Ballistic Shadow Art

Xiaozhong Chen

Master of Science

School of Computer Science

McGill University, Montreal

Quebec, Canada

December, 2016

A thesis submitted to McGill University in partial fulfillment of the requirements of the degree of Master of Science

DEDICATION

This thesis is dedicated to my parents.

ACKNOWLEDGEMENTS

First I would like to thank my supervisor, Dr. Paul G. Kry. Paul has been a very resourceful advisor during the whole research process. Without Paul's experience and advice this thesis could not be initiated and continued.

I also would like to thank Dr. Sheldon Andrews for his help in many ways. Sheldon's aid on theory, experiments and writing are crucial for completing all these work.

I also would like to thank Dr. Derek Nowrouzezahrai for his support and suggestions. Derek has participated in this project since the beginning and has also been helping a lot.

I also would like to thank Vincent Petrella, who provided a French version of abstract for this thesis.

Finally I would like to thank my parents and all my friends who have been supportive on my research and pursuit of my degree. Without them this would be a lot harder. Moreover, I would like express my gratitude for the joy brought by F.C. Barcelona, who won the treble in 2015.

ABSTRACT

This thesis presents a framework for designing shadow art with occluders in ballistic motion. A stochastic optimization is used to find the parameters of a multi-body physics simulation that produce the desired shadow at a specific moment in time. Simulations are performed across many different initial conditions, and a set of carefully designed energy functions are used to evaluate the motion trajectories and shadows of bodies. The best parameters are selected, resulting in a simulation on ballistic motion that produces ephemeral shadow art. Users can design physically plausible artwork that would be extremely challenging or nearly impossible by manual effort. A number of compelling examples are presented and analyzed.

RÉSUMÉ

Ce mémoire thèse présente un ensemble d'outils facilitant la création d'ombres artistiques à l'aide d'objets en mouvement balistique. Nous formulons un problème d'optimisation stochastique afin d'identifier les valeurs des paramètres d'une simulation d'objets solides physiques qui projettent l'ombre désirée à certains temps clés. Nous effectuons les simulations à partir d'une variété de conditions initiales, et employons un ensemble de fonctions soigneusement élaborées pour évaluer les trajectoires et les contours des ombres projetées par les objets. Les meilleurs paramètres qui sont ensuite sélectionnés génèrent une simulation balistique produisant des ombres artistiques éphémères. L'utilisateur peut dès lors élaborer des oeuvres physiquement possibles qu'il serait très difficile de façonner à la main. Plusieurs exemples sont ensuite présentés et analysés.

TABLE OF CONTENTS

LIST OF TABLES

LIST OF FIGURES

CHAPTER 1
Introduction

The use of shadows in artwork dates back to pre-renaissance time periods. Some contemporary artworks have even used shadows as the central visual medium. Specifically, shadow art uses sculpture or arrangements of objects to create distinct silhouettes given just the right lighting conditions. However, constructing these artworks can be a complex and time consuming task.

The method presented in this thesis simplifies the task of 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 1–1 gives an example of how ballistic motions produce shadow art of The Thinker sculpture's profile.

The user provides a binary image as input that represents a target shadow shape, along with a set of occluders and their starting configurations. A stochastic optimization technique is then used to determine the initial velocities for the collection of objects such that, at a specific instant in time, their cast shadows match a target silhouette image. The optimization is challenging because of an objective function that involves a forward multi-body dynamics simulation with contact, giving an inherently sensitive and noisy solution space. The dimensionality also grows linearly with the number of objects, becoming large

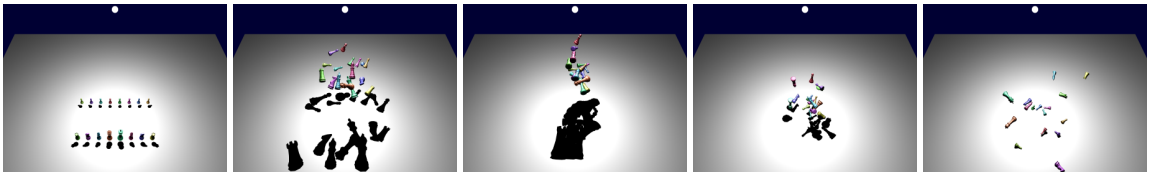


Figure 1–1: A film strip of a ballistic shadow art example. From left to right: starting positions of all chess pieces, launched chess pieces during ballistic motion, the instant that all pieces cast the Thinker-shaped shadow, the pieces continuing on the ballistic motion, and pieces rest on the floor.

for more complex scenes. Our framework therefore makes several accommodations to improve the convergence rate and tractability of the optimization problem.

The remainder of this thesis is organized as follows. In Chapter 2, we discuss about previous work on computer generated shadow art and physics simulation involving boundary value problems. In Chapter 3 on methodology, an overview and formalized version of the optimization problem solved by our framework is presented in Section 3.1, and specific details of the object functions are provided in Section 3.2. Also, the strategies we use to improve the tractability and convergence of the stochastic optimization are discussed in Section 3.3. Compelling examples of ballistic shadow art that have been synthesized with our framework are demonstrated in Chapter 4. In Chapter 5 we provide a discussion of the advantages and limitations of our framework, along with possible future research directions. Finally this thesis concludes in Chapter 6 with a summary of all the work that has been done.

CHAPTER 2
Literature review

Our work is built on the foundation of numerous previous ones. Specifically, it involves three aspects. First of all, existing shadow rendering techniques allow us to have shadows and occlusions as needed. Previous studies on controlling shadows provide inspirations, and so do the studies on trajectory solving.

2.1 Shadow rendering

Rendering shadows has been an important topic in computer graphics since the beginning. Shadows indicate spatial relationship and lighting conditions, and thus they are crucial for realistic rendering effects. For our framework, we need a real-time rendering technique that produces shadows with accurate geometry. Generally there are three options for real-time shadow rendering: projected shadow, shadow mapping, and shadow volume. Blinn [4] proposes an algorithm to create shadows by projecting polygons to a planar surface. This method is straightforward but with artifacts of false shadows when occluders are not completely above the receiver, or 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 ensure visibility.

Williams [23] presents the shadow mapping method. By prerendering the geometry from the light source viewpoint to a depth buffer, it can thus compare the distance information to determine occlusion in the scene, regardless of the caster's and receiver's shape. Zhang [27] introduces a forward shadow mapping method to improve performance, and

Reeves et al. [20] improve shadow quality on boundaries with filtering and self-shadowing with a bias parameter.

Crow [5] introduces the shadow volume algorithm. This algorithm divides the scene into shadowed and unshadowed regions, by constructing cones with one shadow polygon per caster edge from each occluder, and the shadow cones face against the light source. Heidmann [9] improves the performance with a GPU implementation, and Bergeron [2] reduces the volume complexity by eliminating extra shadow polygons.

There are also numerous other work and variations on improving rendering shadows. Woo and Poulin [26] provide a comprehensive survey on this topic. Considering performance, shadow quality, and implementation complexity, we decide to use planar shadow.

2.2 Shadow editing

There are a few researches that have been done on editing shadows in recent years, and they mainly fall into two categories. One category is editing the rendered shadow directly for visual effect, optionally adjusting occluders correspondingly or sacrificing correctness. The other one is to figure out a configuration of shadow occluders that cast shadows matching given targets, and potentially fabricate such configuration.

For direct editing, Poulin and Fournier [19] first investigate on allowing user interaction with shadows and highlights in a rendered image, instead of iterating on adjusting light and rerendering the expensive scene. Pellacini et al. [17] present a user interface that permits an artist to interactively design shadows in animated feature films. DeCoro et al. [6] propose an algorithm rendering stylized shadows. They provide controls on tuning artistic properties for shadow rendering, such as abstraction and softness. Obert et al. [16] describe a method for editing visibility for the design of all-frequency shadows.

Mattausch et al. [13] present a method for editing the boundaries of shadows inspired by freeform deformations. These works inspired us on manipulating shadow geometries, however these strategies can hardly apply to our problem. Considering the complexity, indirectness and precision demand, straightforward editing can not reduce the amount of user's work.

In terms of arranging or generating occluders, Mitra et al. [14] present tools to design a voxel-based occluder that casts different shadows when illuminated from a few different directions. Bermano et al. [3] provide a method of producing a 3-D 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, by changing the light configuration. Won et al. [25] 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 the movement of our dynamic shadow casters, except that we prevent collision with a hard constraint rather than a penalty function.

2.3 Trajectory solving

Applying the dynamical equations of motion to 3D animation problems has been well explored in computer graphics. While most of the focus has been on solving initial value problems, our framework solves a boundary value problem (BVP): for two configurations and two points in time, compute a trajectory that connects them. The prominent work by Witkin and Kass [24] on space-time optimization falls into this category. Popović et al. [18] also describe a method that provides solutions for controlling rigid-body simulations via interactive manipulation of object trajectories. Twigg et al. [22] present a

method that uses backward time stepping to produce animations involving rigid bodies and frictional contact. When solutions are not unique, their method proposes a plausible one.

Our work differs in that the end simulation state is defined implicitly by the target shadow shape. This expands the number of feasible solutions, but introduces non-linearity that can make finding good solutions difficult. In terms of the methods of Popović et al. and Twigg et al., they both require precise knowledge of end state such as positions, orientations, and velocities. These information are pending to solve, and thus lack of them makes these methods inapplicable. Furthermore, Popović et al.'s method allows contacts to occur along the trajectory, and handles each motion between contacts separately. It performs a local discrete search on parameter space to resolve the discontinuity. When the number of rigid-bodies increases and contact becomes frequent, computation will increase dramatically and convergence will become extremely difficult. Twigg et al.'s method allows minor violation of physics, which does not fit in our demand.

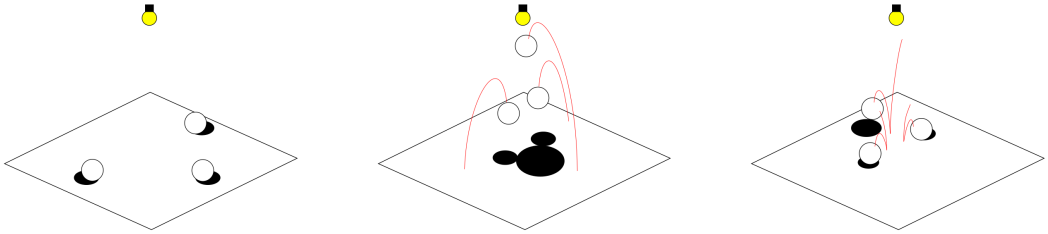
CHAPTER 3
Methodology

In Chapter 2 we reviewed the related work that inspired us in building this framework. However the existing work does not provide exact solutions to the problem we try to solve and therefore we will present our solution in this chapter. First we define the problem formally as a mathematic optimization problem. Then we design energy functions that describe desired solutions. Our energy functions fall into three categories: image comparison, physics simulation and scene settings. We also propose two strategies for organizing optimization. By applying these strategies we expect efficiency and effectiveness that a naive strategy does not have.

3.1 Overview

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. Figure 3–1 indicates the sketch of how input forms up the ballistic shadow art.

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



(a) (b) (c)

Figure 3–1: A sketch of ballistic shadow art. To build such a form of art work it takes a point light source (indicated with a yellow light bulb), a planar receiver and a few projectiles as shadow occluders. In (a) three spherical occluders are at their launching positions awaiting projecting. In (b) occluders are launched and during their ballistic motion, an artistic shadow of Mickey Mouse logo is cast at a predefined time instant. The red lines represent their trajectories. In (c) the occluders continue their motions and bounce on the receiver.

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

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

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

$$\underset{V,\,\Omega}{\operatorname{argmin}}\; E_{\text{image}} + E_{\text{physics}} + E_{\text{scene}}, \tag{3.1}$$

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 [7] method is a stochastic optimization technique that is well suited to these conditions, and so it is used to minimize the problem in Equation 3.1.

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

3.2 Energy functions

We categorize our energy functions into three, based on their input and semantics. They are image comparison, physics simulation and scene settings. The image comparison functions focus on matching captured shadows with the target image, and should reach 0 if a perfect match is encountered. The physics simulation functions are designed to avoid physically correct but unreasonable solutions. Lastly 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, users may want a sharp tip on one specific occluder to fit in a particular corner in the target image.

3.2.1 Image comparison

Energy functions on image comparison take a captured shadow image and a target shape image as input, and both of them I_{shadow} and I_{target} are in binary format as

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

Energy functions on image space are defined as

$$E_{\text{image}} = w_0 d_0 + w_1 d_1 + w_2 d_2 + w_3 E_{\text{XOR}} + w_4 E_{\text{inner}} + w_5 E_{\text{outer}}$$

where w_i denotes the weight of each separated energy, and d_i is the distance between image moments of ith order. Captured shadows and the target image are also compared with E_{XOR} as the exclusive-or operation, as well as E_{inner} and E_{outer} that emphasize on inner and outer boundary matching.

Image moments distance

Image moment [15] is a representative and succinct description of an image. It is widely used in computer vision and computer graphics. For instance, Lee et al. [12] used Hu moments [10] to describe an actor's motion from a camera image for real-time character controls. Ren et al. [21] extend this work by bringing two more cameras, so that recognition is capable for handling complex motions and subtle changes. Image moments are proven effective for carrying low-dimensional information and we include them into our framework.

Our energy functions that involve image moments are the distance between normalized moments of the shadow image and the target shape. The distance is formally represented by

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

with M_i as moment of ith order. We use moments from the 0th order to the 2nd order. Formally they are defined as follows.

The 0th order moment

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

essentially is the total area of the input image expressed in pixels. Thus the distance in between would be the difference of shadow size.

As for the 1st order moment, we use

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

This vector represents the center of mass in image space, and the distance is the Euclidean norm.

Originally the 2nd order image moment should be defined as a 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. However, note that the off-diagonal elements are both μ_{11}. For avoiding repetition in calculating distance, we define the 2nd order image moment as

$$M_2(I) = \Big(\mu_{20},\ \mu_{11},\ \mu_{02}\Big)^{\mathsf{T}},$$

and the distance is still an Euclidean norm, as the 1st order one is.

Image difference

We design energy functions that compare shadows with the target on a pixel level. For the total comparison we use E_{XOR}, which denotes the binary exclusive or operation performed between the captured 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 details in the final result, we also designed energy functions on matching boundaries of the target shape. There are two types of boundary that we use: one is inner boundary and the other one is the outer boundary. The matching is designed differently based on the type. For the inner boundary, E_{inner} is defined as

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

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

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

with I_{outer} as the outer boundary image. Similarly to the previous inner case, the matching is calculated with an "and" operation, but negated in a boolean way. With this design, the shadow is discouraged from intersecting the outline.

To compute these two boundaries we apply erosion and dilation operations on the target image. For the inner boundary it is computed as

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

and for the outer one 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 consisting with all 1s.

3.2.2 Physics simulation

We also design energy functions that take input from the physics simulation parameters, to guide our optimization to convergence and avoid unwanted solutions. There are two parts in our physics-related energy functions

$$E_{\text{physics}} = w_6 E_{\text{contact}} + w_7 E_{\text{reg}}$$

where E_{contact} is a penalty on caster contacts before shaping up artistic shadows at the given moment, and E_{reg} serves as a regularization term on the initial linear and angular velocity of shadow casters, to avoid unreasonable solutions.

Contacts are very difficult to predict and thus add a lot of noise to the solution space. Therefore our framework must avoid contacts before casting the target shadows. The simulation terminates at the exact moment when contacts occur, to ensure target shadow is contact free. In order to provide a smoother guiding, the contact penalty is designed to be a countdown multiply a factor. This countdown is the time left for simulation to reach the given shadow casting moment when contacts occur, so that the countdown is optimally 0

if contacts are not detected all along, the given duration if contacts occur at the beginning, and gradually decrease if contacts are 'postponed'. The factor is defined as the number of contact detected at the terminated moment. Formally, this penalty is computed as

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

where t_{total} is the given duration, that is, the exact instant when we want our occluders to cast the target shadows, and t_{contact} is the actual duration that simulation has run. The factor n_{contact} is the factor decribed above.

For regularization we use the sum of all magnitudes of projectiles' linear and angular velocities, that is

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

We also provide a term α to balance between linear and angular velocities in case their magnitudes are substantially different. We use 0.5 for our examples.

3.2.3 Scene settings

There are considerations on having a reasonable scene setting as well as taking advantage from human perceptual intuition to guide our optimization. With these concerns we designed our energy functions on scene setting as

$$E_{\text{scene}} = w_8 E_{\text{hint}} + w_9 E_{\text{barrier}}$$

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

Specifically, energy for hints are defined as

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

where H denotes a collections of k point hints given by the user, formally $H = \{h_j \mid j \in [1, k]\}$, and h denotes a single point hint that refers one point p in image space, and one point x in model space of a specific occluder O_i, as well as a weight w to prioritize between hints:

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

The projection $P : \mathbb{R}^3 \mapsto \mathbb{R}^2$ maps a three dimensional coordinate in occluder's model space to the corresponding position on captured shadow image. Note that the projected point can possibly be outside of the region for capturing.

Another energy function E_{barrier} is designed to avoid the waste of occluders. There are cases when one or more occluders do not participate in forming up a desirde shape because cast shadows are outside of the region for capturing, and the previous energies may fail in resolving that. For instance, if two occluder shadows run outside, they may cancel the error of each other in terms of first order moments if they happen to balance across the region. The exclusive-or may have a flat change if the shadow has not yet intersected the target shape, and thus would fail to tell any difference among the samples if the shadow is always cast outside.

The barrier function is specifically defined as

$$E_{\text{barrier}} = \sum_{i} B(P(C_i)),$$

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

3.3 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.

3.3.1 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.

3.3.2 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.

3.4 Summary

In this chapter, we define the problem that our framework needs to solve into a mathematic optimization problem. For this problem we also provide energy functions designed carefully and strategies to help converging. Our energy functions have different

focuses such as image comparisons and contact avoiding. Our strategies apply to different circumstances and demands. In the next chapter we will present four examples produced with the framework, and the details in building them.

CHAPTER 4
Results

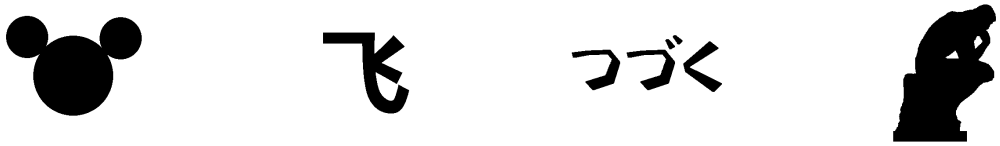
In Chapter 3 we elaborate on all the designs that form up this framework. In this chapter we will present a few compelling examples of ballistic shadow art, and we will also demonstrate details of different aspects in building these examples.

We present four target shapes in Figure 4–1 as example problems. They are a Mickey Mouse logo in Figure 4–1a, a simplified Chinese character of "to fly" in boldface Gothic typeface in Figure 4–1b, a Japanese phrase of "to be continued" (later shortened as "TBC") in an artistic font in Figure 4–1c, and a silhouette of The Thinker by Auguste Rodin in Figure 4–1d.

Table 4–1: Complexity of each example

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

The complexity of each example's occluders is specified inside of Table 4–1. Each example is provided different set of occluders as 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 Thinker example, half a set of chess pieces are provided. These chess pieces are also simulated with their reduced triangle mesh for collision handling.



(a) (b) (c) (d)

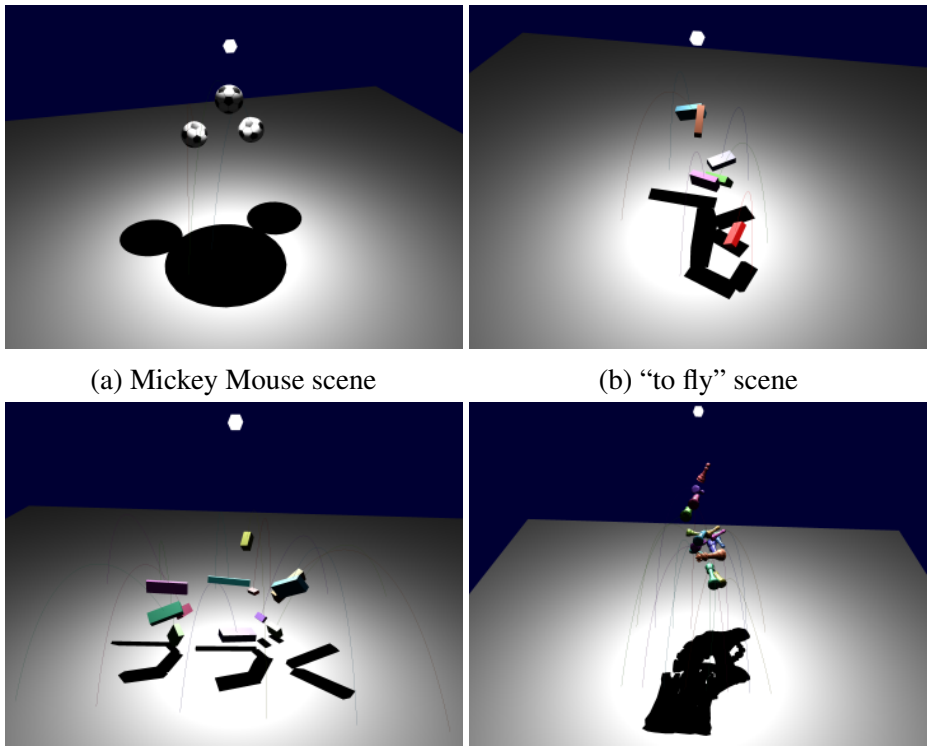
Figure 4–1: The target shadow shapes of the examples. We have (a) the Mickey Mouse logo, (b) Simplified Chinese character of “to fly”, (c) Japanese phrase “to be continued” in an artistic font, and (d) The Thinker silhouette.

The converged scenes are presented with snapshots in Figure 4–2, in each of which there are colored curved lines with one end on occluders indicating the trajectories of their ballistic motions. 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 Figure 4–3, where the black shape indicates the target to reach and the semi-transparent red shape represents the shadow in the scene.

4.1 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 4–2. 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



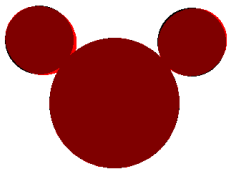
(a) Mickey Mouse scene (b) “to fly” scene

(c) “to be continued” scene (d) The Thinker scene

Figure 4–2: The converged ballistic shadows. In each snapshot the curve lines indicate occluders trajectories, and the white dot at the top indicates the point light source.

Table 4–2: 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



(a) Mickey Mouse shadow

(b) “to fly” shadow

(c) “to be continued” shadow

(d) The Thinker shadow

Figure 4–3: The converged shadows in image space compared with the corresponding targets. In each image the semi-transparent red shape represents the shadow in image space as the converged result, and the black shape in the background is the target shadow shape to match.

Figure 4–4: The results of the two stages of the Mickey example. The initial stage on the left and the final stage on the right.

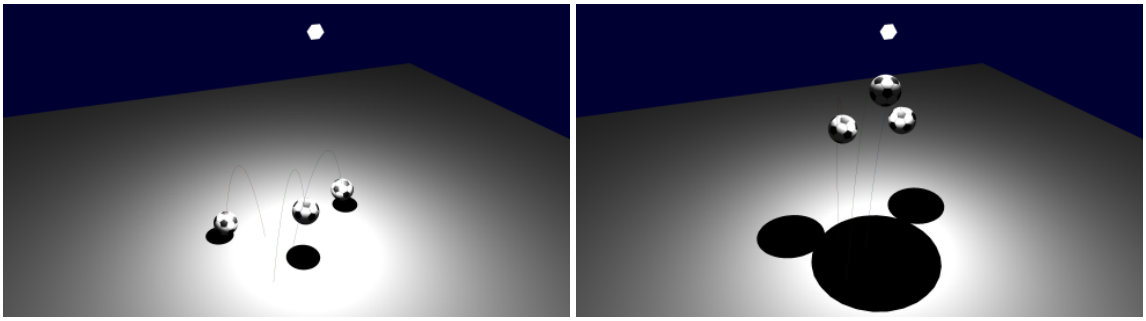


Figure 4–5: The converged scenes of the two stages of Mickey. The left one is from the first stage and the right one is from the second stage. For each one, trajectories are indicated with the curved lines and the light source with the white dot on the top.

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 4–4 on the shadows and Figure 4–5 on the scenes.

In Figure 4–6 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.

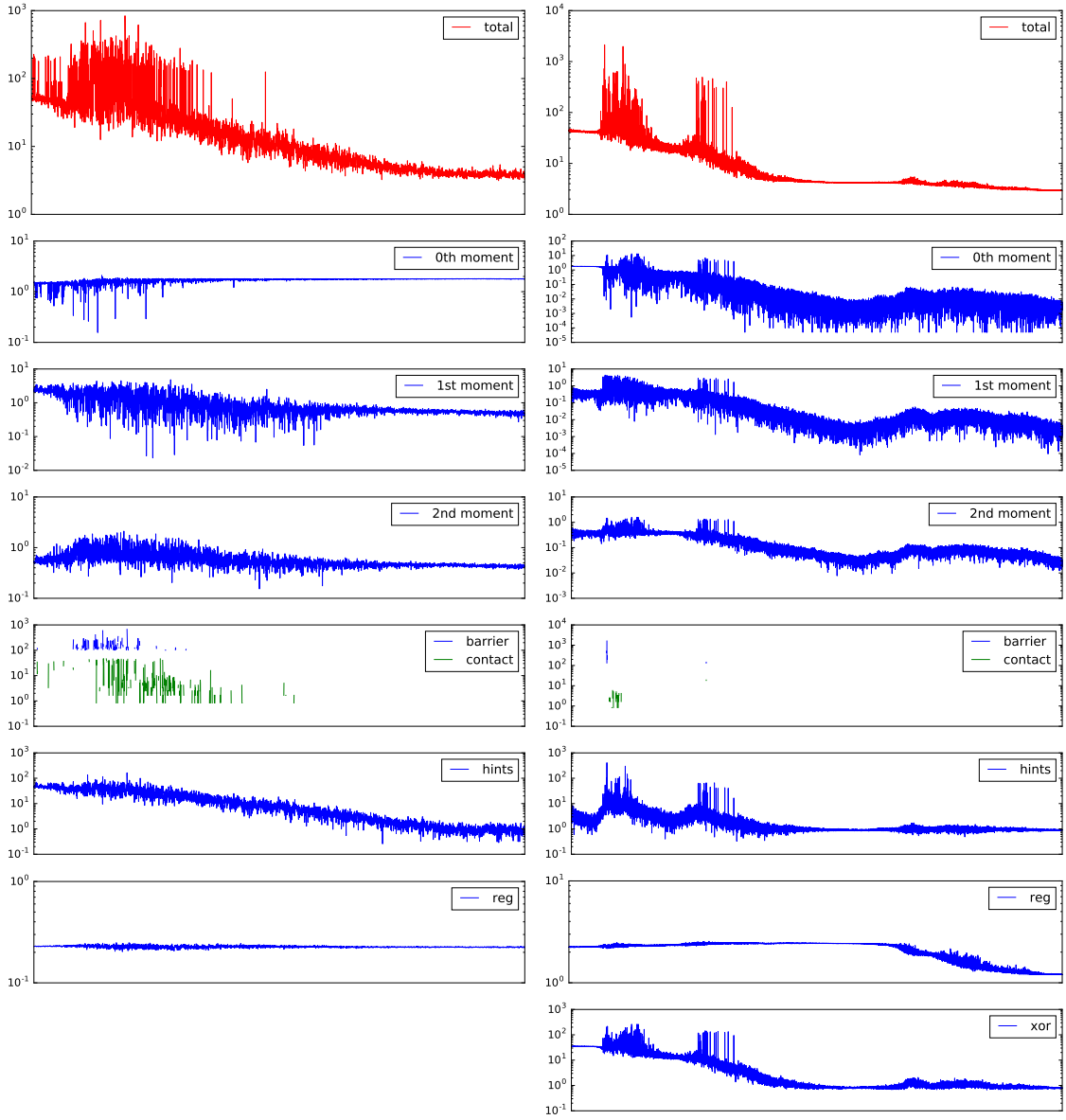


Figure 4–6: Convergence of the two stages of the Mickey example. The left column presents convergence of the first stage and the right presents the second stage. Each figure plots the evaluated value of every sampling instance along the convergence. The red denotes the total of all applied energies, 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.

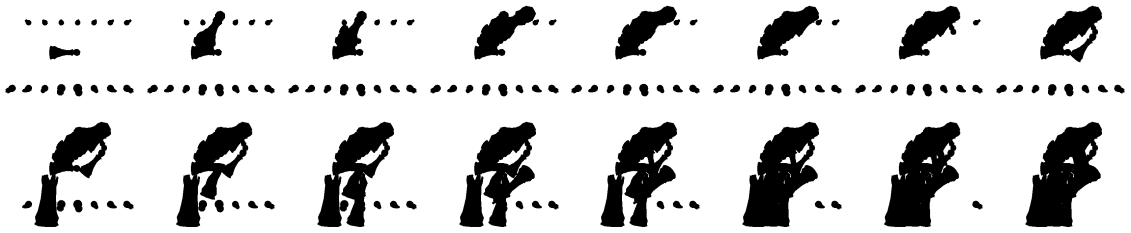


Figure 4–7: The evolution of the Thinker shadows. Each image indicates the converged result of each stage. For every stage there is only one chess piece being manipulated with initial conditions for optimization. The rest of the chess pieces are kept untouched, either remaining still at their launching 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.

The similar convergence process happens in other examples, except that energy function weights remain the same among all iterative stages. We also present the evolution process of The Thinker example. The first 16 stages are optimized for one chess piece in Figure 4–7. In the last stage we increased the weight on boundaries, fixed most of the occluders and optimized only on the pieces that form up the status' back to have a smoother outline. The improvement is demonstrated in Figure 4–8. The back of The Thinker is improved with a smoother outline, with a minor trade-off over the stomach. Meanwhile the rest of The Thinker is kept untouched as the overall shape is satisfactory, with the constraints of the chess piece shapes.

In Figure 4–9 three representative stages of the convergence of The Thinker example are selected and illustrated with plot figures. We select the process of stages which optimizes on the left rook, and on the right rook, as well as the extra stage that adjusts a few chosen pawns that form up the back. In these three convergences, the process always starts with wild sampling penalized by barrier functions or contacts. The hints and exclusive-or are the main factors driving convergence, even though exclusive-or values do

Figure 4–8: 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.

not decrease visually as dramatic as hints. This is because exclusive-or is calculated on the whole image, in which target shadows only take relatively small part, and thus the local changes made by one chess piece appear less substantial than the hints in the figure. The outer boundary function serves as prevention of outline intersection as expected, and the inner boundary function, which replaces the hints in the last stage, decreases gradually, and dominates the convergence.

4.2 Performance

We also provide some performance data in Table 4–3 and Table 4–4 for reference. Table 4–3 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.

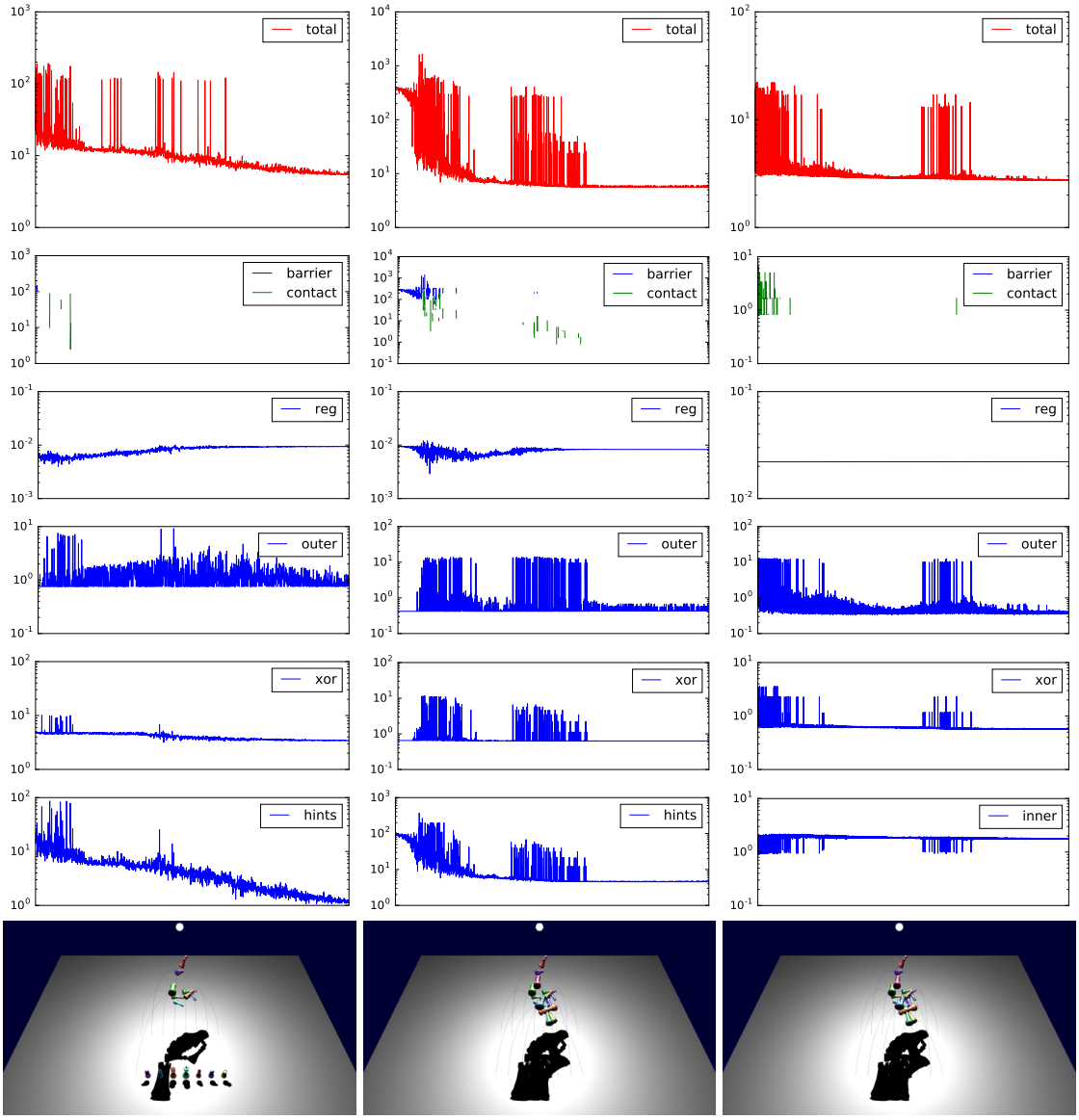


Figure 4–9: Convergence of three selected stages of The Thinker example. The figures are plotted in the same way as Figure 4–6 does and in the bottom row, there are three snapshots of the final converged scenes. Note that right above the snapshots in the bottom, the third stage replaces the hints with the inner boundary function.

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.

Table 4–3: 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

4.3 Implementation

We used projected planar shadows for shadow rendering in our framework, and the discussion of it will be covered in Chapter 5. 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

Table 4–4: 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

performance tests in Table 4–3 and Table 4–4 are run on a PC with processor Intel Core i7-4710HQ @ 2.50GHz and GPU of NVIDIA GeForce GTX 860M.

4.4 Summary

In this chapter we presented our results by demonstrating four successfully-built examples with our framework. Each example has different targets and occluders, and they have different complexities as well. We also demonstrated the convergence process via the weights table and the convergence plots, to illustrate the changes that happened in different stages with different weights. The iterative strategy is also presented with the evolution of The Thinker shadows, and the convergence of three representative stages. Then we analyzed our optimization in the performance aspect, in both convergence rates and profiling results. Finally we presented the technical details in implementing our framework. In the next chapter, we will discuss advantages and disadvantages of a few decisions we made in the framework, and future plans to improve or extend on them.

CHAPTER 5
Discussion

The results in Chapter 4 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 helping to guide the optimization.

We render shadows by planar projection in our framework, rather than shadow mapping or shadow volumes. Projected shadow has a perfect resolution for capturing and image comparison purpose, and it is easy to implement as well as efficient to render. However as a trade-off for fast prototyping, it produces fake shadows when an occluder is located behind the plane, or anti-shadows when it stands behind the light, both of which can be fixed [8] [26]. The major shortcoming is the exclusion to non-planar shadow receivers. To exploit our framework's potentials on different receivers, 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 shadow image with the target shape, perfect matching is the only optimal matching. Therefore the framework has the capability to help the users acquire shadows at an exact desired location. However in the case when location or orientation for the shape is not restricted, our framework does not support less stiff matching. For future work, the framework should provide options to accept translated, rotated or scaled matchings, or to add such energies in the optimization. Furthermore, accepting slightly deformed shapes [11] is a powerful feature to add on.

Our framework only supports optimizing on a single target and as part of future work we plan to investigate synthesizing shadows for multiple targets. One way is using multiple light sources for one static placement of occluders [3][14], and applying this framework to find a set of trajectories. However it normally requires very complex arrangements of occluders and thus the convergence and simulation will be extremely challenging. Another route will be taking advantage from ballistic motions. We can match different targets on different instants in the ballistic process. At one instant all projectiles together cast a shadow of one target shape, and as the motion continues they form up another target shadow at a later instant. Timing becomes critical in this case, as a poor selection of instants may create a problem without feasible solution, given the physics constraints. Determining the exact instants will be difficult and perhaps it should be another variable

in the optimization. Moreover, adding the light source as a projectile may potentially provide more solutions.

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.

Lastly, 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.

CHAPTER 6
Conclusion

This thesis presents our framework of creating ballistic shadow art. Besides the existing shadow art forms, we bring a new form with ballistic motions involved - at an exact given instant, all occluders launched to their trajectories together cast the artistic shadow that is pre-defined by a target image. The initial conditions of occluders are crucial in building the artwork.

To build this form of art in a less time-consuming way, we define the ballistic shadow art formally into a mathematic optimization problem, and a stochastic optimization method is applied to find the answer. For the optimization, we carefully design a set of energy functions that focus on various aspects, including comparing captured shadows with a target shape in image space, avoiding contacts, regularizing on input and penalizing unwanted solutions.

Then we also provide two different strategies to help optimization converge with multiple stages, since optimizing on all parameters and energy functions in one shot is inefficient and sometimes even not effective. These two strategies are scheduled optimization and iterative optimization. The scheduled strategy optimizes on initial conditions of all occluders with adjusting weights of energies in different stages. The iterative optimization generally applies the same energy configuration across stages but in every stage, only one individual occluder's initial condition is optimized.

With these energy functions and optimization strategies, we successfully created four compelling examples. They are given different input of both occluders and targets. For the first example of assembling the Mickey Mouse we use the scheduled optimization with two stages, and for the other three, the “to fly” character, the “to be continued” phrase and The Thinker, we use the iterative optimization. we also compare among the examples on convergence and performance aspects. By demonstrating weights and convergence process of each energy function from different stages, and the results of each iterated stage, we present how the solutions converged to aesthetically pleasing shadow art that matches the target shapes. By comparing the convergence rate and the profiling data of energy function evaluation, we illustrate the performance of our framework. The results presented in this thesis demonstrate that our framework is capable of creating ballistic shadow art, even for complex examples.

References

[1] Ilya Baran, Philipp Keller, Derek Bradley, Stelian Coros, Wojciech Jarosz, Derek Nowrouzezahrai, and Markus Gross. Manufacturing layered attenuators for multiple prescribed shadow images. In *Computer Graphics Forum*, volume 31, pages 603–610. Wiley Online Library, 2012.

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

[3] Amit Bermano, Ilya Baran, Marc Alexa, and Wojciech Matusk. Shadowpix: Multiple images from self shadowing. In *Computer Graphics Forum*, volume 31, pages 593–602. Wiley Online Library, 2012.

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

[5] Franklin C Crow. Shadow algorithms for computer graphics. In *ACM Siggraph Computer Graphics*, volume 11, pages 242–248. ACM, 1977.

[6] Christopher DeCoro, Forrester Cole, Adam Finkelstein, and Szymon Rusinkiewicz. Stylized shadows. In *International Symposium on Non-Photorealistic Animation and Rendering (NPAR)*, number 8. ACM, 2007.

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

[8] Paul S Heckbert and Michael Herf. Simulating soft shadows with graphics hardware. Technical report, DTIC Document, 1997.

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

[10] Ming-Kuei Hu. Visual pattern recognition by moment invariants. *IRE Transactions on Information Theory*, 8(2):179–187, 1962.

[11] Takeo Igarashi, Tomer Moscovich, and John F Hughes. As-rigid-as-possible shape manipulation. In *ACM Transactions on Graphics*, volume 24, pages 1134–1141. ACM, 2005.

[12] Jehee Lee, Jinxiang Chai, Paul SA Reitsma, Jessica K Hodgins, and Nancy S Pollard. Interactive control of avatars animated with human motion data. In *ACM Transactions on Graphics*, volume 21, pages 491–500. ACM, 2002.

[13] Oliver Mattausch, Takeo Igarashi, and Michael Wimmer. Freeform shadow boundary editing. In *Computer Graphics Forum*, volume 32, pages 175–184. Wiley Online Library, 2013.

[14] Niloy J Mitra and Mark Pauly. Shadow art. *ACM Transactions on Graphics*, 28(5), 2009.

[15] Ramakrishnan Mukundan and KR Ramakrishnan. *Moment functions in image analysis: theory and applications*, volume 100. World Scientific.

[16] Juraj Obert, Fabio Pellacini, and Sumanta Pattanaik. Visibility editing for all-frequency shadow design. In *Computer Graphics Forum*, volume 29, pages 1441–1449. Wiley Online Library, 2010.

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

[18] Jovan Popović, Steven M Seitz, Michael Erdmann, Zoran Popović, and Andrew Witkin. Interactive manipulation of rigid body simulations. In *Computer Graphics (Proceedings of SIGGRAPH 2000)*, pages 209–218. ACM, 2000.

[19] Pierre Poulin and Alain Fournier. Lights from highlights and shadows. In *Proceedings of the 1992 symposium on Interactive 3D graphics*, pages 31–38. ACM, 1992.

[20] William T Reeves, David H Salesin, and Robert L Cook. Rendering antialiased shadows with depth maps. In *ACM Siggraph Computer Graphics*, volume 21, pages 283–291. ACM, 1987.

[21] Liu Ren, Gregory Shakhnarovich, Jessica K Hodgins, Hanspeter Pfister, and Paul Viola. Learning silhouette features for control of human motion. *ACM Transactions on Graphics*, 24(4):1303–1331, 2005.

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

[23] Lance Williams. Casting curved shadows on curved surfaces. In *ACM Siggraph Computer Graphics*, volume 12, pages 270–274. ACM, 1978.

[24] Andrew Witkin and Michael Kass. Spacetime constraints. *ACM Siggraph Computer Graphics*, 22(4):159–168, 1988.

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

[26] Andrew Woo and Pierre Poulin. *Shadow algorithms data miner*. CRC Press, 2012.

[27] Hansong Zhang. Forward shadow mapping. In *Rendering Techniques*, pages 131–138. Springer, 1998.