HedgehogAD: Interactive Design Exploration of Omni-Directional Aerodynamics

Tomoya Kumazaki
The University of Tokyo
tomoyakumazaki@g.ecc.u-tokyo.ac.jp

Kazutaka Nakashima
The University of Tokyo
kazutaka.nakashima@n-taka.info

Tsukasa Fukusato
The University of Tokyo
tsukasafukusato@is.s.u-tokyo.ac.jp

Takeo Igarashi
The University of Tokyo
takeo@acm.org

ABSTRACT

OmniAD [Martin et al. 2015] is a data-driven pipeline for physics-based aerodynamics animations. In the pipeline, a real-time aerodynamic model which handles omni-directional airflow has also been introduced. However, their framework requires a captured motion, which implies that their model is not suitable for designing a customized motion. In this paper, we present a method to add user-controllability to the aerodynamic model in OmniAD by allowing the user to directly interact with the aerodynamics model. The system first visualizes the force and torque coefficients, which represent (x, y, z) components of aerodynamic force (i.e., drag and lift force) and torque, as a set of arrows on a sphere. The user then modifies the arrows on the screen as desired. The system updates the internal representation (parameters for spherical harmonics) and shows resulting animation. We run a user study and the participants successfully designed physically plausible falling motions using the proposed method.

CCS CONCEPTS

• **Computing methodologies** → **Physical simulation**; • **Human-centered computing** → **Human computer interaction (HCI)**.

KEYWORDS

Omni-directional aerodynamics, Falling animation, Controllability

ACM Reference Format:
Tomoya Kumazaki, Kazutaka Nakashima, Tsukasa Fukusato, and Takeo Igarashi. 2020. HedgehogAD: Interactive Design Exploration of Omni-Directional Aerodynamics. In *SIGGRAPH Asia 2020 Technical Communications (SA '20 Technical Communications), December 4–13, 2020, Virtual Event, Republic of Korea.* ACM, New York, NY, USA, 4 pages. https://doi.org/10.1145/3410700.3425425

1 INTRODUCTION

Falling lightweight objects interacting with air show and chaotic motions (e.g., leaves fluttering down, or umbrellas spiraling down).

SA '20 Technical Communications, December 4–13, 2020, Virtual Event, Republic of Korea

ACM ISBN 978-1-4503-8080-5/20/11…$15.00
https://doi.org/10.1145/3410700.3425425

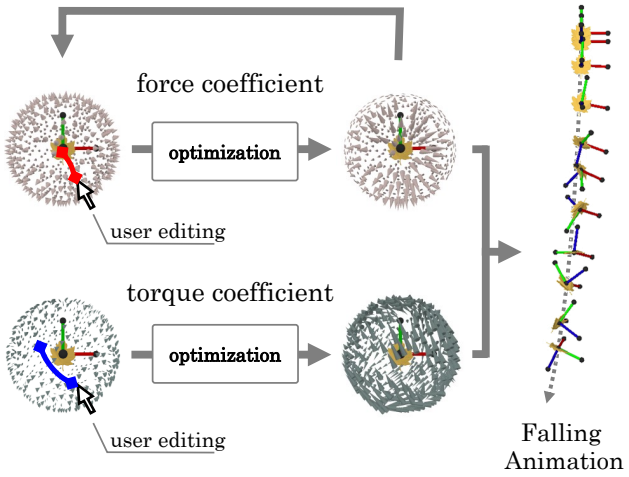

Figure 1: System Overview. The user first selects force and torque coefficients, and edits their magnitudes and directions (left). The system then updates the internal representation (parameters for spherical harmonics) and shows resulting animation (right).

To model three-dimensional (3D) animations, keyframe animation systems are often used. Although these systems allow users to directly control objects' locations and poses, they are unsuitable for designing physically-plausible movements. On the other hand, physics-based simulators (i.e. aerodynamic models) naturally generates physically plausible motions. However, typical physics-based simulation is too slow to iterative design process. In addition, it remains difficult to understand the relationship between the aerodynamic parameters and the chaotic motion.

Martin et al. [2015] introduced an aerodynamic model that handles omni-directional airflow, plus a method to acquire physical parameters (*force coefficients* and *torque coefficients*) based on a captured motion of falling objects in the real world, called OmniAD. These parameters are smoothly interpolated using spherical harmonics, which is used to simulate the chaotic falling motion. However, because their pipeline depends on the captured motion, OmniAD is not suitable to design customized motions in 3D computer graphics scenes.

In this paper, we introduce a method to add controllability to the aerodynamic model in OmniAD. Our method first visualizes

force coefficients and torque coefficients as a set of arrows on a unit sphere. These arrows can be directly modified by simple mouse-based interaction. Then, these modifications are immediately propagated to the internal representation (parameters for spherical harmonics). Consequently, the falling motion simulated by OmniAD's model is also updated and provided to the users. As a design tool, our method provides typical editing functionalities such as undo and redo, allowing rapid design explorations.

Our contributions are as follows.

- A novel method to add controllability to the aerodynamic model in OmniAD [Martin et al. 2015].
- A user interface for designing the chaotic motion of falling lightweight objects.

2 RELATED WORK

2.1 Directable Physics-based Animation

Blending the animator-specified motion, such as keyframes (i.e., shape and pose) and the motion trajectories of objects (i.e., curve editing) with physical simulations, enables us to design physically-plausible animations with good controllability and usability [Kondo et al. 2005; Popović et al. 2000]. However, chaotic behaviors such as falling animations in the air are difficult to imagine key-poses, so these approaches are unsuitable for controlling chaotic behaviors.

Another approach is to simply tune multiple parameters on simulators, but it is difficult to imagine a relationship between parameters and chaotic motions. Twigg et al. [2007] propose a method to find parameters that satisfy user constraints by running physics simulations many times with randomly sampled parameters in a pre-computation stage and visualizing the center-of-mass trajectories of simulated objects (e.g., sphere model). While this method can quickly lead to a dimensional explosion, controlling the simulation is still a problem where much work has to be done. In addition, their browsing does not consider falling objects' rotations. We therefore discuss controlling the simulation in a manner easy to understand.

2.2 Aerodynamic Model

For animating falling lightweight objects in real-time, stochastic models to simulate the interaction between turbulence and a rigid object have been proposed [Xie and Miyata 2014; Yuan et al. 2011]. However, their stochastic process can be difficult to imagine objects' motions, so this is unsuitable for an interactive motion control. Xie et al. [2018] improve a potential theory-based panel method in the aerodynamics, and propose a user interface to design flyable gliders. One problem is that falling animation includes objects' rotations (e.g., screw rotations), but their approach ignores the effects of turbulent flow. OmniAD [Martin et al. 2015] represents an aerodynamic force (drag and lift) and torque for any incoming wind direction based on spherical harmonics. They also analyze trajectories of falling objects from a single video sequence and estimate aerodynamic coefficients to reproduce falling motions in real-worlds. However, this pipeline requires physically-plausible motion trajectories of falling lightweight objects as input. That is, in order to manually edit their parameters, we must prepare the inputs by hand (e.g., drawing trajectories for chaotic motions), but this representation is difficult to understand the effect of force and torque. Furthermore, manually-prepared input motions are not always physically correct. In this paper, we take their aerodynamic model, but consider an alternative approach to edit the parameters.

Figure 2: Screenshot of the proposed system that identifies (a) the force coefficients panel, (b) the torque coefficients panel, and (c) the preview panel showing animation.

3 METHOD

3.1 System Overview

The proposed system visualizes aerodynamic force and torque coefficients used in OmniAD [Martin et al. 2015] as a set of arrows around a unit sphere (Fig. 2 (left)). The user selects arrows within a given radius $r(>0)$ of the mouse cursor, and edit their magnitudes and directions on two modeling panels. These operations (see Fig. 3) are simple but still useful to understand the effect of the air around an object and control it. Next, the system automatically updates spherical harmonics coefficients from the designed vectors. The system animates a falling lightweight object with the updated spherical harmonics coefficients on the preview panel (see Fig. 2 (right)). By repeating this process, users can efficiently design the falling motions that they envision.

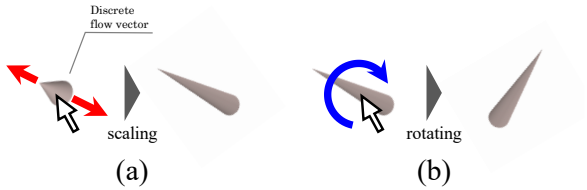

Figure 3: Editing functions: (a) the scaling tool which changes the magnitude of the selected arrow (i.e., force and torque coefficient), and (b) the rotating tool to change its direction.

3.2 Vector Field Editing

As with OmniAD [Martin et al. 2015], we construct two vector fields (i.e., force coefficients and torque coefficients) based on spherical harmonics. We sample M points uniformly on the unit sphere using spherical coordinates $\in (\theta, \phi)$ (in this paper, we empirically set $M = 500$) and define discrete vectors $\boldsymbol{c}(\theta_i, \phi_i)$, $i \in \{0, \cdots, M-1\}$ which correspond to the sampled angles (θ_i, ϕ_i) as follows:

$$\begin{aligned} \boldsymbol{c}(\theta, \phi) &= (Y_0^0(\theta, \phi), Y_1^{-1}(\theta, \phi), \cdots, Y_N^N(\theta, \phi))\boldsymbol{A} \\ &= \boldsymbol{Y}(\theta, \phi)\boldsymbol{A} \end{aligned} \tag{1}$$

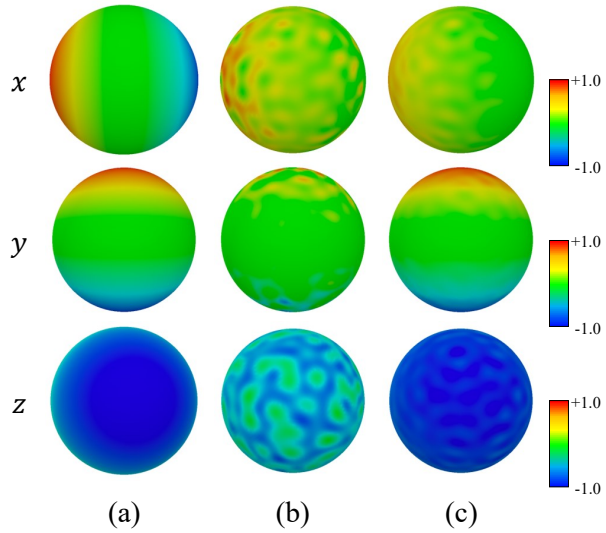

Figure 4: An example of (x, y, z) components of normalized force coefficients based on spherical harmonics: (a) the initial state based on A_0, (b) the updated results without the second term ($\lambda = 0.0$), and (c) our method ($\lambda = 1.0$). Note that the results (b, c) are inferred by editing only the backside arrows.

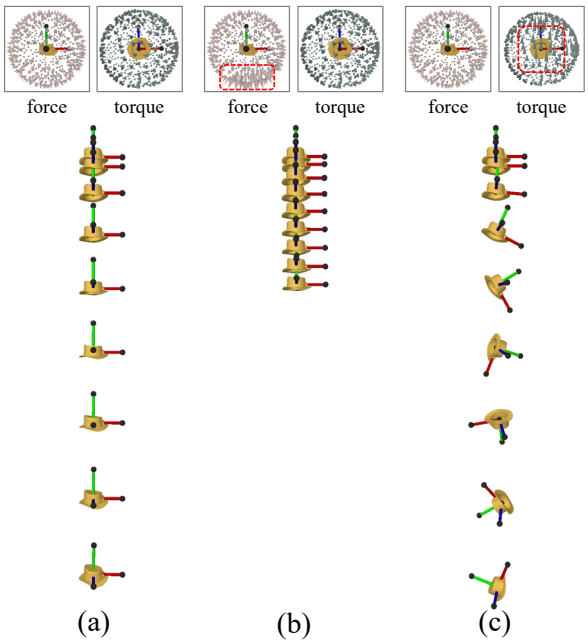

Figure 5: Examples of the user-designed falling motion results from (a) the initial motion. (b) slowing down the speed of falling objects, and (c) emphasizing the rotations around z-axis (blue).

where Y_l^m is a spherical harmonic function of degree l and order m, N is a maximum degree and $\mathbf{A} \in \mathbb{R}^{(N+1)^2 \times 3}$ is a matrix form of spherical harmonics coefficients. This equation corresponds to eq. (7) and eq. (8) in [Martin et al. 2015]. Note that for representing the arrows, it is necessary to set high-degree spherical harmonics because of the large number of control points. In this paper, we empirically set $N = 30$.

When the user edits the magnitudes and directions of the arrows on either of the two sets, the system independently optimizes spherical harmonics coefficients $\boldsymbol{A'}$ for the force or torque coefficients based on the following function.

$$\arg\min_{\boldsymbol{A'}} \sum_i \|Y(\theta_i, \phi_i)\boldsymbol{A'} - \boldsymbol{c}_{\text{edited}}(\theta_i, \phi_i)\|^2 + \lambda \|\boldsymbol{A'} - \boldsymbol{A}_0\|_F^2 \quad (2)$$

where $\|\cdot\|_F$ is the Frobenius norm, $\boldsymbol{c}_{\text{edited}}(\theta_i, \phi_i)$ is the discrete vector at (θ_i, ϕ_i) after user editing, $\boldsymbol{A}_0$ is the coefficients before user editing, and λ is a regularization parameter (in this paper, we empirically set $\lambda = 1.0$).

The first term tries to minimize the difference between the vectors calculated from the updated coefficients $\boldsymbol{A'}$ and the user-edited vectors. The second term is to suppress a drastic change of the spherical harmonics coefficients, and which prevents over-fitting (see Fig. 4). For this optimization problem, an analytical solution can be calculated with interactive speed as follows;

$$\boldsymbol{A'} = (\boldsymbol{Y}^{\mathrm{T}}\boldsymbol{Y} + \lambda \boldsymbol{I})^{-1}(\boldsymbol{Y}^{\mathrm{T}}\boldsymbol{C} + \lambda \boldsymbol{A_0}) \quad (3)$$

$$\boldsymbol{Y} = \begin{pmatrix} Y(\theta_0, \phi_0) \\ \vdots \\ Y(\theta_i, \phi_i) \\ \vdots \\ Y(\theta_{M-1}, \phi_{M-1}) \end{pmatrix}, \quad \boldsymbol{C} = \begin{pmatrix} \boldsymbol{c}_{\text{edited}}(\theta_0, \phi_0) \\ \vdots \\ \boldsymbol{c}_{\text{edited}}(\theta_i, \phi_i) \\ \vdots \\ \boldsymbol{c}_{\text{edited}}(\theta_{M-1}, \phi_{M-1}) \end{pmatrix}$$

4 RESULTS

Our prototype system is implemented using OpenGL and Qt. Note that our system is fast enough for interactive modeling on a MacBook Pro (1.4GHz Quad-Core Intel Core i5 with 16GB RAM). Examples of the designed vector field are shown in Figure 5. Please see our accompanying video for our interface and results.

5 USER STUDY

We invited five participants (P1, $\cdots$, P5) to investigate the potential of the proposed tool. All participants had experience in animation creation. P1 was a professional animation designer with 5+ years experience, in keyframe-based and physics-based designs. P2 and 3 had extensive experience in using keyframe animations and physics-based animations with Blender and Side FX Houdini (> 8 years) to create video games and music video as a hobby. P4 and 5 were experienced users of existing animation software, such as Adobe Flash and After Effects (> 1 year), but had no prior experience in performing physics-based simulators.

First, we gave them a brief overview of our proposed system. Next, we provided them with an example of falling motion and a 3D model. The participants were asked to reproduce the given

Table 1: Questionnaire results.

#	Question item	Mean	SD
1	I thought it was easy to use.	4.20	1.46
2	I did not feel stress.	3.40	1.20
3	I could design the intended animations	4.80	1.17
4	I am satisfied with the visual quality	5.20	1.17

animations as a tutorial task. The participants then design the vector fields to simulate falling objects until obtaining the desired motion. Figure 6 shows examples of the user-created falling animation of a light object. At the end of the field modeling, the participants filled out a questionnaire consisting of four questions about our system's potential (see Table 1) using a seven-point Likert scale (from 1: Extremely dissatisfied to 7: Extremely satisfied).

Table 1 shows the questionnaire results, giving the mean value and standard deviations (SD). The participants' feedback is summarized below.

- P1&2: *Unlike existing software functions such as Unity particle system (without physics-based simulations), I thought that the proposed system is especially useful when controlling the rotation details of falling objects. I want to use this system in game engines.*
- P4: *For designing the falling speed and rotation, the proposed system becomes intuitive in compared with standard keyframe-based systems.*
- P5: *I am glad to qualitatively understand how to improve the visual quality of falling motions through the proposed system (i.e., vector field editing).*

Overall, the participants reported that our system is easy to understand to control aerodynamic coefficients and design chaotic motions. However, some participants had difficulty editing the arrows, as shown below.

- P1: *I have high hopes for further high-speed optimization to return immediate feedback (in less than a second).*
- P2: *With the current operations only, the user-designed arrows might have discontinuous parts. I want to use a smoothing function to set the magnitude and direction of an arrow under the mouse cursor based on the average of its neighboring arrows.*

6 DISCUSSION AND FUTURE WORK

Our system is limited to work with the aerodynamic model in OmniAD, but the concept of editing parameters which are visualized as vector fields can be extended to other simulation scenes. The present paper focuses on direct control of aerodynamic parameters, but it might be worth exploring other operations such as sketching and smoothing as seen in painting tools.

7 CONCLUSION

This paper has presented a method to add user controllability to an aerodynamic model in OmniAD for designing chaotic falling motions. Our system allows users to edit force and torque coefficients around an object, and updates the internal representation (parameters for spherical harmonics). By animating the falling object, users can efficiently explore multiple parameters. We believe that there exists a promising future opportunity to adapt our framework for other aerodynamic models and chaotic motion design process.

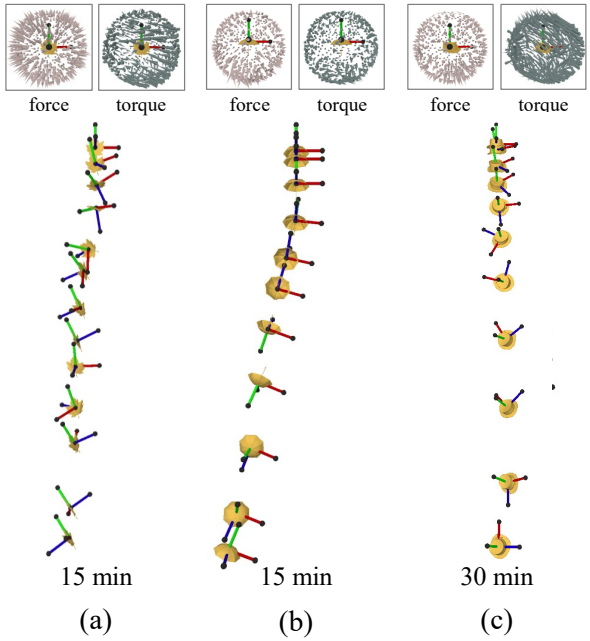

Figure 6: HedgehogAD creations by a group of the user study participants. (a) emphasizing the leaf's rotation around the y-axis (green), (b) slowing down the falling speed when the umbrella falls down with an attitude that its tip (or handle) toward downside and rotate, and (c) changing the falling and rotating speeds based on the hat posture.

ACKNOWLEDGMENTS

We would like to thank Nobuyuki Umetani for many insightful discussions. This work was supported by JST CREST under grant JPMJCR17A1.

REFERENCES

Ryo Kondo, Takashi Kanai, and Ken-ichi Anjyo. 2005. Directable Animation of Elastic Objects. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation (SCA '05)*. ACM, New York, NY, USA, 127–134. https://doi.org/10.1145/1073368.1073385

Tobias Martin, Nobuyuki Umetani, and Bernd Bickel. 2015. OmniAD: Data-Driven Omni-Directional Aerodynamics. *ACM Transactions on Graphics* 34, 4 (2015), 113:1–113:12. https://doi.org/10.1145/2766919

Jovan Popović, Steven M. Seitz, Michael Erdmann, Zoran Popović, and Andrew Witkin. 2000. Interactive Manipulation of Rigid Body Simulations. In *Proceedings of the 27th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '00)*. ACM/Addison-Wesley, USA, 209–217. https://doi.org/10.1145/344779.344880

Christopher D. Twigg and Doug L. James. 2007. Many-Worlds Browsing for Control of Multibody Dynamics. *ACM Transactions on Graphics* 26, 3 (2007), 14:1–14:8. https://doi.org/10.1145/1276377.1276395

Haoran Xie, Takeo Igarashi, and Kazunori Miyata. 2018. Precomputed Panel Solver for Aerodynamics Simulation. *ACM Transactions on Graphics* 37, 2 (2018), 17:1–17:12. https://doi.org/10.1145/3185767

Haoran Xie and Kazunori Miyata. 2014. Real-Time Simulation of Lightweight Rigid Bodies. *The Visual Computer* 30, 1 (2014), 81–92. https://doi.org/10.1007/s00371-013-0783-7

Zhi Yuan, Fan Chen, and Ye Zhao. 2011. Stochastic Modeling of Light-Weight Floating Objects. In *In Proceedings of Symposium on Interactive 3D Graphics and Games*. ACM, New York, NY, USA, 213:1. https://doi.org/10.1145/1944745.1944793