Bringing Computer and Physics Closer

Yonghao Yue
Graduate School of Frontier Sciences, The University of Tokyo, Kiban building 701, 5-1-5
Kashiwa-no-ha, Kashiwa-shi, Chiba 277-8561, Japan
E-mail: yonghao@edu.k.u-tokyo.ac.jp

Abstract. Physically based simulations are important for predicting the future states of the objects in consideration, and have been one of the major research topics in computer graphics. Applications include optical simulations for realistic image synthesis given a 3D scene, and rigid/liquid dynamics for computer animations. The need for the ultimate realism may sometimes ask for more efficient and accurate methods than those available in other fields. In addition, physically based optimizations are useful for designing and fabricating man-made objects given some desirable functionalities governed by physical laws. The ability to incorporate predictive simulations also allows us to build smarter robots that can optimize for their next movement based on the predictions. We introduce some of our work performed from this perspective.

1. Introduction

Our world is governed by physical laws: the way how light travels from the light source to our eye can be described by the radiative transport equation; the way how the water in a cup behaves as we interact with the cup is described by the Navier-Stokes equations. The ability of predicting such physical behaviors would allow us to estimate how objects would deform and how they would look like, and is hence useful for computer animations.

Physical laws also come into play when we want to design and fabricate man-made objects. For example, understanding how the light refracts is essential for designing lenses, and understanding how the contact and frictional forces act against the gravity is important for designing a stable sculpture that is assembled from a number of small bricks. Optimizations with physically based constraints in consideration are thus fundamental for designing and fabricating functional man-made objects.

With the ability to incorporate predictive simulations in planning, we can also build smarter robots that use vision techniques to acquire the external information, perform predictive simulations to estimate the future states of the objects, and determine the appropriate control based on the predicted states. In this paper, we will review and introduce a series of our past work done from the above perspective.

2. Predictive Simulations

As for predictive simulations, we introduce our work on optics- and mechanics-based methods. For optics based method, we present an efficient and unbiased free path sampling technique for radiative transport [1, 2]. For mechanics based method, we present how to simulate non-Newtonian fluids including shear thinning (cream and foam) and shear thickening (oobleck) fluids [3].
the scattering direction and the distance (free path) radiance of the light paths (up to a normalizing constant). The intensity distribution on the image plane is viewed as the marginalized distribution of the light paths that pass through the camera lens and hit the pixel, while in MCMC based methods, we integrate for each pixel on the image plane the contribution (radiance) of the density function related to the contribution of each light path. Modern rendering methods utilize Monte Carlo techniques (e.g., Monte Carlo path tracing [4, 5] or Markov chain Monte Carlo techniques [5, 6]) to draw light paths according to some probability density function related to the contribution of each light path: in methods based on Monte Carlo integration, we integrate for each pixel on the image plane the contribution (radiance) of the density function related to the contribution of each light path.

2.1 Efficient and Unbiased Free Path Sampling

Modern rendering methods utilize Monte Carlo techniques (e.g., Monte Carlo path tracing [4, 5] or Markov chain Monte Carlo techniques [5, 6]) to draw light paths according to some probability density function related to the contribution of each light path: in methods based on Monte Carlo integration, we integrate for each pixel on the image plane the contribution (radiance) of the density function related to the contribution of each light path. Modern rendering methods utilize Monte Carlo techniques (e.g., Monte Carlo path tracing [4, 5] or Markov chain Monte Carlo techniques [5, 6]) to draw light paths according to some probability density function related to the contribution of each light path.

A typical way for generating light paths in a participating medium is to iteratively sample the scattering direction and the distance (free path) \( t \geq 0 \) between successive scattering events. Given the current location \( x_0 \) and the direction \( d \) (in a unit vector), the probability density function \( p_{fp}(x) \) to generate the next scattering event (or equivalently the free path) is given by \( p_{fp}(x(t)|x_0, d) = \exp(-\tau(x_0, x(t)))\sigma(x(t)) \), where \( x(t) = x_0 + td \), \( \sigma \) is the extinction coefficient, \( \tau(x_0, x(t)) = \int_0^t \sigma(x_0 + t'd)dt' \) is the optical depth, and \( \exp(-\tau(x_0, x(t))) \) is the transmittance. \( p_{fp}(\cdot) \) represents the probability that the light travels freely without any interaction in the interval \((x_0, x(t))\) and then gets scattered or absorbed at \( x \). The corresponding cumulative distribution function is given by \( C_{fp}(x(t)|x_0, d) = 1 - \exp(-\tau(x_0, x(t))) \).

A typical way to sample the free path \( t \) is to use the inversion method: draw a random number \( \xi \in (0, 1) \), and solve for \( t \) such that \( C_{fp}(x(t)|x_0, d) = \xi \), or equivalently \( \tau(x_0, x(t)) = -\ln(1 - \xi) \). One way to find \( t \) is to use numerical integration to incrementally compute the optical depth (the resulting algorithm is called ray-marching [7] in the graphics community). However, insufficient number of quadrature points may result in an inaccurate optical depth. On the other hand, using many quadrature points may affect the performance.

In the nuclear science community, there is a technique called delta tracking (also known as Woodcock tracking or pseudo scattering) [8] for unbiased sampling of the free path. Delta tracking first computes an upper bound (can be any upper bound) of the extinction coefficient, called the majorant extinction coefficient \( \sigma_M \). It then fills the entire region with particles that incur null collisions (for convenience we call them ‘null particles’) of an amount of \( \sigma_M - \sigma(x) \) [9, 10], so the extinction coefficient is now \( \sigma_M \) everywhere. Those null particles are transparent: a light hitting a null particle will continue its journey along the same direction. Thus, the addition of the null particles will preserve the appearance of the medium. Since the medium is now uniform with the extinction coefficient \( \sigma_M \), the free path sampling can be performed analytically: the optical depth is now given by \( t\sigma_M \), and can be solved via \( t = -\ln(1 - \xi)/\sigma_M \). Now, we stop only when the light hits a true particle (in the probability \( \sigma(x)/\sigma_M \) and otherwise continue the journey in the same direction.

While delta tracking gives unbiased free path, it becomes less efficient for a more inhomogeneous medium, because we need to fill more null particles to have an homogeneous medium, resulting in a lower probability that the light hits a true particle. The idea of our method is to partition the entire space into a set of disjoint subregions, each with its own
majorant extinction coefficient, which can be usually set much tighter than that for the entire domain.

To perform unbiased sampling in the partitioned space, our approach is to ‘rewind’ the sampling once it crosses a partition. Suppose that we are currently in a subregion $i$, and there is another subregion $j$ adjacent to the subregion $i$ along the ray direction. $\sigma_M^{(i)}$ and $\sigma_M^{(j)}$ are the majorant extinction coefficients for the subregions $i$ and $j$, respectively. We start delta tracking in the subregion $i$ with $\sigma_M^{(i)}$. If we find a true collision in $i$, we accept it. Otherwise, the ray will leave $i$ and enter $j$. If this happens, we reset (rewind) the free path back to the crossing point between $i$ and $j$ along the ray, and perform delta tracking using $\sigma_M^{(j)}$ from there. This approach is proven to give unbiased sampling for the free path [1], and was also used in [11] to treat adjacent medium boundary.

The next question is then how we should partition the space. On one hand, we want to partition the space as much as possible, as long as the partition would bring enough reduction in $\sigma_M^{(i)}$ so we may enjoy performance improvement. But on the other hand, if we partitioned a homogeneous region, the rewinding process for crossing the partition would be a waste compared to the case with no partition. Taking this observation one step further, we have developed a mathematical framework [1, 2] to estimate the performance gain or loss for a given partition, which can then be used to determine the optimal partitioning location. We briefly explain our idea for a one dimensional case.

Suppose that $\sigma_M$ is the maximum value of $\sigma(x)$ in an interval $(s, t)$. In a single iteration of the delta tracking, the free path will proceed $1/\sigma_M$ on average. Thus, it takes $N = (t - s)\sigma_M$ iterations, on average, to go across this interval. This value is exactly the area of the bounding box of $\sigma(x)$ (blue box in the right inset) in the interval $(s, t)$. If we partition this interval at a location $q$, the average number of iterations for passing through each subregion is the bounding area of each subregion (red boxes). On top of their sum, we need an additional iteration for the rewinding process. Then, the reduction $N_r$ in the expected number of iterations when we partition the interval is given by the reduction of the total box areas (the brown box area $R$) minus one: $N_r = R - 1$. Thus, the partition that maximizes $R$ is the candidate for the partitioning location. In addition, we should stop partitioning if $R \leq 1$ because it offers no reduction in the cost. This strategy extends to the case when $\sigma(x)$ has multiple peaks. Formally, the problem to find the partitioning location in the one dimensional case reduces to a 2D largest empty rectangle problem [12] for the region bounded by $\sigma_M$ and $\sigma(x)$ (the gray region). We can also derive an analytical cost function for a 3D volume [2], and make use of various space partitioning data structures.

We now briefly summarize our results. With the ray-marching method with fixed sampling intervals, the results may have bias and may not converge to the exact solution (the exact solution was computed using the delta tracking with 16 cores in parallel in 5 days), as in the right inset. $4 \times$, $1 \times$, $1/4 \times$ and $1/16 \times$ in ray-marching mean the sampling intervals, represented as the multiples of $1/\sigma_M$ of the medium, respectively. ‘time*’ at the bottom of the right inset indicates the computation time to generate $10^6$ rays. Our method is unbiased and is approximately 10 times faster than delta tracking.

With our method, we may obtain an arbitrary speed up, depending on the inhomogeneity of the participating medium; as in the inset in the next page, the performance gain increases as
Cauchy stress fields, respectively. $\frac{\partial b}{\partial t}$ denotes the material derivative, clouds are about 10^2 to 10^3 times as large as those of the atmosphere. In our case, we observed a 380 times speed up over the delta tracking technique.

2.2. A Material Point Method for Shear Dependent Flows

Next, we review our work on how to simulate the motion of complex materials. We have focused on foams, because they arise in many problem domains [13], including firefighting, oil recovery, chemical filtration, industrial textile processing, and computer animation. Dense foams consist of a number of small individual bubbles, each in the order of 40 μm in diameter [14]. The observed macroscopic behavior of a dense foam is a seamless continuum exhibiting a non-Newtonian force response. Foams behave elastically when the applied stress is small. When the stress is large enough, the microscopic stochastic rearrangement of bubbles results in a macroscopically viscoplastic behavior, and the continuum flows more easily under larger stresses, known as the shear thinning effect. Because of these behaviors, foams may experience extreme shape changes.

We adopt the Material Point Method (MPM) (a hybrid particle/grid method) [15] because of its relative ease of handling large deformations and topological changes [16, 17]. The dynamics of the medium over time $t$ is described by the Euler-Lagrange partial differential equation

$p \frac{\partial b}{\partial t} = \nabla \cdot \sigma + \rho g,$

subject to the mass conservation condition $\frac{\partial b}{\partial t} + \rho \nabla \cdot v = 0$, where $\frac{\partial b}{\partial t}$ denotes the material derivative, $\rho$, $v$, and $\sigma$ are the scalar density, vectorial velocity, tensorial Cauchy stress fields, respectively. $g$ is the gravitational acceleration.

For the treatment of elasticity, we use the elastic part of the left Cauchy-Green tensor $b^c$. The application of a slight strain on the foam results in the Kirchhoff stress ([18]-§9.2)

$\tau = J \sigma = \frac{2}{J} \left(J^2 - 1\right) I + \mu \text{dev}[b^c],$

where $\kappa$ and $\mu$ are respectively the bulk and shear moduli,

$J = \sqrt{\text{det}(b^c)}, \quad \text{dev}[x] = x - \frac{1}{3} tr[x] I$ is the deviatoric operator, and $b^c = J^{-2/3} b^c$ is the volume preserving left-Cauchy Green tensor. We adopt the von Mises yield condition to determine the limit of the elastic regime: the material behaves elastically as long as $\Phi(s) = s - \sqrt{2/3} \sigma_Y \leq 0$ is satisfied, where $\sigma_Y$ is the yield stress and $s$ is the magnitude of the shear stress $s = \text{dev}[\tau]$.

We adopt the Herschel-Bulkley model [19] as the plastic constitutive relation for foam [13]. First, the time derivative of $b^c$ is given by the identity

$\dot{b}^c = (\nabla v) b^c + b^c (\nabla v)^T + \mathcal{L}_v b^c,$

where $\mathcal{L}_v b^c$ is the Lie derivative of $b^c$. While $(\nabla v) b^c + b^c (\nabla v)^T$ captures the change in the strain due to the velocity field, $\mathcal{L}_v b^c$ captures the change relative to the velocity (i.e., due to the plastic flow). In rheology, one adopts a flow rule to describe the plastic deformation, where excess elastic strain flows into plastic strain over time. We use the flow rule of Simo and Hughes (see [18]-§9,
\[ L_t b^e = -\frac{2}{3} \text{tr}[b^e] \gamma s, \] where \( \gamma \) is the flow rate, and \( s = s/s \) is the flow direction. With the Herschel-Bulkley model, the flow rate is given by \( \gamma(s) = \max(0, \left( \frac{\Phi(s)}{\eta^{1/h} \Phi^{1-1/h}} \right)^{1/h}) \), where \( \eta \) is the viscosity. For \( \Phi(s) \geq 0 \), we can rewrite \( \gamma(s) = \frac{\Phi}{\eta^{1/h} \Phi^{1-1/h}} \), where the denominator can be viewed as the effective viscosity coefficient from the vantage point of the standard viscoplastic model. This reveals the advantage of the Herschel-Bulkley model: by changing the parameter \( h \), we can capture a wide range of viscoplastic behaviors. Shear thinning and thickening materials are characterized by \( h < 1 \) and \( h > 1 \), respectively. Setting \( h = 1 \) recovers the viscoplastic (Bingham) model; taking the limit \( \eta \to 0 \) recovers perfect plasticity.

When we numerically advect material points, their distribution can become highly nonuniform. This happens, for example, when a sample of foam is compressed along one axis and stretched along a perpendicular axis. In this case, the uniformity of the material point distribution can rapidly degrade, producing a sparse sampling along the stretched axis, resulting in numerically induced, nonphysical voids (the left side of the right inset). We developed a resampling approach for the material point method to maintain a uniform point distribution (the right side of the right inset). We employ Poisson disk sampling to insert new points, and detect and merge points that are too close to one another.

Finally, we observe that an explicit tearing model is required to allow the foam to tear apart, because the grid maintains artificial topological connections at the grid resolution through the stencil of the shape functions. Without a tearing model, regions of foam can become progressively thinner, yielding artificially-slit threads (the left side of the right inset). We therefore developed an extension to directly detect and tear weak regions based on their accumulated plasticity (the right side of the right inset). We first identify regions that are in a ‘weakened’ or torn state. In these regions we directly modify the computed forces and the accumulation of strain so that the material no longer resists separation, nor does the geometric stretching accumulate as additional physical strain. After tearing has occurred, we gradually adjust the accumulated plasticity to model the recovery of bubble neighbor connectivity in these regions, to allow the foam to once again resists stretching.

We demonstrate the method’s efficacy with a comparison to recordings of real foam (the right inset), and with several animated examples (Figure 2). In the comparison with the recorded footage, we performed a real experiment by attaching a sample of real shaving foam to the base of a platform, acquiring an approximated shape of the foam using a 3D scanner (Kinect), and recording the motion of the foam subject to an oscillatory motion. Then, we simulated the acquired geometry, and compared the motion of our simulations to the recorded video. Initially, the foam oscillates and slowly begins to stretch downwards due to accumulated plasticity. As the oscillations continue, the foam exhibits typical nonlinear shear thinning behavior, and rapidly tears apart. Our simulations successfully capture the same characteristic behaviors of real foam. As in Figure 2 left, our method can be used to reproduce a pie throw example with a whipped cream, and as in Figure 2 right, our method can handle a (Bingham) viscoplastic material as
well as a shear-thickening material (like oobleck).

3. Designing Functional Man-Made Objects

Next, we show how the ability to incorporate physical laws can be used for designing functional man-made objects. We introduce our work on optics- and mechanics- based designs. For optics based design, we present our work on goal based lens design [21], with which we can shape the incident light and project an arbitrarily specified caustic image on a screen. For mechanics based design, we present our work for computing a layout of LEGO bricks given a 3D object model [22], that can be assembled in the real world, accounting for the stability of the sculpture.

3.1. Continuous Surface Generation for Goal Based Caustics

Given a desired image of caustics that we want to project onto a planar screen, our method computes the lens geometry that shapes the incident light via refraction and generates the desired caustics on the screen. In particular, our lens can produce high-resolution and complex patterns, such as natural images, with a continuous dynamic range (high quality even in regions of low intensity). In addition, our lens has a wide focus range: the caustic pattern is stable and perceptible even when the object is not perfectly in focus. This is important for the ease of the setting of the lens, light source and the screen.

Our method consists of two steps. In the first step, we compute a map between the light reaching the incident surface and that reaching the screen. This map describes how the light should be shaped (through the refraction) to produce the desired pattern. In the second step, we reconstruct the lens surface from this map.

We consider a collimated setting: the (parallel) light enters the incident surface perpendicularly without refraction (as in the right inset). The light refracts at the other side of the lens, called the refractive surface, and then reaches the screen. Each light path can be described by the relationship between a point \( p(x, y) \) on the incident surface and a point \( q(u, v) \) on the screen. In our method, we consider a one-to-one relationship between \( p(x, y) \) and \( q(u, v) \). We want to find from which regions of the incident surface we need to gather light in order to form the intensity distribution of the desired caustic pattern. This problem can be formulated as the problem of finding a mapping from \( q(u, v) \) to \( p(x, y) \) satisfying the following two conditions. The first condition is that the mapping is continuous (almost everywhere) over the entire domain. With a continuous mapping, the refractive surface also becomes continuous, enabling a stable projection of the caustics image. The second condition is that this mapping preserves the light energy: suppose that the light incident on an infinitesimal region \( dS \) around the point \( p(x, y) \) on the incident surface has intensity \( L(x, y) \), and that the intensity of this light on an infinitesimal region \( dS' \) around the point \( q(u, v) \) on the screen is \( L'(u, v) \), then we have \( LdS = L'dS' \).

We represent the mapping from \( q(u, v) \) to \( p(x, y) \) as a parameterization of \( x \) and \( y \) using \( u \) and \( v \). Under this parameterization, the ratio \( dS/dS' \) is equivalent to the determinant \( J \) of the Jacobian \( \frac{\partial p}{\partial q} \), hence, \( LJ = L' \). Directly computing this parameterization is difficult, we use a geometric flow approach instead. We start from an initial parameterization \( x = u \) and \( y = v \), and then maintain the continuity of the parameterization while gradually modifying it for a better satisfaction of the conservation of light energy. If the parameterization satisfies \( LJ = C \) for every light path, where \( C \) is the intensity of the caustics, then that mapping is the solution to the problem. Otherwise, there is a difference \( D = LJ - C \). We reduce this difference by
continuously modifying the point \( p(x, y) \) on the incident surface, while fixing the corresponding point \( q(u, v) \) on the screen.

First, we observe that there are regions with positive and negative differences \( D \). We regard the positive and negative regions as source and sink, respectively, analogous to those in computational fluid dynamics, where such sources and sinks can be relaxed via pressure gradient: we first solve Poisson’s equation \( \nabla^2 \phi = -D \) to find the pressure field \( \phi \), and then update the points \( p \) along the gradient of \( \phi \), \( \frac{\partial p}{\partial \phi} = \nabla \phi \) (see (a), (b), and (c) in the right inset for an example of the fields \( D \), \( \phi \), and \( \nabla \phi \), respectively). In our method, we repeatedly update the parameterization by computing the difference \( D = LJ - C \), solving the Poisson’s equation \( \nabla^2 \phi = -D \) and updating \( p \) through \( \frac{\partial p}{\partial \phi} = \nabla \phi \) (as in Figure 3).

We can compose our mapping with any divergence-free mapping to produce the same desired caustic pattern, as a divergence-free mapping preserves the area and hence the light energy. Since the parameterization is initially curl free, and our flow \( \nabla \phi \) is curl free, the parameterization obtained using our method contains no divergence-free components. In this sense, our mapping can be regarded as the ‘minimum’ mapping. We also note that as long as the difference \( D \) is non-zero, there will always be a non-zero flow, hence \( D = 0 \) is the only fixed point of the flow constructed above.

To reconstruct the lens surface (right inset), we find the surface in a least square sense, assuming that the refractive surface is a single-valued function, with its \( z \) coordinate represented as \( z = h(x, y) \). A vector \( \mathbf{N} \) normal to the refractive surface can be represented as \( \mathbf{N} = \left( \frac{\partial h}{\partial x}, \frac{\partial h}{\partial y}, -1 \right) \). Considering only the \( x \) and \( y \) components, we have \( \mathbf{N}_{xy} = \nabla h \). Taking the divergence of both sides, we obtain another Poisson’s equation (similar to [23]) \( \nabla^2 h = \nabla \cdot \mathbf{N}_{xy} \). Using the Snell’s law, the normal vector is related to the surface height \( h \) via \( \mathbf{N}_{xy} = k \frac{\mathbf{q} - \mathbf{p}}{\| \mathbf{q} - \mathbf{p} \|^2 + (H - h)^2 - (H - h)} \), where \( k = \eta \sqrt{\| \mathbf{q} - \mathbf{p} \|^2 + (H - h)^2 - (H - h)} \), \( \eta \) is the refractive index (1.49 for an acrylate basin), and \( H \) is the distance between the screen and the lens. We use an iterative approach to compute \( h \): we first initialize the refractive surface as a plane, and then iteratively compute the normal and update \( h \). This process converges usually only after a few iterations.

In the inset at the beginning of this subsection, we show an example of the fabricated lens and its caustics. In the right inset, we show examples with sharp features and characters. In each row, the subfigures from left to right show the photographs of the fabricated objects and their caustics, the simulated caustic patterns, and the caustic patterns of the fabricated objects.

### 3.2. Stability Analysis and Layout Optimization for LEGO Sculptures

Next, we present our work on optimizing designs with mechanical constraints in consideration. We have focused on computing brick layouts for LEGO sculptures. This LEGO construction problem is interesting and challenging [24], first because LEGO bricks are a versatile
building tool that is useful for many applications including rapid prototyping for personal design [25], yet finding a brick layout that can be used for the assembly in the real world requires us to predict the stability of the sculpture and to place bricks only at discretely possible locations, hence the optimization problem naturally has a combinatoric nature.

At the heart of our method is a force-based metric for stability estimation, which allows us to 1) compare the physical stability between different structures, and 2) determine if the resulting structure is stable enough for assembly. In addition, our force-based stability analysis tells us the weakest portion of the structure. Building atop our stability analysis, we iteratively update the layout by locally and randomly reconfiguring the region near the weakest portion. We only accept reconfigurations that give us better stability. This way, we obtain a monotonic increase of the stability. Previous methods [24, 26, 27, 28] lack this force-based metric (they build atop heuristic based metrics like the number of connected bricks), and may fail to predict the stability of the assembled sculpture. Their metrics also do not map monotonically to the physical stability.

Our stability analysis accounts for friction and normal forces: the bricks are held together by friction forces between the knobs (round dots) and cavities (in the back side), and are supported by neighboring bricks through the normal forces. We found it practical to model the maximum friction load using a constant $T$, rather than explicitly model it as a function of the normal force between the knob and cavity (see Appendix A of [22] for the details). Among the normal forces, there is a special set of forces that work horizontally between knobs and cavities. For the ease of computation, we treat the sum of such horizontal forces between each pair of connected bricks as support forces, and exclude them from the normal forces. This way, normal forces become (only) those have the non-negativity constraints, and support forces can take any value, assuming bricks will never fracture. Suppose that the bricks are placed according to the layout, the gravity will then induce forces and torques on the bricks. If the static friction, normal and support forces are able to perfectly counteract the gravity forces, the sculpture will remain static. Our stability analysis computes the force and torque balances in this static picture.

The force and torque balances give rise to a set of equality constraints, stating that the translational and rotational accelerations for every brick should be zero. Namely, for each brick $b_j$, we have $c_T(b_j) : \sum_{F_i \in F_j} F_i + m_{b_j} \mathbf{g} = 0$ as the force balance, and $c_T(b_j) : \sum_{F_i \in F_j} \mathbf{L}_i \times F_i = 0$ as the torque balance, where $F_i$ is the set of forces working on the brick $b_j$, $m_{b_j}$ is the mass of the brick $b_j$, $\mathbf{g}$ is the gravitational acceleration, and $\mathbf{L}_i$ is the arm vector, pointing from the center of the brick to the position where the force is assigned.

The friction and normal forces should satisfy non-negativity (inequality) constraints: $c_{F_j}(i) : 0 \leq F_i \in F_j$ and $c_{F_n}(i) : 0 \leq F_i \in F_n$, where $F_j$ and $F_n$ are the sets of all friction and normal forces, respectively. If all the friction forces are within the maximum friction load $T$, we can conclude the layout is stable. However, this style of stability assessment only tells us a binary result (whether it stays or collapses), and cannot be used to compare the stability between two different layouts. Instead of posing the maximum friction load as an inequality constraint, we consider the capacity $C_i = T - F_i$ of a friction force $F_i \in F_j$. If $C_i > 0$, the corresponding point can still accept additional forces. We define $C_m = \min_i C_i$ to indicate the smallest (weakest) capacity.

Usually, the answer of the force distribution for a static object is not unique (e.g., consider the case of a table with four legs). As long as the forces can be redistributed to make $C_m > 0$, the LEGO sculpture remains stable. We take this concept of force redistribution one step further to estimate what is the highest $C_m$ we can get. Namely, we find a force distribution $\{F_k^M\}$ that maximizes the smallest capacity $C_m$ subject to the linear equality and inequality constraints discussed above (i.e., $c_T, c_R, c_{F_j}$, and $c_{F_n}$), where $F_k^M \in F, F = F_j \cup F_s \cup F_n$, and $F_s$ is...
the set of support forces. With this force distribution \( \{ F_k^M \} \), we can compute the maximum capacity as \( C_M = \min_{F_i \in F} (T - F_i^M) \). \( C_M \) represents how much additional force the structure can accept for the case \( C_M \geq 0 \) or how much the forces are overflowing for the case \( C_M < 0 \), enabling us to compare the stability between different layouts with a larger \( C_M \) meaning more stable. Another important aspect of posing the relationships between the maximum friction load and the friction forces as the objective function is that we can now find a solution not only for stable cases but also for unstable ones (there will be no solution if they were posed as inequality constraints). This enables us to guide the layout refinement from an unstable structure towards a stable one. In addition, \( C_M > 0 \) naturally serves as a threshold for the stability, because it means there is a way to redistribute forces to make all capacities positive. Therefore, we can use \( C_M \) as the stability metric for a layout. Furthermore, we can compute \( F_i^w = \arg \min_{F_i \in F} (T - F_i^M) \), and identify the two bricks that share the contact point corresponding to \( F_i^w \). These two bricks are the weakest portion of the LEGO sculpture.

With the stability analysis at hand, we find a stable configuration via an iterative process. From a randomly initialized layout with the connection graph of all the bricks being single connected, and the fill out of the bricks conforming to the given shape and color, we iteratively perform random reconfiguration around the weakest portion, until \( C_M \) becomes positive. At each random reconfiguration, we only accept a new layout if its \( C_M \) value is larger.

With our method, we can create layout-sensitive sculptures like the Giraffe model with a slender neck to sustain a heavy head, as in the inset at the beginning of this subsection. Besides computing for a stable layout for a given shape, our method can be used to incorporate external weights as in the right inset. Figure 4 shows our result of a real-life sized table sculpture. We imposed external weights of 6kg in total equally divided to the table top. With this setting, we can put a laptop or a book on it, and use it as a real table.

4. Physics for Smart Robots

We have shown that the ability to incorporate physics is important and useful for computer animation to predict the objects’ appearance and motion, as well as for designing functional man-made objects that can shape the light and deal with mechanical constraints. Now, we briefly introduce a series of our work on developing smarter robots with the physics in consideration. We have focused on robotic folding of garments [29, 30, 31].
4.1. Unfolding Laundry Garments by a Robot
Deformable objects such as garments are highly unstructured, making them difficult to recognize and manipulate. In the first work, we focused on how to unfold a garment from a random pose to a desired configuration [32, 33, 34]. To unfold a garment, our approach is to grasp the garment at two desired points with the robot arms (as in Figure 5). These desired points are chosen such that we can likely succeed in flattening the garments by simply laying down the garments on the table while grasping at the desired points. For a sweater, for example, these points are at the middle of the left and right elbows. To grasp the garment at the desired grasping points, our idea is to let the robot iteratively try grasping the garment (each trial may fail to grasp the desired point), until the two robot arms simultaneously succeed in grasping the desired points. With this iterative approach, it is easy to cope with the inaccuracy during the process.

To perform the grasp and to identify the point the robot is actually grasping, a key challenge is the estimation of the garment pose. For this pose estimation, we prepared a garment database in advance using an offline physically based simulation. The database contains 37 garments with different types, including sweater, pants, shorts, dresses, and scarves, each with manually labeled key grasping points such as sleeve end, elbow, shoulder, chest, and waist. We hang each garment for each grasping point in the simulator and simulated the stationary state of the garment.

During the online manipulation, we use Kinect to scan the garment and reconstruct a 3D mesh. Then, we search for the database for the closest garment shape. Because the database is a sparse sample of (the continuously) possible poses, we further perform a non rigid registration to obtain a map between each point on the garment mesh in the database and the corresponding point in the scanned mesh. This way, we can map a desired point on the mesh in the database to the point in the scanned mesh, and tell the robot which point to try grasping next. After grasping at two desired points, the robot will proceed to place the garment on the table.

4.2. Robotic Ironing
Once the garment is flattened on the table, we perform robotic ironing next. We developed a novel solution to analyze the cloth surface by fusing two surface scan techniques: a curvature scan and a discontinuity scan. The curvature scan is performed via Kinect, which is good at capturing regions with large variation in height (which does not require an ironing), while permanent wrinkles, which correspond to high curvatures or discontinuities in the gradient of the garment shape, can be effectively captured via an illumination based approach.

We found that using this combination of a 3D depth scan and an illumination based approach allows us to robustly identify permanent wrinkles from other deformations without a carefully calibrated system. The 3D height map and the 2D illumination image are combined in a probabilistic framework. We then classify and rank the deformations for ironing, and compute
the trajectory for ironing. The robot aligns the principal axis of the iron with the target wrinkle, following the computed trajectory, and performs ironing. After several iterations, we obtain a desired ironed result with most of the permanent wrinkles removed.

4.3. Folding Garments with Path Optimization

Once the garment is flattened, the next task is to fold the garment. Previous methods [35, 36] have focused on folding the garment according to a predefined folding plan (including the path of the robot arm). However, the layout of the same folding action can vary in terms of the material properties such as cloth hardness and the environment such as friction between the garment and the table. Given the starting and ending folding positions, different folding trajectories will lead to different results. Thus, we have developed a novel method that learns optimal folding trajectory parameters from predicted thin shell simulations of similar garments, which can then be applied to a real garment folding task.

The input to our method is the material parameter of the garment, the friction between the garment and the table, the garment shape flattened on the table, the target folded shape, and the order to fold. To estimate the garment pose (and the label of each feature, like the sleeve) on the table, we capture the image and compute the contour of the garment. Then, we match the contour with predefined 2D templates, using a 2D version of the non rigid registration technique, to identify the labels of the feature points of the garment.

To optimize for the trajectory for folding the garment, our prototype approach tries to find the parameters for defining the trajectory curve.

We use a third order Bézier curve to describe the trajectory, and optimize for the control points of the curve, except for the two end points. Our state variable \( \mathbf{x} \) thus consists of the positions of the intermediate control points \( \mathbf{x} = (P_0^T, P_1^T, P_2^T, P_3^T) \). We solve for a minimization problem to find the optimal trajectory \( \mathbf{x}^* \): 

\[
\mathbf{x}^* = \arg \min_{\mathbf{x}} \{ C(\mathbf{x}) \}^2
\]

where \( C(\mathbf{x}) \) is a cost function that accounts for the trajectory length \( l_{\mathbf{x}} \) and the dissimilarity \( D(S_t, S_x) \) between the target folded garment shape \( S_t \) and the simulated folded shape \( S_x \) using the trajectory \( \mathbf{x} \):

\[
C(\mathbf{x}) = \alpha D(S_t, S_x) + l_{\mathbf{x}}.
\]

We used \( \alpha = 10^3 \) in our experiment. By minimizing the trajectory length as well as the dissimilarity, we aim at efficiency (i.e., reducing the time and energy for folding) as well as the correctness of the folding. We define the dissimilarity term as

\[
D(S_t, S_x) = \frac{1}{|S_t|} \int_{S_t} \| \mathbf{q}(\mathbf{y}) - \mathbf{y} \| \, dA,
\]

where \( |S_t| \) is the surface area of the garment, \( \mathbf{y} \in S_t \) is a point on the target folded shape \( S_t \), \( \mathbf{q}(\mathbf{y}) \in S_x \) is the corresponding point on the simulated folded shape, and \( dA \) is the area measure. For the optimization, we used a secant version of the Levenberg-Marquardt algorithm [37]. With our optimized folding trajectory, we can let the robot to follow the trajectory and perform robotic folding in the real world, as in Figure 7.

5. Concluding Remarks

In this paper, we reviewed and introduced a series of our past work done from the perspective of how understanding the physics can help to develop various computing tools for predictive simulations, designing functional man-made objects, and developing smarter robots. Especially,
we have focused on optics and mechanics, which are important for predicting the objects’ appearance and motion, as well as designing functional objects that can shape the light and remain static under gravitational and external forces. We have also shown how optics and mechanics can be incorporated to robotic folding.

The graphics field focuses on such physics problems more at a real world scale in a resolution of our perception level. The need for the ultimate realism may sometimes ask for more efficient and accurate methods than those available in other fields. With no surprise, many state-of-the-art methods have been developed in the graphics field, involving various interesting mathematical problems, like the ones we encountered: statistics, partial differential equations, geometric flow, discrete differential geometry, combinatoric problem, randomized approach, feature registration, and so on. However, there is still a big gap between what we can perceive in the real world and what we can simulate on a computer, and there is still a long and exciting research journey for us to embark. We believe that graphics can serve as a playground to test and develop computational techniques, and its advancement would be beneficial for other fields as well.

Acknowledgments
I would like to deeply appreciate my closest collaborators (listed in the alphabetical order of the last name) Peter K. Allen, Christopher Batty, Shih-Fu Chang, Bing-Yu Chen, Yu-Huan Chung, Yoshinori Dobashi, Eitan Grinspun, Xiuhu Han, Chun-Kai Huang, Sei Imai, Kei Iwasaki, Yinxiao Li, Sheng-Jie Luo, Tomoyuki Nishita, Breannan Smith, Yan Wang, Danfei Xu, and Changxi Zheng, with whom we have worked together for the work presented in this review paper. The work was supported in part by Grant-in-Aid for JSPS Fellows (20-7968), JSPS Grant-in-Aid for Research Activity Start-up (23800011), JSPS Postdoctoral Fellowships for Research Abroad, NSF (Grants IIS-13-19483, CMMI-11-29917, CAREER-1453101, IIS-1217904), NSERC (Grant RGPIN-04360-2014), Ministry of Science and Technology (MOST103-2218-E-002-030, MOST103-2221-E-002-158-MY3), National Science Council of Taiwan (NSC99-2221-E-002-016), Intel, The Walt Disney Company, Autodesk, Side Effects, NVIDIA, Adobe, The Foundry, MTM Corporation (Japan), and Taktktile LLC.

References
[1] Y. Yue, K. Iwasaki, B.-Y. Chen, Y. Dobashi, and T. Nishita. Unbiased, adaptive stochastic sampling for rendering inhomogeneous participating media. ACM Transactions on Graphics (Proc. of SIGGRAPH ASIA 2010), 29(6):177:1–177:8, 2010.
[2] Y. Yue, K. Iwasaki, B.-Y. Chen, Y. Dobashi, and T. Nishita. Toward optimal space partitioning for unbiased, adaptive free path sampling of inhomogeneous participating media. Computer Graphics Forum (Proc. of Pacific Graphics 2011), 30(7):1911–1919, 2011.
[3] Y. Yue, B. Smith, C. Batty, C. Zheng, and E. Grinspun. Continuum foam: A material point method for shear-dependent flows. ACM Transactions on Graphics (Presented at SIGGRAPH Asia 2015), 34(5):160:1–160:20, 2015.
[4] J. T. Kajiya. The rendering equation. Computer Graphics (Proc. of SIGGRAPH 86), 20(4):143–150, 1986.
[5] E. Veach. Robust Monte Carlo methods for light transport simulation. PhD thesis, Stanford University, 1998.
[6] M. Pauly, T. Kollig, and A. Keller. Metropolis light transport for participating media. In Proc. of EUROGRAPHICS Workshop on Rendering 2000, pages 11–22, 2000.
[7] K. Perlin and E. M. Hoffert. Hypertexture. Computer Graphics (Proc. of SIGGRAPH 89), 23(3):253–262, 1989.
[8] E.R. Woodcock, T. Murphy, P.J. Hemnings, and T.C. Longworth. Techniques used in the GEM code for Monte Carlo neutronics calculations in reactors and other systems of complex geometry. In Proc. of Conference on the Application of Computing Methods to Reactor Problems, ANL-7050, pages 557–579, 1965.
[9] M. Galtier, S. Blanco, C. Caliot, C. Coustet, J. Dauchet, M. El Hafi, V. Eymet, R. Fournier, J. Gautrais, A. Khoung, B. Piaud, and G. Terze. Integral formulation of null-collision Monte Carlo algorithms. Journal of Quantitative Spectroscopy and Radiative Transfer, 125:57–68, 2013.
[10] P. Kutz, R. Habel, Y. K. Li, and J. Novák. Spectral and decomposition tracking for rendering heterogeneous volumes. ACM Transactions on Graphics (Proc. of SIGGRAPH 2017), 36(4):111:1–111:16, 2017.
[11] L.L. Carter, E.D. Cashwell, and W.M. Taylor. Monte Carlo sampling with continuously varying cross sections along flight paths. Nuclear Science and Engineering, 48:403–411, 1972.
[12] A. Aggarwal and S. Suri. Fast algorithms for computing the largest empty rectangle. In Proc. of Third Annual Symposium on Computational Geometry, pages 278–290, 1987.

[13] D. Weaire and S. Hutzler. The Physics of Foams. Oxford University Press, 2001.

[14] G. Ovarlez, K. Krishan, and S. Cohen-Addad. Investigation of shear banding in three-dimensional foams. EPL (Europhysics Letters), 91(6):68005:1–68005:6, 2010.

[15] D. Sulsky, S.-J. Zhou, and H. L. Schreyer. Application of a particle-in-cell method to solid mechanics. Computer Physics Communications, 87(1–2):236–252, 1995.

[16] A. Stomakhin, C. Schroeder, L. Chai, J. Teran, and A. Selle. A material point method for snow simulation. ACM Transactions on Graphics (Proc. of SIGGRAPH 2013), 32(4):102:1–102:9, 2013.

[17] A. Stomakhin, C. Schroeder, C. Jiang, L. Chai, J. Teran, and A. Selle. Augmented MPM for phase-change and varied materials. ACM Transactions on Graphics (Proc. of SIGGRAPH 2014), 33(4):138:1–138:11, 2014.

[18] J. C. Simo and T. J. R. Hughes. Computational Inelasticity. Springer, 1998.

[19] W. H. Herschel and R. Bulkley. Konsistenzmessungen von Gummi-Benzollösungen. Kolloid-Zeitschrift & Zeitsschrift für Polymere, 39(4):291–300, 1926.

[20] J. C. Simo. A framework for finite strain elastoplasticity based on maximum plastic dissipation and the multiplicative decomposition: Part I. continuum formulation. Computer Methods in Applied Mechanics and Engineering, 66(2):199–219, 1988.

[21] Y. Yue, K. Iwasaki, B.-Y. Chen, Y. Dobashi, and T. Nishita. Poisson-based continuous surface generation for goal-based caustics. ACM Transactions on Graphics (Presented at SIGGRAPH 2014), 33(3):31:1–31:7, 2014.

[22] S.-J. Luo, Y. Yue, C.-K. Huang, Y.-H. Chung, S. Imai, T. Nishita, and B.-Y. Chen. Legoization: Optimizing LEGO designs. ACM Transactions on Graphics (Proc. of SIGGRAPH ASIA 2015), 34(6):222:1–222:12, 2015.

[23] Y. Yu, K. Zhou, D. Xu, X. Shi, H. Bao, B. Guo, and H.-Y. Shum. Mesh editing with poisson-based gradient field manipulation. ACM Transactions on Graphics (Proc. of SIGGRAPH 2004), 23(3):644–651, 2004.

[24] R. Gower, A. Heydtmann, and H. Petersen. LEGO: automated model construction. In Proc. of 32nd European Study Group with Industry, pages 81–94, 1998.

[25] S. Mueller, T. Mohr, K. Guenther, J. Frohnhofen, and P. Baudisch. faBrickation: fast 3D printing of functional objects by integrating construction kit building blocks. In Proc. of SIGCHI 2014, pages 3827–3834, 2014.

[26] P. Petrovic. Solving the LEGO brick layout problem using evolutionary algorithms. In Proc. of NIK 2001, 2001.

[27] R. Testuz, Y. Schwartzburg, and M. Pauly. Automatic generation of constructable brick sculptures. In Eurographics 2013 Short papers, pages 81–84, 2013.

[28] L. van Zijl and E. Smal. Cellular automata with cell clustering. In Proc. of Automata 2008, pages 425–440, 2008.

[29] Y. Li, D. Xu, Y. Yue, Y. Wang, S.-F. Chang, E. Grinspun, and P. K. Allen. Regrapping and unfolding of garments using predictive thin shell modeling. In Proc. of the IEEE International Conference on Robotics and Automation (ICRA) 2015, pages 1382–1388. IEEE, 2015.

[30] Y. Li, X. Hu, D. Xu, Y. Yue, E. Grinspun, and P. K. Allen. Multi-sensor surface analysis for robotic ironing. In Proc. of the IEEE International Conference on Robotics and Automation (ICRA) 2016, 2016.

[31] Y. Li, Y. Yue, D. Xu, E. Grinspun, and P. K. Allen. Folding deformable objects using predictive simulation and trajectory optimization. In Proc. of the 27th IEEE/RSJ International Conference on Intelligent Robots and Systems (IROS) 2015, 2015.

[32] M. Casasmano-Towner, A. Singh, S. Miller, J. F. O’Brien, and P. Abbeel. Bringing clothing into desired configurations with limited perception. In Proc. of IEEE International Conference on Robotics and Automation (ICRA) 2011, pages 3893–3900, 2011.

[33] A. Doumanoglou, A. Kargakos, T. K. Kim, and S. Malassiotis. Autonomous active recognition and unfolding of clothes using random decision forests and probabilistic planning. In Proc. of IEEE International Conference on Robotics and Automation (ICRA) 2014, pages 987–993, 2014.

[34] F. Osawa, H. Seki, and Y. Kamiya. Unfolding of massive laundry and classification types of dual manipulator. Journal of Advanced and Intelligent Informatics, 11(5), 2007.

[35] S. Miller, M. Fritz, T. Darrel, and P. Abbeel. Parametrized shape models for clothing. In Proc. of IEEE International Conference on Robotics and Automation (ICRA) 2011, 2011.

[36] S. Miller, J. van den Berg, M. Fritz, T. Darrell, K. Goldberg, and P. Abbeel. A geometric approach to robotic laundry folding. The International Journal of Robotics Research, 31(2):249–267, 2012.

[37] K. Madsen, H. B. Nielsen, and O. Tingleff. Methods for non-linear least squares problems (2nd ed.), 2004.