Shape Animation with Combined Captured and Simulated Dynamics

Benjamin Allain, Li Wang, Jean-Sébastien Franco, Franck Hetroy-Wheeler, Edmond Boyer

Shape Animation with Combined Captured and Simulated Dynamics

Benjamin Allain, Li Wang, Jean-Sébastien Franco, Franck Hetroy-Wheeler, and Edmond Boyer
LJK-INRIA Grenoble Rhône-Alpes, France

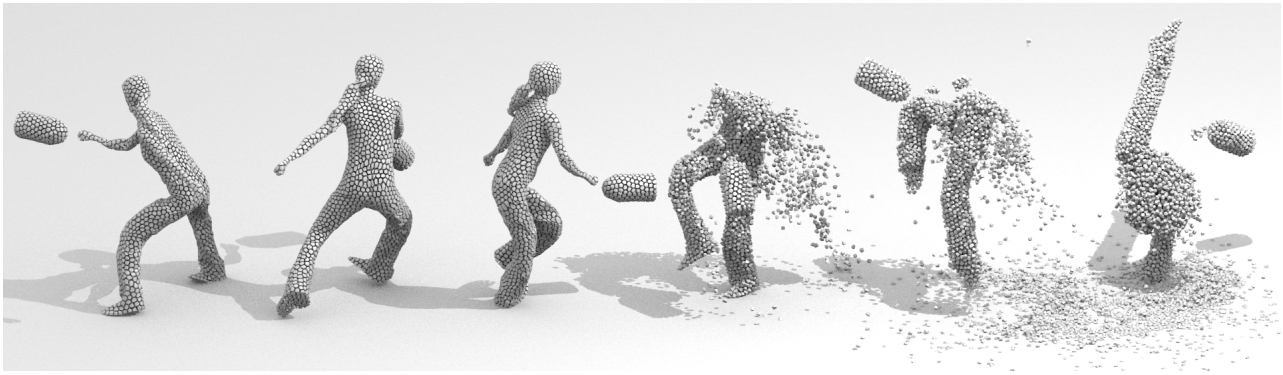


Figure 1: *An animation that combines video-based shape motion (left) and physical simulation (right). Our method allows to apply mechanical effects on captured dynamic shapes and generates therefore plausible animations with real dynamics.*

Abstract

We present a novel volumetric animation generation framework to create new types of animations from raw 3D surface or point cloud sequence of captured real performances. The framework considers as input time incoherent 3D observations of a moving shape, and is thus particularly suitable for the output of performance capture platforms. In our system, a suitable virtual representation of the actor is built from real captures that allows seamless combination and simulation with virtual external forces and objects, in which the original captured actor can be reshaped, disassembled or reassembled from user-specified virtual physics. Instead of using the dominant surface-based geometric representation of the capture, which is less suitable for volumetric effects, our pipeline exploits Centroidal Voronoi tessellation decompositions as unified volumetric representation of the real captured actor, which we show can be used seamlessly as a building block for all processing stages, from capture and tracking to virtual physic simulation. The representation makes no human specific assumption and can be used to capture and resimulate the actor with props or other moving scenery elements. We demonstrate the potential of this pipeline for virtual reanimation of a real captured event with various unprecedented volumetric visual effects, such as volumetric distortion, erosion, morphing, gravity pull, or collisions.

1. Introduction

Creation of animated content has become of major interest for many applications, notably in the entertainment industry, where the ability to produce animated virtual characters is central to video games and special effects. Plausibility of the animations is a significant concern for such productions, as they are critical to the immersion and perception of the audience. Because of the inherent difficulty and necessary time required to produce such plausible animations from scratch, motion capture technologies are now extensively used to obtain kinematic motion data as a basis to produce the animations, and are now standard in the industry.

However, motion capture is usually only the first stage in a complicated process, before the final animation can be obtained. The task requires large amounts of manual labor to rig the kinematic data to a surface model, correct and customize the animation, and produce the specifics of the desired effect. This is why, in recent times, video-based 3D performance capture technologies are gaining more and more attention, as they can be used to directly produce 3D surface animations with more automation, and to circumvent many intermediate stages in this process. They also make it possible to automatically acquire complex scenes, shapes and interactions between characters that may not be possible with the standard sparse-marker capture technologies. Still, the problem of customizing the surface animations produced by such technologies to yield a modified animation or a particular effect has currently no general and widespread solution, as it is a lower level representation to begin with.

In this work, we propose a novel system towards this goal, which produces animations from a stream of 3D observations acquired with a video-based capture system. The system provides a framework to push the automation of animation generation to a new level, dealing with the capture, shape tracking, and animation generation from end-to-end with a unified representation and solution. In particular, we entirely circumvent the need for kinematic rigging and present results in this report obtained without any manual surface correction.

Although the framework opens many effect possibilities, for the purpose of the demonstration here, we focus our effort on combining the real raw surface data captured with physics and procedural animation, in particular using a physics-based engine. To this aim, we propose to use regular Voronoi tessellations to decompose acquired shapes into volumetric cells, as a dense volume representation upon which physical constraints are easily combined with the captured motion constraints. Hence, shape motions can be perturbed with various effects in the animation, through forces or procedural decisions applied on volumetric cells. Motion constraints are obtained from captured multi-view sequences of live actions. We do not consider skeletal or surfacic motion models for that purpose but directly track volumetric cells instead. This ensures high flexibility in both the class of shapes and the class of physical constraints that can be accounted for. We have evaluated our method with various actor performances and effects. We provide both quantitative results for the shape tessellation approach and qualitative results for the generated 3D content. They demonstrate that convincing and, to the best of our knowledge, unprecedented animations can be obtained using video-based captured shape models.

In summary, this work considers video-based animation and takes the field a step further by allowing for physics-based or procedural animation effects. The core innovation that permits the combination of real and simulated dynamics lies in the volumetric shape representation we propose. The associated tessellated volumetric cells can be both tracked and physically perturbed hence enabling new computer animations.

2. Related work

This work deals with the combination of simulated and captured shape motion data. As mentioned earlier, this has already been explored with marker based mocap data as kinematic constraints. Following the work of [Popović and Witkin, 1999], a number of researchers have investigated such combination. They propose methods where mocap data can be used either as reference motion [Popović and Witkin, 1999, Zordan and Hodgins, 2002, Sulejmanpasić and Popović, 2005], or to constrain the physics-based optimization associated to the simulation with human-like motion styles [Liu et al., 2005, Safonova et al., 2004, Ye and Liu, 2008, Wei et al., 2011]. Although sharing conceptual similarities with these methods, our work differs substantially. Since video-based animations already provide natural animations, our primary objective is not to constrain a physical model with captured kinematic constraints but rather to enhance captured animations with user-specified animation constraints based on physics or procedural effects. Consequently, our simulations are not based on biomechanical models but on dynamic simulations of mechanical effects. Nevertheless, our research draws inspiration from these works.

With the aim to create new animations using recorded video-based animations, some works consider the concatenation of elementary segments of animations, *e.g.* [Casas et al., 2013], the local deformation of a given animation, *e.g.* [Cashman and Hormann, 2012], or the transfer of a deformation between captured surfaces, *e.g.* [Sumner and Popović, 2004]. While we also aim at generating new animations, we tackle a different issue in this research with the perturbation of recorded animations according to simulated effects.

Our method builds on results obtained in video-based animations with multi-camera setups, to obtain the input data of our system. Classically, multi-view silhouettes can be used to build free viewpoint videos using visual hulls [Matusik et al., 2000, Gross et al., 2003] or to fit a synthetic body model [Carranza et al., 2003]. Visual quality of reconstructed shape models can be improved by considering photometric information [Starck and Hilton, 2007, Tung et al., 2009] and also by using laser scanned models as templates that are deformed and tracked over temporal sequences [de Aguiar et al., 2007, Vlasic et al., 2008, de Aguiar et al., 2008]. Interestingly, these shape tracking strategies provide temporally coherent 3D models that carry therefore motion information. In addition to geometric and photometric information, considering shading cues allows to recover finer scale surface details as in [Vlasic et al., 2009, Wu et al., 2012] .

More recent approaches have proposed to recover both shapes and motions. They follow various directions depending on the prior information assumed for shapes and their deformations. For instance in the case of human motion, a body of work assumes articulated motions that can be represented by the poses of skeleton based models, *e.g.* [Vlasic et al., 2008, Gall et al., 2009, Straka et al., 2012]. We base our system on a different class of techniques aiming at more general scenarios, with less constrained motion models simply based on locally rigid assumptions in the shape volume [Allain et al., 2015]. This has the advantage that a larger class of shapes and deformations can be considered, in particular motions of humans with loose clothes or props. The technique also has the significant advantage that it allows to track dense volumetric cell decompositions of objects, thereby allowing for consistent captured motion information to be associated and propagated with each cell in the volume, a key property to build our animation generation framework on.

In the following, we provide a system overview, followed by a detailed explanation of how we tessellate 3D input observations into regular polyhedral cells, to be subsequently tracked and used as primary animation entity (§4.). In order to recover kinematic constraints from real actions, our system then tracks polyhedral cells using surface observations and a locally rigid deformation model (§5.). Finally, a physics or procedural simulation integrates the animation constraints over the shape (§6.). To our knowledge, this is the first attempt to propose such an end-to-end system and framework to generate animations from real captured dynamic shapes.

3. System Overview

We generate a physically plausible animation given a sequence of 3D shape observations as well as user specifications for the desired effect to be applied on the animation. 3D observations are transformed into temporally consistent volumetric models using centroidal Voronoi tessellations and shape tracking. Kinematic and physical constraints are then combined using rigid body physics simulation. The approach involves the following main steps depicted in Figure 2.

Video based acquisition Input to our system are 3D observations of a dynamic scene. Traditional and probably most common dynamic scenes in graphics are composed of human movements; however our system can consider a larger class of shapes since only local rigidity is assumed to get temporally consistent shape models. 3D observations are assumed to be obtained using a multi-camera system and can be in any explicit, *e.g.* meshes or implicit, *e.g.* from point clouds through Poisson function, forms. Our own apparatus is composed of 68 calibrated and synchronized cameras with a resolution up to 2048×2048 pixels. The acquisition space is about $8m \times 4m$ and the camera frame rate can go up to 50 fps at full resolution. The outputs of this step are point clouds with around $100k$ points.

Volumetric Representation. Input 3D observations are tessellated into polyhedral cells. This volumetric representation is motivated by two aspects of our animation goal: first, the rep-

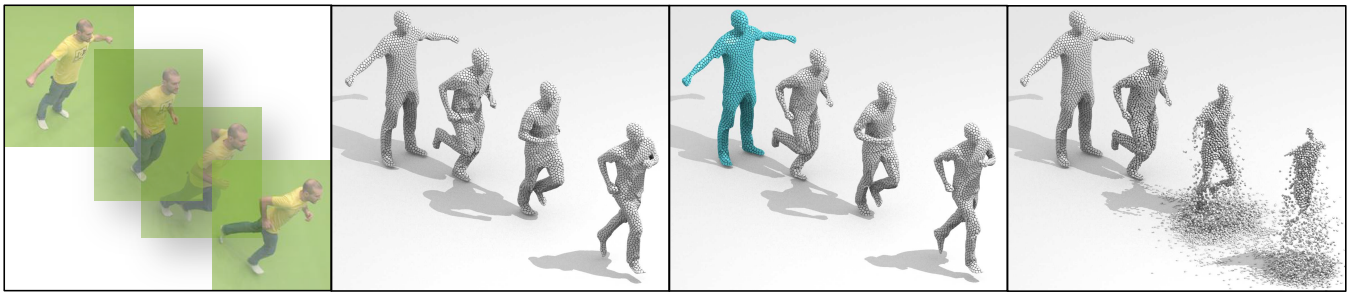


Figure 2: *From video-based shape capture to physic simulation. The approach uses multiple videos and Voronoi tessellations to capture the volumetric kinematic of a shape motion which can then be reanimated with additional mechanical effects, for instance volumetric erosion with gravity in the figure.*

resentation is well suited to physical simulation; second, volumetric deformation models are more flexible than skeleton based models, hence enabling non rigid shape deformations. Still, they allow for locally rigid volumetric constraints, a missing feature with surface deformation models when representing shapes that are volumes. We adopt centroidal Voronoi tessellations that produce regular and uniform polyhedral cells. Several methods have been suggested to clip a Voronoi tessellation to a given surface, *e.g.* [Yan et al., 2013, Lévy, 2014]. However, to the best of our knowledge, none is able to handle point clouds without explicit neighboring information. In section §4. we present a novel clipping method to compute CVT given an indicator function that identifies the two regions inside and outside the shape considered. Such an indicator function can be defined by an implicit function over a point cloud, *e.g.* a Poisson function, or by an explicit form, *e.g.* a mesh.

Tracking. In this step, incoherent volumetric shape models of a temporal sequence are transformed into coherent representations where a single shape model is evolving over time. This provides kinematic information at the cell level that will further be used in the simulation. We use a tracking method [Allain et al., 2015] that finds the poses of a given template shape at each frame. The template shape is taken as one of the volumetric models at a frame. The approach uses a volumetric deformation model, instead of surface or skeleton based model, to track shapes. It optimizes the pose of the template shape cells so as to minimize a distance cost to an input shape model while enforcing rigidity constraints on the local cell configurations.

Simulation. The tracked cell representation is both suitable for tracking and convenient for solid based physics. We embed the tracked volumetric model in a physical simulation, by considering each cell to be a rigid solid object in mechanical interaction with other cells and scene objects. We ensure cohesion of cells by attaching a kinematic recall force in the simulation, and offer various controls as to how the scene may deform, collide, or rupture during contacts and collisions. This simple framework allows for a number of interesting effects demonstrated in §7..

4. Volumetric Shape Modeling

In order to perturb captured moving shapes with simulated mechanical effects, we resort to volumetric discretizations. They enable combined kinematic and physical constraints to be be applied over cells using rigid body simulations. To this goal, we partition shapes into volumetric cells using Voronoi tessellations. Ideally, cells should be regular and uniform to ease the implementation of local constraints such as physical constraints for simulation or local deformation constraints when tracking shapes over time sequences. Volumetric voxel grids [Lorensen and Cline, 1987, Ju et al., 2002], while efficient, are biased towards the grid axes and can therefore produce tessellations with poor quality. Other solutions such as Delaunay tetrahedrizations, *e.g.* [Shewchuk, 1998, Jamin et al., 2014], can be considered however they can present badly shaped cells such as slivers. Moreover, they can not always guarantee a correct topology for the output mesh since the boundary of a tetrahedral structure can always present non manifold parts. In this work, we consider Centroidal Voronoi tessellations (CVTs) to model shapes and their evolutions. Resulting cells in CVTs are known to be uniform, compact, regular and isotropic [Du et al., 1999] . In the following, we explain how to build CVT representations given the indicator function of a 3D shape.

4.1. Mathematical Background

Given a finite set of n points $X = \{x_i\}_{i=1}^n$, called *sites*, in a 3-dimensional Euclidean space $\mathbb{E}^3$, the *Voronoi cell* or *Voronoi region* Ω_i [Okabe et al., 2000] of x_i is defined as follows:

$$\Omega_i = \{x \in \mathbb{E}^3 \mid \|x - x_i\| \leq \|x - x_j\|,\ \forall j \neq i\}.$$

The partition of $\mathbb{E}^3$ into Voronoi cells is called a *Voronoi tessellation*.

A *clipped Voronoi tessellation* [Yan et al., 2013] is the intersection between the Voronoi tessellation and a volume Ω, bounded by the surface S. A *clipped Voronoi cell* is thus defined as:

$$\Omega_i = \{x \in \Omega \mid \|x - x_i\| \leq \|x - x_j\|,\ \forall j \neq i\}.$$

A *centroidal Voronoi tessellation* (CVT) [Du et al., 1999] is a special type of clipped Voronoi tessellation where the site of each Voronoi cell is also its centre of mass. Let the clipped Voronoi cell Ω_i be endowed with a density function ρ such that $\rho(x) > 0\ \forall x \in \Omega_i$. The centre of mass x_i, also called the centroid, of Ω_i is then defined as follows:

$$x_i = \frac{\int_{\Omega_i} \rho(x) x \, \mathrm{d}\sigma}{\int_{\Omega_i} \rho(x) \, \mathrm{d}\sigma},$$

where $\mathrm{d}\sigma$ is the area differential.

CVTs are widely used to discretize 2D or 3D regions. In this respect, CVTs are optimal quantisers that minimise a distortion or quantization error defined as:

$$E(X) = \sum_{i=1}^{n} F_i(X) = \sum_{i=1}^{n} \int_{\Omega_i} \rho(x) \|x - x_i\|^2 \, \mathrm{d}\sigma. \qquad (1)$$

CVTs correspond to local minima of the above function E, also called the CVT energy function [Du et al., 1999].

4.2. Algorithm

Our CVT computation algorithm shares the same pipeline as other CVT computation methods, except it takes as input a shape Ω whose boundary surface S is not necessarily explicitly known. In our case, S can be defined explicitly as a mesh, or implicitly over a point cloud with a function that can be either given or estimated, *e.g.* a Poisson function. From a prescribed number n of sites, our algorithm consists of the following three main steps:

1. Initialization: find initial positions for the n sites inside S.
2. Clipping: compute the Voronoi tessellation of the sites, then restrict it to Ω by computing its intersection with S.
3. Optimization: update the position of the sites by minimizing the CVT energy function.

Steps 2 and 3 are iterated several times. The number of iterations is a user-defined parameter.

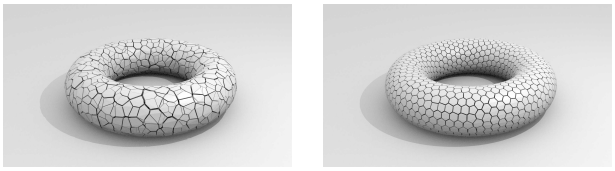
1. Initialization Any initialization can be applied in our framework. In our experiments, we have randomly positioned the sites inside S. These experiments show that such initialization is sufficient to generate better representations than voxel based or Delaunay techniques (see Figure 4 and Table 1).

2. Clipping We have designed an efficient algorithm to compute the clipped Voronoi tessellation of a volumetric shape V bounded by an implicit surface S. Given a 3D Voronoi tessellation $\bigcup_i \Omega_i$ with sites $\{x_i\}$ inside V, the algorithm proceeds as follows:

1. Identify the boundary Voronoi cells Ω_i which intersect the implicit surface S.
2. For each of these cells,
   (a) Compute its intersection with S. This intersection is represented by a set of points P_i obtained by discretizing the facets of Ω_i, and its edges, and identifying the resulting discretized cells that intersect S.
   (b) Construct the boundary clipped Voronoi cell Ω'_i by computing the convex hull of the union of all points in P_i and the vertices of Ω_i inside V.

To identify the boundary cells, all infinite Voronoi cells are first converted to finite cells. This is done by replacing the infinite rays edging the cell by finite length segments, with a length greater than the diameter of a bounding sphere containing the shape. This creates new vertices for adjacent cells. A Voronoi cell is then detected as a boundary cell if at least one of its vertices is outside the shape.

3. Optimization Since the sites are not regularly distributed initially, the cells of the clipped Voronoi tessellation, as obtained with step 2, are not uniform nor regular, as shown in Figure 3 (a). In order to improve cell shapes and to get a uniform and regular volumetric decomposition of V, the site locations are optimized. In the literature, the two main strategies for this optimization are the Lloyd's gradient descent method and the L-BFGS quasi-Newton method. In our approach, we choose the latter for its fast convergence [Liu et al., 2009]. As shown in Figure 3 (b), once convergence in the optimization is reached, all the clipped Voronoi cells present more uniform and regularly distributed shapes.



(a) (b)

Figure 3: *Clipped Voronoi tessellations without (a) and with optimized site locations (b). Cells in displayed CVTs are slightly shrinked for visualization purposes.*

Object	Method	Error (m)	Time (s)
Dancer	MC	61.21	0.945
65386 pts	Delaunay	147.31	6.944
	CVT (0 iter.)	60.17	2.14
	CVT (10 iter.)	42.77	16.650

Table 1: *Distance sum from the input point clouds (Figure 4-(e)) to the estimated shape surface and computation times for: Voxel representation (MC) [Chernyaev, 1995], Delaunay refinement strategy [CGAL,] and our CVT approach.*

4.3. Evaluation

The algorithm was implemented in C++, and uses the libLBFGS library [Okazaki and Nocedal, 2010] for the L-BFGS computation during the optimization. The Tetgen library [Hang, 2015] was used to compute Voronoi tessellations. Our approach was tested on oriented point clouds acquired with a multi-view systems. Implicit functions were estimated from point clouds with the CGAL [CGAL,] implementation of the Poisson reconstruction algorithm. Figure 4 and Table 1 show a comparison of different strategies to get volumetric representations of a dancer. This comparison was performed on various examples with similar results and we only present the dancer example for conciseness. The compared methods are a voxel method with a Marching Cubes algorithm with topology guarantees [Chernyaev, 1995] and the CGAL implementation of the Delaunay refinement approach [CGAL,]. The number of cells was made similar in all approaches by choosing the number of cubes (2cm$\times$2cm) intersecting or fully inside the shape as the number of sites for CVT (14455 in the example) and by making the cube diagonal the length constraint for Delaunay ball diameters in Delaunay refinement. Figure 4 illustrates the benefit of CVTs for regularity. Note in particular the irregular boundary cells with the voxel representation. In addition, Table 1 indicates a better precision for CVTs, where the precision evaluates the quality of the shape approximation by summing the Euclidian distances from the observed points to the generated shape surface. Nevertheless, the table also shows the increased computation time with CVTs, in particular when iterating over site locations. Optimized implementations for CVTs may anyway compensate partially for this additional computation cost.

5. Volumetric Shape Tracking

With the volumetric decomposition proposed above, we now need to define a model by which scene dynamics can be captured through deformations expressed over this decomposition. We consider here as input a time sequence of inconsistent CVT decompositions independently estimated at each frame and we look for a time consistent volumetric decomposition that encode cell motions. We opt for a capture by deformation approach where a template CVT, taken from the input sequence in our case, is tracked throughout

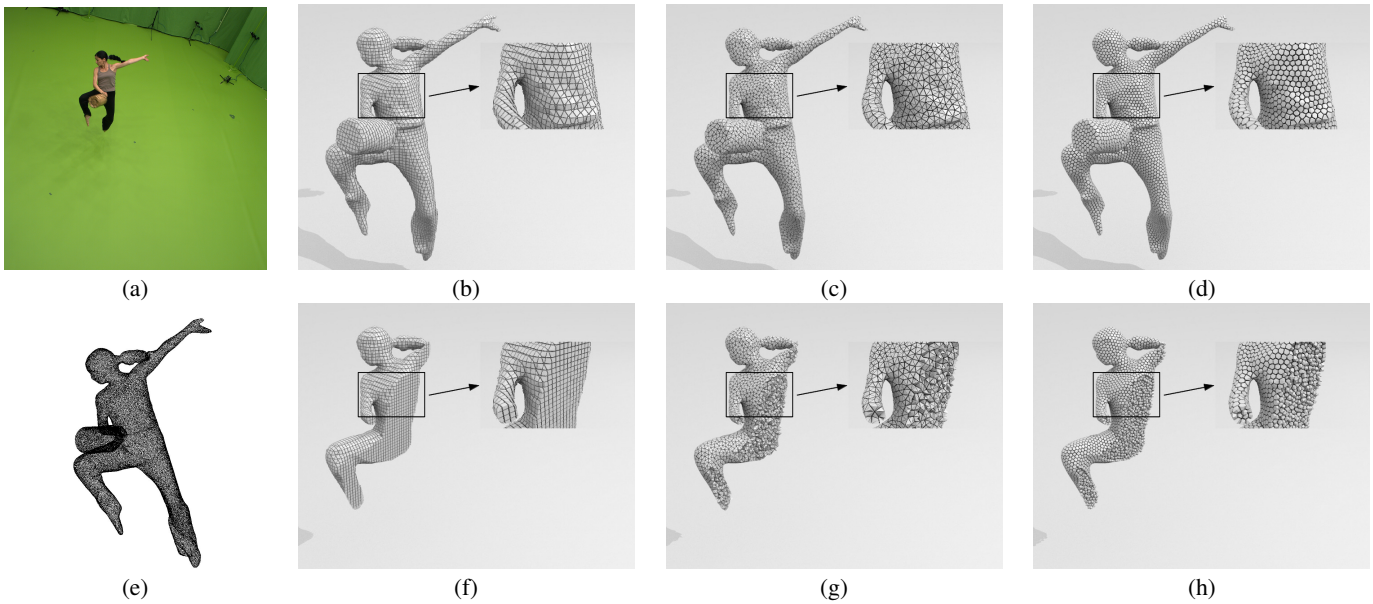


Figure 4: *(a, e) Input multi-camera observation and point clouds (65386 pts). (b,f) Tessellations generated using voxels [Chernyaev, 1995]. (c,g) Tetrahedrisations generated using Delaunay refinement [CGAL,] . (d,h) Clipped Centroidal Voronoi Tessellations (14455 sites).*

the sequence. This volumetric strategy is motivated by two observations. First, attaching the deformation model to the cell shape representation used for the animation directly provides the necessary cell dynamic information to the simulation. It avoids therefore the interpolation between an intermediate motion model, *e.g.* a skeleton or a mesh, and the animation model; Such interpolation being difficult to perform consistently over time. Second, as shown in [Allain et al., 2015], it provides a simple tool for embedding volume-preservation constraints that increase the robustness of the tracking over the dynamic scenes we consider. We describe below the generative approach [Allain et al., 2015] that we follow.

5.1. Tracking Formulation

We are given a sequence of CVTs $\mathcal{V}$ and a template model $\hat{V}$. $\hat{V}$ can be one model taken from the sequence or any other model (*e.g.* a 3D scan) decomposed into a CVT. The tracking consists then in fitting $\hat{V}$ to each $V \in \mathcal{V}$. This can be formulated as a maximum a posteriori (MAP) estimation of the deformation parameters $\hat{\Theta}$ that maximizes the posterior distribution $\mathcal{P}(\Theta|\mathcal{V})$ of the parameters Θ given the observations $\mathcal{V}$:

$$\hat{\Theta} = \arg\max_{\Theta} \; P(\Theta|\mathcal{V}) \simeq \arg\max_{\Theta} \; P(\mathcal{V}|\Theta)\, P(\Theta).$$

Taking the log of the above expression yields the following optimization problem:

$$\hat{\Theta} = \arg\max_{\Theta} \; E_{data}(\mathcal{V}, \Theta) + E_{prior}(\Theta), \qquad (2)$$

where the data term E_{data} evaluates the log-likelihood of a set of deformation parameters Θ given the observations $\mathcal{V}$, and the regularization term E_{prior} enforces prior constraints on the deformation, *e.g.* local rigidity. We detail below the parameterization Θ we use for the deformation and the associated energy terms.

5.2. Motion Parameterization

The deformation model is defined over CVT cells in the shape decomposition and, for efficiency, on aggregates of cells which reduces the number of terms and parameters. To this goal, CVT cells are grouped together as a set of volumetric patches P_k using a k-medoids algorithm, as shown in Figure 5. Such patches can be either adjacent to the surface or completely inside the template shape's volume, which is of particular interest to express non-rigid deformation of the model while preserving the local volume and averting over-compression or dilation. The positional information of a patch is represented as a rigid transform $\mathbf{T}_k^t \in SE(3)$ at every time t. Each position $\mathbf{x}_{k,q}$ of a CVT sample is indiscriminately labeled as a point q. Its position can be written as a transformed version of its template position $\mathbf{x}_q^0$ as follows, once the patch's rigid transform is applied:

Figure 5: *The template model used to recover the runner sequence motion, with its CVT decomposition cells, and the cell clusters in different colors.*

$$\mathbf{x}_{k,q} = \mathbf{T}_k(\mathbf{x}_q^0). \qquad (3)$$

A *pose* of the shape is thus defined as the set of patch transforms $\mathbf{T} = \{\mathbf{T}_k\}_{k\in\mathcal{K}}$, which expresses the deformation of every component in the shape. The parameterization Θ is then the set of pose

parameters of the template over the considered time sequence $\mathcal{T}$:

$$\Theta = \{\mathbf{T}^t\}_{t\in\mathcal{T}}.$$

5.3. Data Term

We assume the observed shape V^t at time t is described by the point cloud $\mathbf{Y}^t = \{\mathbf{y}_o^t\}$. We assume this point cloud to include inner volume points and surface points, *i.e.* the CVT sites as well as the outside surface points.

In order to measure how a deformed version of the template explains the observed shape, first the associations between the observations and the template must be determined. This is achieved via a soft ICP strategy that iteratively reassigns each observation $\mathbf{y}_o^t$ to the template volumetric patches. For simplicity, each observation $\mathbf{y}_o^t$ is associated to the patch P_k via the best candidate point $\mathbf{x}_{k,q}$ of patch P_k and with an association penalty $\alpha_{o,k}$.

The matching penalty $E(\mathbf{y}_o^t, \mathbf{T}_{\mathrm{k}}^t)$ that evaluates how well P_k explains $\mathbf{y}_o^t$ is then the weighted distance between $\mathbf{y}_o^t$ and the best candidate $\mathbf{x}_{k,q}$, that is the template point $\mathbf{x}_q^0$ transformed by the current pose $\mathbf{T}_{\mathrm{k}}^t$ of the template model:

$$E(\mathbf{y}_o^t, \mathbf{T}_{\mathrm{k}}^t) \;=\; \alpha_{o,k}^t \, \|\mathbf{y}_o^t - \mathbf{T}_{\mathrm{k}}^t(\mathbf{x}_q^0)\|. \tag{4}$$

Associations are additionally filtered using a compatibility test. Observed surface points are associated to template surface points with similar orientations with respect to a user defined threshold θ_{max}; Observed inner points are associated to template inner points that present similar distances to the surface up to a user defined tolerance ϵ. If there is no compatible candidate in a patch P_k, then P_k is discarded for the association with $\mathbf{y}_o^t$, *i.e.* $\alpha_{o,k}^t = 0$. Finally:

$$E_{data}(V^t, \Theta^t) = \sum_{o,k} E(\mathbf{y}_o^t, \mathbf{T}_{\mathrm{k}}^t). \tag{5}$$

5.4. Regularization Term

The pose of a shape is defined by the set of rigid motion parameters of the shape volumetric patches. While these parameters hardly constraint the patch motions, they do not define a coherent shape motion since each patch moves independently of the others. In order to enforce shape cohesion, soft local rigidity constraints, reflecting additional prior knowledge on shape deformation parameters, are considered. These constraints rely on a pose distance function that evaluates how distant from a rigid transformation a deformation between two poses is. Once such a distance is defined, the regularization term is defined as the sum of distances between all poses in the sequence and a given pose that can be a reference pose or an estimated mean pose, as explained below.

Pose Distance To simplify the estimation, the shape distance is expressed over coordinates of points belonging to patches and not on the parameters of the pose itself:

$$\mathcal{D}(\mathbf{T}^i, \mathbf{T}^j) = \sum_{(P_k,P_l)\in\mathcal{N}} \mathcal{D}_{kl}(\mathbf{T}^i, \mathbf{T}^j), \quad \text{with} \tag{6}$$

$$\mathcal{D}_{kl}(\mathbf{T}^i, \mathbf{T}^j) = \sum_{q\in P_k\cup P_l} \|\mathbf{T}_{k-l}^i(\mathbf{x}_q^0) - \mathbf{T}_{k-l}^j(\mathbf{x}_q^0)\|^2,$$

where $\mathbf{T}_{k-l}^i = \mathbf{T}_l^{i\,-1} \circ \mathbf{T}_k^i$ is the relative transformation between patches P_k and P_l for pose i, and $\mathcal{N}$ is the set of neighboring patch pairs within the shape. Intuitively, this distance measures whether the relatives poses between neighboring volumetric patches are preserved during motion between two shape poses.

Deformation Energy The pose distance above allows to compute a deformation energy between two poses and with respect to a reference pose from which the patch transformations $\mathbf{T}_t^k$ are expressed. This reference pose can be taken as the identity pose of the initial template model (see Figure 5 for instance). However such a strategy is biased toward the template pose and discourage locally rigid motions between poses that are distant from the template pose. A more appropriate approach is to exploit the pose distance function to first define a mean pose $\bar{\mathbf{T}}$ over a time window $\{t\}$:

$$\bar{\mathbf{T}} = \arg\min_{\mathbf{T}} \sum_t \mathcal{D}(\mathbf{T}^t, \mathbf{T}).$$

This averaged pose can then be taken as an evolving reference pose from which non-rigid deformations are measured to define the deformation energy over the associated time interval:

$$E(\{\mathbf{T}^t\}, \bar{\mathbf{T}}) \;=\; \sum_t \mathcal{D}(\mathbf{T}^t, \bar{\mathbf{T}}). \tag{7}$$

This imposes general proximity of poses to a sequence specific "rest" pose. These rest poses must also be constrained to some form of inner cohesion. This is ensured by minimizing the distance from the mean pose to the identity pose of our initial template:

$$E(\bar{\mathbf{T}}) \;=\; \mathcal{D}(\bar{\mathbf{T}}, \mathbf{Id}), \tag{8}$$

This definition of deformation has a number of advantages: first it enforces geometric cohesion and feature preservation, and second it is quite simple to formulate and to optimize, as the minimization of (7) and (8) translates to a sum of least square constraints over the set of CVT sites.The prior energy term finally writes:

$$E_{prior}(\Theta = \{\mathbf{T}^t\}) = \; \mathcal{D}(\bar{\mathbf{T}}, \mathbf{Id}) \;+\; E(\{\mathbf{T}^t\}.\bar{\mathbf{T}}), \tag{9}$$

We jointly extract poses and a sequence mean pose by minimizing the sum of the data terms (5) and the prior term (9). Section §7. shows results on various sequences and gives run time performances.

6. Combined Animation

The template representation of the subject now being consistently tracked across the sequence, we can use the tracked cells as input for solid physics-based animation. We have purposely chosen CVTs as a common representation as they can be made suitable for shape and motion capture as we have shown, while being straightforwardly convenient for physics-based computations. In fact CVT cells are compact, convex or easily approximated by their convex hull. This is an advantage for the necessary collision detection phase of physics models, as specific and efficient algorithms exist for this case [Gilbert et al., 1988, Rabbitz, 1994]. We here describe the common principles of our animation model, with more specific applications being explored and reported on in the following sections.

As our animation framework is solid-based, we base our description on commonly available solid-based physics models, *e.g.* [Baraff, 1997]. Each CVT cell is considered a homogeneous rigid body, whose *simulated* state is parameterized by its 3D position, rotation, linear momentum and angular momentum. The cell motion is determined by Newton's laws through a differential equation involving the cell state, the sum of forces and sum of torques that are applied to the cell.

The animation is thus obtained by defining the set of forces and torques applied at each instant, and iteratively solving these differential equations to obtain a new cell position and orientation for a target time step, using one of many available techniques. For our demonstrator, we use the simple and efficient off-the-shelf Bullet Physics engine [Bullet, , Catto, 2005].

6.1. Ordinary Applied Forces

We classically apply the constant gravity force $F_g = Mg$. We apply additional external forces or constraints as needed for the target application, as will be detailed in the coming sections. Additionally, contact forces such as collisions are handled with scene objects, as well as between different cells of the CVT. For this purpose the physics engine first needs to detect the existence of such contacts. It relies on a hierarchical space decomposition structure, such as an AABB-tree, for broad-phase collision detection, *i.e.* coarse elimination of collision possibilities. A narrow-phase collision test follows, between objects lying in the same region of space as determined by the AABB-tree traversal. In this narrow phase the full geometry of objects is examined, *e.g.* using the GJK algorithm [Gilbert et al., 1988] for pairs of convex polyhedra. Once the existence of a contact is established, various strategies exist to deal with the collision, *e.g.* by introducing impulse repulsion forces to produce a collision rebound. We follow the common approach of modeling contacts as a linear complementary problem (LCP) popularized by [Baraff, 1994], which derives contact forces as the solution of a linear system that satisfies certain inequality constraints. These constraints are typically formulated using a constraint Jacobian over the combined state spaces of rigid bodies. [Catto, 2005] expose the specific variant applied in the context of the Bullet Physics engine.

6.2. Physical Modeling of Kinematic Control

To relate the physical simulation to the acquired non-rigid poses of the model, we need to introduce coupling constraints. Our goal is to allow the model to materialize and control the tradeoff between the purely kinematic behavior acquired from visual inputs for the cell, and the purely mechanically induced behavior in the simulation. First it is important to note that the temporal discretization used for acquisition and for simulation and rendering of the effects are generally different. Consequently the first stage in achieving our goal is to compute a re-sampling of the pose sequence, to the target simulation and rendering frequency, using position and quaternion interpolation. The poses so obtained are here referred as the acquired cell poses $\hat{x}^a(t)$ and $R^a(t)$. Second, we formulate the coupling by introducing a new kinematic recall force, in the form of a damped spring between the acquired cell poses and the simulated cell poses:

$$F_r(t) = k.(\hat{x}^a(t) - \hat{x}(t)) - \lambda.\frac{\mathrm{d}}{\mathrm{d}t}(\hat{x}^a(t) - \hat{x}(t)), \qquad (10)$$

where k and λ are respectively the rigidity and damping coefficients of the spring, which control the strength and numerical stability of the coupling.

7. Visual Effects

This section presents various animation results on three captured animations of variable nature and speed (see also the accompanying video). RUNNER shows a male character running in a straight line, during 3 motion cycles. This animation lasts 2.5 s. In BIRDCAGEDANCE a female dancer moves while holding a bird cage. This sequence is 56 s long and shows a complex sequence of motions which would be difficult to synthetize without sensors. Finally, in SLACKLINE a male acrobat evolves on a non rigid line above the ground, for 25 s. The input sequences of temporally inconsistent 3D point clouds are made respectively of 126 (RUNNER), 2800 (BIRDCAGEDANCE) and 1240 (SLACKLINE) temporal frames.

Parameters The interior of all shapes has been tessellated according to §4. using 5000 cells. 10 iterations of the L-BFGS quasi-Newton algorithm were applied, except for the template shape where 50 iterations were applied. We use a temporal window of 10 frames for the tracking, and cluster the 5000 cells into 200 patches. 60 iterations were applied.

We list below examples of visual effects that we were able to generate using the proposed approach. First, we present animations combining tracking results with solid dynamics simulation (§7.1.). Then we present other visual effects that also exploit the volumetric tracking information (§7.2. and §7.3.).

7.1. Asynchronous Kinematic Control Deactivation

In order to show the effect of gravity while keeping the dynamic motion of the input sequence, we deactivate kinematic control forces independently for each cell. After being deactivated, a cell usually falls to the ground, since it follows a trajectory determined only by gravity, collision forces and its initial velocity. Asynchronous cell deactivations result in an animation combining cells that follow their tracking trajectory and cells that fall. By choosing diverse strategies for scheduling cell deactivations, a wide variety of animations can be obtained.

We explore here three possible deactivation strategies that are based on different criteria. Note that while these effects are straightforward to produce with our volumetric framework, it would be difficult to obtain them if only surface or skeleton-based tracking was available.

7.1.1. Rupture under Collisions

Collisions with obstacles sometimes deviate a cell from its theoretical trajectory, which results in an increase of the recall force magnitude (see Eq. 10). This phenomenon can be detected and used for simulating the rupture of the material: when the recall force magnitude of a cell is above a given threshold, we deactivate the recall force (for this cell only). This makes the rupture looks like the consequence of the collision.

Figure 8 shows a heavy pendulum that hits the subject and makes a hole in it. Under the intensity of the collision, several cells are ejected (rupture) and fall to the ground.

7.1.2. Heat Diffusion

In order to make the cell deactivation both temporally and spatially progressive, we rely on a diffusion algorithm. We diffuse an initial temperature distribution inside the volume according to the diffusion equation. Deactivation is triggered when the cell temperature is above a given threshold.

Heat diffusion in a CVT The CVT provides a graph structure on centroids, which is a subgraph of the Delaunay tetrahedralization of the cells centroids. The heat diffusion on a graph structure is expressed by the heat equation:

$$(\partial/\partial t + \mathbf{L})\mathbf{F}_t = 0$$

where $\mathbf{F}_t$ is the column vector of centroids temperatures at time t, and $\mathbf{L}$ is the combinatorial graph Laplacian matrix (note that a geometric Laplacian is not necessary since centroids are regularly distributed in space). Given an initial temperature distribution $\mathbf{F}_0$, this equation has a unique solution

$$\mathbf{F}_t = \mathbf{H}_t\mathbf{F}_0$$

where $\mathbf{H}_t = e^{t\mathbf{L}}$ is the heat diffusion kernel, which can be computed by means of the spectral decomposition of $\mathbf{L}$.

We compute the temperature evolution on the BIRDCAGEDANCE sequence for an initial temperature distribution where all cell temperatures are zero, except for the dancer's head

Figure 6: *Time persistence on the* RUNNER *sequence: a slower copy of the shape that erodes over time is generated at regular intervals.*

top cells (which are set to 1). We observe in Figure 1 that cells fall progressively across time, from the upper to the lower body parts. Note that the cage cells remain kinematically controlled since heat is not transferred between different connected components.

7.1.3. Morphological Erosion

The discrete cell decomposition of shapes allows to apply morphological operators. To illustrate this principle, we have experimented erosion as shown in Figure 2 . In this example, cells are progressively eroded starting from the outside. The morphological erosion is performed by deactivating each cell after a delay proportional to the distance between the cell centroid and the subject's surface. The distance is computed only once for each cell, on the template shape. Figure 2 shows the erosion animation on the RUNNER sequence. Note that the operation progressively reveals the dynamic of the inner part of the shape.

7.2. Time Persistence

In this example, we experiment time effects over cell decompositions. To this aim, dynamic copies of the model are generated at regular time intervals. These copies are equipped with deceleration and erosion effects over time and create therefore ghost avatars that vanish with time (see Figure 6 and the accompanying video). The benefit of the tracked volumetric representation in this simulation is the ability to attach time effect to the model behavior at the cell level, for instance lifetime and deceleration in the example.

7.3. Morphing

Our dynamic representations allows to apply volumetric morphing between evolving shapes, enabling therefore new visual effects with real dynamic scenes. To this purpose, cells of the source shape are first matched to the target shape. Second, each cell is individually morphed to its target cell at a given time within the sequence. Time ordering is chosen such that cells in the source shape are ordered from the outside to the inside, and associated with the cells of the target shape ordered from the inside to the outside. Cells are transformed from the source to the destination by interpolating their positions and using [Kent et al., 1992] to morph their polyhedral shapes. Figure 7 shows the dynamic morphing of the RUNNER sequence onto the BIRDCAGEDANCE sequence.

7.4. Run Time Performance

Our approach has been tested on a dual Intel Xeon E5-2665 processor with 2.40 GHz each. For each animation, the Poisson runs in 0.67 s per frame on average, and the volumetric decomposition for each frame runs in 6.27 s, except for the template model for which it runs in 25.52 s since more iterations are applied. Our tracking algorithm runs in 45 s per frame on average. The physical simulation usually needs about 350 ms per simulation step on a single thread. The morphing runs on multiple threads in about 2.35 s.

8. Limitations

As shown in the previous examples, our approach generates plausible results for a variety of captured and simulated motions. However, a few limitations must be noted. First, the true captured shape must be volumetric in nature, since we tessellate its interior into 3D cells. Thin shapes such as clothes may cause some cells to be flat, leading to volume variation among cells and ill-defined cohesion constraints that would cause difficulty to the tracking model.

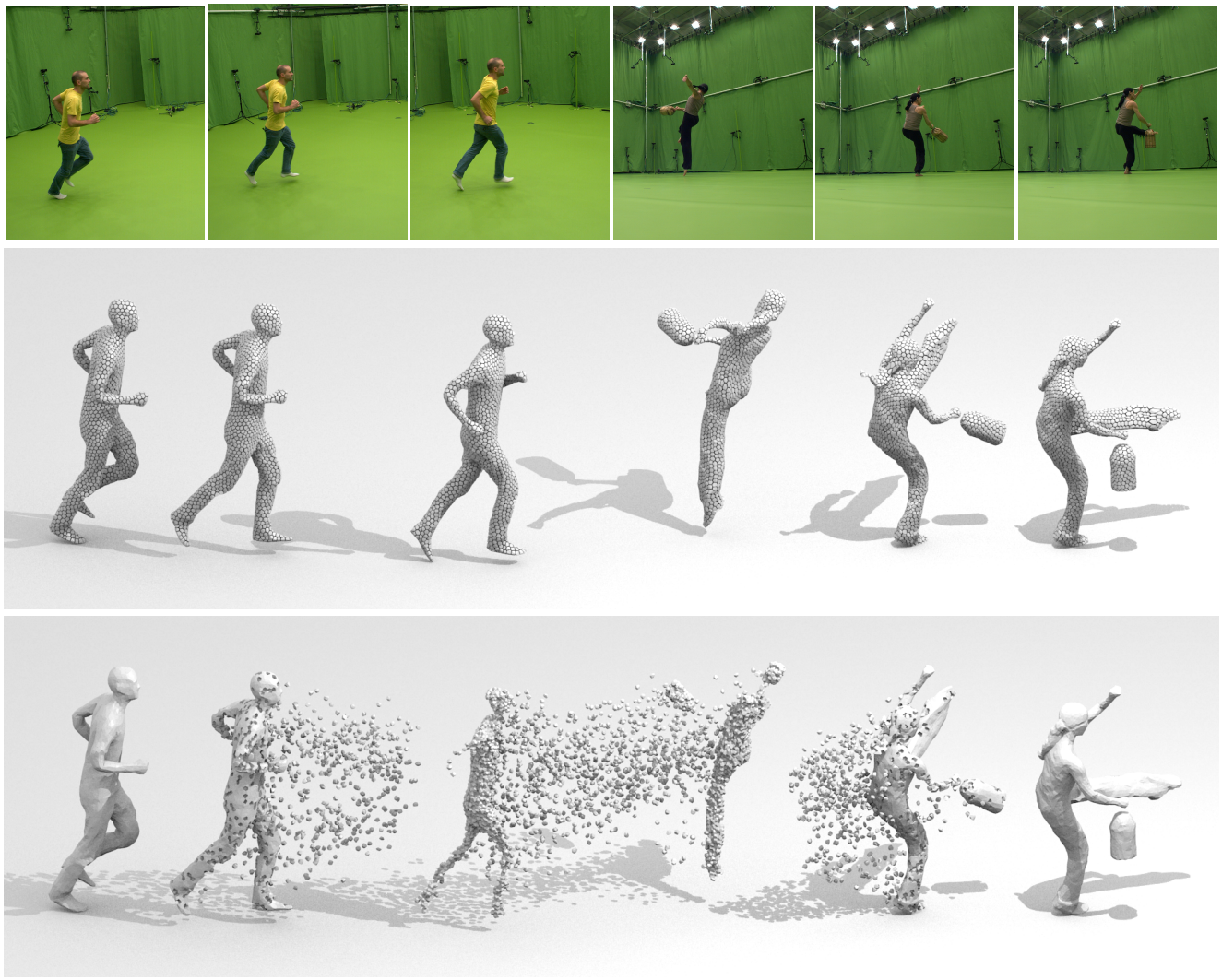


Figure 7: *Tracking result of the* RUNNER *and the* BIRDCAGEDANCE *sequences (middle) and combination with volumetric morphing with 5000 cells (bottom).*

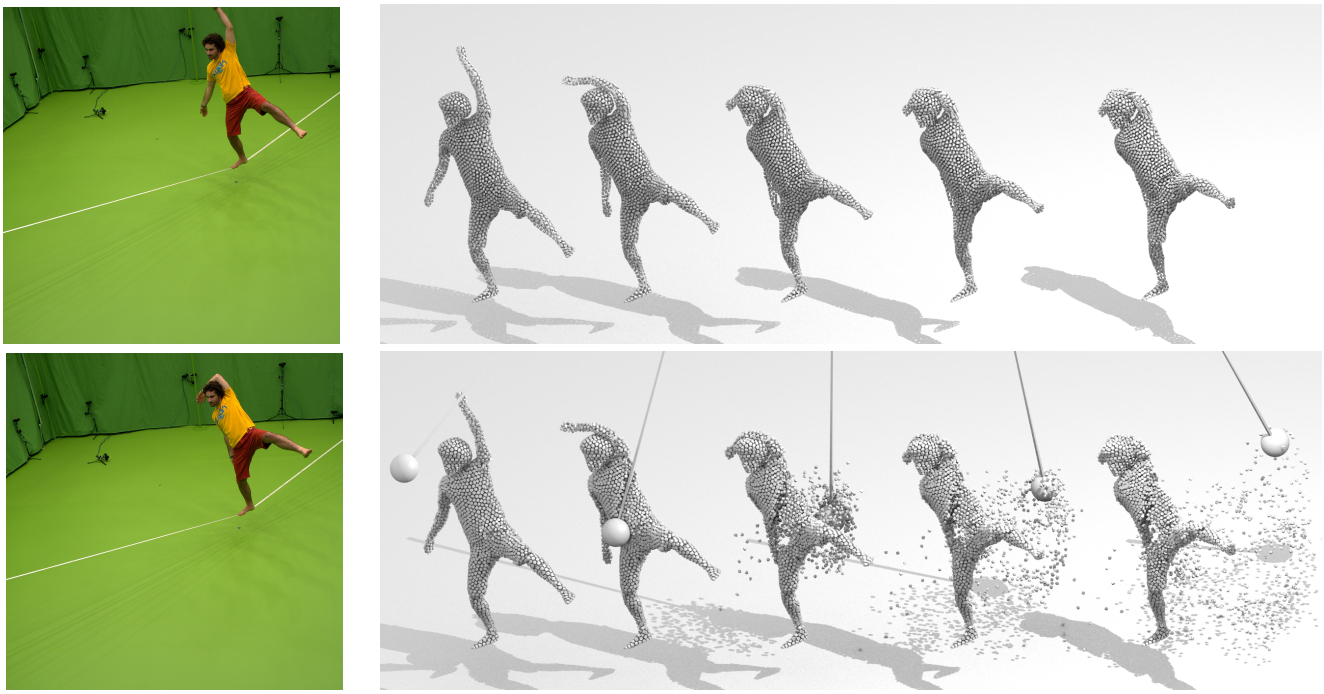


Figure 8: *Input* SLACKLINE *Multi-camera observations (left), tracking result of the* SLACKLINE *sequence (top) and combination with the effect of collision with a pendulum (bottom).*

Regarding the physical simulation, our current demonstrator is limited to rigid body interactions, but could be extended to other physical models such as soft body physics and fluid simulation. Since a CVT provides neighboring information, soft body simulation could be achieved by introducing soft constraints between neighboring cells. This would lead to animations where cell sets behave more like a whole rather than independent bodies.

9. Conclusion

This document reports on a framework that allows video-based animations to be combined with physical simulation to create unprecedented and plausible animations. The interest is to take benefit of both modalities and to bring complementary properties to computer animations. Our approach relies on volumetric tessellations within which kinematic constraints can be associated to mechanical forces or procedural tasks at a cell level. Because of its simple and unified nature, relying on centroidal Voronoi tessellations, the system opens various new opportunities to reconsider how animations can be generated and automated from raw captured 3D data, and it paves the way for other effects such as soft-body and fluid mechanics, which we will explore in future work.

References

[Allain et al., 2015] Allain, B., Franco, J.-S., and Boyer, E. (2015). An Efficient Volumetric Framework for Shape Tracking. In *Proc. of CVPR*.

[Baraff, 1994] Baraff, D. (1994). Fast contact force computation for nonpenetrating rigid bodies. In *Proc. of SIGGRAPH*.

[Baraff, 1997] Baraff, D. (1997). An introduction to physically based modeling: Rigid body simulation. In *SIGGRAPH Course Notes*.

[Bullet,] Bullet. Bullet Physics Library. http://bulletphysics.org/.

[Carranza et al., 2003] Carranza, J., Theobalt, C., Magnor, M., and Seidel, H.-P. (2003). Free-viewpoint Video of Human Actors. *ACM Trans. on Graph.*, 22(3).

[Casas et al., 2013] Casas, D., Tejera, M., Guillemaut, J., and Hilton, A. (2013). Interactive Animation of 4D Performance Capture. *IEEE Trans. on Visualization and Computer Graphics*, 19(5).

[Cashman and Hormann, 2012] Cashman, T. J. and Hormann, K. (2012). A continuous, editable representation for deforming mesh sequences with separate signals for time, pose and shape. *Comput. Graph. Forum*, 31(2).

[Catto, 2005] Catto, E. (2005). Iterative Dynamics with Temporal Coherence. In *Proc. of Game Developer Conference*.

[CGAL,] CGAL. Computational Geometry Algorithms Library. http://www.cgal.org/.

[Chernyaev, 1995] Chernyaev, E. V. (1995). Marching Cubes 33: Construction of topologically correct isosurfaces. Technical Report CN 95–17, CERN.

[de Aguiar et al., 2008] de Aguiar, E., Stoll, C., Theobalt, C., Ahmed, N., Seidel, H.-P., and Thrun, S. (2008). Performance capture from sparse multi-view video. *ACM Trans. on Graph.*, 27(3).

[de Aguiar et al., 2007] de Aguiar, E., Theobalt, C., Stoll, C., and Seidel, H. (2007). Marker-less Deformable Mesh Tracking for Human Shape and Motion Capture. In *Proc. of CVPR*.

[Du et al., 1999] Du, Q., Faber, V., and Gunzburger, M. (1999). Centroidal Voronoi Tessellations: Applications and Algorithms. *SIAM review*, 41.

[Gall et al., 2009] Gall, J., Stoll, C., De Aguiar, E., Theobalt, C., Rosenhahn, B., and Seidel, H.-P. (2009). Motion capture using joint skeleton tracking and surface estimation. In *Proc. of CVPR*.

[Gilbert et al., 1988] Gilbert, E. G., Johnson, D. W., and Keerthi, S. S. (1988). A Fast Procedure for Computing the Distance Between Complex Objects in Three-dimensional Space. *IEEE Jour. on Robotics and Automation*, 4.

[Gross et al., 2003] Gross, M., Würmlin, S., Naef, M., Lamboray, E., Spagno, C., Kunz, A., Koller-Meier, E., Svoboda, T., Goll, L. V., Lang, S., Strehlke, K., Moere, A. V., and Staadt, O. (2003). blue-c: A Spatially Immersive Display and 3D Video Portal for Telepresence. *ACM Trans. on Graph.*, 22(3).

[Hang, 2015] Hang, S. (2015). Tetgen, a Delaunay-based quality tetrahedral mesh generator. *ACM Trans. on Mathematical Software*, 41(2).

[Jamin et al., 2014] Jamin, C., Alliez, P., Yvinec, M., and Boissonnat, J.-D. (2014). CGALmesh: a generic framework for delaunay mesh generation. *ACM Trans. on Mathematical Software*.

[Ju et al., 2002] Ju, T., Losasso, F., Schaefer, S., and Warren, J. (2002). Dual contouring of Hermite data. *ACM Trans. on Graph.*, 21(3).

[Kent et al., 1992] Kent, J. R., Carlson, W. E., and Parent, R. E. (1992). Shape transformation for polyhedral objects. In *Proc. of SIGGRAPH*.

[Lévy, 2014] Lévy, B. (2014). Restricted Voronoi diagrams for (re)-meshing surfaces and volumes. In *8th International Conference on Curves and Surfaces*.

[Liu et al., 2005] Liu, C. K., Hertzmann, A., and Popović, Z. (2005). Learning Physics-based Motion Style with Nonlinear Inverse Optimization. *ACM Trans. on Graph.*, 24(3).

[Liu et al., 2009] Liu, Y., Wang, W., Lévy, B., Sun, F., Yan, D.-M., Liu, L., and Yang, C. (2009). On centroidal voronoi tessellation - energy smoothness and fast computation. *ACM Trans. on Graph.*, 28(101).

[Lorensen and Cline, 1987] Lorensen, W. E. and Cline, H. E. (1987). Marching Cubes: A high resolution 3d surface construction algorithm. In *Proc. of SIGGRAPH*.

[Matusik et al., 2000] Matusik, W., Buehler, C., Raskar, R., Gortler, S., and McMillan, L. (2000). Image Based Visual Hulls. In *Proc. of SIGGRAPH*.

[Okabe et al., 2000] Okabe, A., Boots, B., Sugihara, K., and Chi, S. N. (2000). *Spatial Tessellations: Concepts and Applications of Voronoi Diagrams*. John Wiley.

[Okazaki and Nocedal, 2010] Okazaki, N. and Nocedal, J. (2010). libLBFGS: a library of Limited-memory Broyden-Fletcher-Goldfarb-Shanno (L-BFGS). http://www.chokkan.org/software/liblbfgs/.

[Popović and Witkin, 1999] Popović, Z. and Witkin, A. (1999). Physically Based Motion Transformation. In *Proc. of SIGGRAPH*.

[Rabbitz, 1994] Rabbitz, R. (1994). Graphics gems iv. chapter Fast Collision Detection of Moving Convex Polyhedra. Academic Press.

[Safonova et al., 2004] Safonova, A., Hodgins, J. K., and Pollard, N. S. (2004). Synthesizing Physically Realistic Human Motion in Low-dimensional, Behavior-specific Spaces. *ACM Trans. on Graph.*, 23(3).

[Shewchuk, 1998] Shewchuk, J. R. (1998). Tetrahedral mesh generation by Delaunay refinement. In *Symposium on Computational Geometry (SoCG)*. ACM.

[Starck and Hilton, 2007] Starck, J. and Hilton, A. (2007). Surface Capture for Performance Based Animation. *IEEE Comp. Graph. and Applications*, 27(3).

[Straka et al., 2012] Straka, M., Hauswiesner, S., Rüther, M., and Bischof, H. (2012). Simultaneous shape and pose adaption of articulated models using linear optimization. In *Proc. of ECCV*.

[Sulejmanpasić and Popović, 2005] Sulejmanpasić, A. and Popović, J. (2005). Adaptation of Performed Ballistic Motion. *ACM Trans. on Graph.*, 24(1).

[Sumner and Popović, 2004] Sumner, R. W. and Popović, J. (2004). Deformation Transfer for Triangle Meshes. *ACM Trans. on Graph.*, 23(3).

[Tung et al., 2009] Tung, T., Nobuhara, S., and Matsuyama, T. (2009). Complete Multi-view Reconstruction of Dynamic Scenes from Probabilistic Fusion of Narrow and Wide Baseline Stereo. In *Proc. of ICCV*.

[Vlasic et al., 2008] Vlasic, D., Baran, I., Matusik, W., and Popović, J. (2008). Articulated Mesh Animation from Multi-view Silhouettes. *ACM Trans. on Graph.*, 27(3).

[Vlasic et al., 2009] Vlasic, D., Peers, P., Baran, I., Debevec, P., Popović, J., Rusinkiewicz, S., and Matusik, W. (2009). Dynamic Shape Capture using Multi-view Photometric Stereo. *ACM Trans. on Graph.*, 28(5).

[Wei et al., 2011] Wei, X., Min, J., and Chai, J. (2011). Physically Valid Statistical Models for Human Motion Generation. *ACM Trans. on Graph.*, 30(3).

[Wu et al., 2012] Wu, C., Varanasi, K., and Theobalt, C. (2012). Full-body Performance Capture under Uncontrolled and Varying Illumination : A Shading-based Approach. In *Proc. of ECCV*.

[Yan et al., 2013] Yan, D.-M., Wang, W., Lévy, B., and Liu, Y. (2013). Efficient computation of 3d clipped Voronoi diagram for mesh generation. *Computer-Aided Design*, 45.

[Ye and Liu, 2008] Ye, Y. and Liu, C. K. (2008). Animating responsive characters with dynamic constraints in near-unactuated coordinates. *ACM Trans. on Graph.*, 27(5).

[Zordan and Hodgins, 2002] Zordan, V. B. and Hodgins, J. K. (2002). Motion Capture-driven Simulations That Hit and React. In *Proc. of SCA*.