Cloth in the Wind
A Case Study of Physical Measurement through Simulation
Runia, T.F.H.; Gavrilyuk, K.; Snoek, C.G.M.; Smeulders, A.W.M.

DOI
10.1109/CVPR42600.2020.01051

Publication date
2020

Document Version
Final published version

Published in
2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition

License
Article 25fa Dutch Copyright Act (https://www.openaccess.nl/en/in-the-netherlands/you-share-we-take-care)

Citation for published version (APA):
https://doi.org/10.1109/CVPR42600.2020.01051
Abstract

For many of the physical phenomena around us, we have developed sophisticated models explaining their behavior. Nevertheless, measuring physical properties from visual observations is challenging due to the high number of causally underlying physical parameters – including material properties and external forces. In this paper, we propose to measure latent physical properties for cloth in the wind without ever having seen a real example before. Our solution is an iterative refinement procedure with simulation at its core. The algorithm gradually updates the physical model parameters by running a simulation of the observed phenomenon and comparing the current simulation to a real-world observation. The correspondence is measured using an embedding function that maps physically similar examples to nearby points. We consider a case study of cloth in the wind, with curling flags as our leading example – a seemingly simple phenomena but physically highly involved. Based on the physics of cloth and its visual manifestation, we propose an instantiation of the embedding function. For this mapping, modeled as a deep network, we introduce a spectral layer that decomposes a video volume into its temporal spectral power and corresponding frequencies. Our experiments demonstrate that the proposed method compares favorably to prior work on the task of measuring cloth material properties and external wind force from a real-world video.

1. Introduction

There is substantial evidence [17, 10] that humans run mental models to predict physical phenomena. We predict the trajectory of objects in mid-air, estimate a liquid’s viscosity and gauge the velocity at which an object slides down a ramp. In analogy, simulation models usually optimize their parameters by performing trial runs and selecting the best. Over the years, physical models of the world have become so visually appealing through simulations and rendering [46, 28, 7, 36] that it is worthwhile to consider them for physical scene understanding. This alleviates the need for meticulous annotation of the pose, illumination, texture and scene dynamics as the model delivers them for free.
Accordingly, we propose to learn a world observation of the particular phenomenon (Figure 1). To evaluate our method, we record real-world video of flags and a 3D engine with unknown parameters $\theta$. 3D meshes, points clouds, and simulations (e.g., tree branches [53, 41], water surfaces [40], and hanging cloth [6, 54, 47, 9]. Our leading example of a flag curling in the wind may appear simple at first, but its motion is highly complex. Its dynamics are an important and well-studied topic in the field of fluid-body interactions [37, 42, 43]. Inspired by this work and existing visual cloth representations that characterize wrinkles, folds and silhouette [4, 14, 49, 55], we propose a novel spectral decomposition layer which encodes the frequency distribution over the cloth’s surface.

Previous work has considered the task of measuring intrinsic cloth parameters [4, 6, 54] or external forces [9] from images or video. Notably, Bouman et al. [6] use complex steerable pyramids to describe hanging cloth in a video, while both Yang et al. [54] and Cardona et al. [9] propose a learning-based approach by combining a convolutional network and recurrent network. In our experiments we will compare our cloth frequency-based representations with Cardona et al. [9] on flags while Yang et al. [54] is a reference on the hanging cloth dataset of Bouman et al. [6].

Our approach of measuring physical parameters by iterative refinement of simulations shares similarity to the Monte Carlo-based parameter optimization of [51] and the particle swarm refinement of clothing parameters from static images [55]. In particular, the work of [55] resembles ours as they infer garment properties from images for the purpose of virtual clothing try-on. However, our work is different in an important aspect: we estimate intrinsic and extrinsic physical parameters from video while their work focuses on estimating intrinsic cloth properties from static equilibrium images. Recently, Liang et al. [24] have proposed a differentiable cloth simulator which could potentially be used as an alternative to our approach for cloth parameter estimation.

2. Related Work

Previous work has measured physical properties by perceiving real-world objects or phenomena – including material properties [11], cloth stiffness and bending parameters [6, 54], mechanical features [51, 26, 27, 23], fluid characteristics [50, 40, 35] and surface properties [25]. The primary focus of the existing literature has been on estimating intrinsic material properties from visual input. However, physical phenomena are often described by the interaction between intrinsic and extrinsic properties. Therefore, we consider the more complex scenario of jointly estimating intrinsic material properties and extrinsic forces from a single real-world video through the iterative refinement of physics simulations.

Our case study focuses on the physics of cloth and flags, both of which belong to the broader category of wind-excited bodies. The visual manifestation of wind has received modest attention in computer vision, e.g., the oscillation of tree branches [53, 41], water surfaces [40], and hanging cloth [6, 54, 47, 9]. Previous work on flags while Yang et al. [54] and Cardona et al. [9] propose a learning-based approach by combining a convolutional network and recurrent network. In our experiments we will compare our cloth frequency-based representations with Cardona et al. [9] on flags while Yang et al. [54] is a reference on the hanging cloth dataset of Bouman et al. [6].

Our approach of measuring physical parameters by iterative refinement of simulations shares similarity to the Monte Carlo-based parameter optimization of [51] and the particle swarm refinement of clothing parameters from static images [55]. In particular, the work of [55] resembles ours as they infer garment properties from images for the purpose of virtual clothing try-on. However, our work is different in an important aspect: we estimate intrinsic and extrinsic physical parameters from video while their work focuses on estimating intrinsic cloth properties from static equilibrium images. Recently, Liang et al. [24] have proposed a differentiable cloth simulator which could potentially be used as an alternative to our approach for cloth parameter estimation.

3. Method

We consider the scenario in which we make an observation of some phenomena with a physical model explaining its manifestation available to us. Based on the perception of reality, our goal is to measure the $D_p$ unknown continuous parameters of the physical model $\theta \in \mathbb{R}^{D_p}$, consisting of intrinsic parameters $\theta_i$ and extrinsic parameters $\theta_e$ through an iterative refinement of a computer simulation that implements the physical phenomena at hand. In particular, we consider observations in the form of short video clips $X_{\text{target}} \in \mathbb{R}^{C \times N_t \times H \times W}$, with $C$ denoting the number of image channels and $N_t$ the number of $H \times W$ frames. In each
iteration, the simulator runs with current model parameters $\theta$ to produce some intermediate representation (e.g. 3D meshes, point clouds or flow vectors), succeeded by a render engine with parameters $\zeta$ that yields a simulated video clip $x_{\text{sim}} \in \mathbb{R}^{C \times N_i \times H \times W}$. Our insight is that the physical similarity between real-world observation and simulation can be measured in some embedding space using pairwise distance:

$$D_{i,j} = D(s_{\phi}(x_i), s_{\phi}(x_j)) : \mathbb{R}^{D_{\theta}} \times \mathbb{R}^{D_{\zeta}} \rightarrow \mathbb{R}$$  \hspace{1cm} (1)

where $s_{\phi}(x) : \mathbb{R}^{C \times N_i \times H \times W} \rightarrow \mathbb{R}^{D_{\theta}}$ an embedding function parametrized by $\phi$ that maps the data manifold $\mathbb{R}^{C \times N_i \times H \times W}$ to some embedding manifold $\mathbb{R}^{D_{\theta}}$ on which physically similar examples should lie close. In each iteration, guided by the pairwise distance (1) between real and simulated instance, the physical model is refined to maximize physical similarity. This procedure ends whenever the physical model parameters have been measured accurately enough or when the evaluation budget is finished. The output comprises the measured physical parameters $\theta^*$ and corresponding simulation $x_{\text{sim}}^*$ of the real-world phenomenon. An overview of the proposed method is presented in Figure 3.

3.1. Physical Similarity

For the measurement to be successful, it is crucial to measure the similarity between simulation $x_{\text{sim}}$ and real-world observation $x_{\text{target}}$. The similarity function must reflect correspondence in physical dynamics between the two instances. The prerequisite is that the physical model must describe the phenomenon’s behavior at the scale that coincides with the observational scale. For example, the quantum mechanical understanding of a pendulum will be less meaningful than its formulation in classical mechanics when capturing its appearance using a regular video camera.

Given the physical model and its implementation as a simulation engine, we generate a dataset of simulations with its parameters $\theta$ randomly sampled from some predefined search space. For each of these simulated representations of the physical phenomenon, we use a 3D render engine to generate multiple video clips $x_{\text{sim}}^i$ with different render parameters $\zeta^i$. As a result, we obtain a dataset with multiple renders for each simulation instance. Given this dataset we propose the following training strategy to learn a distance metric quantifying the physical similarity between observations.

We employ a contrastive loss [15] and consider positive example pairs to be rendered video clips originating from the same simulation (i.e. sharing physical parameters) while negative example pairs have different physical parameters. Both rendered video clips of an example pair are mapped to the embedding space through $s_{\phi}(x)$ in Siamese fashion [8]. In the embedding space, the physical similarity will be evaluated using the squared Euclidean distance: $D_{i,j} = D(s_{\phi}(x_i), s_{\phi}(x_j)) = \|s_{\phi}(x_i) - s_{\phi}(x_j)\|_2^2$. If optimized over a collection of rendered video clips, the contrastive loss asserts that physically similar examples are pulled together, whereas physically dissimilar points will be pushed apart. As a result, by training on simulations only, we can learn to measure the similarity between simulations and the real-world pairs.

3.2. Simulation Parameter Optimization

We will arrive at a measurement through gradual refinement of the simulations (Figure 3). To optimize the physical parameters we draw the parallel with the problem of hyperpa-
rameter optimization [39, 3]. In light of this correspondence, our collection of model parameters is analogous to the hyperparameters involved by training deep neural networks (e.g., learning rate, weight decay, dropout). Formally, we seek to find the global optimum of physical parameters:

\[
\theta^* = \arg \min_{\theta} D \left( s_\theta(x_{\text{target}}), s_\theta(x_{\text{sim}}) \right),
\]

where the target example is fixed and the simulated example depends on the current set of physical parameters \( \theta \). Adjusting the parameters \( \theta \) at each iteration is challenging as it is hard to make parametric assumptions on (2) as function of \( \theta \) and accessing the gradient is costly due to the simulations’ computational complexity. Our goal is, therefore, to estimate the global minimum with as few evaluations as possible. Considering this, we adopt Bayesian optimization [39] for updating parameters \( \theta \). Its philosophy is to leverage all available information from previous observations of (2) and not only use local gradient information. We treat the optimization as-is and use a modified implementation of Spearmint [39] with the Matérn52 kernel and improved initialization of the acquisition function [29]. Note that the embedding function \( s_\theta(x) \) is fixed throughout this optimization.

4. Physics, Simulation and Appearance of Cloth

Up until now, we have discussed the proposed method in general terms and made no assumptions on physical phenomena. In this paper, we will consider two cases of cloth exposed to the wind: curling flags and hanging cloth (Figure 4). To proceed, we need to confine the parameters \( \theta \) and design an appropriate embedding function \( s_\theta(x) \).

4.1. Physical Model

The physical understanding of cloth and its interaction with external forces has been assimilated by the computer graphics community. Most successful methods treat cloth as a mass-spring model: a dense grid of point masses organized in a planar structure, inter-connected with different types of springs which properties determine the fabric’s behavior [1, 33, 46, 2, 28]. We adopt Wang’s et al. [46] non-linear and anisotropic mass-spring model for cloth. This model uses a piecewise linear bending and stretching formulation. The stretching model is a generalization of Hooke’s law for continuous media [38]. As our experiments focus on flags in the wind for which the stretching properties are of minimal relevance, our experiments will focus on flags in the wind, typically made of strong weather-resistant material such as polyester and nylon. Therefore, the material’s stretching properties are of minimal relevance and we will emphasize on the cloth’s bending model [46] and external forces [48].

Bending Model (\( \theta_e \)). The bending model is based on the linear bending force equation first proposed in [7]. The model formulates the elastic bending force \( F_e \) over triangular meshes sharing an edge (Figure 4). For two triangles separated by the dihedral angle \( \varphi \), the bending force reads:

\[
F_e = k_e \sin(\varphi/2)(N_1 + N_2)^{-1}|E|u,
\]

where \( k_e \) is the material dependent bending stiffness, \( N_1, N_2 \) are the weighted surface normals of the two triangles, \( E \) represents the edge vector and \( u \) is the bending mode (see Figure 1 in [7]). The bending stiffness \( k_e \) is non-linearly related to the dihedral angle \( \varphi \). This is realized by treating \( k_e \) as piecewise linear function of the reparametrization \( \alpha = \sin(\varphi/2)/(N_1 + N_2)^{-1} \). After this reparametrization, for a certain fabric, the parameter space is sampled for \( N_0 \) angles yielding a total of \( 3N_0 \) parameters across the three directions. Wang et al. [46] empirically found that 5 measurements are sufficient for most fabrics, producing 15 bending parameters.

External Forces (\( \theta_d \)). For the dynamics of cloth, we consider two external forces acting upon its planar surface. First, the Earth’s gravitational acceleration \( (F_g = ma_g) \) naturally pushes down the fabric. The total mass is defined by the cloth’s area weight \( \rho_A \) multiplied by surface area. More interestingly, we consider the fabric exposed to a constant wind field. Again, modeling the cloth as a grid of point masses, the drag force on each mass is stipulated by Stokes’s equation \( F_d = 6\pi R n v_w \) in terms of the surface area, the air’s dynamic viscosity and the wind velocity \( v_w \) [48, 28]. By all means, this is a simplification of reality. Our model ignores terms associated with the Reynolds number (such as the cloth’s drag coefficient), which will also affect a real cloth’s dynamics. However, it appears that the model is accurate enough to cover the spectrum of cloth dynamics.
we independently optimize the 15 bending coefficients instead of only tuning the one-dimensional multiplier. The full parameter search space is listed in Table 1.

<table>
<thead>
<tr>
<th>Parameter</th>
<th>Params</th>
<th>Search space</th>
</tr>
</thead>
<tbody>
<tr>
<td>$\theta_i$</td>
<td>15</td>
<td>$k_e \in [10^{-1} k_e, 10 k_e]$</td>
</tr>
<tr>
<td>$\theta_f$</td>
<td>1</td>
<td>$\rho_A \in [0.10, 0.17] , \text{kg m}^{-2}$</td>
</tr>
<tr>
<td>$\theta_e$</td>
<td>1</td>
<td>$\gamma_w \in [0, 10] , \text{m s}^{-1}$</td>
</tr>
</tbody>
</table>

### 4.2. Simulation Engine

We employ the non-differentiable ArcSim simulation engine [28] which efficiently implements the complex physical model described in Section 4.1. On top of the physical model, the simulator incorporates anisotropic remeshing to improve detail in densely wrinkled regions while coarsening flat regions. As input, the simulator expects the cloth’s initial mesh, its material properties and the configuration of external forces. At each time step, the engine solves the system for implicit time integration using a sparse Cholesky-based solver. This produces a sequence of 3D cloth meshes based on the physical properties of the scene. As our goal is to learn a physical distance metric in image space between simulation and real-world observation, we pass the sequence of meshes through a 3D render engine [5]. Given render parameters $\zeta$ comprising of camera position, scene geometry, lighting conditions and the cloth’s visual texture, the renderer produces a simulated video clip ($x_{\text{sim}}$) which we can compare directly to the real-world observation ($x_{\text{target}}$). We emphasize that our focus is neither on inferring render parameters $\zeta$ from observations nor on attaining visual realism for our renders.

### Parameter Search Space ($\theta_i, \theta_e$).

The ArcSim simulator [28] operates in metric units, enabling convenient comparison with real-world dynamics. As the base material for our flag experiments, we use “Camel Ponte Roma” from [46]. Made of 60% polyester and 40% nylon, this material closely resembles widely used flag fabrics [46]. The fabric’s bending coefficients, stretching coefficients, and area weight were accurately measured in a mechanical setup by the authors. We adopt and fix their stretching parameters and use the bending stiffness and area weight as initialization for our cloth material. Specifically, using their respective parameters we confine a search space that is used during our parameter refinement. We determine $\rho_A \sim \text{Uniform}(0.10, 0.17) \, \text{kg m}^{-2}$ after consulting various flag materials at online retailers. And, we restrict the range of the bending stiffness coefficients by multiplying the base material’s $k_e$ in (3) by $10^{-1}$ and 10 to obtain the most flexible and stiffest material respectively. As the bending coefficients have a complex effect on the cloth’s appearance, we independently optimize the 15 bending coefficients instead of only tuning the one-dimensional multiplier. The full parameter search space is listed in Table 1.

### 4.3. Spectral Decomposition Network

The dominant source of variation is in the geometry of the waves in cloth rather than in its texture. Therefore, we seek a perceptual model that can encode the cloth’s dynamics such as high-frequency streamwise waves, the number of nodes in the fabric, violent flapping at the trailing edge, rolling motion of the corners and its silhouette [37, 42, 13]. As our goal is to measure sim-to-sim and sim-to-real similarity, a crucial underpinning is that our embedding function is able to disentangle and extract the relevant signal for domain adaptation [32, 20]. Therefore, we propose modeling the spatial distribution of temporal spectral power over the cloth’s surface. Together with direction awareness, this effectively characterizes the traveling waves and flapping behavior from visual observations.

#### Spectral Decomposition Layer.

The proposed solution is a novel spectral decomposition layer that distills temporal frequencies from a video. Specifically, similar to [34], we treat an input video volume as a collection of signals for each spatial position ($i.e., H \times W$ signals) and map the signals into the frequency domain using the Discrete Fourier Transform (DFT) to estimate the videos’ spatial distribution of temporal spectral power. The DFT maps a signal $f[n]$ for $n \in [0, N_t-1]$ into the frequency domain [30] as formalized by:

$$ F(\omega) = \sum_{n=0}^{N_t-1} f[n] e^{-j \omega n T}. \quad (4) $$

We proceed by mapping the DFT’s complex output to a real-valued representation. The periodogram of a signal is a representation of its spectral power and is defined as $I(\omega) = \frac{1}{N_t} |F(j \omega)|^2$ with $F(j \omega)$ as defined in (4). This provides the spectral power magnitude at each sampled frequency. To effectively reduce the dimensionality and emphasize on the videos’ discriminative frequencies, we select the top-$k$ strongest frequencies and corresponding spectral power from the periodogram. Given a signal of arbitrary length, this produces $k$ pairs containing $I(\omega_{\text{max}}, i)$ and $\omega_{\text{max}}$, for $i \in [0, k]$ yielding a total of $2k$ scalar values.

Considering an input video volume, treated as a collection of $H \times W$ signals of length $N_t$, the procedure extracts the discriminative frequency and its corresponding power at each spatial position. In other words, the spectral decomposition layer performs the mapping $\mathbb{R}^{C \times N_t \times H \times W} \rightarrow \mathbb{R}^{2kC \times H \times W}$. The videos’ temporal dimension is squeezed and the result can be considered a multi-channel feature map – to be further processed by any 2D convolutional layer. We reduce spectral leakage using a Hanning window before applying the DFT. The batched version of the proposed layer is formalized as algorithm in the supplementary material.
Figure 5. Overview of our SDN architecture $s_\phi(x)$ for learning the physical correspondence between the simulation and real-world observation of dynamic flags. Given a 3D video volume as input, we first apply a 0th-order temporal Gaussian filter followed by two directional 1st-order Gaussian derivative filters and then spatially subsample both filtered video volumes by a factor two. The proposed spectral decomposition layer then applies the Fourier transform and selects the maximum power and corresponding frequencies densely for all spatial locations. This produces 2D multi-channel feature maps which we process with 2D ResNet blocks to learn the embedding.

Embedding Function. The specification of $s_\phi(x)$, with the spectral decomposition layer at its core, is illustrated in Figure 5. First, our model convolves the input video $x$ with a temporal Gaussian filter followed by two spatially oriented first-order derivative filters. Both resulting video volumes are two-times spatially subsampled by means of max-pooling. Successively, the filtered video representations are fed through the spectral decomposition layer to produce spectral power and frequency maps. The outputs are stacked into a multi-channel feature map to be further processed by a number of 2D convolutional filters with trainable weights $\phi$. We use 3 standard ResNet blocks [16] and a final linear layer that maps to the $\mathbb{R}^{D_c}$ embedding space. We refer to our network as Spectral Decomposition Network (SDN).

Network Details. Our network is implemented in PyTorch [31] and is publicly available¹. Unless mentioned otherwise, all network inputs are temporally sampled at 25 fps. After that, we use a temporal Gaussian with $\sigma_t = 1$ and first-order Gaussian derivative filters with $\sigma_{x,y} = 2$. For training the embedding function with the contrastive loss, we adopt a margin of 1 and use the BatchAll sampling strategy [18, 12]. The spectral decomposition layer selects the single most discriminative frequency (i.e., $k = 1$). Adding secondary frequency peaks to the feature maps did not yield substantial performance gains. The size of our embeddings is fixed ($D_c = 512$) for the paper. Input video clips of size $224 \times 224$ are converted to grayscale. We optimize the weights of the trainable ResNet blocks using Adam [22] with mini-batches of 32, learning rate $10^{-2}$ and a weight decay of $2 \cdot 10^{-3}$.

5. Real and Simulated Datasets

Real-world Flag Videos. To evaluate our method’s ability to infer physical parameters from real-world observations, we have set out to collect video recordings of real-world flags with ground-truth wind speed. We used two anemometers (Figure 6) to measure the wind speed at the flag’s position. After calibration and verification of the meters, we hoisted one of them in the flagpole to the center height of the flag to ensure accurate and local measurements. A Panasonic HC-V770 camera was used for video recording. In total, we have acquired more than an hour of video over the course of 5 days in varying wind and weather conditions. We divide the dataset in 2.7K train and 1.3K non-overlapping test video clips and use 1-minute average wind speeds as ground-truth. The train and test video clips are recorded on different days with varying weather conditions. Examples are displayed in Figure 6 and the dataset is available through our website.

FlagSim Dataset. To train the embedding function $s_\phi(x)$ as discussed in Section 3.1, we introduce the FlagSim dataset consisting of flag simulations and their rendered animations. We simulate flags by random sampling a set of physical parameters $\theta$ from Table 1 and feed them to ArcSim. For each flag simulation, represented as sequence of 3D meshes, we use Blender [5] to render multiple flag animations $x_{\text{sim}}^i$ at different render settings $\zeta^i$. We position the camera at a varying distance from the flagpole and assert that the cloth surface is visible by keeping a minimum angle of $15^\circ$ between the wind direction and camera axis. From

Figure 6. Left: Two anemometers used for gauging the wind speed. Right top: Real flag recordings with corresponding wind speeds measured by the anemometer hoisted in the flagpole. Right bottom: simulated examples from our FlagSim dataset.

---

¹https://tomrunia.github.io/projects/cloth/
work on the task of real-world wind speed regression. This involves using RMSE and accuracy within 0.5 m/s to predict wind speed. 

Table 2. External wind speed prediction from real-world flag observations on the dataset of Cardona et al. [9]. We regress the wind speed ($v_w \in \theta_c$) in the range 0 m/s to 15.5 m/s and report numbers on the evaluation set.

<table>
<thead>
<tr>
<th>Model</th>
<th>Input Modality</th>
<th>RMSE ↓</th>
<th>Acc@0.5 ↑</th>
</tr>
</thead>
<tbody>
<tr>
<td>Cardona et al. [9]</td>
<td>30 × 224 × 224</td>
<td>1.458</td>
<td>0.301</td>
</tr>
<tr>
<td>ResNet-18</td>
<td>1 × 224 × 224</td>
<td>1.390</td>
<td>0.274</td>
</tr>
<tr>
<td>ResNet-18</td>
<td>10 × 224 × 224</td>
<td>1.237</td>
<td>0.314</td>
</tr>
<tr>
<td>ResNet-18</td>
<td>20 × 224 × 224</td>
<td>1.347</td>
<td>0.296</td>
</tr>
<tr>
<td>SDN (ours)</td>
<td>30 × 224 × 224</td>
<td><strong>1.179</strong></td>
<td><strong>0.337</strong></td>
</tr>
</tbody>
</table>

a collection of 12 countries, we randomly sample a flag texture. Background images are selected from the SUN397 dataset [52]. Each simulation produces 60 cloth meshes at step size $\Delta T = 0.04$ s (i.e., 25 fps) which we render at 300 × 300 resolution. Following this procedure, we generate 1,000 mesh sequences and render a total of 14,000 training examples. We additionally generate validation and test sets of 150/3,800 and 85/3,500 mesh sequences/renders respectively. Some examples are visualized in Figure 6.

6. Results and Discussion

Real-world Extrinsic Wind Speed Measurement ($\theta_c$). We first assess the effectiveness of the proposed spectral decomposition network by measuring the wind speed on the recently proposed real-world flag dataset by Cardona et al. [9]. Their method, consisting of an ImageNet-pretrained ResNet-18 [16] with LSTM, will be the main comparison. We also train ResNet-18 with multiple input frames, followed by temporal average pooling of the final activations [21]. After training all methods, we report the root mean squared error (RMSE) and accuracy within 0.5 m/s ($\text{Acc} \geq 0.5$) in Table 2. While our method has significantly fewer parameters (2.6M versus 11.2M and 42.1M), the SDN outperforms the existing work on the task of real-world wind speed regression. This indicates the SDN’s effectiveness in modeling the spatial distribution of spectral power over the cloth's surface and its descriptiveness for the task at hand. The supplementary material contains the results on our FlagSim dataset.

SDN’s Physical Similarity Quality ($\theta_i, \theta_c$). We evaluate the physical similarity embeddings after training with contrastive loss. To quantify the ability to separate examples with similar and dissimilar physical parameters, we report the triplet accuracy [45]. We construct 3.5K FlagSim triplets from the test set as described in Section 3.1. We consider the SDN trained for video clips of a varying number of input frames and report its accuracies in Table 3. The results indicate the effectiveness of the learned distance metric to quantify the physical similarity between different observations. When considering flags, we conclude that 30 input frames are best with a triplet accuracy of 96.3% and therefore use 30 input frames in the remainder of this paper. In Figure 7 we visualize a subset of the embedding space and observe that the flag instances with low wind speeds are clustered in the top-right corner whereas strong wind speeds live in the bottom-left.

Real-world Intrinsic Cloth Parameter Recovery ($\theta_i$). In this experiment, we assess the effectiveness of our SDN for estimating intrinsic cloth material properties from a real-world video. We compare against Yang et al. [54] on the hanging cloth dataset of Bouman et al. [6] (Figure 2). Each of the 90 videos shows one of 30 cloth types hanging down while being excited by a fan at 3 wind speeds (W1-3). The goal is to infer the cloth’s stiffness and area weight. From our SDN trained on FlagSim with contrastive loss, we extract the embedding vectors for the 90 videos and project them into a 50-dimensional space using PCA. Then we train a linear regression model using leave-one-out following [6]. The results are displayed in Figure 9. While not outperforming the specialized method of [54], we find that our flag-based features generalize to intrinsic cloth material recovery. This is noteworthy, as our SDN was trained on flags of lightweight materials exhibiting predominantly horizontal motion.

Table 3. Evaluation of our physical similarity $s_\phi(x)$ for FlagSim test examples. We report average triplet accuracies [45].

<table>
<thead>
<tr>
<th>Input Frames</th>
<th>10</th>
<th>20</th>
<th>30</th>
<th>40</th>
<th>50</th>
</tr>
</thead>
<tbody>
<tr>
<td>FlagSim Accuracy</td>
<td>89.3</td>
<td>92.1</td>
<td><strong>96.3</strong></td>
<td>90.1</td>
<td>92.4</td>
</tr>
</tbody>
</table>
Putting everything together, our goal is measuring physics parameters based on real-world observations. We demonstrate the full measurement procedure (Figure 3) by optimizing over intrinsic and extrinsic model parameters \((\theta_i, \theta_e)\) from real-world flag videos (and present hanging cloth refinement results in the supplementary material). First, we randomly sample a real-world flag recording as subject of the measurement. The parameter range of the intrinsic \((16\times)\) and extrinsic \((1\times)\) is normalized to the domain \([-1, +1]\) and are all initialized to 0, i.e. their center values. We fix the render parameters \(\zeta\) manually as our focus is not on inferring those from real-video. However, these parameters are not carefully determined as the residual blocks in the embedding function can handle such variation (Figure 7). In each step, we simulate the cloth meshes with current parameters \(\theta_i, \theta_e\) and render its video clip with fixed render parameters \(\zeta\). Both the simulation and real-world video clips are then projected onto the embedding space using \(s_\phi(x)\), and we compute their pairwise distance (1). Finally, the Bayesian optimization’s acquisition function (Section 3.2) determines where to make the next evaluation \(\theta_i, \theta_e \in [-1, +1]\) to maximize the expected improvement, i.e. improving the measurement. The next iteration starts by denormalizing the parameters and running the simulation. We run the algorithm for 50 refinement steps. In Figure 8, we demonstrate our method’s measurements throughout optimization. Most importantly, we observe a gradual decrease in the pairwise distance between simulation and real-world example, indicating a successful measurement of the physical parameters. Importantly, we note that the wind speed converges towards the ground-truth wind speed within a few iterations, as indicated with a dashed line. More examples are given in the supplementary material.

7. Conclusion

We have presented a method for measuring intrinsic and extrinsic physical parameters for cloth in the wind without perceiving real cloth before. The iterative measurement gradually improves by assessing the similarity between the current cloth simulation and the real-world observation. By leveraging only simulations, we have proposed a method to train a physical similarity function. This enables measuring the physical correspondence between real and simulated data. To encode cloth dynamics, we have introduced a spectral decomposition layer that extracts the relevant features from the signal and generalizes from simulation to real observations. We compare the proposed method to prior work that considers flags in the wind and hanging cloth and obtain favorable results. For future work, given an appropriate physical embedding function, our method could be considered for other physical phenomena such as fire, smoke, fluid or mechanical problems.
References

[19] Roozbeh Mottaghi, Mohammad Rastegari, Abhinav Gupta, and Ali Farhadi. What happens if... learning to predict the effect of forces in images. In ECCV, 2016. 2


