Pacific Graphics 2016
E. Grinspun, B. Bickel, and Y. Dobashi
(Guest Editors)

Volume 35 (2016), Number 7

Physically Based Video Editing

J.-C. Bazin[1], C. Plüss (Kuster)[1], G. Yu[4], T. Martin[1], A. Jacobson[1,2,3] and M. Gross[1]

[1]ETH Zurich, Switzerland [2]Columbia University, USA [3]University of Toronto, Canada [4]Nanyang Technological University, Singapore

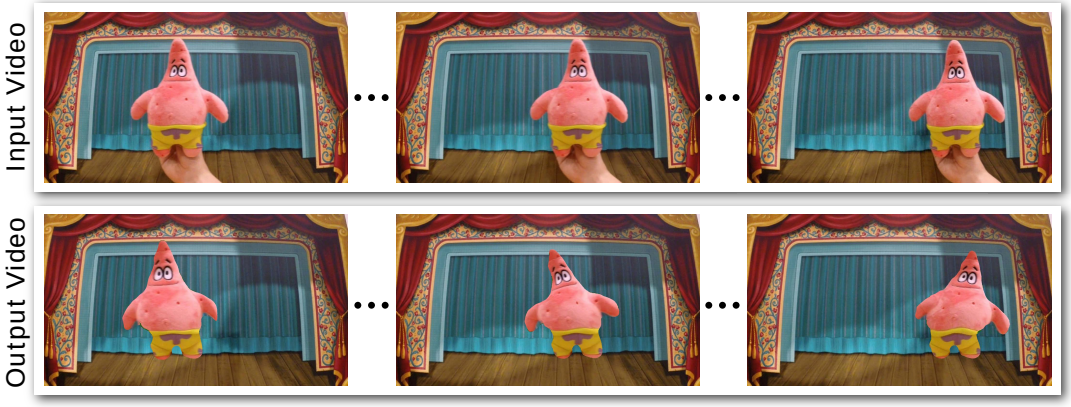


Figure 1: *Our proposed physically based video editing approach outputs an edited version of an input video where the physical properties of an object have been manipulated, such as exaggerating the elastic behavior of a plush toy. A key aspect of our approach is that we append an image-based visibility constraint to the physics-based simulation to ensure that the simulated object can be appropriately textured from the input video.*

Abstract

Convincing manipulation of objects in live action videos is a difficult and often tedious task. Skilled video editors achieve this with the help of modern professional tools, but complex motions might still lack physical realism since existing tools do not consider the laws of physics. On the other hand, physically based simulation promises a high degree of realism, but typically creates a virtual 3D scene animation rather than returning an edited version of an input live action video. We propose a framework that combines video editing and physics-based simulation. Our tool assists unskilled users in editing an input image or video while respecting the laws of physics and also leveraging the image content. We first fit a physically based simulation that approximates the object's motion in the input video. We then allow the user to edit the physical parameters of the object, generating a new physical behavior for it. The core of our work is the formulation of an image-aware constraint within physics simulations. This constraint manifests as external control forces to guide the object in a way that encourages proper texturing at every frame, yet producing physically plausible motions. We demonstrate the generality of our method on a variety of physical interactions: rigid motion, multi-body collisions, clothes and elastic bodies.

Categories and Subject Descriptors (according to ACM CCS): I.3.3 [Computer Graphics]: Picture/Image Generation—Line and curve generation

1. Introduction

Photograph and video manipulation is nearly as old as photographs and videos themselves. Modern professional tools, such as Adobe Photoshop or After Effects, allow highly skilled users to manipulate images and videos. For example, with rotoscoping, hole-filling and keyframing, an animator could alter a foreground object's motion to create a desired effect, such as following a certain trajectory. However, these manipulations are based largely on low-level edits without parameters that correspond to the physical world captured

in the images [ZFL*10]. On the other hand, physically based simulation has made leaping advances in recent years, diminishing the visible difference between rendered animations and captured live action videos. However, the typical input to and the output from physically based simulation is a description of the virtual 3D scene geometry and associated properties: that is, not directly inputting and outputting live action videos.

We propose to combine video editing and physically based simulation to achieve physics-based video editing. By bootstrapping physically based simulation with image-based constraints, we inherit a level of motion realism difficult to achieve without detailed and tedious manual intervention. Given an input video of an object, our approach permits to generate a new version of the video where the physical properties of the object have been edited (Figure 1).

Problem statement and contributions. Our goal is to modify an input video such that the captured object appears with new physical properties. We assume a geometric 3D model of the object is available. This (untextured) model can be known *a priori* (e.g., from measurements or 3D printing), automatically computed from a video [PR12, SSS06, TKM*13], obtained from a single image [CZS*13, HWK15] or available in online public repositories of 3D models (e.g., 3D Warehouse or Turbosquid). We represent the object using its 3D model and apply a physics-based simulation. The challenge that arises is the following: when the simulation is applied to the model, object parts which were previously occluded or facing away from the camera might become visible (see Figure 2(d)). The appearance of these areas is unknown. Therefore, there is no texture information available to render the physically modified object and insert it back into the video.

If the texture of the model was known entirely, then it could be directly manipulated using any standard 3D animation tool. However, this is not the case when combining a 2D video with a 3D physical simulation. The appearance, and thus the texture of the model, is known only partially, depending on the camera viewpoint.

A straightforward approach to infer missing texture information is inpainting. However, despite the large existing literature in the field [GKT*12, NAF*13, KWB*15], inpainting in a spatio-temporally consistent way for moving objects in a video is still an open challenge. Other naive solutions, such as creating a static texture and generating a rendering of the direct, unaltered simulation, suffer from obvious pitfalls. Static textures cannot incorporate visual changes present in the input video, such as changing lighting conditions. Another approach for inpainting the unknown texture area is to rely on special properties of the object, such as symmetry and then "mirror" the texture [KSES14]. However this does not apply to general objects (see Figure 2(c)). Generally speaking, estimating the unobserved texture of an object is a research challenge in itself [MNZ*15].

Instead of attempting to fill in the missing texture areas, we investigate and propose an alternative approach which is orthogonal to the inpainting paradigm. Our approach computes control forces for the physical simulation, which encourage that objects parts for which no texture information exists do not become visible. This constitutes the main contribution of our work.

In this paper, we focus on the manipulation of videos of a rigid object moving in front of a static background and acquired by a fixed camera (Figure 1). Our method estimates the pose of selected objects from an input image or video and represents them in 3D space using a 3D model of the objects. The user then draws from the rich set of physically based simulation algorithms to generate a physically edited version of the model, which is then transferred back into the input video (Figure 1). Our contribution is the use of an image-based visibility constraint that we append to the physics simulation to ensure that the simulated object can be appropriately textured (see Figure 2(e) and Figure 2(f)). Our solution acknowledges that the output of the simulation will be another video. Consequently, it gently guides the simulation in such a way that the object can be textured convincingly. Our framework does not depend on a particular type of simulation technique: we implement it as a wrapper around the general-purpose physics engine BULLET [Cou05]. We demonstrate the effectiveness of our method for various typical physical models: rigid motion, multi-body collisions, cloth and elastic bodies.

2. Related work

Our work is related to image/video editing, physics-based simulation and physics behavior control. Given the vastness of these fields, we focus only on work most relevant to ours.

Image and video editing. To manipulate the motion or pose of an object in an image or a video, deformation techniques are generally applied. Image deformation is often expressed as mesh deformation, such as [SMW06, ACOL00], among many others. Some techniques are guided with the aid of the 3D model of the object such as [KSES14, ZFL*10]. For video editing, mesh deformation techniques can be applied in the temporal domain such as [KWSH*13], video mesh [CPW*11], and also with the aid of the 3D model [JTST10]. Related to our texture visibility constraint, [SKPSH13] deform a mesh while avoiding inverting elements such as mesh triangles.

Appearance manipulation in images has also been investigated. For example, [YJHS12, BLDA11] transfer edits between images, or [GCZ*12, HE07] in the context of image appearance manipulation, such as scene completion, using photo collection. Image and video manipulation was also studied in the context of blending and composition [PGB03, CZSH13]. Some works manipulate objects in videos by modifying the video timeline and the video frames order [LZW*13, SSSE00].

While visually attractive and entertaining results can be obtained, these techniques are physics-blind in the sense that they edit images and videos without considering the laws of physics. This might result in manipulated objects whose motion lacks physical realism. In contrast, our physics-aware video editing approach can deal with this issue by building upon physics engines.

Simulation. Physics based simulation has continued to promote realism in the context of computer animation. It is an active area of research both in academia and industry, and many different techniques have been proposed during the last few decades (see [NMK*06] for a survey). The output of physics-based simulation is generally a virtual 3D scene or a rendered animation. A few

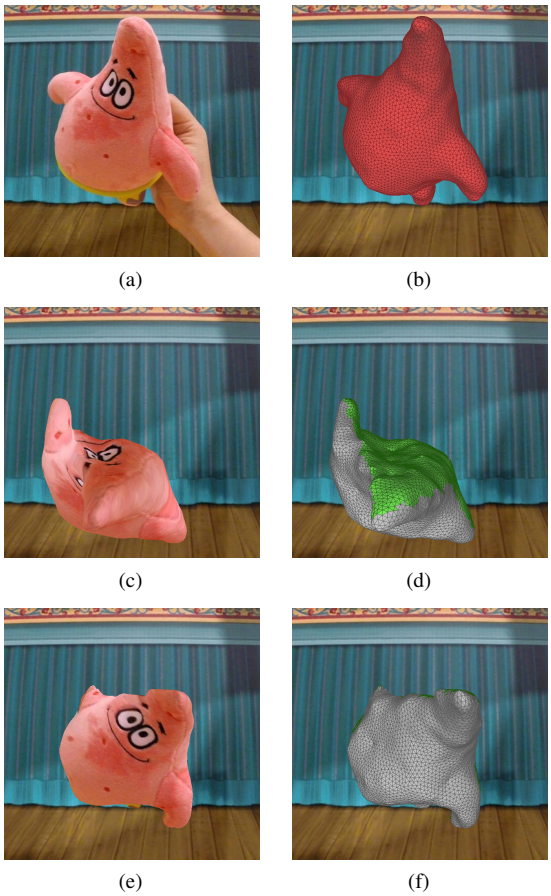


Figure 2: *Examples of video editing in combination with physics-based simulation with and without our image-based visibility constraint, in the context of puppetry (see Figure 1). (a,b): image from the input video and the aligned object mesh. Result of object simulation without (d) and with our image constraint (f), and their respective textured version (c,e). The object mesh triangles with unknown texture are displayed in green. In (c), these triangles are textured by "mirroring". Note how our approach (e,f) can recover from an extreme pose with many invisible parts, while still returning a physically plausible motion (see supplementary video).*

systems accept a video as input, for example the methods incorporating physics simulation to guide the 3D reconstruction process [SGdA*10], however they still return a synthetic scene.

A key difference with the typical physics simulation methods is that our approach returns a physically-edited version of the input real footage, rather than creating a synthetic scene. Moreover we incorporate an image-based constraint into the physics simulator. Our image-aware physics simulation is guided and controlled such that the image-based visibility constraint is encouraged. In the following we review methods which are related to this scenario.

At a high level, our constraint is similar in spirit to those arising during contact and collision handling, see for example [HVS*09]. Rather, than preventing interpenetrations of geometric models, we prevent unseen parts of the model from turning toward the camera.

Physical behavior control. There exists a large body of work on how to control motion and behavior of physics based animation. The framework of spacetime constraints introduced by Witkin [WK88] is a powerful tool that allows a user to specify motion constraints, such as "jump from here to there" for character animation. The result is a motion which best preserves the input motion, i.e., it ensures interpolation of user edits without deviating too severely from the original physics. The seminal spacetime constraints work of Witkin paved the way to several subsequent methods such as [Coh92, Gle97, BSG12, LHDJ13], among many others. Similar to standard (uncontrolled) physics-based simulation, spacetime constraints output a 3D synthetic animation rather than an edited version of a live action video.

External control forces have been used to alter trajectories [BP08], control fluids [MTPS04] and guide smoke animation [TMPS03], among many others. Our approach follows the same line, but we compute control forces to fulfill image-based constraints, rather than geometry constraints.

3. Proposed approach

The input to our method is a video of a moving rigid object captured by a static standard color camera. Our method returns an edited version of the input video in which the object follows a certain physical behavior desired by the user. To achieve this, we append image-aware external control forces in the physically based simulation to encourage proper texturing of the object (see Figure 4). In the following, we review the equations of motion, define the setting, and then introduce our energy formulation.

3.1. Equations of motion

The equations of motion of a given object with Rayleigh damping are $\mathbf{M}\ddot{\mathbf{x}} + \mathbf{C}\dot{\mathbf{x}} - \mathbf{f}_{int}(\mathbf{x}) = \mathbf{f}_{ext}(\mathbf{x}, \dot{\mathbf{x}})$ [Sha97]. The vector $\mathbf{x}$ represents the positions of the object points, and $\dot{\mathbf{x}}$ and $\ddot{\mathbf{x}}$ are the first and second derivatives of $\mathbf{x}$ with respect to time. $\mathbf{M}$ is the physical mass matrix, $\mathbf{C}$ is the Rayleigh damping matrix, $\mathbf{f}_{int}$ is the vector of internal forces, and $\mathbf{f}_{ext}$ is the vector of external forces such as gravitational forces. The equations of motion can be used to simulate rigid bodies, as well as soft bodies depending on $\mathbf{f}_{int}$.

The equations of motion may be numerically solved by discretizing them with respect to time using an integration scheme such as implicit Euler [BW98]. Then position vector $\mathbf{x}$ can be integrated over time starting from a given rest pose $\mathbf{x}_{rest}$. In this paper, we do not assume any specific integration scheme or physics engine, and consider the simulation procedure as a black box physics simulator.

3.2. Settings

The equations of motion are applied to the 3D points of the object. The 3D model of this object is represented by a triangle mesh consisting of N vertices $\hat{x}_i$ with $i = 1 \dots N$, and M triangles with corresponding centroids $\hat{c}_j$ and normals $\hat{n}_j$ with $j = 1 \dots M$. In a preprocess stage, this 3D model is aligned with the object in each frame of the input video [RRB12]. Let x_i^t, c_j^t and n_j^t be the corresponding entities $\hat{x}_i$, $\hat{c}_j$ and $\hat{n}_j$ expressed with respect to the camera at frame t. All the mesh vertices at frame t are described in $\mathbf{x}^t = \{x_1^t, \dots, x_N^t\}$.

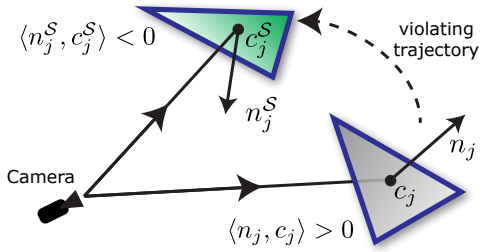


Figure 3: *Illustration of the image-based visibility constraint. The triangle (grey) of the input frame initially points away from the camera (i.e., unobserved triangle), while its transformation (green) now faces the camera (i.e., the triangle gets visible).*

As motivated above, we influence the physical behavior with the help of external control forces. These forces are added to the set of external forces $\mathbf{f}_{ext}$ for the physics simulation. The simulation starts with the rest pose $\mathbf{x}_{rest} = \mathbf{x}^1$, and in our experiments, we set the simulation timestep size to the framerate of the input video. After performing a time step at frame t, the physics simulator returns the state $\mathbf{x}^{\mathcal{S}} = \{x_1^{\mathcal{S}}, \ldots, x_N^{\mathcal{S}}\}$ of the object, where the superscript $\mathcal{S}$ emphasizes the fact that the values are computed by the simulator. From these vertex positions, triangle centroids $c_i^{\mathcal{S}}$ and triangle normals $n_i^{\mathcal{S}}$ can be computed. Each $x_i^{\mathcal{S}}$, $c_i^{\mathcal{S}}$, and $n_i^{\mathcal{S}}$ has a corresponding x_i^t, c_i^t, and n_i^t in the aligned mesh, as described above. For writing simplification, we omit the superscript t in the following.

3.3. Energy formulation

For each object vertex x_i, we compute a control force f_i such that occluded parts in the input will not become visible in the output video, while yet producing a physical behavior which appears physically plausible. We express this problem as an energy minimization over the set of free parameters f_i:

$$E = \lambda_{\mathcal{C}} E_{\mathcal{C}} + \lambda_{\mathcal{F}} E_{\mathcal{F}} + \lambda_{\mathcal{T}} E_{\mathcal{T}}, \tag{1}$$

where $E_{\mathcal{C}}$ prevents violating trajectories, $E_{\mathcal{F}}$ favors control forces of small magnitude to deviate only slightly from the original simulation and $E_{\mathcal{T}}$ encourages temporal continuity of the forces. The weights λ balance the importance of these terms. In the following paragraphs we discuss these energies in more detail.

The image-aware energy $E_{\mathcal{C}}$ prevents violating trajectories and is a key aspect of this paper. A trajectory is violated when a triangle occluded in the current input frame becomes visible at the corresponding time step of the simulation (because its texture is unknown). We define the binary visibility variable v of a triangle as 1 when it is visible, and 0 otherwise. The visibility of the triangle j in the input video and obtained by simulation are respectively written v_j and $v_j^{\mathcal{S}}$. A triangle j is considered visible when $\langle n_j, c_j \rangle < 0$, where $\langle .,. \rangle$ is the inner product operator (see Figure 3). The control force f influences the vertices position $x_j^{\mathcal{S}}$, and thus in turn $n_j^{\mathcal{S}}$, $c_j^{\mathcal{S}}$ and $v_j^{\mathcal{S}}$, and therefore can be optimized to avoid violating trajectories. Over all triangles, this is expressed as

$$E_{\mathcal{C}} = \sum_j^M \delta(v_j^{\mathcal{S}}, 1 - v_j) \left(w_j \langle n_j^{\mathcal{S}}, c_j^{\mathcal{S}} \rangle \right), \tag{2}$$

where $\delta(a,b) = 1$ when $a = b = 1$ and returns 0 otherwise. It activates the constraint for each triangle j leading to a violating trajectory, and non-offending triangles do not contribute to the cost. An offending triangle is penalized by how "visible" it is, that is proportionally to $\langle n_j^{\mathcal{S}}, c_j^{\mathcal{S}} \rangle$. The weights w are chosen based on the triangle size, where larger triangles are penalized more than smaller triangles. Each weight w_j is computed by $w_j = a_j / A$, where a_j is the area of the triangle j obtained by the simulator in the image, and A is the total surface area.

To remain faithful to the physical motion, we would like our control forces to remain small. Thus, $E_{\mathcal{F}}$ punishes large control force magnitudes:

$$E_{\mathcal{F}} = \sum_i^N \|f_i\| \tag{3}$$

where $\|.\|$ represents the l^2 norm. Finally, the energy $E_{\mathcal{T}}$ encourages temporal consistency:

$$E_{\mathcal{T}} = \sum_i^N \|f_i - \tilde{f}_i\|. \tag{4}$$

where $\tilde{f}_i$ is the force corresponding to f_i at the previous frame.

The total energy E is optimized at each frame. Because $E_{\mathcal{C}}$ depends on the black box physics engine, analytic gradients of the energy E may not be easily computed. Instead, we compute gradient by taking central differences. Then we compute f using an off-the-shelf nonlinear optimization solver (KNITRO [BNW06], see details in Section 4.1).

Rigid body. For deformable bodies, a control force is computed for each vertex. In the special case of rigid body dynamics, only a force acting on the center of mass and the torque acting on the rotational degrees of freedom needs to be computed per rigid object. This greatly reduces the amount of variables down to 6.

Constraints. In early experiments, we expressed the violating triangles as hard constraints. However the optimization often took too long to converge. A first reason is that a feasible solution is not readily available as initial solution. In contrast, when using soft constraints, there is no concept of "feasible" solution per se, and since we search for the control force vectors with low magnitude (Eq. 3), we can simply initialize them with zero. A second reason is that hard constraints generally lead to a sudden discontinuity when a constraint is activated/deactivated [SKPSH13]. Instead, we combine the soft constraint of the triangle visibility (Eq. 2) with barrier functions [SKPSH13]: the cost increases to infinity as the point approaches the boundary of the feasible region. In rare cases, it can still occur that a violating triangle becomes visible, but it is generally small, and in that case, we simply linearly interpolate the texture from the neighborhood triangles as a fail safe.

3.4. Fitting the input video

To initialize the simulation, we consider that objects start their motion from a state of rest, and thus the initial velocity of the first frame $\mathbf{x}^t$ is set to 0. In the case the object is moved by an external force (e.g., by a hand moving an object through space), the input sequence is tied to the physical simulator by specifying a subset

of the object vertices as *anchor* vertices. The user can interactively select them in our program by directly clicking on the object mesh vertices, and this set of anchor vertices is used along the video. Anchor vertices hold the object in place, i.e., given the example above, they prevent the object to *slip off* the hand and just fall down. The locations of these anchor vertices change from frame-to-frame according to their locations specified in the position vector $\mathbf{x}^t$, and are not modified by the simulator.

The non-anchor vertices of the object are the degrees of freedom whose trajectories are governed by the equations of motion, and if required, are influenced by our optimized control forces.

4. Results

As in any physics simulation, the user assigns the physical parameters of the object (e.g., deformation, stiffness, etc). The result obtained by the physics engine is displayed, and the user can adjust the physical parameters if wanted. Then our approach runs over this engine. For casual users not familiar with simulation, we also suggest some default parameters based on the intended general behavior that the user wants to create, e.g., elastic look.

We encapsulate our method over an existing physics engine. We intentionally did not write a specific simulation method and we do not assume any particular engines or simulation methods. Therefore any simulation tool can be used as black box and our method can be applied on top of these different engines. On the user side, this physics engine bootstrapping also allows the user to continue using her favorite physics engine.

In the following, we provide implementation details and then show a variety of applications of our approach. They illustrate results with different kinds of input and target edits. All these results have been created by the same implementation code of the approach described in Section 3.

4.1. Implementation details

To facilitate follow-up work, we provide on our project website all the input data such as the videos, object 3D models, physics parameters as well as the estimated object poses. In addition, we also provide source code templates as well as wrappers of our method for the popular physics engine BULLET [Cou05].

Preprocessing. In preprocessing, we estimate the 3D model pose with respect to the image by silhouette alignment [RRB12] or camera pose estimation [SSS06, LPT13]. Other object pose estimation methods can be also used. The 3D model can be known a priori (e.g., from measurements or when 3D printing) or captured in many different ways such as from RGBD cameras [IKH*11], smartphone cameras [TKM*13], from a single image [CZS*13], or from online public repositories of object 3D models (e.g., 3D Warehouse or Turbosquid) [AME*14]. We will show results with models obtained for some of these cases. Our main contribution (Section 3) does not rely on or assume a specific preprocessing technique.

The silhouette of the object is cut out from the input video, and can be filled from a given background plate to create a foreground completed video. In some of our examples, the background plate was not available (Figure 1 and Figure 6), so we compute it using a patch-based approach [XLYD11], like Chen et al. [CZS*13] in the context of single image-based object manipulation.

In practice, the object silhouette might have a few pixel inaccuracies. We found that a trivial-but-effective method is to slightly increase the model size (e.g., by 1 percent). This size change is visually unnoticeable and makes sure that the object in the input video is entirely covered.

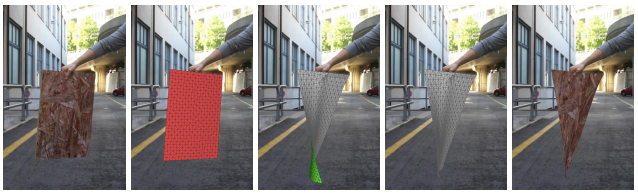
Control force estimation. To minimize the energy of Eq. 1, we use the interior-point algorithm of Waltz et al. [WMNO06] available in KNITRO [BNW06]. We set $f_i = 0$ for all vertices as initial solution since the forces are expected to have a small magnitude. We set λ to default values that could be adjusted by the user if needed. The physics simulation of the forces is performed with the physics engine BULLET that is directly called from the optimization routine. The physics engine returns the position of the mesh vertices obtained by the optimal control forces. Finally, this mesh is rendered in OpenGL with texture mapping from the input image (the position of the triangles in the input image is known) and is then displayed in the foreground completed video.

4.2. Applications for videos

We invite the readers to see the video results available in the supplemental material. The results, detailed in the following, depict different properties and cases, such as recovery of extreme pose with deformable object ("puppetry" Figure 1), smooth motion and small required rectification ("board" Figure 4), and sudden motion with rigid pieces ("fracturing" Figure 6).

Puppetry. We show an application relevant to post processing recorded sequences of puppetry as an artist is moving a puppet through the scene. Here we imitate a classic cartoon effect [TJ81], i.e., when the artist stops the motion, the puppet still moves in its deformable regions. We obtained the 3D model of the puppet by KinectFusion [IKH*11]. Figure 2 presents a result, and Figure 1 shows some frames of the resulting sequence. To visually hide to young viewers the fact that this puppet is just an object manipulated by a person, we also inpainted the artist's hand with the computed background plate. The result for the full sequence is provided in the supplemental video.

Board sequence. We present an additional example where we turn a flat, rigid board observed in a given input video into a piece of cloth (see Figure 4). The 3D model of the board is simply obtained by defining a box with the real measurements of the board. The user chooses anchor points on the parts which should not be affected by the physical simulator. In this example, the anchor points are the board vertices where the board part is held by the hand (otherwise the simulated object will just fall down). As in any physics simulation tool, the user specifies the physical parameters of her choice, such as material properties, to mimic a cloth effect. The optimization is then automatically performed, and our program generates an output video showing the rigid board moving like a cloth with a physically plausible motion. This is an effect which would require a tremendous amount of user interaction in existing video editing tools. Also note that the cloth moves in agreement with the board motion. Constraints on normals are implicitly enforced to prevent



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

Figure 4: *(a): frame from the input video, (b): model pose alignment, (c): standard unconstrained simulation, (d): our image-aware physics simulation, (e): our textured result from (d).*

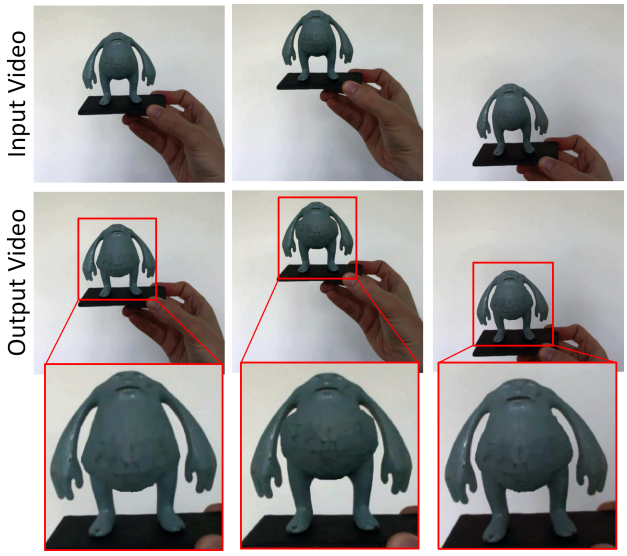


Figure 5: *Editing the stiffness properties of the belly of a 3D printed figurine.*

the board from flipping to its backside via Eq. 2, as those parts are not visible in any frame of the input sequence.

3D printed characters alive. In our last application use case, we modify a video of a 3D printed figurine. By virtually changing its material properties, we can turn it into an "alive character", quoting Coros et al. [CMT*12]. The 3D model is obtained from the online public repository Turbosquid and we use it for 3D printing. Our program allows a wide range of deformation, such as changing the stiffness of the belly (see Figure 5). This visual effect is entertaining since regular 3D printed objects are inherently rigid and look inert, and they suddenly appear as alive in the output video.

4.3. Applications for single image

Our method can also be naturally applied on single pictures. We show here an example of object fracturing from a single picture. The object model is first decomposed into a set of rigid pieces by a standard fracturing tool (Shatter tool in Maya). These pieces are sent to a rigid body simulator with multi-body collision BULLET. The challenge is to let each of these pieces fall in a way to not violate the image constraint. The constraints are applied to the triangles corresponding to the surface of the object model and their

Figure 6: *Object fracturing from a single image. Input picture of a 3D printed object (very left) and some snapshots of the output fracturing video obtained by our approach.*

	"more" appealing	"same"	"less" appealing
small required rectification	22%	78%	0%
middle required rectification	57%	43%	0%
large required rectification	85%	15%	0%

Table 1: *User study results of our approach compared to the unconstrained method for videos with different amounts of required rectification. See text for details.*

texture is taken from the input picture. Triangles which belong to the inside of the object are not considered in the optimization and we assign them a uniform texture. We apply our method on the same model as the belly stiffness experiment (Figure 5) to demonstrate that our method can be applied to a same object in different contexts, i.e., non-rigid deformation with a single object and rigid body simulation with multiple objects. Figure 6 shows a representative result where a single picture of the 3D printed object is automatically converted into a fracturing video.

4.4. Evaluation

User study. We conducted an evaluation with a user study. We asked participants to compare the textured video results obtained by the unconstrained simulation and our proposed method. We recruited 13 participants, and each participant watched 9 pairs of videos (unconstrained and ours). For each pair, the participants were asked to choose the more appealing video. In the case the participants could not see a difference between the two videos, they had the option to choose "same" quality.

The 9 videos were classified into 3 main groups (3 videos per group) based on the number of triangles becoming visible during the (unconstrained) physics-based simulation, i.e. the amount of needed rectification: small (e.g., "board" Figure 4), middle (e.g., "fracturing" Figure 6) and large (e.g., "puppetry" Figure 1).

The results are contained in Table 1. "more", "same" and "less" respectively refer to when our results were rated more, same or less appealing than the unconstrained result. For example, for the group where a large number of triangles becomes visible (large required rectification), the visual appeal was rated "same" in 15%, and our results were preferred in 85% of the experiments. Table 1 also shows that our method is more preferred when the amount of required rectification becomes larger. This is because the texturing artifacts become more visible when more violating triangles are not handled properly by the unconstrained approach. On the other end of the rectification spectrum, Table 1 also indicates that when the

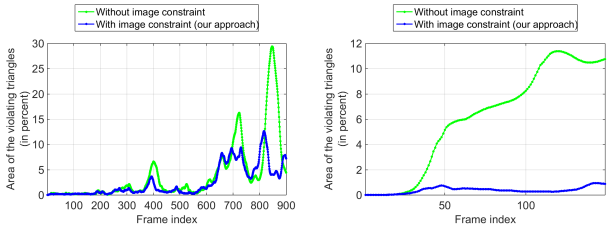


Figure 7: *Comparison of the area of the violating triangles (i.e., the triangles that are occluded in the input frames but become visible) obtained without and with the image-based visibility constraint. The area is computed in percent of the total visible area. Left: result for the puppetry sequence (Figure 1). Right: result for the fracturing sequence (Figure 6) on the linear force.*

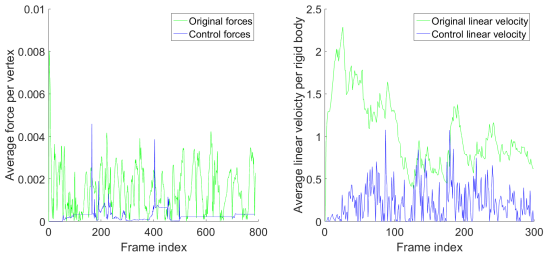


Figure 8: *Comparison of how strong the control forces were in comparison to the original forces. Each data point corresponds to the magnitude of the respective global force vector (Figure 4). Left: result for board sequence. Right: result for fracturing sequence (Figure 6).*

amount of required rectification is small, the visual appeal is often rated "same" (78% of the experiments). It shows that when the number of triangles to be fixed is small, our method appropriately modifies the physics only slightly, and thus the results look similar.

Quantitative evaluation. We compare the screen space error of the violating triangles, i.e., the triangles which should not become visible, for the results obtained by the image-blind (unconstrained) approach and our approach. For each image, we define the error as the percentage of the object area corrupted by violating triangles in the image, i.e., the area of the violating triangles occupied in the image over the total area of the object (all the visible triangles) in the image. Figure 7 shows the resulting errors for the puppetry and the fracturing sequences. In the initial frames when the object does not move a lot, both approaches yield a similar error. However, the error and hence the visual artifacts of the unconstrained approach highly increases along the frames, contrary to our approach. For the fracturing sequence composed of several object pieces and without anchor points, this behavior is even more amplified and our approach significantly outperforms the unconstrained approach.

For occasional frames of the puppetry sequence, the error of our approach is temporarily larger than for the unconstrained approach. We believe that it is simply due to the fact that, considering the force magnitude and temporal consistency terms (Eq. 3 and Eq. 4), the global energy at Eq. 1 allows a bit higher error for occasional frames (e.g., frame index 800) since it then permits to obtain a much lower error over many frames (e.g., frame index 820-880) later in time.

We further compare, how strong the control forces were in comparison to the original forces. Figure 8 shows the forces for the board and the fracturing sequences. Each datapoint corresponds to the magnitude of the global force vector. Overall, the original, i.e., the physical forces dominate over the control forces. While the latter forces are zero for most vertices, they get activated when triangles may transition into an invalid state.

On the whole, these experiment demonstrate that our approach can successfully drive the physics simulation so that the object parts occluded in the input video do not become visible during the simulation, which in turn provides more visually appealing results.

4.5. Future Extensions

Our current system runs a physically based simulation on triangle meshes, and yields plausible physically manipulated videos for a variety of physical interactions, such as rigid motions, cloths and elastic bodies. The proposed formulation could also be naturally extended for tetrahedral meshes, where only the triangles on the object surface are considered for the visibility constraint.

Our method can successfully handle challenging situations in terms of triangles becoming visible (see Figure 2). However, for complex cases where the motion is too extreme, preventing unobserved triangles to become visible can result in unnatural physical behavior, which in turn might lead to visually unpleasing results (see bouncing ball in the supplementary video).

In our setup, solving for the external control forces is rather computationally involved, in the order of about 10 seconds per frame. The main reason is that the gradient of the energy defined in Eq. 1 cannot be expressed analytically. Thus each individual energy partial derivative is computed via central differences. This is the cost to ensure generality, not assume any particular engines or simulation methods, and allow the user to use her favorite physics engine. If execution time is an issue, a potential approach is to perform optimization on a relatively low-resolution mesh that could be then upsampled. Another potential strategy is a coarse-to-fine optimization scheme with a multi-resolution mesh, i.e., the optimized control forces obtained at a certain mesh resolution level is used as initial solution for optimization on a finer resolution of the mesh. An alternative optimization approach is to conduct random sampling over the control forces parameters [BB12, TJ07].

Our current preprocessing is dedicated to rigid objects. Tracking of deformable objects is outside the scope of this paper. Extension of our approach to such models, e.g. with the aid of non-rigid alignment techniques [NFS15, SSP07], is an interesting direction for future work.

4.6. Failure cases and limitations

As discussed, the energy is optimized at each frame (Section 3.3). A failure scenario is when an object becomes rapidly unoccluded, and therefore large forces need to be suddenly exerted. If different control forces had been applied earlier in time, this could have led

to a different, eventually better, configuration. Such configuration could be obtained by performing the optimization over the entire sequence, rather than per frame. However this is computationally expensive due to the high number of accumulated DOFs. In practice we just optimized over a temporal window of 10 frames, which is also used for temporal consistency (Eq. 4).

More thorough analysis of our external force magnitudes could lead to an improved solution or user interface. For example, we could vary the temporal consistency and optimization window depending on the maximum external force. This could help diffuse forces at the cost of computation time. Alternative, the external force magnitudes could be interpreted as a measure of *how non-physical* the constrained simulation is. This information could guide automatic optimization of physical parameters (e.g., stiffness) or notify a user that a parameter choice has lead to an implausible result. This remains for future work; for now, we use the difference between the number of violating triangles in the unconstrained and constrained simulation (see Figure 7) as an approximate measure of *non-physicalness*.

As discussed, our approach assumes a complete 3D model of the object. Running physics based simulation and our optimization with an incomplete 3D model, which may be generated from the input video directly [SSS06, TKM*13], is subject to future work.

A triangle could be occluded in one frame, but may be visible in an earlier or later frame. Hence, the visible texture of this frame *could* be reused. However, we observed that this quickly introduces shading inconsistencies. Instead, our approach deliberately considers the triangle visibility only in the current frame. An interesting direction for future work is to analyze and exploit shading consistency throughout the image sequence.

5. Conclusion

In this paper we present a method which allows editing the content of an image or a video, while leveraging modern physically based simulation techniques. This is achieved by computing a set of external control forces to enforce an image-based visibility constraint which in turn ensures proper texturing. We have demonstrated our framework to various examples, such as clothes, soft bodies, and rigid bodies. An exciting future direction is the investigation of tracking and image constraints for very different physical phenomena such as fluids.

Acknowledgments

We thank Sebastian Martin and Kaan Yücer for insightful discussions. This research, which is carried out in part at BeingThere Centre, is supported in part by the Singapore National Research Foundation under its International Research Centre@Singapore Funding Initiative and administered by the IDM Programme Office.

References

[ACOL00] ALEXA M., COHEN-OR D., LEVIN D.: As-rigid-as-possible shape interpolation. In *SIGGRAPH* (2000). 2

[AME*14] AUBRY M., MATURANA D., EFROS A. A., RUSSELL B. C., SIVIC J.: Seeing 3D chairs: Exemplar part-based 2D-3D alignment using a large dataset of CAD models. In *CVPR* (2014). 5

[BB12] BERGSTRA J., BENGIO Y.: Random search for hyper-parameter optimization. Journal of Machine Learning Research (2012). 7

[BLDA11] BERTHOUZOZ F., LI W., DONTCHEVA M., AGRAWALA M.: A framework for content-adaptive photo manipulation macros: Application to face, landscape, and global manipulations. TOG (2011). 2

[BNW06] BYRD R. H., NOCEDAL J., WALTZ R.: KNITRO: An integrated package for nonlinear optimization. In *Large-Scale Nonlinear Optimization* (2006). 4, 5

[BP08] BARBIČ J., POPOVIĆ J.: Real-time control of physically based simulations using gentle forces. TOG (SIGGRAPH Asia) (2008). 3

[BSG12] BARBIČ J., SIN F., GRINSPUN E.: Interactive editing of deformable simulations. TOG (SIGGRAPH) (2012). 3

[BW98] BARAFF D., WITKIN A.: Large steps in cloth simulation. In *SIGGRAPH* (1998). 3

[CMT*12] COROS S., MARTIN S., THOMASZEWSKI B., SCHUMACHER C., SUMNER R. W., GROSS M.: Deformable objects alive! TOG (SIGGRAPH) (2012). 6

[Coh92] COHEN M. F.: Interactive spacetime control for animation. SIGGRAPH (1992). 3

[Cou05] COUMANS E.: The Bullet physics library. http://www.bulletphysics.org, 2005. 2, 5

[CPW*11] CHEN J., PARIS S., WANG J., MATUSIK W., COHEN M., DURAND F.: The video mesh: A data structure for image-based three-dimensional video editing. In *ICCP* (2011). 2

[CZS*13] CHEN T., ZHU Z., SHAMIR A., HU S.-M., COHEN-OR D.: 3-sweep: Extracting editable objects from a single photo. TOG (2013). 2, 5

[CZSH13] CHEN T., ZHU J., SHAMIR A., HU S.: Motion-aware gradient domain video composition. TIP (2013). 2

[GCZ*12] GOLDBERG C., CHEN T., ZHANG F., SHAMIR A., HU S.: Data-driven object manipulation in images. CGF (Eurographics) (2012). 2

[GKT*12] GRANADOS M., KIM K. I., TOMPKIN J., KAUTZ J., THEOBALT C.: Background inpainting for videos with dynamic objects and a free-moving camera. In *ECCV* (2012). 2

[Gle97] GLEICHER M.: Motion editing with spacetime constraints. In *I3D* (1997). 3

[HE07] HAYS J., EFROS A. A.: Scene completion using millions of photographs. TOG (SIGGRAPH) (2007). 2

[HVS*09] HARMON D., VOUGA E., SMITH B., TAMSTORF R., GRINSPUN E.: Asynchronous contact mechanics. TOG (SIGGRAPH) (2009). 3

[HWK15] HUANG Q., WANG H., KOLTUN V.: Single-view reconstruction via joint analysis of image and shape collections. TOG (SIGGRAPH) (2015). 2

[IKH*11] IZADI S., KIM D., HILLIGES O., MOLYNEAUX D., NEWCOMBE R., KOHLI P., SHOTTON J., HODGES S., FREEMAN D., DAVISON A., ET AL.: KinectFusion: real-time 3D reconstruction and interaction using a moving depth camera. UIST (2011). 5

[JTST10] JAIN A., THORMÄHLEN T., SEIDEL H.-P., THEOBALT C.: MovieReshape: tracking and reshaping of humans in videos. TOG (SIGGRAPH Asia) (2010). 2

[KSES14] KHOLGADE N., SIMON T., EFROS A. A., SHEIKH Y.: 3D object manipulation in a single photograph using stock 3D models. TOG (SIGGRAPH) (2014). 2

[KWB*15] KLOSE F., WANG O., BAZIN J.-C., MAGNOR M. A., SORKINE-HORNUNG A.: Sampling based scene-space video processing. TOG (SIGGRAPH) (2015). 2

[KWSH*13] KAUFMANN P., WANG O., SORKINE-HORNUNG A., SORKINE-HORNUNG O., SMOLIC A., GROSS M.: Finite element image warping. CGF (Eurographics) (2013). 2

[LHDJ13] LI S., HUANG J., DESBRUN M., JIN X.: Interactive elastic motion editing through space-time position constraints. Computer Animation and Virtual Worlds (CASA) (2013). 3

[LPT13] LIM J. J., PIRSIAVASH H., TORRALBA A.: Parsing IKEA objects: Fine pose estimation. In *ICCV* (2013). 5

[LZW*13] LU S., ZHANG S., WEI J., HU S., MARTIN R. R.: Timeline editing of objects in video. TVCG (2013). 2

[MNZ*15] MAGNENAT S., NGO D. T., ZÜND F., RYFFEL M., NORIS G., ROETHLIN G., MARRA A., NITTI M., FUA P., GROSS M., SUMNER R. W.: Live texturing of augmented reality characters from colored drawings. TVCG (2015). 2

[MTPS04] MCNAMARA A., TREUILLE A., POPOVIĆ Z., STAM J.: Fluid control using the adjoint method. TOG (SIGGRAPH) (2004). 3

[NAF*13] NEWSON A., ALMANSA A., FRADET M., GOUSSEAU Y., PÉREZ P.: Towards fast, generic video inpainting. In *Proceedings of European Conference on Visual Media Production (CVMP)* (2013). 2

[NFS15] NEWCOMBE R. A., FOX D., SEITZ S. M.: DynamicFusion: reconstruction and tracking of non-rigid scenes in real-time. In *CVPR* (2015). 7

[NMK*06] NEALEN A., MUELLER M., KEISER R., BOXERMAN E., CARLSON M.: Physically based deformable models in computer graphics. CGF (2006). 2

[PGB03] PÉREZ P., GANGNET M., BLAKE A.: Poisson image editing. TOG (SIGGRAPH) (2003). 2

[PR12] PRISACARIU V., REID I.: PWP3D: Real-time segmentation and tracking of 3D objects. IJCV (2012). 2

[RRB12] REINBACHER C., RÜTHER M., BISCHOF H.: Fast pose estimation from silhouettes in video sequences. In *Computer Vision Winter Workshop* (2012). 3, 5

[SGdA*10] STOLL C., GALL J., DE AGUIAR E., THRUN S., THEOBALT C.: Video-based reconstruction of animatable human characters. TOG (SIGGRAPH Asia) (2010). 3

[Sha97] SHABANA A.: *Vibration of Discrete and Continuous Systems*. Mechanical Engineering Series. Springer, 1997. 3

[SKPSH13] SCHÜLLER C., KAVAN L., PANOZZO D., SORKINE-HORNUNG O.: Locally injective mappings. CGF (SGP) (2013). 2, 4

[SMW06] SCHAEFER S., MCPHAIL T., WARREN J.: Image deformation using moving least squares. TOG (SIGGRAPH) (2006). 2

[SSP07] SUMNER R. W., SCHMID J., PAULY M.: Embedded deformation for shape manipulation. TOG (SIGGRAPH) (2007). 7

[SSS06] SNAVELY N., SEITZ S. M., SZELISKI R.: Photo tourism: Exploring photo collections in 3D. TOG (SIGGRAPH) (2006). 2, 5, 8

[SSSE00] SCHÖDL A., SZELISKI R., SALESIN D., ESSA I. A.: Video textures. In *SIGGRAPH* (2000). 2

[TJ81] THOMAS F., JOHNSTON O.: *The Illusion of Life: Disney Animation*. Hyperion, New York, 1981. 5

[TJ07] TWIGG C. D., JAMES D. L.: Many-worlds browsing for control of multibody dynamics. TOG (SIGGRAPH) (2007). 7

[TKM*13] TANSKANEN P., KOLEV K., MEIER L., PAULSEN F. C., SAURER O., POLLEFEYS M.: Live metric 3D reconstruction on mobile phones. In *ICCV* (2013). 2, 5, 8

[TMPS03] TREUILLE A., MCNAMARA A., POPOVIĆ Z., STAM J.: Keyframe control of smoke simulations. TOG (SIGGRAPH) (2003). 3

[WK88] WITKIN A., KASS M.: Spacetime constraints. In *SIGGRAPH* (1988). 3

[WMNO06] WALTZ R. A., MORALES J. L., NOCEDAL J., ORBAN D.: An interior algorithm for nonlinear optimization that combines line search and trust region steps. Math. Program. (2006). 5

[XLYD11] XIAO C., LIU M., YONGWEI N., DONG Z.: Fast exact nearest patch matching for patch-based image editing and processing. TVCG (2011). 5

[YJHS12] YÜCER K., JACOBSON A., HORNUNG A., SORKINE O.: Transfusive image manipulation. TOG (SIGGRAPH Asia) (2012). 2

[ZFL*10] ZHOU S., FU H., LIU L., COHEN-OR D., HAN X.: Parametric reshaping of human bodies in images. TOG (SIGGRAPH) (2010). 2