Ballistic Shadow Art

Xiaozhong Chen Sheldon Andrews Derek Nowrouzezahrai Paul G. Kry
McGill University, Montreal, Canada

Figure 1: An example of our ballistic shadow art: a set of chess pieces in an initial arrangement (left) undergo ballistic motion until they cast a targeted THE THINKER-shaped shadow (middle), before continuing on through their ballistic trajectories.

ABSTRACT

We present a framework for generating animated shadow art using occluders under ballistic motion. We apply a stochastic optimization to find the parameters of a multi-body physics simulation that produce a desired shadow at a specific instant in time. We perform simulations across many different initial conditions, applying a set of carefully crafted energy functions to evaluate the motion trajectory and multi-body shadows. We select the optimal parameters, resulting in a ballistics simulation that produces ephemeral shadow art. Users can design physically-plausible dynamic artwork that would be extremely challengingif even possible to achieve manually. We present and analyze number of compelling examples.

Keywords: Shadows, animation, optimization, physics simulation.

Index Terms: I.3.7 [Computer Graphics]: Three-Dimensional Graphics and Realism—Animation

1 INTRODUCTION

The use of shadows in artwork dates back to pre-renaissance time periods, and many contemporary artworks have even relied on the use of shadows as their central visual medium. Specifically, *shadow art* uses sculptures or arrangements of objects to create desired target silhouettes under precise lighting conditions. Contructing these artworks can, however, be a complex and time consuming task.

We propose a method that simplifies the task of dynamic shadow art creation by partially automating the process. Furthermore, our approach creates shadows that form recognizable silhouettes while the objects are in motion, introducing a dynamic aspect that allows the visualized result to be appreciated as both shadow art and kinetic sculpture. Figure 1 gives an example of how ballistic motions produce shadow art of THE THINKERsculpture's profile.

The user provides a binary image that represents a target shadow shape, along with a set of occluders and their starting configurations (i.e., static positions and orientations). We then apply a stochastic optimization technique to determine the initial velocities for the collection of objects such that, at a specific instant in time, they cast a shadow that matches the target silhouette image. This optimization is challenging because the objective function involves a forward multi-body dynamics simulation with contact, giving rise to an inherently sensitive and noisy solution space. The dimensionality of this space also grows linearly with the number of objects, quickly becoming large for more complex scenes. Our framework therefore makes several accommodations that improve the convergence rate and tractability of the optimization problem.

Our paper is organized as follows. In Section 2 we discuss previous work on computer generated shadow art and physics-based simulation involving boundary value problems. We present an overview and formalized version of the optimization problem solved by our framework in Section 3, with specific details of the object functions provided in Section 4. Moreover, we discuss the strategies we use to improve the tractability and convergence of the stochastic optimization in Section 5. Compelling examples of ballistic shadow art synthesized with our framework are demonstrated in Section 6. In Section 7 we provide a discussion of the advantages and limitations of our framework, along with possible directions for future research.

2 RELATED WORK

Our work is built on the foundation prior art in many domains. Specifically, our work involves three aspects. First, shadow rendering techniques allow us to efficiently produce shadows for use in our image objectives. Second, we are inspired by previous work on controlling shadows . Finally, certain techniques from the trajectory optimization literature are of particular interest to our ballistic shadow art problem.

Shadow rendering. Shadow rendering remains a fundamental problem in interactive and offline rendering. Shadows help disambiguate spatial relationships and lighting conditions, and are thus crucial to realistic image synthesis. We require an efficient and accurate shadow rendering technique. Generally there are three options for such high-performance shadow rendering: projected shadows, shadow mapping and shadow volumes.

Blinn [5] proposes an algorithm to create shadows by projecting polygons onto a planar surface. This method is straightforward but has artifacts. It produces false shadows when occluders are not completely above the receiver, and anti-shadows when light source is between occluders and the planar receiver. Furthermore, it requires an offset between the shadow geometry and the planar receiver to avoid co-planar polygon fighting.

Williams [30] first proposed the shadow mapping method. By prerendering the geometry from the light source viewpoint to a depth buffer, this technique can compute light-scene occlusion regardless of the geometric complexity. Zhang [34] introduce a forward shadow

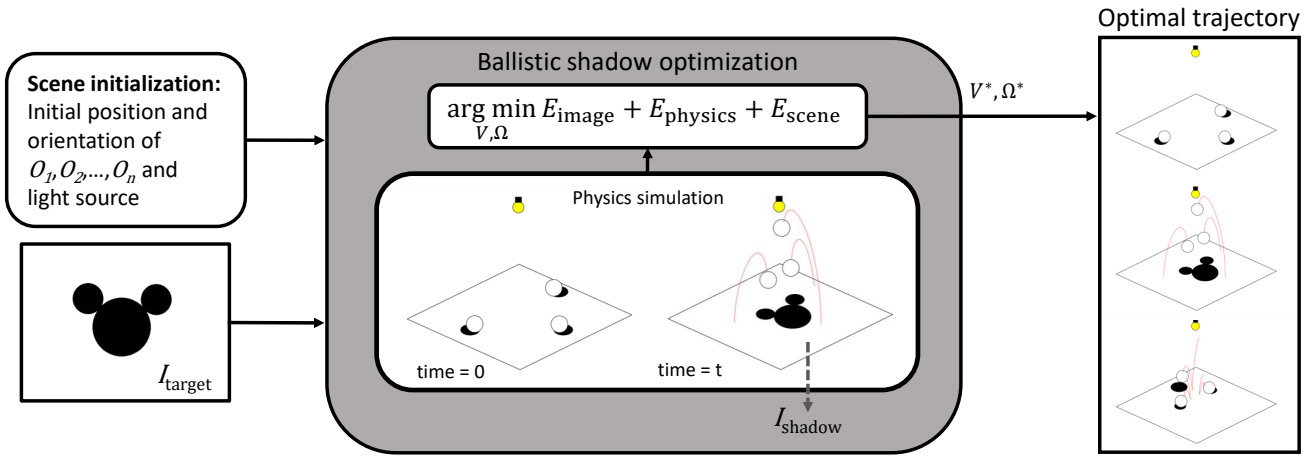

Figure 2: An overview of our method. A point light source (yellow light bulb), planar receiver, and the initial configuration of the light occluders are provided as input. We apply our stochastic optimization technique to find the optimal trajectory that produces the desired shadow shape at time t, minimizing image, physical, and scene objectives. We visualize the optimal trajectory generated by our underlying ballistics simulation (right), and the resulting shadow image I_{shadow} closely resembles that of the target image I_{target}.

mapping method to improve performance, and Reeves et al. [21] improve shadow quality with a filtering and self-shadowing bias.

Crow [6] introduced the shadow volume algorithmcapable of computing shadows from point and directional sources by extruding the volume along the lighting direction from the geometry silhouettes visible to the light. Heidmann [10] propose GPU optimizations for this approach, and Bergeron [3] reduce the volume complexity by eliminating superfluous shadow polygons.

There are many other techniques and variations for improving the rendering of shadows. Woo and Poulin [33] provide a comprehensive survey on this topic. Given the performance, shadow quality, implementation complexity and technical requirements of our application, we opt for planar projection shadows; these are efficient to compute, especially within the inner loop of an optimization.

Shadow editing. Research on editing shadows mainly falls into two categories. The first treats the editing of rendered shadows for visual effects, and optionally adjusting occluders correspondingly or sacrificing correctness. Another research area involves finding a configuration of shadow occluders that cast shadows matching an input target, and potentially fabricating physical examples of such a configuration.

For direct editing, Poulin and Fournier [20] describe how to allow a user to interact with shadows and highlights in a rendered image, providing an efficient alternative to manual light parameter edits and re-rendering. Pellacini et al. [18] present a user interface to permit artists to interactively design shadows in animated feature films. DeCoro et al. [7] propose an algorithm for rendering stylized shadows. They provide controls on tuning artistic properties for shadow rendering, such as abstraction and softness. Obert et al. [17] describe a method for editing visibility for the design of all-frequency shadows. Mattausch et al. [14] present a method for editing the boundaries of shadows inspired by freeform deformations. More advanced techniques also allow for direct and intuitive control of complex illumination and reflection effects, even in the context of global illumination [23], and these works inspire our shadow geometry manipulation solution. We refer interested readers to Schmidt et al.'s survey on the topic [24] For ballistic shadow art, the complexity, indirectness, and demand for precision requires an alternative approach.

In terms of arranging or generating occluders, Mitra et al. [15] present tools to design voxel-based occluders that cast different shadows when illuminated from different directions. Bermano et al. [4] provide a method of producing a 3D printable height field illustrating multiple images from self-shadowing. Baran et al. [1] fabricate a multi-layer light attenuator that casts multiple colored shadows of several target images under different light configurations. Won et al. [32] solve the very difficult optimization problem of using human forms to create shadows. Inspired by shadow theater, they are able to produce silhouettes and subtle animations. We use a similar technique for solving for the movement of our dynamic shadow casters, except that our results involve large motions and we prevent collisions with hard constraints rather than a penalty function.

Trajectory optimization. The application of physics-based dynamics equation to the synthesis of 3D animation sequence is a foundational topic in computer graphics research. While a good deal of this work focusses on solving initial value problems, our framework solves a boundary value problem (BVP): for two scene geometry configurations and two points in time, we search for a trajectory that connects the two together. The prominent work by Witkin and Kass [31] on space-time optimization falls into this category. Popović et al. [19] also describe a method that provides solutions for controlling rigid body simulations via interactive manipulation of object trajectories. The method allows contacts to occur in the trajectory, and handles each motion between contacts separately, performing a local discrete search on parameter space to resolve the discontinuities. Such a method could be adapted to work with shadows. However, when the number of rigid bodies increases and contact becomes frequent, it may become very difficult to converge to a solution. In contrast, Twigg et al. [29] present a method that uses backward time stepping to produce animations involving rigid bodies and frictional contact. Given a final target configuration of objects that produce a target shadow, their method could produce occluder trajectories, though the method allows for minor violations of physics. In our work we use a shooting method to find physically correct trajectories while searching for the parameters that produce the target image.

Finally, we note that there is a collection of other works that strive to control physics simulations. In fluid simulation, various methods have been explored for having the surface momentarily match a target shape mesh or mesh animation [25–27]. Likewise, there is related work in controlling smoke such that it interpolates keyframes or forms target shapes [2, 13, 28].

3 OVERVIEW

Figure 2 provides a sketch of how our method generates ballistic shadow art. The input to our framework is a binary image I_{target} defining the desired shadow shape and a set of n occluders $(O_1, O_2, \ldots, O_n)$ with user-specified launching positions, orientations, geometries, and masses. The user also specifies a duration t which is the time in seconds when the shadows cast by the occluders should match the desired shape. Shadows are projected onto a planar surface by a static point light source and observed from a fixed viewpoint with a pinhole camera model. The configuration of the light source, projection surface, and camera are part of the scene definition.

The core of our framework is a multi-objective optimization that determines the initial velocities of each occluder such that, at time t, the shadows cast by the occluders closely resemble the target image I_{target}. The vectors $v_i \in \mathbb{R}^3$ and $\omega_i \in \mathbb{R}^3$ store the initial linear and angular velocities, respectively, of each occluder body i, and collectively for all occluders this is denoted

$$V = (v_1, v_2, \ldots, v_n) \text{ and } \Omega = (\omega_1, \omega_2, \ldots, \omega_n).$$

The optimization finds velocities that minimize a multi-objective cost function, or formally

$$\underset{V,\,\Omega}{\operatorname{argmin}}\; E_{\text{image}} + E_{\text{physics}} + E_{\text{scene}}, \tag{1}$$

where E_{image}, E_{physics} and E_{scene} are energy functions that introduce penalties pertaining to image, physics, and scene criteria.

A forward dynamics simulation with gravity is used to update the position and orientation of each body. This produces ballistic trajectories that are the signature feature of our framework. Collision detection is also enabled, and so intersecting bodies generate contact forces to resolve penetration. Since we are optimizing for the initial conditions of a physics simulation involving contacts, the solution space is non-convex and highly discontinuous. The CMA-ES [8] method is a stochastic optimization technique that is well suited to these conditions, and so it is used to minimize the problem in Equation 1.

In the next section, details are provided on the terms that compose each energy function and how their values are computed.

4 ENERGY FUNCTIONS

The energy functions are categorized into three groups: image comparison, physics simulation, and scene settings. The image comparison functions focus on matching the simulated shadow with the target image. The physics simulation functions are designed to avoid unreasonable or implausible solutions. Finally, the scene setting functions penalize unwanted scene arrangements, and also provide opportunity for users to give guidance or specify demands in building such dynamic scene. For instance, the user may want a sharp feature of a specific occluder to correspond with a particular corner in the target image.

4.1 Image comparison

Energy functions on image comparison take a simulated shadow image, I_{shadow}, and a target shape image, I_{target}, as input. Both images are represented in binary format, such that

$$I(x,y) = \begin{cases} 1, & \text{pixel at } (x,y) \text{ is in shadow,} \\ 0, & \text{pixel at } (x,y) \text{ is not in shadow.} \end{cases}$$

The energy function on image space is defined as

$$E_{\text{image}} = \sum_{i=0}^{2} w_i d_i + w_{\text{XOR}} E_{\text{XOR}} + w_{\text{in}} E_{\text{in}} + w_{\text{out}} E_{\text{out}}$$

where w denotes the weight of the energy term with the corresponding subscript, and d_i is the distance between image moments of ith order. The simulated shadow and target image are compared with an exclusive-OR operation in E_{XOR}. As well E_{inner} and E_{outer} encourage that the inner and outer boundaries of the images match, respectively.

Image moments distance

Image moments [16] are succinct descriptors of an image and they are widely used in computer vision and computer graphics applications [12, 22]. They are effective for carrying low-dimensional information and we include them into our framework.

The energy functions d_0, d_1, d_2 are the distance between moments of the shadow image and the target shape,

$$d_i = \left\| M_i(I_{\text{shadow}}) - M_i(I_{\text{target}}) \right\|,$$

where M_i is a function computing the ith order moment of an image.

The 0th order moment is the total area of the image expressed in pixels. Thus the distance gives the difference of the shadow size. This moment is computed as

$$M_0(I) = \sum_x \sum_y I(x,y).$$

The 1st order moment represents the 2D centroid of a binary image in pixels and the distance is computed by the Euclidean norm. This moment is computed by

$$M_1(I) = \begin{pmatrix} \sum_x \sum_y x\, I(x,y) \;/\; M_0(I) \\ \sum_x \sum_y y\, I(x,y) \;/\; M_0(I) \end{pmatrix} = \begin{pmatrix} \bar{x} \\ \bar{y} \end{pmatrix}.$$

Finally the 2nd order image moment is a 2×2 matrix

$$\begin{bmatrix} \mu_{20} & \mu_{11} \\ \mu_{11} & \mu_{02} \end{bmatrix},$$

where

$$\mu_{pq} = \sum_x \sum_y (x - \bar{x})^p (y - \bar{y})^q I(x,y) \;/\; M_0(I).$$

This matrix represents the inertia tensor of the input image in two dimensional space. Note that the off-diagonal elements are both μ_{11}. To avoiding repeated calculations in the distance, the 2nd order image moment is defined as a vector

$$M_2(I) = \left(\mu_{20},\, \mu_{11},\, \mu_{02}\right),$$

and the distance computed by the Euclidean norm.

Image difference

The image moments help during early stages of the optimization to ensure that the shadow coarsely matches the target image. To achieve more exact alignment, we design energy functions that compare shadow images per pixel. For comparison of the full image, a binary exclusive-OR operation is performed between the simulated shadow image and the desired shape image:

$$E_{\text{XOR}} = \sum_x \sum_y I_{\text{shadow}}(x,y) \veebar I_{\text{target}}(x,y).$$

To improve detail matching, we also design energy functions to match the boundaries of the target shape. There are two types of boundary considered: the inner boundary and the outer boundary. For the inner boundary, E_{in} is defined as

$$E_{\text{in}} = \sum_x \sum_y I_{\text{shadow}}(x,y) \wedge I_{\text{in}}(x,y),$$

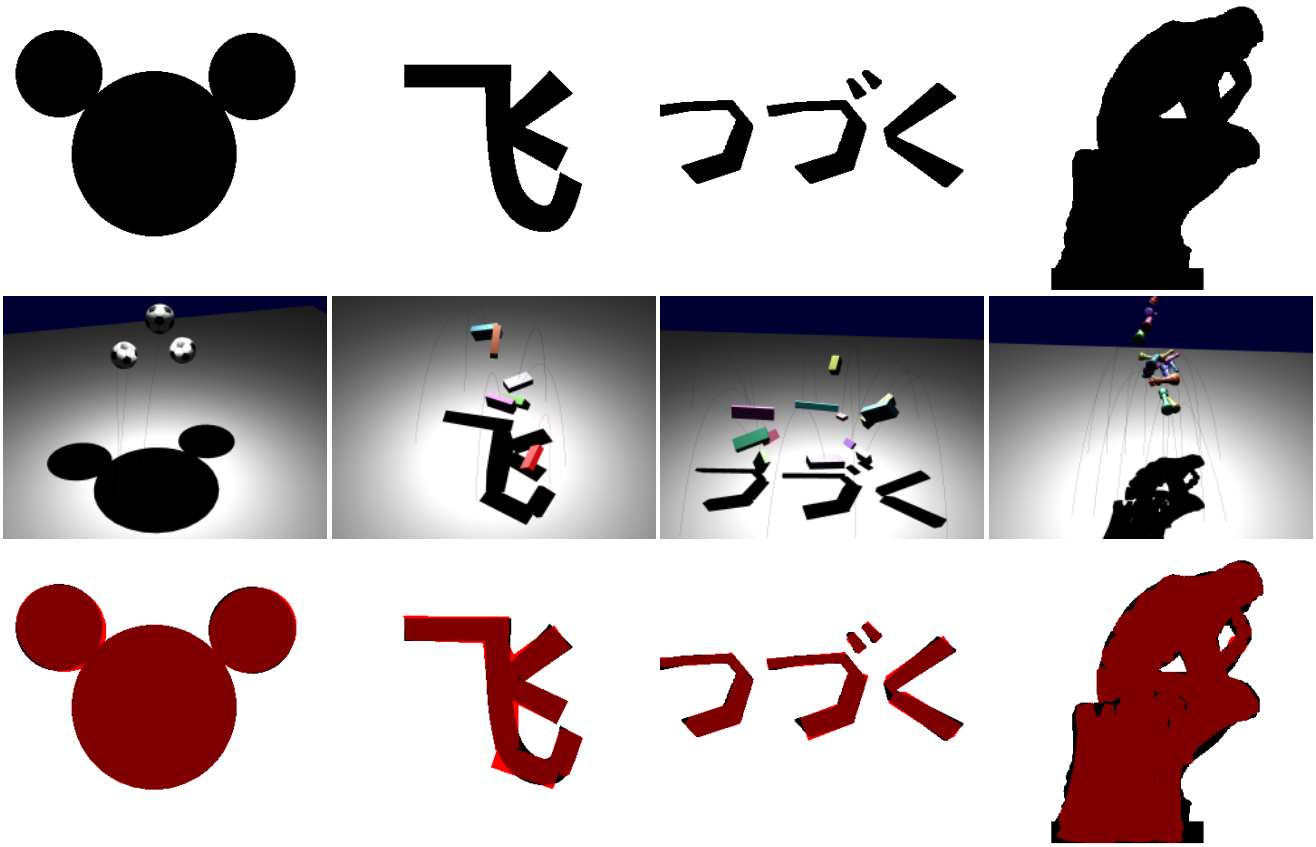

Figure 3: The results obtained with our method for various examples. From left-to-right: Mickey, "fly" Chinese character, "to be continued" in Japanese, and THE THINKER. The target images provided as input (top row), are closely matched by the simulated results found by running the optimization algorithm (middle row). A comparison of the target image and shadow image for each example (bottom row) clearly demonstrates the accuracy. The target images, drawn in black, differ from the simulated shadow images, drawn in red, in only a few small regions.

with I_{in} as the inner boundary image. This energy function encourages the shadow to cover the target silhouette outline by using the AND operation. For outer boundary matching, we define E_{out} as

$$E_{\text{out}} = \sum_x \sum_y 1 - (I_{\text{shadow}}(x,y) \wedge I_{\text{out}}(x,y)),$$

with I_{out} as the outer boundary image. Similar to the inner boundary, the matching is calculated with a negative-AND operation. With this design, the shadow is discouraged from intersecting the outline.

To compute the boundary images, erosion and dilation operations are applied to the target image. For the inner boundary it is computed as

$$I_{\text{in}} = I_{\text{target}} - I_{\text{target}} \ominus K,$$

and for the outer boundary as

$$I_{\text{outer}} = I_{\text{target}} \oplus K - I_{\text{target}}.$$

We apply K as the kernel in image dilation and erosion. In this case K is a 5×5 matrix with all elements set to 1.

4.2 Physics simulation

The physics simulation objectives guide the optimization toward plausible solutions and avoids unwanted ballistic trajectories. There are two terms in this energy function

$$E_{\text{physics}} = w_{\text{contact}} E_{\text{contact}} + w_{\text{reg}} E_{\text{reg}}$$

where E_{contact} penalizes contact between occluder objects before time t, and E_{reg} serves as a regularization term on the initial linear and angular velocity of shadow casters.

Since contacts are difficult to predict and add noise to the solution space, we try to avoid solutions where contact occurs before the moment at which the target shadow shape is formed. The simulation therefore terminates whenever (i) contact occurs or (ii) the simulation time reaches t. This ensures a collision free trajectory before time t.

To smoothly guide the optimizer towards a collision free solution, the contact penalty is computed as different between the actual simulation time and the target time t. In other words, this is the time remaining for the simulation to reach the designated shadow casting moment, and so it will reach zero if contacts do not occur. Formally, this penalty is computed as

$$E_{\text{sim}} = (t - t_{\text{sim}}) n_{\text{contact}}$$

where t_{sim} is the actual duration of the simulation. A factor n_{contact} is used to scale the time difference by the number of contacts detected at termination.

A regularization term, E_{reg}, is also used to avoid large velocities for the occluders, such that

$$E_{\text{reg}} = \alpha \sum_{v \in V} \|v\| + (1-\alpha) \sum_{\omega \in \Omega} \|\omega\|.$$

The scalar $\alpha \in [0,1]$ is used to balance linear and angular velocities in case their magnitudes are significantly different. A value of $\alpha = 0.5$ for our examples.

4.3 Scene settings

The scene setting function ensures that the occluder objects contribute to the target image in a reasonable way, and we also take advantage of human intuition to guide the final configuration of occluders. With these objectives in mind, the scene setting energy function is defined as

$$E_{\text{scene}} = w_{\text{hint}} E_{\text{hint}} + w_{\text{bar}} E_{\text{bar}}$$

where E_{hint} denotes how well the occluders in the scene make use of user defined hints, and E_{bar} is a barrier function that penalizes occluders casting shadows out of the desired region.

The energy term for hints is similar to the one used by Won and Lee [32], which is used to align the projected 3D features of a character model to points on the target shadow contour. In our case, we match points on the occluder surface to pixels in the target image. The energy term is defined as

$$E_{\text{hint}} = \sum_{h \in H} w_h \| P(x_h) - p_h \|,$$

where $H = \{h_1, h_2, \ldots, h_k\}$ denotes a collection of k hint points given by the user. Each h is defined as

$$h = \{O_i, x, p, w \mid i \in [1,n], x \in \mathbb{R}^3, p \in \mathbb{R}^2, w \in \mathbb{R}\}.$$

and denotes a single hint that matches a location p in image space with a position x on occluder O_i; a weighting factor w is used to prioritize hints. A projection $P : \mathbb{R}^3 \mapsto \mathbb{R}^2$ is used to map the 3D coordinate x to a position in the shadow image. Note that the projected point can be outside the region of the shadow image. The hints used for THE THINKER example are shown in Figure 4.

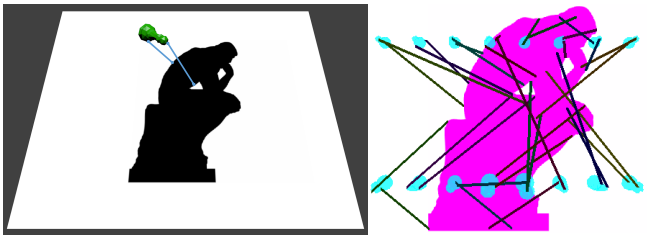

Figure 4: The diagram illustrates how hints are specified as positions on an occluder object, and then mapped to corresponding locations in the target shadow images (left). All hints used for THE THINKERexample are shown in image space (right).

Another energy function E_{bar} is designed to avoid the "waste" of occluders. There are cases when one or more occluders do not contribute to the shadow image because they cast shadows outside the image region. Previous energy functions may fail to prevent this. For instance, if two occluder shadows move outside the region, it's possible that nearby solutions also move the occluders outside shadow casting region. The energy function could plateau for these solutions and convergence will be difficult.

The barrier function is specifically designed to avoid these situations and encourage viable solutions. It is defined as

$$E_{\text{bar}} = \sum_i B(P(C_i)),$$

with $P : \mathbb{R}^3 \mapsto \mathbb{R}^2$ the same projection as above, C_i denoting the center of mass of occluder i in model space, and $B : \mathbb{R}^2 \mapsto \mathbb{R}$ the barrier function on a single occluder:

$$B(x) = \begin{cases} 0, & \|x-c\| < r, \\ \|x-c\|, & \|x-c\| \geq r. \end{cases}$$

Table 1: complexity of each examples

	Mickey	"to fly"	TBC	The Thinker
occluder number	3	6	12	16
hint number	3	6	12	32

Here, c is a point in image space, in this cases we use the center of the whole image, and r is a user-defined radius. In our cases, we choose half of the image height for the radius.

5 Optimization Strategies

As previously indicated, the optimization problem being minimized by our framework is non-convex, involves discontinuities, and may also be high-dimensional if n is large. For these reasons we applied different kinds of strategies in building scenes to cast desired shadows, especially ones with bigger amount of occluders to manipulate, and we integrated these strategies in our framework. These two types of strategies that we employed that can produce solutions to the problem: scheduled optimization and iterative optimization.

5.1 Scheduled optimization

At the early stages of optimization with bigger deviations, it would be useful to start with simpler energy functions and leave others aside for later refinement. Energy functions such as image moments and the barrier function have generally less noise in a wide range of samplings. The overall smooth shape helps guiding the sampling range quickly to narrow down into a smaller area that contains desired solution. By then users can restart the optimization, enabling energy functions that focus on local details, such as the exclusive-or comparison. Optionally users can adjust energy weights to emphasize on different aspects of convergence.

5.2 Iterative optimization

As the number of projectile increases, contacts between each other become more difficult to avoid. Especially when the shadow occluders are launched from clustered positions, most of the early sample evaluations will end up with penalty on contact in substantial amount, and they change dramatically with very small deviations. Therefore, the solution space is filled with high frequency changes, the sampling results will appear noisy, and the convergence becomes inefficient.

For this reason we made attempts to improve the convergence rate by a greedy strategy with optional "back-tracing", which actually produces nice results and converges more efficiently in our experiments. The strategy is actually simple: with an order of all occluders, either generated, randomized or user-defined, it iterates all occluders individually by sampling, simulating and optimizing on only one pair of initial linear and angular velocities, meanwhile keeping the rest fixed, until all of them have reached optimality.

The iterative optimization can also fit in the scheduled strategy. After iterating on all occluders, users can set up another stage that optimizes on all projectiles at once, but with smaller sampling deviations. By this means we carried out an overall refinement of previous result, and potentially can escape local minima.

6 Results

The top row of Figure 3 shows the four target shapes that we use as example problems. They are a Mickey Mouse logo, a simplified Chinese character of "to fly" in boldface Gothic typeface, a Japanese phrase of "to be continued" (later shortened as "TBC") in an artistic font, and a silhouette of THE THINKERby Auguste Rodin.

The complexity of each example's occluders is specified inside of Table 1. Each example is provided different set of occluders as

Figure 5: The results of the two stages of the Mickey example. The initial stage on the left and the final stage on the right.

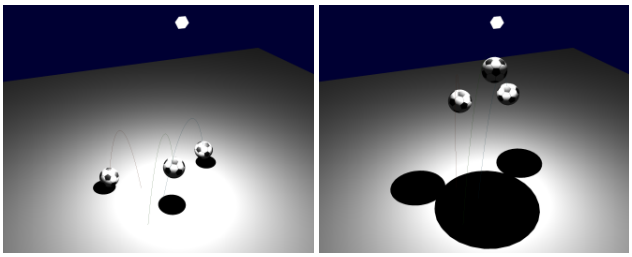

Figure 6: The converged scenes of the two stages of Mickey. The initial stage on the left and the final stage on the right. For each one, trajectories are indicated with the curved lines and the light source with the white dot on the top.

Table 2: Weight of each energy function of the Mickey example

energy	stage 1	stage 2
0th moment	2	2
1st moment	10	10
2nd moment	5	5
XOR	0	300
regularization	0.01	0.1
barrier function	100	100
contact	50	50
hints	0.1	0.1
inner boundary	0	0
outer boundary	0	0

input. For the Mickey example, they are three spheres. For the "to fly" example we have a few brick-shaped cuboids and for the "to be continued" we have diverse dimensions of bricks. For THE THINKERexample, half a set of chess pieces are provided. These chess pieces are also simulated with their reduced triangle mesh for collision handling.

The converged scenes are presented with snapshots in the middle row of Figure 3. Each snapshot shows the instant when the shadow forms the target, and includes colored curved lines indicating the the ballistic motion of the occluders. The point light source is indicated with a white dot at the top of each snapshot. The converged shadows in image space are also presented in the bottom row of Figure 3, where the black shape indicates the target to reach and the semi-transparent red highlight represents the shadow in the scene.

6.1 Convergence

We apply different optimization strategies for these four examples. For the Mickey example, we schedule a two-stage optimization, and for the other three examples we used the iterative strategy that optimizes for one individual occluder at one stage.

For the Mickey example, the weights of two stages are presented in Table 2. First we started with energies that focus on less detailed aspects, such as moments and hints, along with penalties on wild solutions, such as the barrier function. When the shadows converge to a generally matched shape, we added in more energy functions to refine the details, optionally starting from where we left off with small sampling deviations. In this case we only use exclusive-or comparison since the Mickey logo does not require many details to recognize. Note that in this Mickey example, the sphere projectiles still have angular velocities. They are included for framework consistency concerns, but they have no effect on convergence. Therefore we also increase the weight for regularization to reduce the unnecessary angular velocities in the second stage. The converged results of each stage are given by Figure 5 on the shadows and Figure 6 on the scenes.

In Figure 7 we also present the process of convergence of two stages. In the first stage, the total value of all energies dropped quickly with the guidance of hints and the first moments. Besides, at the early stages there are a few wild samples, potentially with occluders running outside of the region for capturing, or into each other, penalized by the barrier function and the contact penalty. After adding the exclusive-or and increasing regularization, we started the second stage from the previous position. Less wild sampling occurred this time and the shadows converged to a satisfactory shape, as the exclusive-or decreased to a small magnitude. By then all other energy functions started to plateau except the regularization, which led to the solution with smaller angular velocity but the same visual effect.

The similar convergence process happens in other examples, except that energy function weights remain the same among all iterative stages. Note that image moments are not as useful as before in the iterative stages, since in every stage the shadows do not need to fit the general shape of the target, but instead matching part of the target can be sufficient, and thus image moments will bring unwanted side effects. Furthermore, as the problem dimension has decreased, guiding convergence to a subspace via rough information is marginally useful. For these concerns we drop the image moment based energies and use local detail oriented energies directly. As an example we present the evolution process of THE THINKERexample. The first 16 stages are optimized for one chess piece in Figure 8. In the last stage we increased the weight on boundaries, fixed the initial conditions of most of the occluders and optimized only on the pieces that form the statue's back to have a smoother outline. The improvement is demonstrated in Figure 9. The back of THE THINKERis improved with a smoother outline, with a minor trade-off over the stomach. Meanwhile the rest of THE THINKERis kept untouched as the overall shape is satisfactory, with the constraints of the chess piece shapes.

6.2 Performance

We also provide some performance data in Table 3 and Table 4 for reference. Table 3 presents some data indicating how far it takes to converge for each example. Apparently for more complicated problems, we need more iterations and samplings to converge. Moreover, using different optimization strategies makes a difference on its convergence, as the latter three examples have a low average sampling and iteration number on each stage.

In the other table, we list the time it takes for computing each energy function in milliseconds, except for the energies of regularization and contacts, which both take less than 0.1 milliseconds. Besides energies we also conducted profiling on ballistic simulations and included the results together. One obvious bottle-neck on performance is the physics simulation. Besides that other energies have a decent rate of computation. Synchronization of threads, optimization algorithm, hard-drive IO, and other potential sources of significant time consumptions are not taken into account.

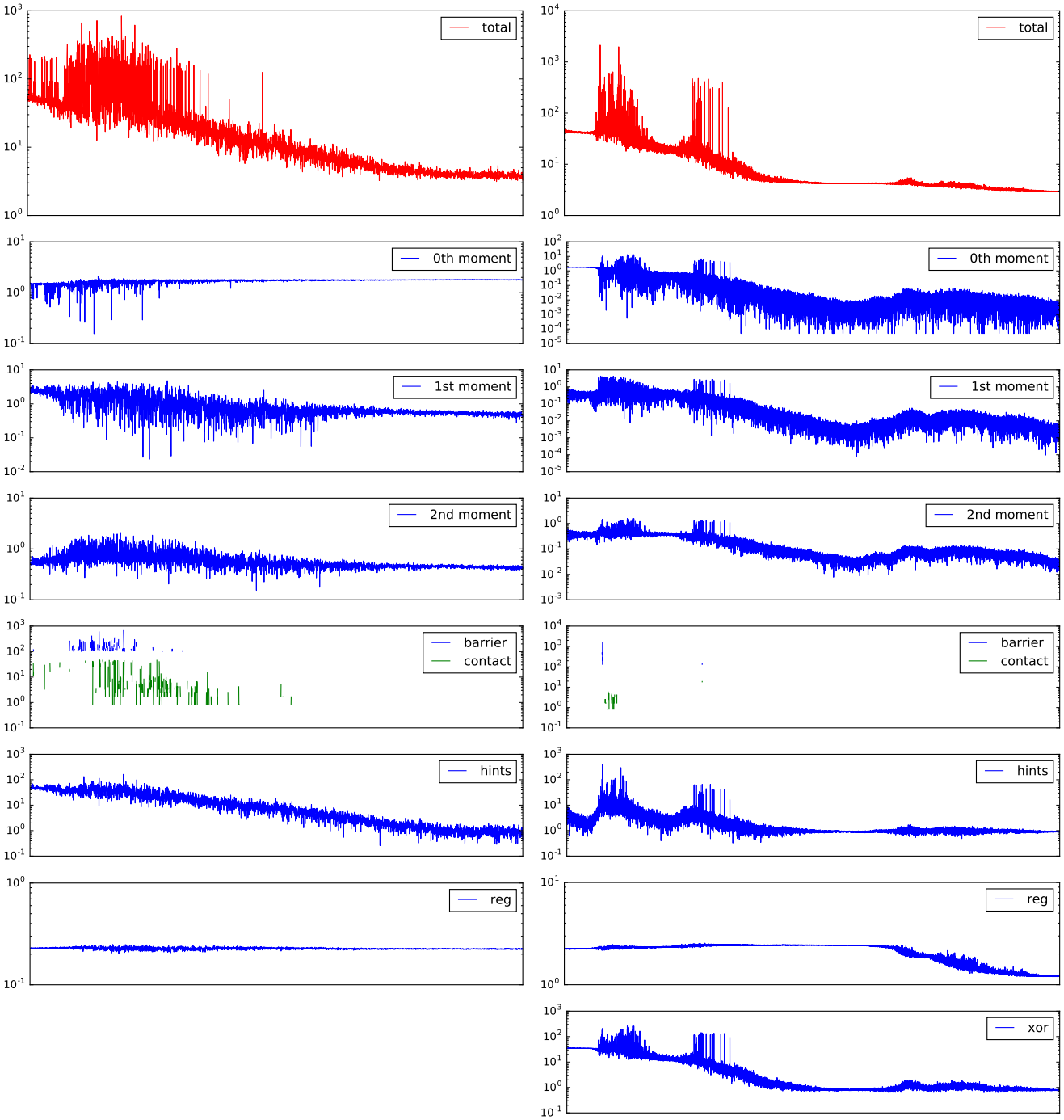

Figure 7: Convergence of the two stages of the Mickey example. The left column presents convergence of the first stage and the right presents the second stage. Each figure plots the evaluated value of every sampling instance along the convergence. The red denotes the total of all applied energies, and the blues denote the weighted value of each energy, short named in the legends. The contact and barrier function are plotted together for space saving purpose. Each plot has a y-axis on log10 scale and therefore the missing values indicate 0. Note the newly added exclusive-or function's plot at the right bottom, and the y-axis with scale may have changed.

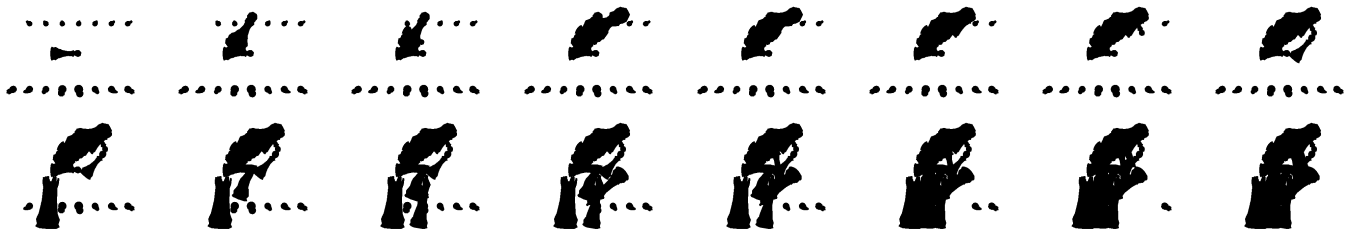

Figure 8: The evolution of the Thinker shadows. Each image indicates the converged result of each stage. For every stage there is only one chess piece being manipulated with initial conditions for optimization. The rest of the chess pieces are kept untouched, either remaining still at their launching positions, casting the small pieces of shadow that line up straight, or adopting the same velocities from last stage convergence and casting the same shadows.

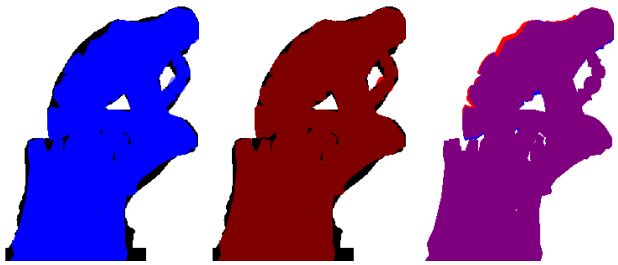

Figure 9: THE THINKER shadow from the extra stage and comparisons with the last stage and target. From left to right: the comparison between the shadow from last stage (blue) and the target (black); the comparison between the shadow of the extra stage (red) and the target (black); the comparison between the shadows of the extra stage (red) and the last stage (blue) and purple indicates overlapping.

Table 3: Performance of optimizations. The optimization process of each example is presented with the numbers of stages, iterations, and samplings. The convergences are also timed and the results are denoted in minutes and seconds.

	Mickey	"to fly"	TBC	The Thinker
stages	2	6	12	17
total iterations	283	458	987	1572
average iterations	141.5	76.3	82.25	98.25
total samplings	20376	10992	23688	54048
average samplings	10188.0	1832.0	1974.0	3378.0
total running time	29m07s	16m55s	44m57s	396m15s

6.3 Implementation

We used projected planar shadows for shadow rendering in our framework, and we discuss the implications of this in Section 7. For specifying the target shape and capturing shadows, we used a image of 640×480 pixels. For simulating the ballistic motions, we used the semi-implicit Euler method for integration and the time step is $1/60$ second.

The framework is mainly written in Python 2.7.12. We used OpenGL 4.3 for rendering, Python Image Library and OpenCV for image processing, Vortex for physics simulation, and NumPy and Python Computer Graphics Kit (cgKit) for matrix computation. The operating system is Windows 10 Home. In terms of hardware, all our results, as well as performance tests in Table 3 and Table 4 are run on a PC with processor Intel Core i7-4710HQ @ 2.50 GHz and GPU of NVIDIA GeForce GTX 860M.

Table 4: Performance of energy functions. All functions in different examples are all timed in milliseconds. For those energy functions that are not applied in some examples, the value is labeled as "N/A"; for those not included in this table, the evaluation is trivial and lasts less than 0.1 milliseconds. Besides energy functions, the physics simulation is also timed and presented.

	Mickey	"to fly"	TBC	The Thinker
simulation	15.2	19.8	31.7	621.0
0th moment	1.6	N/A	N/A	N/A
1st moment	4.9	N/A	N/A	N/A
2nd moment	16.1	N/A	N/A	N/A
XOR	8.4	8.3	6.2	7.4
inner boundary	N/A	N/A	N/A	18.7
outer boundary	N/A	N/A	N/A	17.2
hints	1.2	1.4	1.4	3.9
barrier	1.0	1.6	1.6	2.3

7 DISCUSSION AND FUTURE WORK

The results in Section 6 demonstrate that our framework is capable to synthesize compelling examples of ballistic shadow art, even when the simulation involves more than a dozen objects and the target shadow is complex. The user-in-the-loop aspect of our framework allows visually pleasing solutions to be found by guiding the optimization.

We render shadows by planar projection. Compared to shadow maps or shadow volumes, projected shadows are easy to implement, efficient to render, and have a perfect resolution, which is useful for image comparison purposes. However, the technique produces fake shadows when an occluder is located behind the plane, or anti-shadows when it is located behind the light. We note that it is possible to fix these issues using existing methods [9, 33], but does not address a major shortcoming, which is the exclusion to non-planar shadow receivers. To exploit the full potential of our framework, we plan to use shadow mapping or shadow volume instead.

There are some factors defining the shadow art problem other than target or occluders that we did not enumerate in our experiments. For instance, so far our shadow art effects are only supposed to be captured from the same static camera, which is pointing to the receiver plane perpendicularly. The light source generally lies above the shadow center in all cases. To demonstrate the power, there should be experiments on problems with more diverse configurations. Scene construction also could further be automated by optimizing for the light configuration and camera parameters. However, this would introduce additional non-linearities to the optimization, not to mention increasing the dimensionality of the problem. This deserves further investigation so that tractability is not severely impacted.

When comparing the captured and target shadow images, the

optimization attempts to find a perfect match at the target location that is specified by the user. As part of future work, we intend to investigate translated, rotated or scaled matching. For example, by accepting slightly deformed shapes [11]. Our framework also only supports optimizing only a single target and we plan to investigate synthesizing shadows for multiple targets. One possiblity is to use multiple light sources for the static placement of occluders [4,15] and apply our framework to find a set of ballistic trajectories. However, this may require very complex arrangements of the occluders and thus the convergence and simulation will be challenging. Another route could be to match different target images at different instances along the ballistic trajectory. For example, the projectiles cast a shadow of a target shape at one instant, and at a later instant they form another target shadow. Timing becomes critical in such cases, as a poor selection of instants by the user may mean there is no feasible physical solution.

Another drawback of our framework is that the resulting effect is transient, and the desired shadow image is formed during only a few frames of animation. However, we hypothesis that the optimization could be coaxed to find solutions where the perceived effect lasts for a longer duration. For example, by adding an energy function that minimizes the spatial velocity of the occluders at time t this would implicitly lengthen the duration of the effect. Alternatively, an energy function that explicitly tries to increase the duration of the effect could also be introduced by evaluating the image difference function, E_{XOR}, over a window of frames in the neighborhood of t.

Contacts have been one factor we try to suppress in the framework. The friction and the bounce that contacts produce will lead to a tremendous amount of differences in the final status of ballistic motions, even with minor adjustments to starting conditions. Therefore in the solution space, regions with contacts are filled with high frequency changes. The sampling on this type of region is not representative unless with small enough deviations. However, because of the dramatical changes that contacts can bring, allowing contacts to occur before matching the target may produce more solutions. For example, we can have two projectiles deflect from each other to change their trajectories, so that they can reach places in a way that used to be impossible. Therefore this may be very useful for multiple target problems, but it requires the capability of optimization algorithms to search within the contact subspace effectively.

Finally, successfully fabricating this ballistic shadow art in reality will be persuasive but difficult. To build it we need more precise physics simulation, including smaller time steps, bringing drag and accurate measuring projectile contact properties if it applies. We also need stable, reliable and accurate methods for launching occluders into desired ballistic trajectories. Furthermore audience needs a more effective expression to demonstrate the transitory art effect.

Acknowledgments

The authors gratefully acknowledge the financial support of NSERC and FRQNT.

References

[1] I. Baran, P. Keller, D. Bradley, S. Coros, W. Jarosz, D. Nowrouzezahrai, and M. Gross. Manufacturing layered attenuators for multiple prescribed shadow images. In *Computer Graphics Forum*, vol. 31, pp. 603–610. Wiley Online Library, 2012.

[2] A. Barnat, Z. Li, J. McCann, and N. S. Pollard. Mid-level smoke control for 2d animation. In *Proceedings of Graphics Interface 2011*, GI '11, pp. 25–32, 2011.

[3] P. Bergeron. A general version of Crow's shadow volumes. *IEEE Computer Graphics and Applications*, 6(9):17–28, 1986.

[4] A. Bermano, I. Baran, M. Alexa, and W. Matusk. Shadowpix: Multiple images from self shadowing. In *Computer Graphics Forum*, vol. 31, pp. 593–602. Wiley Online Library, 2012.

[5] J. Blinn. Jim Blinn's corner: Me and my (fake) shadow. *IEEE Computer Graphics and Applications*, 8(1):82–86, 1988.

[6] F. C. Crow. Shadow algorithms for computer graphics. In *ACM Siggraph Computer Graphics*, vol. 11, pp. 242–248. ACM, 1977.

[7] C. DeCoro, F. Cole, A. Finkelstein, and S. Rusinkiewicz. Stylized shadows. In *International Symposium on Non-Photorealistic Animation and Rendering (NPAR)*, number 8. ACM, 2007.

[8] N. Hansen and A. Ostermeier. Adapting arbitrary normal mutation distributions in evolution strategies: The covariance matrix adaptation. In *Proceedings of the 1996 IEEE International Conference on Evolutionary Computation*, pp. 312–317. IEEE, 1996.

[9] P. S. Heckbert and M. Herf. Simulating soft shadows with graphics hardware. Technical report, DTIC Document, 1997.

[10] T. Heidmann. Real shadows, real time. *Iris Universe*, 18:28–31, 1991.

[11] T. Igarashi, T. Moscovich, and J. F. Hughes. As-rigid-as-possible shape manipulation. In *ACM Transactions on Graphics*, vol. 24, pp. 1134–1141. ACM, 2005.

[12] J. Lee, J. Chai, P. S. Reitsma, J. K. Hodgins, and N. S. Pollard. Interactive control of avatars animated with human motion data. In *ACM Transactions on Graphics*, vol. 21, pp. 491–500. ACM, 2002.

[13] J. Madill and D. Mould. Target particle control of smoke simulation. In *Proceedings of the 2013 Graphics Interface Conference*, GI '13, pp. 125–132, 2013.

[14] O. Mattausch, T. Igarashi, and M. Wimmer. Freeform shadow boundary editing. In *Computer Graphics Forum*, vol. 32, pp. 175–184. Wiley Online Library, 2013.

[15] N. J. Mitra and M. Pauly. Shadow art. *ACM Transactions on Graphics*, 28(5), 2009.

[16] R. Mukundan and K. Ramakrishnan. *Moment functions in image analysis: theory and applications*, vol. 100. World Scientific.

[17] J. Obert, F. Pellacini, and S. Pattanaik. Visibility editing for all-frequency shadow design. In *Computer Graphics Forum*, vol. 29, pp. 1441–1449. Wiley Online Library, 2010.

[18] F. Pellacini, P. Tole, and D. P. Greenberg. A user interface for interactive cinematic shadow design. *ACM Transactions on Graphics*, 21(3):563–566, 2002.

[19] J. Popović, S. M. Seitz, M. Erdmann, Z. Popović, and A. Witkin. Interactive manipulation of rigid body simulations. In *Computer Graphics (Proceedings of SIGGRAPH 2000)*, pp. 209–218. ACM, 2000.

[20] P. Poulin and A. Fournier. Lights from highlights and shadows. In *Proceedings of the 1992 symposium on Interactive 3D graphics*, pp. 31–38. ACM, 1992.

[21] W. T. Reeves, D. H. Salesin, and R. L. Cook. Rendering antialiased shadows with depth maps. In *ACM Siggraph Computer Graphics*, vol. 21, pp. 283–291. ACM, 1987.

[22] L. Ren, G. Shakhnarovich, J. K. Hodgins, H. Pfister, and P. Viola. Learning silhouette features for control of human motion. *ACM Transactions on Graphics*, 24(4):1303–1331, 2005.

[23] T.-W. Schmidt, J. Novak, J. Meng, A. S. Kaplanyan, T. Reiner, D. Nowrouzezahrai, and C. Dachsbacher. Path-space manipulation of physically-based light transport. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH 2013)*, 32(4), Aug. 2013.

[24] T.-W. Schmidt, F. Pellacini, D. Nowrouzezahrai, W. Jarosz, and C. Dachsbacher. State of the art in artistic editing of appearance, lighting, and material. In *Eurographics 2014 - State of the Art Reports*. Eurographics Association, Strasbourg, France, Apr. 2014.

[25] L. Shi and Y. Yu. Controllable smoke animation with guiding objects. *ACM Trans. Graph.*, 24(1):140–164, Jan. 2005.

[26] L. Shi and Y. Yu. Taming liquids for rapidly changing targets. In *Proceedings of the 2005 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '05, pp. 229–236. ACM, New York, NY, USA, 2005.

[27] N. Thürey, R. Keiser, M. Pauly, and U. Rüde. Detail-preserving fluid control. In *Proceedings of the 2006 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, SCA '06, pp. 7–12, 2006.

[28] A. Treuille, A. McNamara, Z. Popović, and J. Stam. Keyframe control of smoke simulations. *ACM Trans. Graph.*, 22(3):716–723, July 2003.

[29] C. D. Twigg and D. L. James. Backward steps in rigid body simulation. *ACM Transactions on Graphics*, 27(3):25, 2008.

[30] L. Williams. Casting curved shadows on curved surfaces. In *ACM Siggraph Computer Graphics*, vol. 12, pp. 270–274. ACM, 1978.

[31] A. Witkin and M. Kass. Spacetime constraints. *ACM Siggraph Computer Graphics*, 22(4):159–168, 1988.

[32] J. Won and J. Lee. Shadow theatre: discovering human motion from a sequence of silhouettes. *ACM Transactions on Graphics*, 35(4):147, 2016.

[33] A. Woo and P. Poulin. *Shadow algorithms data miner*. CRC Press, 2012.

[34] H. Zhang. Forward shadow mapping. In *Rendering Techniques*, pp. 131–138. Springer, 1998.