process $\{X_n\}$ is a Markov chain.

Let the state space N_0 be the motion group G, and let process $\{X_n\}$ on $\{X_t\}$ be a discrete time set, then the transition probability $p_{ij} = P[X_{t+1} = L_j | X_t = L_i]$ is the conditional probability to transition from primitive motion L_i to primitive motion L_j. The transition matrix is given as $P = (p_{ij})$ (Figure 11).

Next, we discuss the realization of Markov chain $\{X_t\}$ and the transition matrix P. To obtain the process $\{X_t\}$, the model starts with an initial state at time $t_0 = 0$. Then an iteration step is executed from state L_i at time t to state L_j at time $t + 1$, and the calculation depends on the probabilities at the i-th row of the transition matrix P [i.e., $P_i = (p_{ij} | j = 1, 2..., 6)$]. The state transition probabilities for the transition matrix P are found by counting the state transitions that occurred in experimental data. Let set N_i be the number of all transitions from state L_i in the experimental data, N_{ij} be the number of transitions from state L_i to state L_j, then the probability is given as $p_{ij} = \frac{N_{ij}}{N_i}$.

The advantages of using the 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; to create a realistic simulation of free fall motion, the features of each primitive motion can be used to obtain a valid transition matrix from experimental data; and the computational cost of the model is low.

3.3 Graph construction

A special free fall motion graph (Fig. 12) is based on the motion graph described in ref. [7]. This graph is a complete directed graph: each node of the graph is connected to other nodes in the same graph. We use $G = (V, E)$ to represent the motion graph, where V is the node set, and E is the edge set. Every frame in a motion sequence of primitive motions appears as a node in the motion graph; a transition splice in a motion sequence appears as an edge between nodes. We only search the graph in one direction (from the top to down) in the order of m_i in the hypothesis presented in Sect. 3.1. Therefore, for example, it is impossible for free fall motion to become tumbling motion after spiral motion. The transition probability is attached to the edge between nodes using the discrete-time Markov chain model presented in Sect. 3.2.

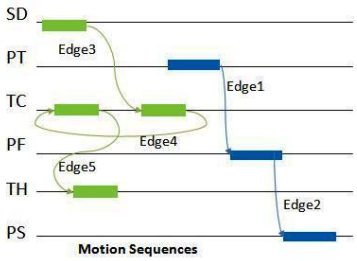


Fig. 12 Free fall motion graph. A motion path represents a collection of splices between sequences. Here, two example motions are shown.

4 Falling in wind

4.1 Wind field

Let the velocity of wind be $V = (V_u, V_v, V_w)$, where V_u, V_v, and V_w describe the wind velocity components along the x-, y- and z- axes of the coordinate system in the free fall simulation. In addition, let $U(h)$ be the mean wind velocity at height h. According to the logarithmic wind law [20], $U(h)$ is given by

$$U(h) = \frac{u_*}{k} \ln(\frac{h}{z_0}) \tag{10}$$

where u_* is the friction velocity (m/s), k is von Karman's constant ($k = 0.40$), and z_0 is the roughness parameter (conceptually it is the height where V goes to zero). The value of z_0 depends on the type of ground terrain (we choose $z_0 = 0.3$).

The fBm method can suitably represent the wind

[11]. The spectral density function of the wind field is given as follows based on Kolmogoroff's law:

$$S_u(n) = u_*^2(\frac{U(h)\phi}{h})^{2/3}\frac{C}{n^{5/3}} \tag{11}$$

where $\phi = \epsilon kh/u_*^3$, ϵ is the dissipation rate according to Kolmogoroff's law, and C is a constant equal to $\alpha(2\pi k)^{-2/3}$, where α is determined experimentally to be 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 $S(f) = A/f^{\beta}$ [where A is the amplitude in wind direction (u, v, w)], we adopt the approximations of A_u, A_v, and A_w from ref. [10] as follows:

$$\begin{aligned} A_u &= u_*(\frac{U(h)}{h})^{2/3}, \beta = 5/3 \\ A_v &= 0.88A_u \\ A_w &= 0.55A_u \end{aligned} \tag{12}$$

where u_* is calculated from Eq. (10). To obtain the wind velocities, we apply the inverse Fourier transform to the following equation:

$$S_p(f_1, ..., f_n) = \frac{A_p}{(\sqrt{\sum_{i=1}^{n} f_i^2})^{\beta+n-1}} \tag{13}$$

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

Considering the computational costs of two-dimensional (2D) and three dimensional (3D) wind field (a 2D grid size of 100×100 requires 8ms; a 3D grid size of $100 \times 100 \times 100$ requires 1363ms), we use a 2D wind field to approximate a 3D wind field by using the height of the falling object. The wind field $u(p, t)$ is represented as:

$$u(p,t) \equiv u(p', h, t) = \frac{\ln(h) - \ln(z_0)}{\ln(h_0) - \ln(z_0)}u(p', h_0, t) \tag{14}$$

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. 13 (mean wind velocity: $U = 4.0m/s$; grid size: 100×100).

4.2 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 $u(p_0, z_0, t_0) = (u_x, u_y, u_z)$ [u_x, u_y, and u_z are corresponding to the u,v, and w components in Eq. (12)].

To represent the computed trajectory of a falling object in a still fluid, we set the trajectory to be 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. 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, $\widehat{p_0p_1} = u(p_0, z_0, t_0) + \widehat{p_0p'}$ (Fig. 14). If p_1 does not coincide with any grid node of the wind field, assuming the neighboring 2D grid nodes around p_1 are $Pi(0 \leq i \leq 3)$, 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 synthesized using a Bezier curve to produce a smooth path with control points $p_i(i = 0, 1, ...)$.

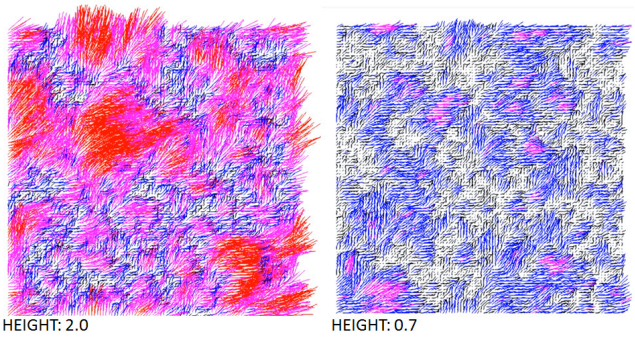


Fig. 13 Two-dimensional wind field. HEIGHT represents the distance(m) above the ground. The color represents the velocity V compared to the mean wind velocity U(red: $V > 2U$; pink: $U < V \leq 2U$; blue: $U/2 < V \leq U$; black: $V \leq U/2$). The wind direction is along the x-axis.

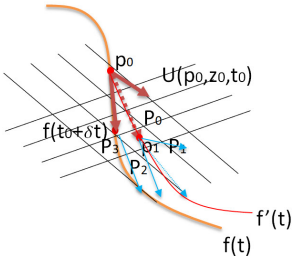


Fig. 14 Trajectory of freely falling behaviour in wind field

Next, we consider the rotation of an object under the influence of wind. Note that a falling object such as a leaf or piece of paper, can change its angle in a wind field. To achieve a realistic effect, we apply a noise function into the orientation calculation of the falling object:

$$\theta(t) = WN(t) \tag{15}$$

where $N(t) = \sqrt{u_x^2 + u_y^2 + u_z^2}$ is obtained from the fBm noise function in Sect. 4.1, and W is the maximum motion angle, which is designated by the designer.

5 Results

From the input parameters in Sect. 2.1, the main primitive motion L_i is determined by using a calculated

Re and I^* phase diagram. According to the hypothesis in Sect. 3.1 and the Markov chain model, global path synthesis starts from a random primitive motion $L_j(i < j)$, and the next primitive motion is estimated from the transition matrix. The primitive motion segments are determined from a precomputed trajectory database. To efficiently evaluate our simulations, we compare them with experimental videos of free falling objects.

The simulation results presented in Fig. 15 suggest that our simulations are realistic and that the methods used in our simulation are applicable in various fluids, such as water and air. Figure 15 (a) shows an aluminum circular disk (radius: 1.0cm; thickness: 0.15cm) free falling in still water from a height of 50cm. We use the $Re - I^*$ phase diagram to determine that the main primitive motion is fluttering for $I^* = 10^{-2}$ and $Re = 3.55 \times 10^3$. Note that both the simulation and the video show fluttering motion as the global free fall path.

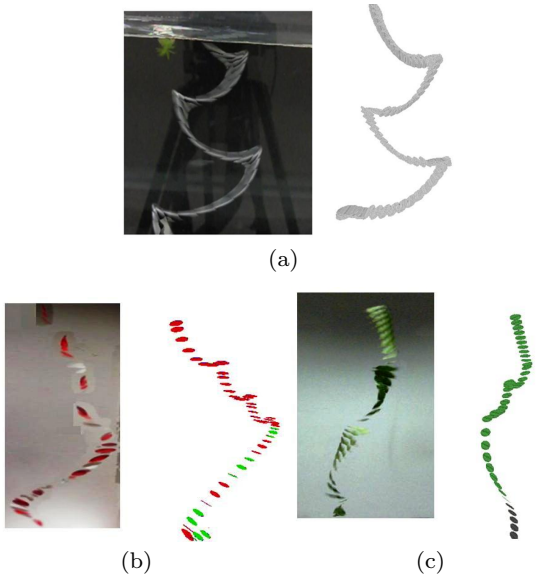


Fig. 15 Comparison of our simulations(right) with the ground truths (left). (a): A Japanese yen coin freely falls in water. (b) A paper freely falls in air. (c) A leaf freely falls in air.

Figure 15 (b) shows an elliptical piece of paper (major axis: 8.0 cm; minor axis: 2.0 cm; thickness: 0.01 cm) freely falling 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 phase diagram indicates that the main primitive motion is a spiral motion. Note that the falling motion consists of both tumbling and spiral motions, which is consistent with the hypothesis in Sect. 3.1.

Figure 15 (c) shows a leaf (major axis: 7.3cm; minor axis: 4.2cm; thickness: 0.03cm) freely falling 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 phase diagram indicates that the main primitive motion is a transitional helix motion. Note that the falling motion consists of steady descent, tumbling and helix motions.

In Fig. 16, free fall motion paths are coupled with different wind fields. The black dots represent the control points of Bezier curves. Figure 16(a) has a low wind field, therefore, the motion path is similar to that in Fig. 15 (c). Figure 16(b) and (c) have higher wind fields, and the influences of the wind are apparent.

Figure 17 shows the final simulated free fall motion in no wind [Fig. 17(a)] and in two strong wind fields [Fig. 17(b) and (c)] that correspond to the conditions in Fig. 16(b) and (c), respectively. We found that under a strong wind field, tumbling motion disappears and the object travels far.

All simulations were implemented in C++ on an Intel Core i7 CPU 3.20 GHz with 12.0 GB RAM in real time (around 50fps). Because most of our method was executed offline, the online motion synthesis and optimization process were rarely memory consuming; therefore, our simulation is not only realistic but also feasible for interactive applications.

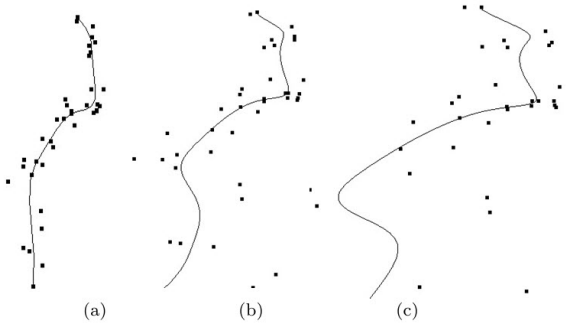


Fig. 16 Free fall motion paths for different values of the mean wind velocity U.(a) $U = 1.0m/s$, (b) $U = 3.0m/s$, and (c) $U = 5.0m/s$). The wind direction is from right to left.

6 Conclusion

This paper presents a framework for simulating realistic free fall motion in both still fluids and wind fields. In addition, it presents the first research on the physical details of free fall motion. Furthermore, it proposes an efficient motion synthesis method to achieve realistic free fall simulations in real time.

This work is limited to the study of objects with regular geometries (such as rectangular, circular, and

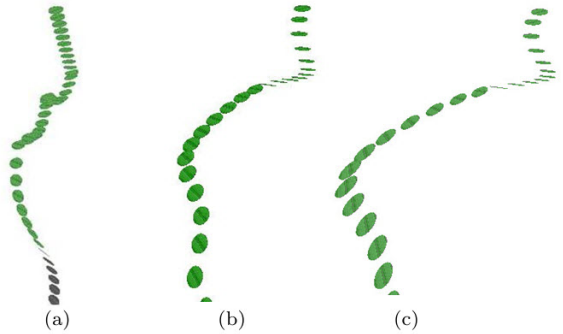


Fig. 17 Final synthesized free fall motion in different wind fields. (a) No wind as reference, (b) Mean wind velocity $U = 3.0m/s$, and (c) $U = 5.0m/s$. The wind direction is from right to left.

elliptical) and constant densities. For objects with irregular geometries and uneven density distributions, it is difficult to determine the influences of the geometry and density distribution modifications on the free fall motion. When a paper or plastic object falls freely, the object can change its shape, we omit the effect of this shape deformation in this work.

In the future, we could use the pattern-based method to apply the current simulations as patterns in fluid simulation. Because free fall is a common phenomenon for lightweight objects, it is promising to couple our simulation method with simulations used in the game industry and other areas of computer graphics.

Simulating oscillations is an intriguing topic. For example, oscillations could occur in different directions if an object has a high number of degree-of-freedom. The quantitative analysis of oscillations of an object in free fall is another feasible method for the motion synthesis of free fall motion.

References

1. Andersen, A., Pesavento, U., Wang, Z.J.: Unsteady aerodynamics of fluttering and tumbling plates. Journal of Fluid Mechanics **541**, 65–90 (2005)
2. Aoki, K., Hasegawa, O., Nagahashi, H.: Behavior learning and animation synthesis of falling flat objects. JACIII **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: N. Chiba, E. Galin (eds.) NPH, pp. 75–82. Eurographics Association (2006)
6. Haiyan, L., Qingjie, S., Hui, Z., Qin, Z.: Example-based motion generation of falling leaf. In: Computer Design and Applications (ICCDA), 2010 International Conference on, vol. 5, pp. V5–217 –V5–221 (2010)
7. Kovar, L., Gleicher, M., Pighin, F.: Motion graphs. ACM Trans. Graph. **21**, 473–482 (2002)
8. Mahadevan, L., Ryu, W.S., Samuel, A.D.T.: Tumbling cards. Physics of Fluids **11**, 1–3 (1999)
9. Maxwell, J.C.: On a particular case of the descent of a heavy body in a resisting medium. Camb. and Dublin Math.J. **9**, 145–148 (1854)
10. O.Khorloo Z.Gunjee, B.S., N.Chiba: Wind field synthesis for animating wind-induced vibration. Int. J Virtual Reality **10**, 53–60 (2011)
11. Olesen, H.R., Larsen, S.E., Hjstrup, J.: Modelling velocity spectra in the lower part of the planetary boundary layer. Boundary-Layer Meteor. **29**, 285–312 (1984)
12. Ota, S., Tamura, M., Fujimoto, T., Muraoka, K., Chiba, N.: A hybrid method for real-time animation of trees swaying in wind fields. The Vis. Comput. **20**, 613–623 (2004)
13. Razavi, P.: On the motion of falling leaves. ArXiv e-prints (2010)
14. Reissell, L.M., Pai, D.K.: Modeling stochastic dynamical systems for interactive simulation. Comput. Graph. Forum **20**(3) (2001)
15. Shi, L., Yu, Y., Wojtan, C., Chenney, S.: Controllable motion synthesis in a gaseous medium. The Visu. Comput. **21**(7), 474–487 (2005)
16. Shinya, M., Fournier, A.: Stochastic motionmotion under the influence of wind. Computer Graphics Forum **11**(3), 119–128 (1992)
17. Stam, J., Eugene, F.: Turbulent wind fields for gaseous phenomena. In: Proceedings of the 20th annual conference on Computer graphics and interactive techniques, SIGGRAPH '93, pp. 369–376. ACM, New York, NY, USA (1993)
18. STRINGHAM G.E., S.D.B., GUY, H.P.: The behavior of large particles falling in quiescent liquids. GEOL SURV PROF PAP 562-C **562**, C1–C36 (1969)
19. Tanabe, Y., Kaneko, K.: Behavior of a falling paper. Phys. Rev. Lett. **73**, 1372–1375 (1994)
20. Thuillier, R.H., Lappe, U.O.: Wind and temperature profile characteristics from observations on a 1400 ft tower. J. of Appl. Meteor. **3**, 299–306 (1964)
21. Vázquez, P.P., Balsa, M.: Rendering falling leaves on graphics hardware. JVRB **5**(2) (2008)
22. Wei, X., Zhao, Y., Fan, Z., Li, W., Qiu, F., Yoakum-Stover, S., Kaufman, A.E.: Lattice-based flow field modeling. IEEE Transactions on Visualization and Computer Graphics **10**, 719–729 (2004)
23. Willmarth, W.W., Hawk, N.E., Harvey, R.L.: Steady and Unsteady Motions and Wakes of Freely Falling Disks. Physics of Fluids **7**, 197–208 (1964)
24. Zhang, L., Zhang, Y., Jiang, Z., Li, L., Chen, W., Peng, Q.: Precomputing data-driven tree animation. Journal of Visualization and Computer Animation **18**(4-5), 371–382 (2007)
25. Zhong, H., Chen, S., Lee, C.: Experimental study of freely falling thin disks: Transition from planar zigzag to spiral. Physics of Fluids **23**(1), 011702 (2011)