SMASH: Data-driven Reconstruction of Physically Valid Collisions

Aron Monszpart
University College London

Nils Thuerey
TU Munich

Niloy J. Mitra
University College London

Abstract

Collision sequences are commonly used in games and entertainment to add drama and excitement. Authoring even two body collisions in real world can be difficult as one has to get timing and the object trajectories to be correctly synchronized. After trial-and-error iterations, when objects can actually be made to collide, then they are difficult to acquire in 3D. In contrast, synthetically generating plausible collisions is difficult as it requires adjusting different collision parameters (e.g., object mass ratio, coefficient of restitution, etc.) and appropriate initial parameters. We present SMASH to directly 'read off' appropriate collision parameters simply based on input video recordings. Specifically, we describe how to use laws of rigid body collision to regularize the problem of lifting 2D annotated poses to 3D reconstruction of collision sequences. The reconstructed sequences can then be modified and combined to easily author novel and plausible collision sequences. We demonstrate the system on various complex collision sequences.

1 Introduction

Collisions capture suspense, build anticipation, and pack drama. Naturally, they remain an integral part of movies, games, and entertainment. Creating a good real-world collision sequence involving multiple objects, however, is difficult. While the act of smashing objects into each other so that they collide in a certain way is already non-trivial, the setup quickly becomes unmanageable when additional colliding objects are to be collided, or adjustments are required to object trajectories. Such changes can easily require many further iterations and recordings, and become a tedious trail-and-error process. Moreover, trying out multiple collision iterations with expensive or fragile objects may not even be a realistic option.

Accurately capturing real-world collision sequences poses further challenges. On one hand, such sequences necessitate high to very high framerate capture, thus making state-of-the-art methods like Kinect Fusion, etc. unsuitable candidates for 3D acquisition. On the other hand, high-framerate video data only provides partial 2D information, both in space and in time (see Figure 2). A more fundamental problem arises due to unavoidable occlusions near collision instances, which hinder direct observation of the actual collision processes in any acquisition setup.

While the physics of object collisions is a challenging problem in itself, well-developed high-level models exist to reduce its complexity. One widely used assumption is that of infinite object stiffness, i.e., ideally rigid motions. Whereas such rigid body simulations are widely used in games and movies, the task of setting up a collision with the right initial conditions remains tough: many parameters, such as velocities, mass ratios, and coefficient of resitiution, have to be correctly guessed and adjusted. Given the nonlinear nature of the underlying physics, such a rigging up a desirable sequence is problematic and typically requires extensive prior experience. Further, physical parameters (e.g., coefficient of restitution) may not be readily available for the objects at hand.

In this work, we propose to marry the benefits of the above setups. The user records collisions between pairs of objects simply using a smartphone. The video data, however, lacks depth information and is noisy. We expect the user to roughly annotate object poses in a few keyframes away from the collision time. Then, we use laws of physics to regularize the reconstruction problem to produce space-time recording of the collision sequence. This step utilizes the original 3D models of the colliding objects, assumed to be captured in a pre-collision stage. As output, we can directly *read off* physical collision parameters from the reconstruction that can readily be incorporated into existing physics engines to recreate and reauthor modifications of the recorded collisions. For example, in Figure 1, we show the replay of the reconstructed collision sequence happening behind the curtain.

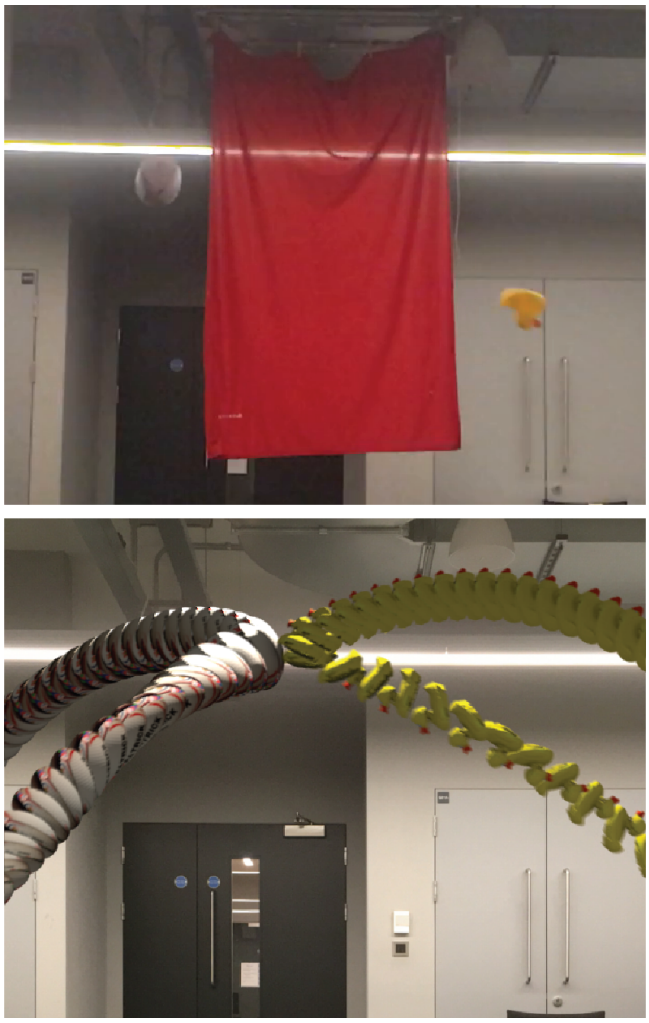

Figure 1: *Starting from a smartphone video recording of a collision sequence behind a curtain (top), we 3D reconstruct a physically valid collision (bottom) using laws of rigid body collisions. Note the reconstructed spin (i.e., angular velocity) of the objects. The algorithm also estimates mass ratio and coefficient of restitution.*

We evaluate our method on a range of complex examples, and generate new collision sequences using an easy authoring workflow. In summary, we propose an algorithm to reconstruct physically valid collisions from sparse input poses from raw video footage, and perform collision analysis without access to exact object geometry.

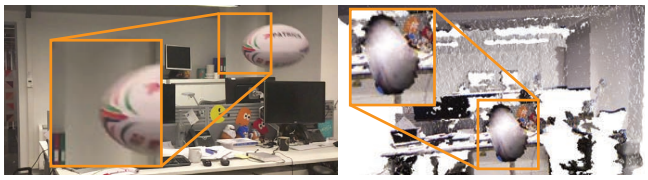

Figure 2: *Accurate acquisition of collision sequences is difficult as objects move fast and are typically occluded around collision time: (left) video captured by a smartphone showing motion blur, while (right) RGBD scan using Kinect is noisy and partial.*

The estimated collision parameters can then be used to author new collisions by mixing reconstructed collisions with synthetic ones.

2 Related work

Rigid body simulation has a long and successful history within Computer Graphics. Starting with the seminal work of Baraff [2001], they are now widely used in all forms of computer animation. A good overview can be found, e.g., in the book by Eberly [2010]. While the forward simulation problem is far from trivial, it is well studied. We are targeting an inverse problem, and in the following we restrict discussion to the relevant works.

First methods to edit or modify simulations proposed space-time methods to compute optimal motions in constrained systems [Witkin and Kass 1988; Liu et al. 1994]. Variants with neural networks [Grzeszczuk et al. 1998], and genetic algorithms were proposed [Tang et al. 1995] to reduce the complexity or synthesize directable motions. A randomized approach to re-construct motions for target configurations of rigid bodies was proposed by Chenney et al. [2000], and an interactive variant was developed by Popović et al. [2000]. While these algorithms offer varying levels of control and parameter estimation for rigid bodies, they focus on virtual situations in a simulator. In contrast, we propose a method that works robustly with only sparse, unreliable data from a real world source.

Areas that would be highly interesting, but which we have currently not taken into account are deformable objects [James and Pai 1999; Debunne et al. 2001], and fracture [Müller et al. 2001; Su et al. 2009]. A closely related challenge is editing deformable simulations [Barbic et al. 2012]. In the following, we assumed that the observed objects do not deform or change topology.

Others have used the complexity of collision events to randomize the solution space, in this way giving control over the outcome of a simulation. [Twigg and James 2007]. Recently, Smith et al. [2012] investigated the even tougher case of multiple, simultaneous collisions. While we focus on the two body case, these papers highlight the complexities of collisions, which are amplified for imperfect real-world objects. A good general survey of collision modeling is the report by Gilardi and Sharf [2002].

An interesting general direction of research investigates the retrieval of modeling equations from data, by employing learning techniques [Bongard and Lipson 2007]. While our method is not directly using machine learning, it represents a *data-driven* approach to recover complex real-world phenomena. In this context, will demonstrate that suitable physical models are powerful regularizers for underdetermined problems.

Geometry acquisition has been simplified with affordable and portable scanning options. Hence, in the recent years, significant efforts have gone towards capturing and parsing 3D scenes. Approaches include using classifiers on scene objects [Schlecht and Barnard 2009; Xiong and Huber 2010; Anand et al. 2011; Koppula et al. 2011; Silberman et al. 2012], interactive 3D modeling from raw RGBD scans [Shao et al. 2012], interleaving segmentation and classification [Nan et al. 2012], unsupervised algorithms to identify and consolidate scans [Mattausch et al. 2014], proxy geometry based scene understanding [Lafarge and Alliez 2013], or studying spatial layout of scenes [Gupta et al. 2010; Lee et al. 2010; Hartley et al. 2012]. Physical constraints have also been used for scene understanding: for example, [Jia et al. 2013; Jiang and Xiao 2013; Zheng et al. 2013; Shao et al. 2014] consider local and global physical stability to predict occluded geometry in scenes. These methods, however, primarily focus on static scenes. In case of dynamic scenes, even state-of-the-art systems like Kinect Fusion [Izadi et al. 2011] mainly capture the static parts of the scenes. Capturing 3D geometry of dynamic scenes remain very challenging. While impressive results have been demonstrated in case of template-based solution for human faces, hair, body, etc., they focus on application specific contexts where object behavior and dynamics can be captured and learned in a training phase. In contrast, we focus on reconstructing 3D geometry of collisions directly from raw videos. Note that due to the nature of the problem, collisions happen fast and cannot be slowed down for recording.

3 Formulation

Our goal is to acquire the parameters of object collisions based on a roughly annotated video input, and reconstruct a physically valid motion. This yields a space-time recording of the collision event that can be re-used in a variety of ways: to set up new collisions that behave faithfully to the original recording, to introduce new objects, or even to author complex interactions between objects from multiple recordings.

We first briefly review rigid body motion in Section 3.1, as the equations provide important building blocks for our optimization problem. We will then explain how we state the problem Section 3.2, and introduce the conservation laws to constrain solutions to the space of physically-plausible ones (Section 3.3 to 3.5). Based on these components, we will formulate the energy we minimize. Below, we will use bold lower-case letters for three- and four-dimensional vectors (e.g., $\mathbf{p}$), and reserve upper case letters for matrices (such as R). We typically run our calculations with world space units, but will omit them for most equations.

3.1 Rigid Bodies

By using a rigid body model we represent an object by its motion around its center of mass $\mathbf{p}$. In addition to $\mathbf{p}$, each body has a pose that captures its orientation w.r.t. global directions, represented as a quaternion $\mathbf{q}$. Due to rigidity, we only need to consider linear and angular velocity ($\mathbf{v}$ and $\mathbf{w}$). A body further has a mass, and a moment of inertia (calculated for a reference pose), denoted by m and $\mathbf{I}_0$, respectively. While $\mathbf{p}, \mathbf{q}, \mathbf{v}$, and $\mathbf{w}$ are changing over time, m and $\mathbf{I}_0$ are assumed to be constant. We will omit the time dependence, e.g. $\mathbf{p}(t)$, as much as possible for brevity, and write $\mathbf{p}$ instead.

Collisions between two objects are typically modelled with a scalar impulse j acting along a collision normal $\mathbf{n}$. We now have two bodies a and b, and we will denote their variables with a subscript, e.g., $\mathbf{p}_a$ is the center of mass of object a. The impulse changes pre-collision velocities into a set of post-collision velocities (denoted with superscript $^{\text{post}}$) such that the scalar relative velocity $v_r = \mathbf{v}_r \cdot \mathbf{n}$ at the point of collision satisfies $v_r^{\text{post}}/v_r = -c$. Here, c is the coefficient of restitution, which is related to the amount of energy that is transferred into a reversal of the object's motion. The remainder is "lost" for the simulation, and dissipated into heat, sound, or work to deform the internal structure of the objects. We

can compute the velocity at the collision point on body a with $\mathbf{x}_c^a$ (relative to the center of mass for body a) as $\mathbf{v}_a + \mathbf{x}_c^a \times \mathbf{w}_a$. This velocity will be important later on to compute the coefficient of restitution.

In rigid body solvers we typically know the pre-collision state, and use a chosen value for c to compute the impulse j, in turn giving the post-collision velocities of both objects. As linear and angular momentum are conserved upon impact, the impulse must act symmetrically upon both objects. The impulse acts upon a and b with opposite signs:

$$\mathbf{v}^{\text{post,a}} = \mathbf{v}^a + j\mathbf{n}/m^a \tag{1}$$

$$\mathbf{v}^{\text{post,b}} = \mathbf{v}^b - j\mathbf{n}/m^b \tag{2}$$

Correspondingly, the instantaneous change of angular momentum $\mathbf{k} = \mathbf{I}\mathbf{w}$ is given by:

$$\mathbf{k}^{\text{post,a}} = \mathbf{k}^a + (\mathbf{x}_c^a \times j\mathbf{n}) \tag{3}$$

$$\mathbf{k}^{\text{post,b}} = \mathbf{k}^b - \left(\mathbf{x}_c^b \times j\mathbf{n}\right) \tag{4}$$

Regular rigid body solver then continue to integrate the motion for both objects with the post collision state.

3.2 Problem Statement

In contrast to the typical procedure above, our goal is to retrieve the physical parameters from a real collision of objects. This would normally involve a large amount of tedious manual work or complicated capturing setups. Thus, we will now explain our *inverse approach* to retrieve the physical parameters based on observations of the objects' trajectories. It should be pointed out here that computing absolute quantities is not possible based on a single video. We will not to able to compute an absolute position of the objects on earth from a single video input, and, correspondingly, we cannot compute their absolute masses. However, we can retrieve relative quantities, i.e., the relative positions, and the relative mass of the objects. We assume that there are no external forces at work, except for a known gravitational acceleration with magnitude $g = 9.81m/s^2$.

Interestingly, the actual shape of the objects does not play a direct role. Only a related quantity, the distribution of mass is important in the form of the moment of inertia. We approximate the moment of inertia of the objects with three diagonal entries, and zero off-diagonals. We will demonstrate in Section 5 that this simplification is general enough to solve a variety of complex cases for more irregular object shapes (see Figure 8).

As input we use a sparse set of cuboid poses for a and b, at arbitrary points before and after the collision. These come from the recorded video, and we use *Blender* to place approximate bounding boxes around the objects. Thus, at n points in time $t_0^{\text{obs}}, ..., t_{n-1}^{\text{obs}}$ we have input i with position $\mathbf{p}_i^{\text{obs},a}$ and pose $\mathbf{q}_i^{\text{obs},a}$ for object a, and $\mathbf{p}_i^{\text{obs},b}$ and $\mathbf{q}_i^{\text{obs},b}$ for object b. We will use superscripts ^{a} and ^{b} for variables related to each of the two objects, and ^{c} for variables related to the collision event. We additionally use the frame rate of the input video f_{fps} to determine the acceleration from gravity over time.

Given this set of positions and poses, our goal is to compute: a parametrization of the trajectories of two bodies a and b, their relative mass, the pre- and post-collision velocities (linear as well as angular), the time of collision, the collision impulse, and the coefficient of restitution. We will describe our approach to incorporate these unknowns in our solve in this order. As our input annotations are potentially inaccurate and sparse, we do not rely purely on the corresponding data-terms in our calculations, but use the physical laws as regularizers.

3.3 Center of Mass Trajectories

The center of mass of an object experiencing a constant acceleration is given by a parabola. This description becomes invalid at the time of the collision, but it is an excellent model for the trajectories before and after, as long as effects such as air drag are negligible. For the two objects, with pre- and post-collision trajectories, we thus need to parametrize and extract four 3D parabolas from the inputs. We parametrize each parabola in 3D space with

$$\mathbf{p}(t) = \mathbf{R}\begin{pmatrix} b_3 t \\ b_1/2\, t^2 + b_2 t \\ 0 \end{pmatrix} + \mathbf{b}_4 \ , \tag{5}$$

where b_1, b_2, b_3 parametrize the curve over time, the vector $\mathbf{b}_4$ determines its offset, and the rotation matrix R its orientation. Together we can significantly reduce the total number of unknowns as we know that all parabolas where caused by the same acceleration.

A first consequence is that the magnitude of the acceleration is shared by all objects. For the 3D gravity vector $\mathbf{g}$, we use the following constant:

$$||\mathbf{g}|| = 9.81 = b_1 f_{\text{fps}}^2 \tag{6}$$

This means that the coefficient b_1 is the same for all four parabolas.

In addition, the different parabolas can only rotate about the axis of gravity, which, as follows from Equation 5, is the y-axis in our formulation. We thus have a global rotation with two degrees of freedom shared by all four parabolas, and individual rotations around the gravity direction. As a consequence we construct the $\mathbf{R}$ matrix as a sequence of Euler angle rotations. We chose the proper Euler angle representation Y,X,Y, the angles of which we denote with β_{y0}, β_x, and β_{y1}, respectively. β_{y0} is different for every parabola, while the other two are shared.

Additionally the two parabolas for an object share the position at the time of collision. This means we can express both parabolas uniquely with a single $\mathbf{b}_4$ offset. In total, that leaves us with two global angles, four individual angles, two offsets, one global factor b_1, and four times b_2, b_3 as unknowns. The geometric setup of the four parabolas is illustrated in Figure 3.

We can directly estimate the linear velocity from these curves with the temporal derivative of Equation 5. The most important velocity in our setting is the velocity at collision time t^c:

$$\mathbf{v}(t^c) = (\mathbf{v}_0 + b_1 t^c) f_{\text{fps}} \ , \quad \text{with} \quad \mathbf{v}_0 = \mathbf{R}\begin{pmatrix} b_2 \\ b_3 \\ 0 \end{pmatrix} \tag{7}$$

We will outline how we compute the collision time from the trajectories in more detail below.

3.4 Pose and Angular Momentum

It is a well studied fact that the angular velocity of a rigid body can change without external forces, but its angular momentum remains constant [Kleppner and Kolenkow 2013]. Thus, for each of the four trajectories, we calculate an angular momentum $\mathbf{k}$ that best explains the poses observed over time. Angular momentum and velocity relate to each other with:

$$\mathbf{k}(t) = \mathbf{q}(t)\mathbf{I}_0\mathbf{q}(t)^{-1}\mathbf{w}(t) \tag{8}$$

$$\mathbf{w}(t) = \mathbf{q}(t)\mathbf{I}_0^{-1}\mathbf{q}^{-1}(t)\mathbf{k}(t) \tag{9}$$

Once we know the angular momentum at time t, we can compute the corresponding angular velocity $\mathbf{w}$, and use it to integrate a pose

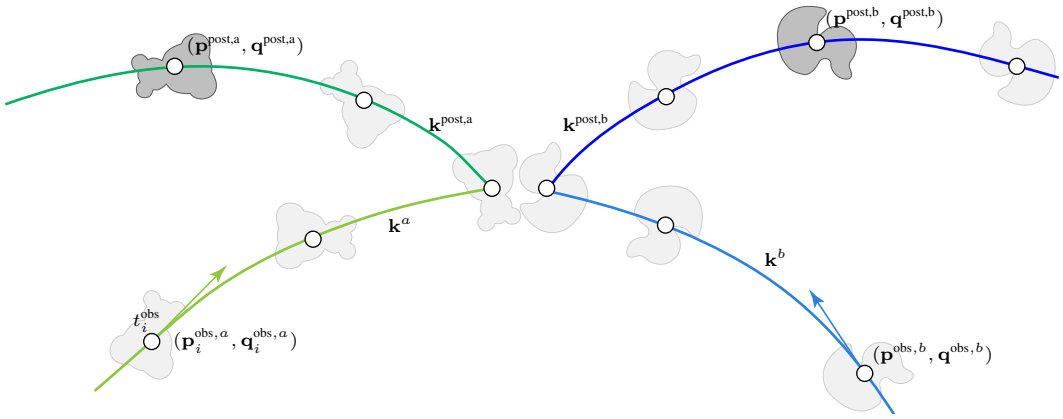

Figure 3: *Modeling two body collisions. The annotated observations from a few time instances t_i^{obs} are linked together with laws of physics: the individual trajectories should be parabolas as we assume them to be bodies in ballistic motion and they get coupled around (unknown) collision point based on linear/angular momentum conservation.*

forward in time. With $\mathbf{w}$ as the imaginary part of a quaternion, an Euler step is given by:

$$\mathbf{q}(t+\Delta t) = \mathbf{q}(t) + \frac{\Delta t}{2}\begin{pmatrix}0\\ \mathbf{w}\end{pmatrix} \otimes \mathbf{q}(t)\,, \tag{10}$$

where $\otimes$ denotes the quaternion dot product.

Similar to the shared offset of the parabolas, we introduce a pose $\mathbf{q}^{\text{c}}$ for each body at collision time that is shared by the two parabola segments. Our goal is not to use one of the input poses directly, due to their potential unreliability. Instead we find an improved pose that, together with an angular momentum $\mathbf{k}$, explains the observed poses as good as possible. Starting with $\mathbf{q}^{\text{c}}$ at collision time t^{c} we can integrate this pose backward or forward to time t with explicit integration. We perform a series of integration steps with Equation 10 from t^{c} to a given time t. This yields the pose of body a or b at any instance in time based on the estimated pose at collision time. We will later on use this step to calculate the estimated object pose at different times, based on our current estimate for the collision pose and object motion. Thus it will later on serve to guide the optimization towards the annotated poses.

3.5 Conservation of Momentum

Throughout the objects' trajectories, angular momentum is conserved in the absence of external torques. Linear momentum is not exactly conserved in our case, as gravity acts on all bodies. Instantaneous rigid body collisions, however, fully preserve linear and angular momentum of the objects involved [Kleppner and Kolenkow 2013]. As such, the equations for an impulse-based collision are ideal to couple the four separate trajectories at the time of collision. We have already exploited the fact the two parabolas for each object have to connect in the collision point, but the conservation laws for momentum provide a much tighter coupling.

Conservation of linear momentum means that the sum of linear momenta before and after collision has to be equal. Since we cannot recover the absolute mass, we set the mass of body a to one, and introduce a mass ratio $m^{\text{b,a}} = m^b/m^a$, (we will revisit the case of a potentially infinitely heavy object in Section 5.1). Formulating the equations in terms of $m^{\text{b,a}}$ yields

$$\mathbf{v}^a(t^{\text{c}}) + \mathbf{v}^b(t^{\text{c}})m^{\text{b,a}} = \mathbf{v}^{\text{post,a}}(t^{\text{c}}) + \mathbf{v}^{\text{post,b}}(t^{\text{c}})m^{\text{b,a}}\,. \tag{11}$$

As the velocities are determined from the parabolas, this equation ensures that these curves do not create or lose linear momentum, and effectively couple the planes of the four parabolas.

We can similarly ensure that the angular momentum we compute does not violate conservation of angular momentum. For the angular momenta, we have to consider their sum, and additionally the instantaneous angular momentum around the origin:

$$\begin{aligned}&\mathbf{p}^a \times \mathbf{v}^a + \mathbf{k}^a + m^{\text{b,a}}\mathbf{p}^b \times \mathbf{v}^b + \mathbf{k}^b = \\ &\mathbf{p}^a \times \mathbf{v}^{\text{post,a}} + \mathbf{k}^{\text{post,a}} + m^{\text{b,a}}\mathbf{p}^b \times \mathbf{v}^{\text{post,b}} + \mathbf{k}^{\text{post,b}}\end{aligned} \tag{12}$$

Note that all quantities in Equation 12 are evaluated at collision time t^{c}, this is omitted for brevity. The two equations Equation 11 and Equation 12 constrain our solution space to a physically plausible ones, and provide a first coupling between objects a and b. We can improve this coupling by using the impulse equations from Section 3.1. With the equation for linear impulse (Equation 1) we can introduce a relationship between pre- and post-collision velocities that takes into account the exchange of momentum between the two bodies relative to their mass:

$$\begin{aligned}\mathbf{v}^{\text{post,a}}(t^{\text{c}}) &= \mathbf{v}^a(t^{\text{c}}) + \frac{j\mathbf{n}}{1}\\ \mathbf{v}^{\text{post,b}}(t^{\text{c}}) &= \mathbf{v}^b(t^{\text{c}}) - \frac{j\mathbf{n}}{m^{\text{b,a}}}\end{aligned} \tag{13}$$

This equation turned out to be crucial for retrieving the mass ratio $m^{\text{b,a}}$ during the combined optimization, which we will detail below.

The angular impulse equations allow us to enforce a shared collision point $\mathbf{x}_{\text{c}}$. Equation 3 connects the angular momenta of the two bodies via the relative offset of the collision point:

$$\begin{aligned}\mathbf{k}^{\text{post,a}}(t^{\text{c}}) &= \mathbf{k}^a(t^{\text{c}}) + ((\mathbf{x}_{\text{c}} - \mathbf{b}_4^a) \times j\mathbf{n})\\ \mathbf{k}^{\text{post,b}}(t^{\text{c}}) &= \mathbf{k}^b(t^{\text{c}}) - \Big((\mathbf{x}_{\text{c}} - \mathbf{b}_4^b) \times j\mathbf{n}\Big)\end{aligned} \tag{14}$$

Deduced from relative velocities at collision point and the collision normal: We can complete the set of necessary equations by computing the relative velocities at the collision point to compute the coefficient of restitution. We only need consider the velocity com-

ponents along the collision normal direction:

$$c = \frac{\left(\hat{\mathbf{v}}^{\text{post,b}} - \hat{\mathbf{v}}^{\text{post,a}}\right) \cdot \mathbf{n}}{\left(\hat{\mathbf{v}}^{a} - \hat{\mathbf{v}}^{b}\right) \cdot \mathbf{n}} \quad \text{with} \tag{15}$$
$$\hat{\mathbf{v}} = \mathbf{v}(t^{\text{c}}) + \mathbf{w}(t^{\text{c}}) \times \mathbf{x}_{\text{c}}$$
$$\mathbf{w}(t^{\text{c}}) = \mathbf{I}^{-1}(t^{\text{c}})\, \mathbf{k}(t^{\text{c}})$$

With these equations at hand we can formulate an optimization problem to compute the unknown quantities.

4 Method

The terms of the previous section can be divided into those related to the input data, and those related to the underlying physics. In terms of unknowns, we can distinguish the parameters of the four trajectory segments of the two objects, which determine position and velocities, and the unknowns of the collision event. We will use a superscript $^{(*)}$ to distinguish the four trajectories, i.e. $* \in$ (a, b, post-a, post-b).

Per trajectory we have the unknowns of each parabola.They are $b_2^{(*)}, b_3^{(*)}, \beta_{y0}^{(*)}$, while $\mathbf{b}_4^{(\text{a,b})}$ exist once for each body. Finally, we have a single b_1, and the global angles β_x, β_{y1}. Unknowns for angular momentum exist for each segment: $\mathbf{k}^{(*)}$. Additionally, we have the unknowns for the collision: orientation for both objects at collision time $\mathbf{q}^{\text{c (a,b)}}$, location $\mathbf{x}_{\text{c}}$, impulse $j\mathbf{n}$ and the mass ratio $m^{\text{b,a}}$, as well as the coefficient of restitution c and the time of the collision t^{c}.

As input we typically have nine tuples of position and orientation for both bodies: $\mathbf{p}_i^{\text{obs}}, \mathbf{q}_i^{\text{obs}}$ with $i \in (1, ..., 9)$, at times t_i^{obs}. Additionally, we require a bounding box estimate for each body $\mathbf{s}^{\text{a,b}}$, and earth's gravity constant. Summing these numbers up we have 47 unknowns, and 115 inputs, i.e. equations from the input data. Finally, we have a set of additional equations from the conservation laws, namely equations (11) to (15), and (6) to tie the observations together according to the physical model. They represent our physics-based regularization encoded as 7 additional equations.

122 equations with 47 unknowns mean that our problem is over-determined. As the equations are highly non-linear, we compute a solution using non-linear least-squares. We summarize the 47 unknowns in the vector $\mathbf{x}$ for this least-squares solve, and re-formulate the equations to take the form

$$\underset{x}{\text{argmin}} \frac{1}{2} \sum_i w_i \|f_i(\mathbf{x})\|^2 \,, \tag{16}$$

where w_i is an weighting factor for the corresponding energy term f_i.

Physics. We can directly turn the physics constraints (Eq.s 6,11-15) into energy terms by defining the least squares energy to be their residual. We combine the equations for momentum conservation (11,12) into f^{mom}, and the impulse equations (13,14) into f^{imp}.

Instead of imposing Equation 6 directly, we re-formulate it to align the gravity direction with the y-axis, but allow for slight deviations. We found this to work more robustly in the presence of not fully aligned data, e.g., the phone being held slightly at an angle. Thus, we formulate f^g as

$$f^g = 9.81 - R_{0,\beta_x,\beta_{y1}} \begin{pmatrix} 0 \\ b_1 f_{\text{fps}}^2 \\ 0 \end{pmatrix} \cdot \begin{pmatrix} 0 \\ 1 \\ 0 \end{pmatrix} \,. \tag{17}$$

Note that β_{y0} is zero here, as this value represents rotation around the axis of gravity. Including the gravity term Equation 17, our physical regularizers can be summarized as $f^{\text{mom}} + f^{\text{imp}} + f^{\text{g}}$.

Observations. Next, we formulate the data terms. Based on Equation 5 we add terms for position fidelity. For each input position $\mathbf{p}_i^{\text{obs}}$ at time t_i^{obs} we add an energy term penalizing its deviation from the corresponding parabola:

$$f_i^{\text{pos}} = \mathbf{R}_{\beta_{y0}^*,\beta_x,\beta_{y1}} \begin{pmatrix} b_3^*(t_i^{\text{obs}} - t^{\text{c}}) \\ b_1/2(t_i^{\text{obs}} - t^{\text{c}})^2 + b_2^*(t_i^{\text{obs}} - t^{\text{c}}) \\ 0 \end{pmatrix} + \mathbf{b}_4^* - \mathbf{p}_i^{\text{obs}} \tag{18}$$

where $* \in$ (a, b, post-a, post-b) depending on the input i. Note that $\mathbf{b}_4$ is shared among pre- and post-collision parabolas. We likewise add energy terms for the poses with Equation 10:

$$f_i^{\text{ori}} = \mathbf{q}^*(t_i^{\text{obs}}) - \mathbf{q}_i^{\text{obs}} \,, \tag{19}$$

where $\mathbf{q}^*(t_i^{\text{obs}})$ is expressed as a series of explicit integration steps starting from $\mathbf{q}^{\text{c},*}$. We also calculate the moment of inertia of the objects based on the bounding boxes of the annotations. We use the analytic equations for a solid cuboid, and scale the resulting moment of inertia with the masses of the objects, i.e., one and $m^{\text{b,a}}$. For a specific instance in time, $\mathbf{I}_0$ is transformed with the orientation at this time according to Equation 9.

Combining all the pieces, our least-squares energy for Equation 16 has the physics terms f^{mom}, f^{imp}, and f^{g}, in addition to one data term f_i^{pos} and f_i^{ori} for every input pose. We additionally have weights for these terms, e.g., w^{pos} for the position terms f_i^{pos}, the values for which we will give below.

4.1 Optimization Phases

As our optimization has to navigate a highly non-linear energy landscape, a good initialization is crucial. Additionally, we found that our optimization profits from an alternation scheme, that keeps different sets of unknowns constant for three different solving phases. In this way, each step provides an improved initial guess for the next step of the alternation. We work with three phases, from large, geometric properties in phase 1, towards the details of the collision in phase 3. For the different terms of our formulation, we use the following weights throughout all of our examples: $w^{\text{mom}} = w^{\text{imp}} = 10^{-1}$, $w^{\text{g}} = 1$, and $w_i^{\text{pos}} = w_i^{\text{ori}} = 4$.

We start with $m^{\text{b,a}} = 1$, and compute velocities and momenta from a picked segment $\mathbf{p}_i^{\text{obs}}$ to $\mathbf{p}_{i+1}^{\text{obs}}$ of the input poses. How to choose this segment will be detailed below. We set $t^{\text{c}} = (t_i^{\text{obs}} + t_{i+1}^{\text{obs}})/2$, and compute collision pose and angular momentum with a spherical linear interpolation for the segment at t^{c}. We found that the initialization of the parabola parameters did not play a significant role. We simply initialize default parabolas with $b_2 = -0.05$, $b_3 = 1/f_{\text{fps}}$, $\beta_{y0} = 20^\circ$ (slightly facing away from the camera for our setup), and $\beta_x = \beta_{y1} = 0^\circ$. $b1$ is a constant (Equation 6). Initialization of $j\mathbf{n}$ is likewise uncritical, we use random values around 0.1.

Phase 1: We first solve for the trajectories in combination with the time of collision t^{c}. For this step, we include f^{mom} and f^{g}, but discard the impulse term f^{imp}. We define t^{c} to be the point in time where the connection points $\mathbf{b}_4^a$ and $\mathbf{b}_4^b$ of the parabolas have their closest distance. We do not add an energy term for the optimization here, but instead iterate over the different segments of our input poses, until we find the closest value (in the worst case, this will take eight iterations). We re-initialize the optimization as described

above for each segment, and once we have established the minimal distance configuration for the parabolas, t^c is fixed for the subsequent phases.

Interestingly, the collision point $\mathbf{x}_c$ forms a zero set in solution space. Thus, there is a whole set of collision points which explain a set of inputs, and lead to the same minima, meaning identical results for the unknowns. Closed form solutions for this set turned out to be too complicated to be useful. For restricted cases, the set consists of a polynomial curve in the velocities and the collision normal. Even without knowing the exact form of this zero set, it means that we can retrieve the correct solution without having to know the exact point of collision. The only thing to prevent is a collision point very far away from the objects, which would lead to potentially large angular velocities, where small deviations from the correct angular velocity lead to suboptimal solutions. Thus, for phase 2, we set the collision point to be located at the midpoint of the objects' centers at the time of collision, i.e. $\mathbf{x}_c = (\mathbf{b}_4^a + \mathbf{b}_4^b)/2$ for phase 2. In this phase, we start solving for collision properties by including f^{imp} in the optimization. Removing the degrees of freedom of the point of collision increases robustness, especially for cases with lower quality annotations.

In the last step, phase 3, we remove the hard constraint for collision point, and let the optimization refine its position and the remainder of the solution based on the initialization from phase 2. This phase yields our final values for the collision. While mass ratio, velocities etc. have been computed during the solve, we evaluate Equation 15 to compute our final estimate for the coefficient of restitution c.

4.2 Implementation

We use the *Ceres* library for our implementation, as it provides an efficient solver for non-linear least squares problems. More specifically, we use its sparse-Cholesky variant with a Jacobian preconditioner. In addition, Ceres provides a set of tools for quaternion parametrizations, which helps to compute stable derivatives.

Due to the non-linearity of the angular motion in Equation 10 we restrict the integration step to $\Delta t/4$. Note that the sequence of integration steps is taken into account during the auto differentiation performed in the solver. Thus more sub-steps lead to an increased difficulty computing derivatives for the Jacobian. While higher-order integrators or even closed-form Poinsot solutions would be available to express a rigid body pose over time, we found that the explicit integration yields stable solutions in practice, and leave the other alternatives for future work.

In our formulation above, we took care to minimize the number of free variables and energy terms. It turned out to be crucial for a good solution to use a formulation with shared variables, instead of enforcing equalities with highly weighted penalty terms. E.g., instead of adding a term $\mathbf{b}_4^a - \mathbf{b}_4^{\text{post},a}$, we formulate the parabolas with a shared variable, which inherently enforces the equality of the offsets. Likewise, shared angles and poses are implemented with shared variables instead of additional penalty terms.

5 Results

We will first show a series of evaluation examples to illustrate the accuracy of our method. We then move to more complex examples, and authored collisions. For all of the following examples, we captured 1280x720 video with 120fps or 240fps using a regular iPhone6. The videos were typically downsampled to 60fps for annotation. These were done in Blender, using a set of camera intrinsics for the iPhone camera.

5.1 Evaluation and Validation

Below we compare our calculations to different setups with known quantities. In each case, the measured or estimated real-world quantity is denoted with an m subscript, e.g., c_m for an estimated coefficient of restitution, to be compared with our calculated results, e.g. c.

Potential Energy Comparison One way to measure the coefficient of restitution is to measure the ratio of energy lost during a collision. This can be performed in a very controlled setting, where where we can measure the change of energy using only the objects' potential energy, i.e. when they have zero kinetic energy. We use different balls dropped from a controlled height h, measuring the apex of their trajectories after they bounce off the floor. When no energy is transferred into an angular motion, we can compute $c = \sqrt{\frac{h_{\text{after}}}{h_{\text{before}}}}$.

We performed this experiment for the three types of balls shown on the right (from left to right: rugby, tennis, and ping-pong), and computed the resulting c values. We then annotated the scenes, to compare the c estimates computed with our algorithm. As this setup involves one static object (the floor), we remove all unknowns of the second object from our solve. In total we have 3,4, and 3 versions for rugby, tennis, and ping-pong ball respectively. In this order, our computed c values have an error of only 1%, 5%, and 4%. All measured and computed values can be found in Table 1. Given the rough annotations, we believe these are very good results.

Name	c_{ours}	c_m
Ping Pong	0.857, 0.846 , 0.868	0.812
Tennis	0.76, 0.816, 0.787, 0.745	0.750
Rugby	0.697, 0.684, 0.767	0.711

Table 1: *Results of the c estimation.*

We can consistently measure c (see Table 1), even, when some energy is stored/converted to kinetic energy throughout the recording, *i.e.,* the ball is or starts spinning (Figure 4). The simple estimation with potential energy is not valid for this case, while our formulation is still able to compute an estimate very close to the non-spinning case (with an error of ca. 6%).

Figure 4: *Rotating rugby ball: Estimated c= 0.767, matching earlier measurements around* 0.711.

Mass Ratio While we cannot measure c for more complicated collisions of two bodies in mid air, we can measure a ground truth for the mass ratio of the objects. We did this experiment for two sets of boxes a small (S) and medium size one (M).

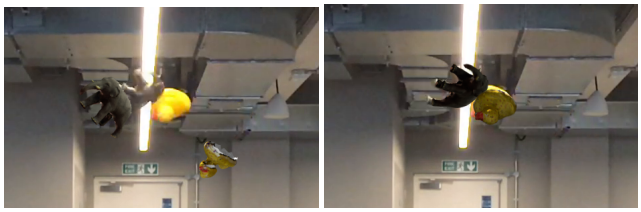

Figure 5: *A direct interpolation in Blender (left), versus our reconstructed result (right). The interpolated result is far away from the right position (shown in the background).*

Finite mass. While most of the estimated mass ratios are very close

Name	$m^{\text{b,a}}_{\text{ours}}$	$m^{\text{b,a}}_{m}$
S & S	1.062, 0.98, 1.155	1.0
S & M	2.275, 2.3336, 1.967, 1.577	2.223

Table 2: *Results of the mass ratio estimation.*

to the measured one (usually within 10%), the last measurement of the small-medium cases stands out as a negative example. This example illustrates a possible, but rare, failure case for which the relative depths of the objects was inaccurate. This leads to a shifted parabola reconstruction, which in turn typically leads to noticeable under- or over-estimations of the mass ratios.

Our annotations are very coarse, and not sufficient for a play-back of the collision event. To illustrate this, Figure 5 shows a comparison of a direct interpolation in Blender, and our reconstructed result. The video is shown in the background, and the direct interpolation gives obviously non-physical, and unexpected behavior, especially around the time of collision.

Synthetic Data For our last evaluation we used inputs for a forward simulation to test our method across a broad range of inputs. We used the *Bullet* physics engine in Blender to evaluate the robustness of our algorithm with respect to the interval between annotated poses. We created a scene with two colliding cuboids under gravity. We carefully turned off friction and all artificial damping parameters (linear, angular per object, and global minimum thresholds) in order to achieve the most plausible setup. We then exported the state of both cuboids at every time step, and sub-sampled this data when using it as input for our solver. which we then subsampled with regular intervals starting from the collision time to mimic annotated input.

We measured, how our algorithm is capable of estimating $m^{\text{b,a}}$ and c with increasing distances between the annotations. Figure 6 shows of our estimations for growing intervals. Note that there is a gap of twice the interval size around the time of collision. The consistency of our estimates for both parameters is illustrated with the blue dots in Figure 6. As the sequence is 90 frames long in total, an interval of 19 frames means only four annotated poses per object are available to our solver. Up to this extreme case, our solver retrieves excellent estimates.

To test robustness, we added uniform random noise (5 and 10 percent) to each 3D and quaternion coordinate to simulate inaccurate annotations. This is noticeable in the growing spread of the measurements around the ground truth, but our method is still able to recover reasonable estimates for most cases. Considering the roughness and sparsity of the data, we believe this still represents an impressive result. We noticed that failure cases can be detected by inspecting the mass ratio being set to its minimum or maximum bound $(10^{-5}, 10)$, or the condition $0 \leq c \leq 1$ being violated.

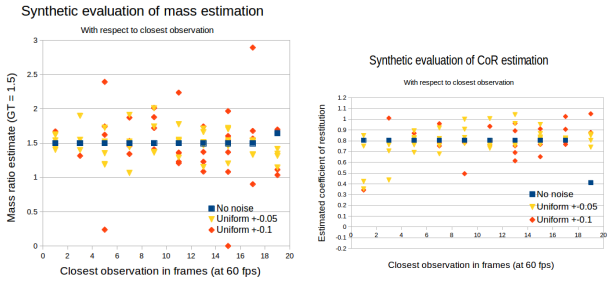

Figure 6: *Synthetic evaluation of robustness towards elapsed time to first annotation. We added uniform noise coefficient-wise to the simulated input annotations.*

5.2 Real-world Cases

Figure 7 shows one of several experiments, where we successfully reconstructed the collision of box shaped objects. Given only 3 poses before and after collision, the shape of the fitted parabolas allow an accurate estimation of the objects' mass ratios. These boxes have different shape, but are made of the same material, hence we see similar c values in the reconstructions. Our calculated relative mass is $m^{\text{b,a}} = 2.275$, while we measured a close agreement of $m^{\text{b,a}}_{m} = 2.223$. We calculate a $c = 0.303$ for this case. Some deviation is caused by the fact, that the smaller packages were less rigid. In these cases, more energy is absorbed due to the deformation of the bodies, so we expect lower c values, which is exactly how our optimization explained the collision happening over a longer period of time.

Figure 7: *Reconstruction of colliding cuboid shapes from sparsely annotated poses. Top half shows the video at collision time, while the bottom half shows our reconstructed trajectories and selected frames.*

In Figure 8 we demonstrate, how we can reconstruct collisions with arbitrarily shaped objects. The optimization is capable of approximating the observed motion with the motion of a box shape, and reconstructs the input collision from the annotations. Bounding boxes can be used to annotate the input, or for higher accuracy, one can use scans to create semi-transparent models, that then are used for annotation in Blender (and for the renderings shown).

Our calculated relative mass is $m^{b,a} = 1.924$, while we measured $m^{b,a}_m = 1.996$, and we computed $c = 0.407$.

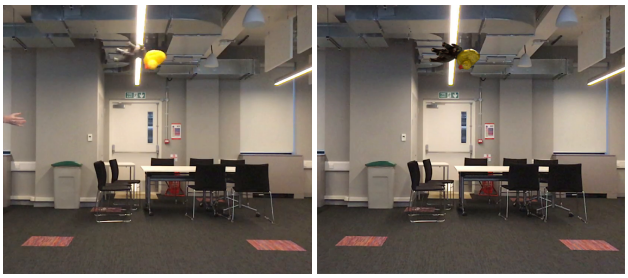

Figure 8: *A reconstruction (right) of collision of complex shapes from sparsely annotated poses (5+8) from video recorded with a smartphone (left).*

5.3 Authoring

To show the how to author complex new collision situations with our solutions, we developed a Blender plug-in (Figure 9), that allows the import of our reconstruction. This way a user has the possibility of combining several observed collisions, and time them to co-incide, or later interact. For the first collisions, we use the highly accurate reconstructions of the collisions. For later collisions, we switch to Bullet to predict the object trajectories as accurately as possible. Pairs of objects ('collisions') can be moved around, and rotated around the gravity vector. The position of the objects at the current frame is shown by the coloured spheres. To time extra collisions, one can offset an observed collision in time. The timing can be done manually, or automatically (closest point on parabolas, later collision adjusted to the position of the earlier).

In Figure 10 we construct a complex case of three pairs of rigid bodies colliding in mid air, and two of them colliding again afterwards on their downward trajectories. Based on our reconstructed solutions, the new simulations behaves naturally, and the object collide as we would expect them to do from their real-world counterparts.

Limitations. A key limitation of our method is that it applied to rigid body collisions. In case of deformable bodies, the physics priors we use are not valid. Empirically we observed that our estimates are still reasonable for near rigid collisions (see Figure 11). Our method assumes access to initial scans of participating 3D models. Finally, the user is expected to annotate a few poses of the two objects. In case the object is feature less or symmetric such annotations can easily miss object rotations. For example, if the rugby ball was featureless, it will be nearly impossible to disambiguate the pose estimates. An optical tracking based approach may help here, but the motion blur in the inputs pose challenges.

6 Conclusions

We presented SMASH a data-driven framework for capturing, reconstructing, and authoring collision sequences. The key idea is regularize the problem of reconstructing raw videos of collision of object pairs using laws of physics dealing with rigid body collisions. We demonstrate how to reconstruct (in 3D) plausible collision sequences just by observing objects in motion away from the collision instant, while assuming access to static version of the models and some annotations. The method allows us to directly 'read off' collision parameters like mass ratio, coefficient of restitution, and initial condition for the colliding objects. The information can then be readily used to author new collision sequences,

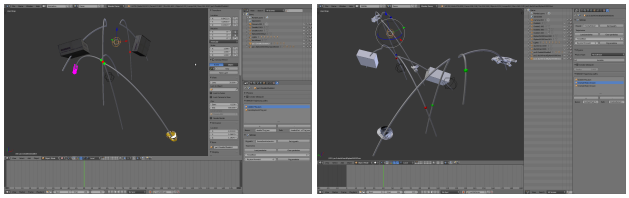

Figure 9: *Our Blender plugin to author and synchronize collisions using the recovered parameters from our reconstruction. Object trajectories, mass ratios and c values are read in from the optimization automatically.*

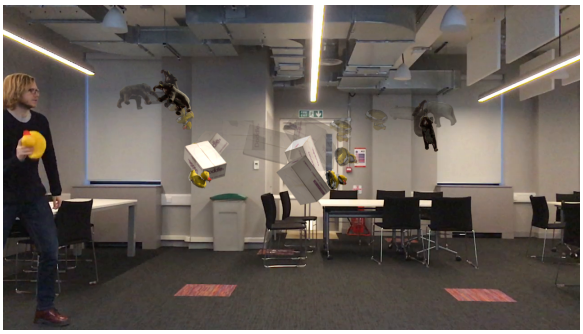

Figure 10: *Result of authoring process. 5 collisions were created using the retrieved information from two reconstructions.*

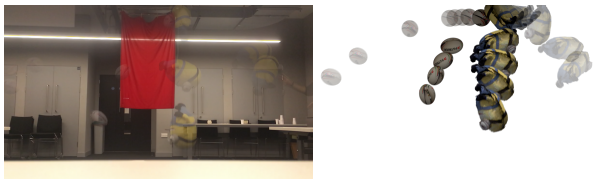

Figure 11: *Collision reconstruction for near-rigid objects using our framework. Even if the rigid body physics priors are not theoretically valid, for near-rigid colliding objects we found our method to give (visually) plausible reconstructions. However, just around collision time, the reconstructed poses inter-penetrated (see video).*

thus allowing users to combine recordings and synthetic collision sequences to create non-trivial collision effects.

While we presented a first workflow for reconstructing collision sequences, many avenues for improvements remain. Our current implementation expects the author to annotate a few of the object poses. An interesting next step will be to allow the optimization to progressively refine initial annotations based on the current reconstruction. Further, the annotations can be pushed forward (or backward) in time using a physics-guided optical frame using RGB information for similarity. We anticipate motion blur in the recordings to limit the extent of such interpolation. Finally, it will be valuable to generalize our method to handle non-rigid body collisions [Müller et al. 2005; Barbic et al. 2012]. While this is going to be very challenging, we are hopeful as the initial results even with rigid body assumptions are promising (see Figure 11).

References

ANAND, A., KOPPULA, H. S., JOACHIMS, T., AND SAXENA, A. 2011. Contextually guided semantic labeling and search for 3d point clouds. *CoRR abs/1111.5358*.

BARAFF, D. 2001. Physically based modeling: Rigid body simulation. *SIGGRAPH Course Notes, ACM SIGGRAPH 2*, 1, 2–1.

BARBIC, J., SIN, F., AND GRINSPUN, E. 2012. Interactive editing of deformable simulations. *ACM Trans. on Graphics (SIGGRAPH 2012) 31*, 4.

BONGARD, J., AND LIPSON, H. 2007. Automated reverse engineering of nonlinear dynamical systems. *Proceedings of the National Academy of Sciences 104*, 24, 9943–9948.

CHENNEY, S., AND FORSYTH, D. A. 2000. Sampling plausible solutions to multi-body constraint problems. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, ACM Press/Addison-Wesley Publishing Co., 219–228.

DEBUNNE, G., DESBRUN, M., CANI, M.-P., AND BARR, A. H. 2001. Dynamic real-time deformations using space & time adaptive sampling. In *Proceedings of the 28th annual conference on Computer graphics and interactive techniques*, ACM, 31–36.

EBERLY, D. H. 2010. *Game physics*. Taylor and Francis.

GILARDI, G., AND SHARF, I. 2002. Literature survey of contact dynamics modelling. *Mechanism and machine theory 37*, 10, 1213–1239.

GRZESZCZUK, R., TERZOPOULOS, D., AND HINTON, G. 1998. Neuroanimator: Fast neural network emulation and control of physics-based models. In *Proceedings of the 25th annual conference on Computer graphics and interactive techniques*, ACM, 9–20.

GUPTA, A., EFROS, A. A., AND HEBERT, M. 2010. Blocks world revisited: Image understanding using qualitative geometry and mechanics. In *ECCV*.

HARTLEY, E., KERMGARD, B., FRIED, D., BOWDISH, J., PERO, L. D., AND BARNARD, K. 2012. Bayesian geometric modeling of indoor scenes. *IEEE CVPR*, 2719–2726.

IZADI, S., KIM, D., HILLIGES, O., MOLYNEAUX, D., NEWCOMBE, R., KOHLI, P., SHOTTON, J., HODGES, S., FREEMAN, D., DAVISON, A., AND FITZGIBBON, A. 2011. Kinectfusion: Real-time 3d reconstruction and interaction using a moving depth camera. In *Proc. UIST*, 559–568.

JAMES, D. L., AND PAI, D. K. 1999. Artdefo: accurate real time deformable objects. In *Proceedings of the 26th annual conference on Computer graphics and interactive techniques*, ACM Press/Addison-Wesley Publishing Co., 65–72.

JIA, Z., GALLAGHER, A., SAXENA, A., AND CHEN, T. 2013. 3d-based reasoning with blocks, support, and stability. In *IEEE CVPR*, 1–8.

JIANG, H., AND XIAO, J. 2013. A linear approach to matching cuboids in rgbd images. In *IEEE CVPR*.

KLEPPNER, D., AND KOLENKOW, R. 2013. *Introduction to Mechanics, 2nd Ed.* Cambridge University Press.

KOPPULA, H., ANAND, A., JOACHIMS, T., AND SAXENA, A. 2011. Semantic labeling of 3d point clouds for indoor scenes. In *NIPS*.

LAFARGE, F., AND ALLIEZ, P. 2013. Surface reconstruction through point set structuring. *CGF*.

LEE, D. C., GUPTA, A., HEBERT, M., AND KANADE, T. 2010. Estimating spatial layout of rooms using volumetric reasoning about objects and surfaces. In *NIPS*, vol. 24.

LIU, Z., GORTLER, S. J., AND COHEN, M. F. 1994. Hierarchical spacetime control. In *Proceedings of the 21st annual conference on Computer graphics and interactive techniques*, ACM, 35–42.

MATTAUSCH, O., PANOZZO, D., MURA, C., SORKINE-HORNUNG, O., AND PAJAROLA, R. 2014. Object detection and classification from large-scale cluttered indoor scans. *CGF Eurographics*.

MÜLLER, M., MCMILLAN, L., DORSEY, J., AND JAGNOW, R. 2001. Real-time simulation of deformation and fracture of stiff materials. In *Proceedings of the Eurographic Workshop on Computer Animation and Simulation*, Springer-Verlag New York, Inc., New York, NY, USA, 113–124.

MÜLLER, M., HEIDELBERGER, B., TESCHNER, M., AND GROSS, M. 2005. Meshless deformations based on shape matching. In *ACM SIGGRAPH 2005 Papers*, SIGGRAPH '05, 471–478.

NAN, L., XIE, K., AND SHARF, A. 2012. A search-classify approach for cluttered indoor scene understanding. *ACM SIGGRAPH Asia 31*, 6, 137:1–137:10.

POPOVIĆ, J., SEITZ, S. M., ERDMANN, M., POPOVIĆ, Z., AND WITKIN, A. 2000. Interactive manipulation of rigid body simulations. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, ACM Press/Addison-Wesley Publishing Co., 209–217.

SCHLECHT, J., AND BARNARD, K. 2009. Learning models of object structure. In *NIPS*.

SHAO, T., XU, W., ZHOU, K., WANG, J., LI, D., AND GUO, B. 2012. An interactive approach to semantic modeling of indoor scenes with an rgbd camera. *ACM SIGGRAPH Asia 31*, 6, 136:1–136:11.

SHAO, T., MONSZPART, A., ZHENG, Y., KOO, B., XU, W., ZHOU, K., AND MITRA, N. J. 2014. Imagining the unseen: Stability-based cuboid arrangements for scene understanding. *ACM SIGGRAPH Asia*.

SILBERMAN, N., HOIEM, D., KOHLI, P., AND FERGUS, R. 2012. Indoor segmentation and support inference from rgbd images. In *ECCV*.

SMITH, B., KAUFMAN, D. M., VOUGA, E., TAMSTORF, R., AND GRINSPUN, E. 2012. Reflections on simultaneous impact. *ACM Trans. Graph. 31*, 4 (July), 106:1–106:12.

SU, J., SCHROEDER, C., AND FEDKIW, R. 2009. Energy stability and fracture for frame rate rigid body simulations. In *Proceedings of the 2009 ACM SIGGRAPH/Eurographics Symposium on Computer Animation*, ACM, 155–164.

TANG, D., NGO, J. T., AND MARKS, J. 1995. N-body spacetime constraints. *The Journal of Visualization and Computer Animation 6*, 3, 143–154.

TWIGG, C. D., AND JAMES, D. L. 2007. Many-worlds browsing for control of multibody dynamics. In *ACM Transactions on Graphics (TOG)*, vol. 26, ACM, 14.

WITKIN, A., AND KASS, M. 1988. Spacetime constraints. In *ACM Siggraph Computer Graphics*, vol. 22, ACM, 159–168.

XIONG, X., AND HUBER, D. 2010. Using context to create semantic 3d models of indoor environments. In *BMVC*, 1–11.

ZHENG, B., ZHAO, Y., YU, J. C., IKEUCHI, K., AND ZHU, S.-C. 2013. Beyond point clouds: Scene understanding by reasoning geometry and physics. In *IEEE CVPR*.