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 challenging if 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 still rely 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. Constructing 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 THINKER sculpture'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.

16-19 May, Edmonton, Alberta, Canada

2 RELATED WORK

Our work is built on a foundation of 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 formation 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 pre-rendering 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 mapping method to improve performance, and Reeves et al. [21] improve shadow quality with a filtering and self-shadowing bias.

Crow [6] introduces the shadow volume algorithm capable 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] proposes GPU optimizations for this approach, and Bergeron [3] reduces 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,

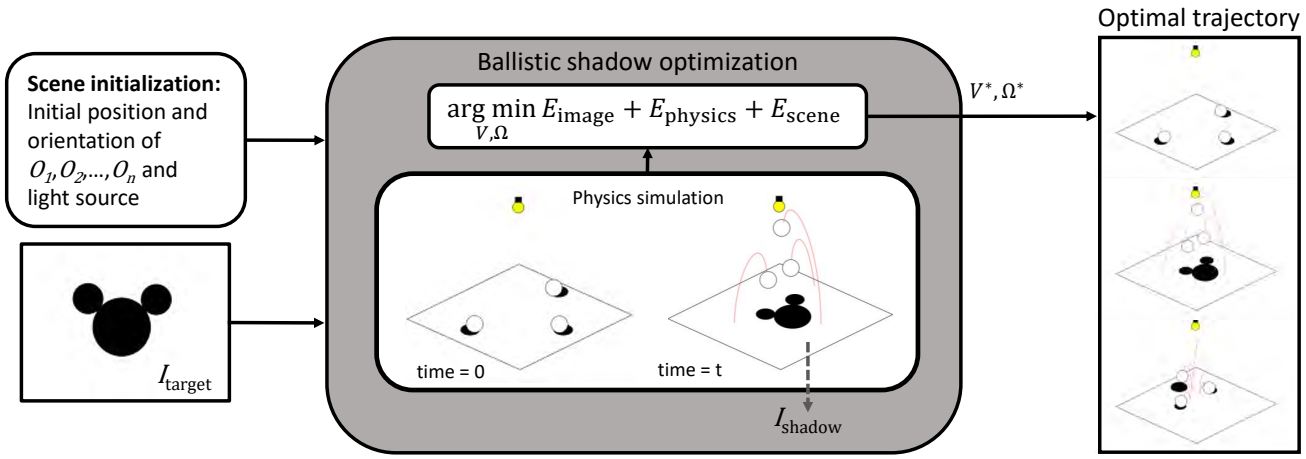

Figure 2: An overview of our method. A point light source (yellow light bulb), planar receiver, and the initial configuration of the light occluders are provided as input. We apply our stochastic optimization technique to find the optimal trajectory that produces the desired shadow shape at time t, minimizing 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}.

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 free-form 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 equations to the synthesis of 3D animation sequences is a fundamental topic in computer graphics research. While a good deal of this work focuses on solving initial value problems, our framework solves a boundary value problem (BVP): for two configurations of scene geometry and two points in time, we search for a trajectory that connects the two together. Witkin and Kass [31] propose a space-time optimization technique that falls into this category. Popović et al. [19] also provide a solution for controlling rigid body simulations via interactive manipulation of object trajectories, and their method allows for contacts in the trajectory, handling 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, as the number of rigid bodies increases and contact becomes more 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, although their model does not fully respect the laws of motion and contact. 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 (in seconds) during which shadows cast by the occluders should match the desired shape. Shadows are projected onto a planar surface due to lighting from a static point light source. 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}. 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.

We apply a forward dynamics simulation with gravity 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 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 we use it to minimize Equation 1.

In the next section, we detail the terms that compose each energy function and how we compute their values.

4 Energy functions

Our 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 an opportunity for users to optionally guide the solution. For instance, the user may want a sharp feature of a specific occluder to correspond with a particular feature in the target shadow 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 i^{th}-order. The simulated shadow and target image are compared with an exclusive-OR operation in E_{XOR}. Finally, E_{in} and E_{out} encourage the inner and outer boundaries of the images to 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]. Because they are effective for carrying low-dimensional information, we include them in our framework.

The energy functions d_0, d_1, and d_2 are the distance between the 0^{th}- through to 2^{nd}-order 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 i^{th}-order moment of an image.

The 0^{th}-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 1^{st}-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 2^{nd}-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. Due to symmetry, the off-diagonal elements are both μ_{11}. To avoiding repeated calculations in the distance, the 2^{nd}-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 by encouraging the shadow to coarsely match the target image. To subsequently improve the alignment, we use energy functions that compare shadow images at the pixel level. 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. The inner boundary, E_{in} is defined as

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

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.

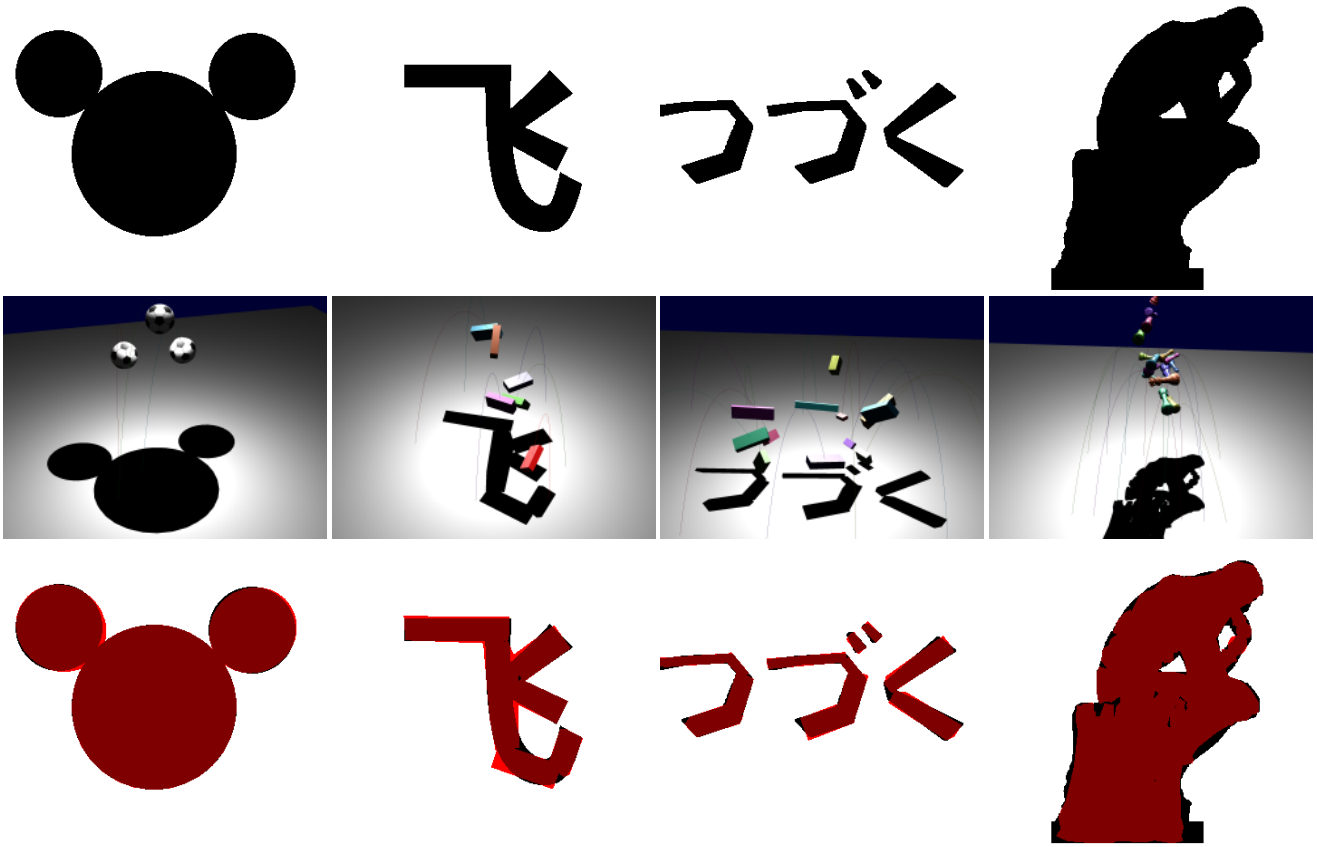

Figure 3: The results obtained with our method for various examples. From left-to-right: Mickey, "fly" Chinese character, "to be continued" in Japanese, and THE THINKER. The target images (top row) are closely matched by the simulated results found by running the optimization algorithm (middle row). A comparison of the target and shadow images (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.

To compute the boundary images, we apply erosion and dilation operations to the target image. For the inner and outer boundary, they are computed as

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

We use a simple kernel for image dilation and erosion. We let K be a 5×5 matrix with all elements set to 1.

4.2 Physics simulation

The physics simulation objective guide our 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 the difference between the actual simulation time and the target time t. In other words, it consists of the time remaining for the simulation to reach the designated shadow casting moment, and therefore it will be zero if contacts do not occur, or if they only happen at the moment of shadow formation. Formally, this penalty is computed as

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

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

We also apply a regularization term, E_{reg}, 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. The light to plane projection $P : \mathbb{R}^3 \mapsto \mathbb{R}^2$ is used to map the 3D coordinate x on the occluder at the time of image formation to its 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.

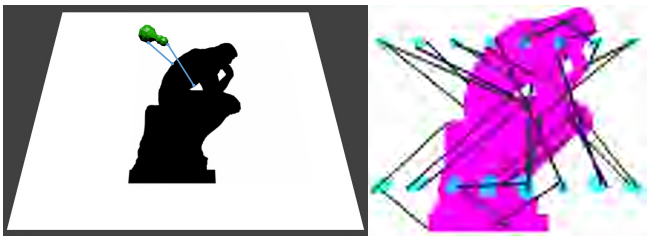

Figure 4: The diagram illustrates how hints are specified as positions on an occluder object, and then mapped to corresponding locations in the target shadow images (left). All hints used for THE THINKER example are shown in image space (right).

We design the energy function E_{bar} to avoid "wasting" occluders. There are cases when one or more occluders do not contribute to the shadow image because they cast shadows completely outside the image region. Previous energy functions may fail to prevent this. For instance, if two occluder shadows move outside the region, it is 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:

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

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 case, we choose half of the image height for the radius.

5 Optimization Strategies

As previously noted, the optimization is highly non-linear, due mainly to discontinuities introduced by contacts and the use of energy functions computed over binary images. The optimization is also affected by the "curse of dimensionality" if n is large. For these reasons, we apply various strategies to guide the optimization and improve its convergence toward viable solutions. This is especially helpful when there are more than just a few occluders.

We use two main strategies to improve convergence: (i) a scheduled optimization that explores the solution space in stages, and (ii) sequentially solving for trajectories one occluder at a time. We describe these strategies in the following sections.

5.1 Scheduled optimization

Early on in the optimization, the solution space is unexplored. It can therefore be difficult for the optimizer to make progress towards a viable solution in an error landscape that is wrought with local minima. We therefore perform a preliminary stage of optimization using only a subset of the energy functions. This tends to produce a smoother landscape dominated by the image moment and barrier functions, which tend to be less noisy and less prone to high frequency variations. This initial stage helps to guide the parameter sampling and narrow down the solution space.

A subsequent stage of optimization is then seeded using the solution of the first stage. High frequency energy functions, specifically the XOR comparison, are enabled during this phase to refine the solution and improve fine scale detail. Figure 6 shows the solutions at the end of the first and second stages for the Mickey example. The energy function weights for this example are also provided in Table 2. Note that the XOR energy is assigned a zero weight during the first stage, which is then enabled during the second stage to refine the shadow shape.

During each stage of the optimization, we visualize the best result of the most recent iteration. The user then stops the current stage and moves onto the next stage once they are satisfied with the results. The accompanying videos demonstrates this visualization. Alternatively, CMA-ES iterations can be stopped once the total energy function is sufficiently small, or stagnation occurs and reduction in its value is below some small tolerance.

5.2 Sequential optimization of occluders

For complex scenes involving many projectiles, avoiding contacts between them becomes difficult if we try to optimize all trajectories simultaneously. In particular, when the shadow occluders are launched from clustered positions, most of the early solutions produce motions that cause the occluders to collide. Even small deviations in the parameters result in dramatically different motions. The optimizer converges poorly in such cases since the error landscape is noisy.

We improve convergence by optimizing for only one occluder at a time. That is, the initial linear and angular velocity of just a single occluder is considered during each step of the optimization. Meanwhile, the velocities and trajectories for all other occluders remains fixed. This compromises the automation of our system, but in practice we found that plausible solutions are found much more quickly. This strategy also facilitates backtracking solutions found for a specific occluder, since they may be revisited. The order of occluders may be automatically generated or defined by the user. This further emphasizes the human-in-the-loop aspect of our system, and provides additional artistic control.

After iterating on all occluders, the user can perform a final stage of optimization that considers all projectiles at once, but with smaller sampling deviations.This refines the previous result, and can potentially help to escape local minima.

Table 1: The complexity of each example. A different set of occluders is used for each example: Mickey uses three spheres; "to fly" uses brick-shaped cuboids; the TBC uses a larger number of bricks; and THE THINKER uses a collection of chess pieces.

	Mickey	"to fly"	TBC	THE THINKER
occluders	3	6	12	16
hints	3	6	12	32
geometry	sphere	box	box	triangle mesh

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

6 RESULTS

Our framework is programmed in Python and uses OpenGL 4.3 for rendering. Projected planar shadows are used for shadow rendering, and we discuss the implications of this in Section 7. The target shape and rendered shadows are represented by 640×480 binary images. Hints are stored separately and manually defined in a text file that is loaded alongside the 3D meshes and target images used in our examples. For ballistic simulation and collision detection, we use the Vortex Dynamics toolkit and time step of $1/60$ s. All experiments were conducted on a PC with Intel Core i7 2.50 GHz processor and NVIDIA GeForce GTX 860M GPU.

The top row of Figure 3 shows the four target shapes that we use as example problems: a Mickey Mouse logo, a simplified Chinese character "to fly" in boldface Gothic typeface, a Japanese phrase for "to be continued" (TBC) in an artistic font, and a silhouette of THE THINKER by Auguste Rodin. The solutions at the time of image formation are presented with snapshots in the middle row of Figure 3, where each snapshot also shows colored curved lines indicating the ballistic motion of the occluders. The point light source is indicated with a white dot at the top of each snapshot. We illustrate the converged solution's shadows in image space in the bottom row. The black shape indicates the target to reach and the semi-transparent red highlight represents the shadow in the scene. The complexity of each scene, in terms of the number of occluders and hints and types of geometry, is given in Table 1.

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 provided by the barrier function. Once the shadows converge to a reasonable match, we introduce the other energy functions to refine the details, continuing the optimization 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 the sphere projectiles in this Mickey example still have angular velocities. They are included for the simplicity of framework consistency, 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 shown in Figure 5 for the shadows and in Figure 6 for the scenes.

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.

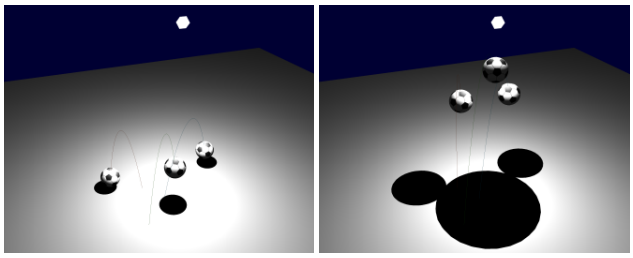

Figure 6: The converged solution scenes at time of image formation of the two stages of the Mickey optimization. The initial stage on the left and the final stage on the right. For each, trajectories are indicated with the curved lines and the light source with the white dot on the top.

In Figure 7 we also present the process of convergence of the two stages. In the first stage, the total value of all energies dropped quickly with the guidance of hints and the first moments. At the start of the first stage there are a few wild samples with large error, involving occluders running outside of the image formation region or into each other, penalized by the barrier function and the contact penalty. After adding the exclusive-or objective function and increasing regularization, we started the second stage from the previous position. In this second phase, the error values involved fewer spikes and the shadows converged to a satisfactory shape, as the exclusive-or decreased to a small magnitude. Toward the end of the second stage, all the other energy functions started to plateau except the regularization, which led to a solution with smaller angular velocity having the same visual effect.

A similar convergence process happens in the other examples, except that energy function weights remain the same among all iterative stages. Note that image moments are not as useful for the iterative stages because the shadows do not need to fit the overall shape of the target, but instead need only match part of the target, and thus image moments will bring unwanted side effects. Furthermore, as the problem dimension is smaller, guiding convergence to a subspace via rough information is marginally useful. Given 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 THINKER example. 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

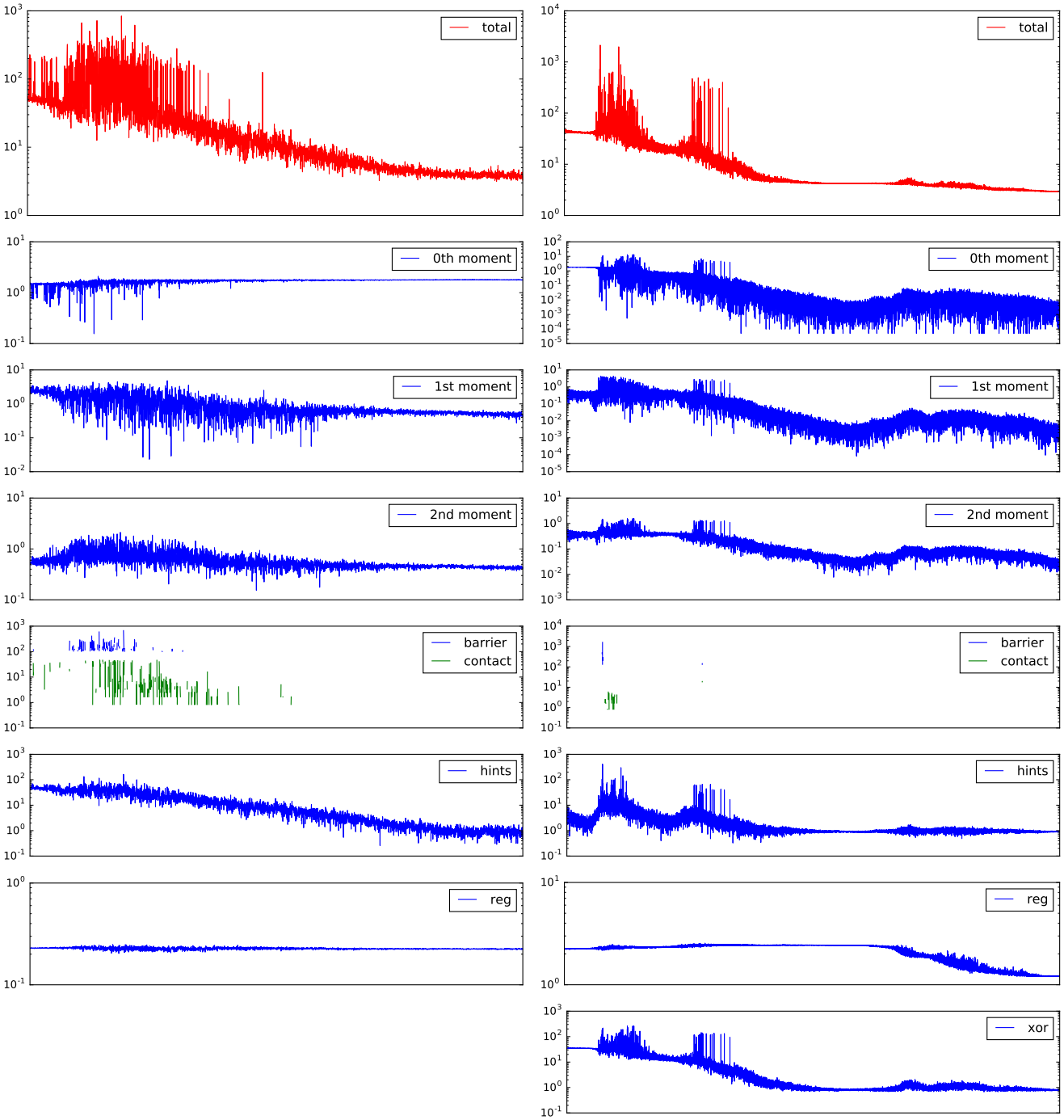

Figure 7: Convergence of the two stages of the Mickey example. The left column presents convergence of the first stage and the right presents the second stage. Each figure plots the evaluated value of every sampling instance along the convergence. The red denotes the total of all applied energies, and 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.

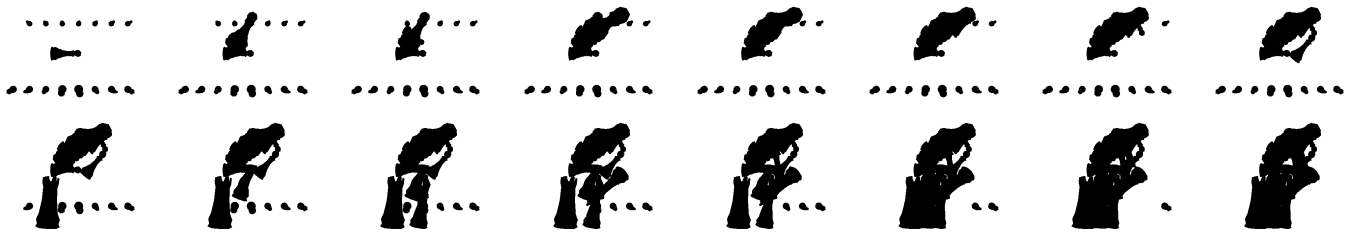

Figure 8: The evolution of THE THINKER shadows. Each image indicates the converged result of each stage. For every stage there is only one chess piece being manipulated with initial conditions for optimization. The rest of the chess pieces are kept untouched, either remaining still at their launching 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.

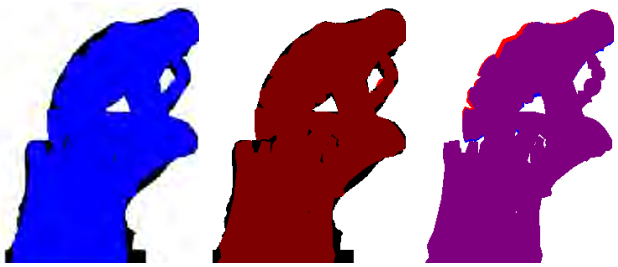

Figure 9: THE THINKER shadow from the extra stage and comparisons with the last stage and target. From left to right: the comparison between the shadow from last stage (blue) and the target (black); the comparison between the shadow of the extra stage (red) and the target (black); the comparison between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

demonstrated in Figure 9. The back of THE THINKER becomes smoother, with a minor trade-off over the stomach. Meanwhile, the rest of THE THINKER is left untouched as the overall shape is satisfactory given the optimized trajectories of the other chess pieces.

6.2 Performance

In Table 3, we present data indicating how long it takes to converge for each example. Notice that for more complicated problems, we need more iterations and samplings to converge. Moreover, using different optimization strategies makes a difference on convergence time, as the latter three examples have a low average sampling and iteration number on each stage.

In Table 4, 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. We also conducted profiling on ballistic simulations and included these timings in the first row of the table. From this we can identify that one obvious bottle-neck on performance is the physics simulation. Synchronization of threads, the optimization algorithm, hard-drive IO, and other explanations of compute time are not taken into account.

7 Discussion and Future Work

The results in Section 6 demonstrate that our framework is capable of synthesizing 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.

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

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

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, this method 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 our technique remains limited to shadows cast onto planar receivers. To exploit the full potential of our framework, we believe it would be interesting to experiment with shadow mapping or shadow volumes.

There are some factors defining the shadow art problem other than target or occluders that we did not enumerate in our experiments. For instance, we assume that our shadow art effects are only supposed to be captured from a 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 of our framework, there should be experiments on problems with

more diverse configurations. Scene construction could be further 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.

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. An interesting extension for future work, could investigate translated, rotated or scaled matching. We could further improve our framework by accepting slightly deformed shapes [11]. Our framework supports optimizing only a single target, while it is interesting to consider synthesizing shadows for multiple targets. One possibility 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 hypothesize 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 small enough deviations are employed. However, because of the dramatic 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 was otherwise impossible. Therefore, this may be useful for multi-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 would be exciting but difficult. To build it we need the physics simulations to be accurate with respect to air drag and collision events. Finally, we would also need to design reliable and accurate methods for physically launching occluders into desired ballistic trajectories.

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. *Computer Graphics Forum*, 2012.

[2] A. Barnat, Z. Li, J. McCann, and N. S. Pollard. Mid-level smoke control for 2d animation. In *Proc. of Graphics Interface 2011*, 2011.

[3] P. Bergeron. A general version of Crow's shadow volumes. *IEEE Computer Graphics and Applications*, 1986.

[4] A. Bermano, I. Baran, M. Alexa, and W. Matusk. Shadowpix: Multiple images from self shadowing. *Computer Graphics Forum*, 2012.

[5] J. Blinn. Me and my (fake) shadow. *IEEE Comput. Graph. and Applications*, 1988.

[6] F. C. Crow. Shadow algorithms for computer graphics. *SIGGRAPH Comput. Graph.*, 1977.

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

[8] N. Hansen and A. Ostermeier. Adapting arbitrary normal mutation distributions in evolution strategies: The covariance matrix adaptation. In *Proc. of the IEEE Intl. Conf. on Evolutionary Computation*, 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*, 1991.

[11] T. Igarashi, T. Moscovich, and J. F. Hughes. As-rigid-as-possible shape manipulation. *ACM Trans. on Graph.*, 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. *ACM Trans. on Graph.*, 2002.

[13] J. Madill and D. Mould. Target particle control of smoke simulation. In *Proc. of Graphics Interface 2013*, 2013.

[14] O. Mattausch, T. Igarashi, and M. Wimmer. Freeform shadow boundary editing. *Computer Graphics Forum*, 2013.

[15] N. J. Mitra and M. Pauly. Shadow art. *ACM Trans. on Graph.*, 2009.

[16] R. Mukundan and K. Ramakrishnan. *Moment functions in image analysis: theory and applications*. World Scientific, 1998.

[17] J. Obert, F. Pellacini, and S. Pattanaik. Visibility editing for all-frequency shadow design. *Computer Graphics Forum*, 2010.

[18] F. Pellacini, P. Tole, and D. P. Greenberg. A user interface for interactive cinematic shadow design. *ACM Trans. on Graph.*, 2002.

[19] J. Popović, S. M. Seitz, M. Erdmann, Z. Popović, and A. Witkin. Interactive manipulation of rigid body simulations. In *Proc. of SIGGRAPH 2000*, 2000.

[20] P. Poulin and A. Fournier. Lights from highlights and shadows. In *Proc. of the Symp. on Interactive 3D Graphics*, 1992.

[21] W. T. Reeves, D. H. Salesin, and R. L. Cook. Rendering antialiased shadows with depth maps. *SIGGRAPH Comput. Graph.*, 1987.

[22] L. Ren, G. Shakhnarovich, J. K. Hodgins, H. Pfister, and P. Viola. Learning silhouette features for control of human motion. *ACM Trans. on Graph.*, 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 Trans. on Graph.*, 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. *Computer Graphics Forum*, 2015.

[25] L. Shi and Y. Yu. Controllable smoke animation with guiding objects. *ACM Trans. Graph.*, 2005.

[26] L. Shi and Y. Yu. Taming liquids for rapidly changing targets. In *Proc. of the ACM SIGGRAPH/Eurographics Symp. on Comput. Anim.*, 2005.

[27] N. Thürey, R. Keiser, M. Pauly, and U. Rüde. Detail-preserving fluid control. In *Proc. of the ACM SIGGRAPH/Eurographics Symp. on Comput. Anim.*, 2006.

[28] A. Treuille, A. McNamara, Z. Popović, and J. Stam. Keyframe control of smoke simulations. *ACM Trans. Graph.*, 2003.

[29] C. D. Twigg and D. L. James. Backward steps in rigid body simulation. *ACM Trans. on Graph.*, 2008.

[30] L. Williams. Casting curved shadows on curved surfaces. *SIGGRAPH Comput. Graph.*, 1978.

[31] A. Witkin and M. Kass. Spacetime constraints. *SIGGRAPH Comput. Graph.*, 1988.

[32] J. Won and J. Lee. Shadow theatre: discovering human motion from a sequence of silhouettes. *ACM Trans. on Graph.*, 2016.

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

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