Cloth in the Wind:

A Case Study of Physical Measurement through Simulation

Tom F. H. Runia Kirill Gavrilyuk Cees G. M. Snoek Arnold W. M. Smeulders
QUVA Deep Vision Lab, University of Amsterdam
{runia,kgavrilyuk,cgmsnoek,a.w.m.smeulders}@uva.nl

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.

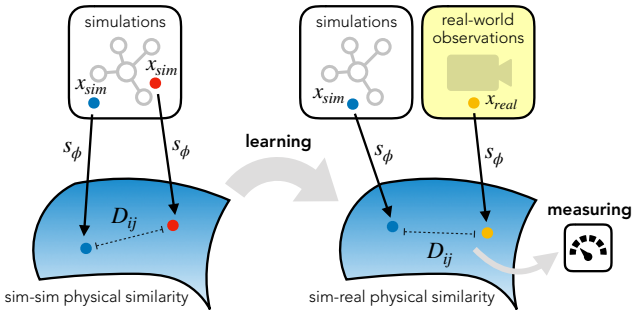

Figure 1. We propose to measure real-world physical cloth parameters without ever having seen the phenomena before. From cloth simulations only, we learn a distance metric that encodes both intrinsic and extrinsic physical properties. After learning, we use the embedding function to measure physical parameters from real-world video by comparison to its simulated counterpart.

In this paper, we consider flags and cloth in the wind as a case study. Measurements and visual models of flags and cloth are important for virtual clothing try-on [55], energy harvesting and biological systems [37, 19]. The cloth's intrinsic material properties, together with the external wind force, determine its dynamics. Untangling the dynamics of fabric is challenging due to the involved nature of the air-cloth interaction: a flag exerts inertial and elastic forces on the surrounding air, while the air acts on the fabric through pressure and viscosity [19]. As we seek to measure both the cloth's intrinsic material properties and the external wind force, our physical model couples a non-linear cloth model [46] with external wind force [48].

The task is challenging, as physical models of cloth tend to have high numbers of unknown parameters and bear intricate coupling of intrinsic and external forces. Our solution is to compare pairs of real and simulated observations and measure their physical similarity. As there is a fundamental caveat in the use of simulation and rendering for learning: "visually appealing" does not necessarily imply the result is realistic, the main question is how to assess the similarity of the causally underlying physical parameters rather than visual correspondence. It might be the case that the image looks real but never occurs in reality.

Figure 2. We consider two cases of cloth in the wind. Top row: random still images from our video recordings of real flags. Bottom row: examples from the Bouman *et al*. hanging cloth dataset [6].

At the core of our measurement is a cloth simulation engine with unknown parameters $\boldsymbol{\theta}$ to be determined. The outcome of a simulation (*e.g*. 3D meshes, points clouds, flow vectors) is converted to the image space using a render engine. We then compare the simulated visual data with a real-world observation of the particular phenomenon (Figure 1). Accordingly, we propose to learn a *physical similarity* metric from simulations only, without ever perceiving a real-world example. In the learned embedding space, observations with similar physical parameters will wind up close, while dissimilar example pairs will be further away. Guided by the physical similarity, the simulation's parameters are refined in each step. As a result, we obtain a complete computational solution for the refined measurements of physical parameters.

Our contributions are as follows: (1) We propose to train a perception-based physical cloth measurement device from simulations only, without ever observing a real-world manifestation of the phenomena. Our measurement device is formulated as a comparison between two visual observations implemented as a Siamese network that we train with contrastive loss. (2) In a case study of cloth, we propose a specific instantiation of the physical embedding function. At its core is a new spectral decomposition layer that measures the spectral power over the cloth's surface. Our solution compares favorably to existing work that recovers intrinsic and extrinsic physical properties from visual observations. (3) To evaluate our method, we record real-world video of flags with the ground-truth wind speed gauged using an anemometer. (4) Finally, we iteratively refine physics simulations from a single real-world observation towards maximizing the physical similarity between the real-world and its simulation.

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]. 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.

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 $\boldsymbol{\theta} \in \mathbb{R}^{D_p}$, consisting of intrinsic parameters $\boldsymbol{\theta}_i$ and extrinsic parameters $\boldsymbol{\theta}_e$ 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 $\boldsymbol{x}_{\text{target}} \in \mathbb{R}^{C \times N_t \times H \times W}$, with C denoting the number of image channels and N_t the number of $H \times W$ frames. In each

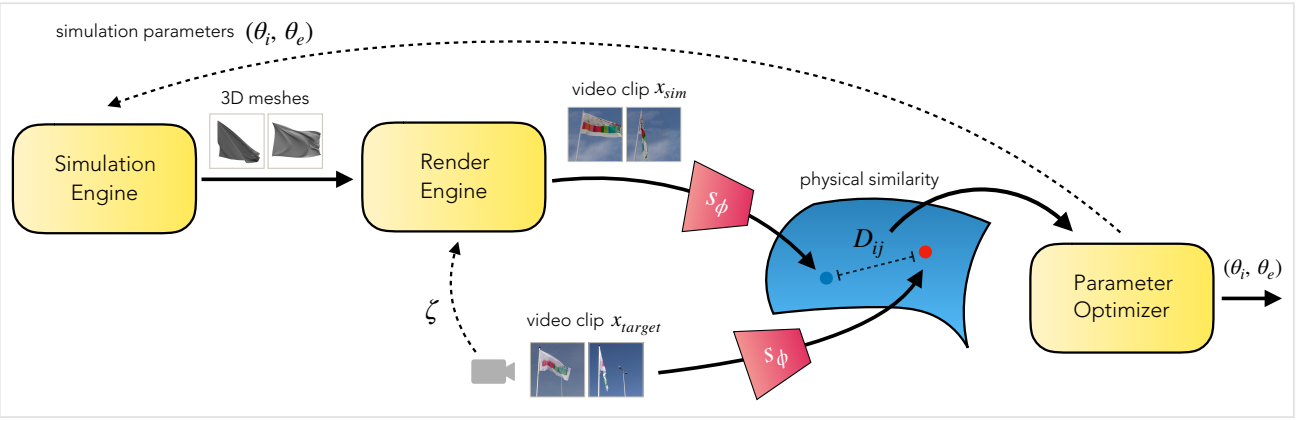

Figure 3. We propose the perception-based measurement of physical scene properties. Given an observation of a real-world physical phenomenon, here represented as video clip $\boldsymbol{x}_{\text{target}}$, our algorithm measures the underlying parameters of the physical scene. Central is a simulation engine implementing the physical model, parametrized by intrinsic material properties $\boldsymbol{\theta}_i$ and the characterization of external forces $\boldsymbol{\theta}_e$. A render engine, with render parameters $\boldsymbol{\zeta}$, maps the simulator's output to the image space producing video clip $\boldsymbol{x}_{\text{sim}}$. Using an embedding function $s_\phi(\boldsymbol{x})$, both real and simulated examples are mapped to a manifold on which physically similar examples are assigned to nearby points. To measure the similarity between both clips, we evaluate a distance metric $D_{ij}(\cdot,\cdot)$ in the embedding space. Its result serves as the objective for an optimization module that refines the physical parameters $\boldsymbol{\theta}$ towards the actual observation.

iteration, the simulator runs with current model parameters $\boldsymbol{\theta}$ to produce some intermediate representation (*e.g.* 3D meshes, point clouds or flow vectors), succeeded by a render engine with parameters $\boldsymbol{\zeta}$ that yields a simulated video clip $\boldsymbol{x}_{\text{sim}} \in \mathbb{R}^{C\times N_t\times H\times W}$. Our insight is that the physical similarity between real-world observation and simulation can be measured in some embedding space using pairwise distance:

$$D_{i,j} = D\left(s_\phi(\boldsymbol{x}_i), s_\phi(\boldsymbol{x}_j)\right) : \mathbb{R}^{D_e} \times \mathbb{R}^{D_e} \rightarrow \mathbb{R} \qquad (1)$$

where $s_\phi(\boldsymbol{x}) : \mathbb{R}^{C\times N_t\times H\times W} \rightarrow \mathbb{R}^{D_e}$ an embedding function parametrized by ϕ that maps the data manifold $\mathbb{R}^{C\times N_t\times H\times W}$ to some embedding manifold $\mathbb{R}^{D_e}$ 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 $\boldsymbol{\theta}^*$ and corresponding simulation $\boldsymbol{x}^*_{\text{sim}}$ 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 $\boldsymbol{x}_{\text{sim}}$ and real-world observation $\boldsymbol{x}_{\text{target}}$. 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 $\boldsymbol{\theta}$ 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 $\boldsymbol{x}^i_{\text{sim},}$ with different render parameters $\boldsymbol{\zeta}^i$. 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 $s_\phi(\boldsymbol{x})$ in Siamese fashion [8]. In the embedding space, the physical similarity will be evaluated using the squared Euclidean distance: $D_{i,j} = D\left(s_\phi(\boldsymbol{x}_i), s_\phi(\boldsymbol{x}_j)\right) = \|s_\phi(\boldsymbol{x}_i) - s_\phi(\boldsymbol{x}_j)\|_2^2$. 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 hyper-parameters involved by training deep neural networks (*e.g.* learning rate, weight decay, dropout). Formally, we seek to find the global optimum of physical parameters:

$$\boldsymbol{\theta}^* = \arg\min_{\boldsymbol{\theta}} D\left(s_\phi(\boldsymbol{x}_{\text{target}}), s_\phi(\boldsymbol{x}_{\text{sim}}(\boldsymbol{\theta}))\right), \qquad (2)$$

where the target example is fixed and the simulated example depends on the current set of physical parameters $\boldsymbol{\theta}$. Adjusting the parameters $\boldsymbol{\theta}$ at each iteration is challenging as it is hard to make parametric assumptions on (2) as function of $\boldsymbol{\theta}$ 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 $\boldsymbol{\theta}$. 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 $s_\phi(\boldsymbol{x})$ 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 $\boldsymbol{\theta}$ and design an appropriate embedding function $s_\phi(\boldsymbol{x})$.

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 ($\boldsymbol{\theta}_i$). The bending model is based on the linear bending force equation first proposed in [7]. The model

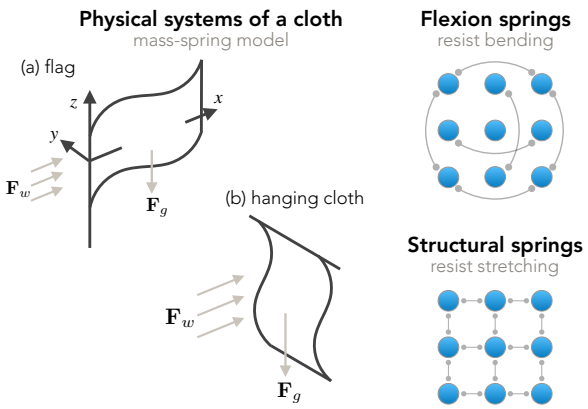

Figure 4. *Left:* we consider two cases of cloth exposed in the wind: (a) a flag curling in the wind; and (b) cloth fabric hanging from a rod. In both cases, the fabric fabric is treated as a mass-spring model in which a dense grid of point masses is inter-connected with multiple springs. *Right:* the bending and stretching springs determine the materials behavior. Flexion springs act over shared edges whereas structural springs connect to direct neighbors.

formulates the elastic bending force $\boldsymbol{F}_e$ over triangular meshes sharing an edge (Figure 4). For two triangles separated by the dihedral angle φ, the bending force reads:

$$\boldsymbol{F}_e = k_e \sin(\varphi/2)(N_1 + N_2)^{-1}|\boldsymbol{E}|\boldsymbol{u}, \qquad (3)$$

where k_e is the material dependent bending stiffness, N_1, N_2 are the weighted surface normals of the two triangles, $\boldsymbol{E}$ represents the edge vector and $\boldsymbol{u}$ is the bending mode (see Figure 1 in [7]). The bending stiffness k_e is non-linearly related to the dihedral angle φ. 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_b angles yielding a total of $3N_b$ 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 ($\boldsymbol{\theta}_e$). For the dynamics of cloth, we consider two external forces acting upon its planar surface. First, the Earth's gravitational acceleration ($\boldsymbol{F}_g = m\boldsymbol{a}_g$) naturally pushes down the fabric. The total mass is defined by the cloth's area weight ρ_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 $\boldsymbol{F}_d = 6\pi R\eta\boldsymbol{v}_w$ in terms of the surface area, the air's dynamic viscosity and the wind velocity $\boldsymbol{v}_w$ [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.

Table 1. The predefined parameter range for optimization of $\boldsymbol{\theta} = (\boldsymbol{\theta}_i, \boldsymbol{\theta}_e)$ given the physical model of a flag curling in the wind. The bending parameters $\bar{k}_e$ correspond to the "Camel Ponte Roma" base material from [46].

	Parameter	Params	Search space
θ_i	Bending stiffness	15	$k_e \in [10^{-1}\bar{k}_e, 10\bar{k}_e]$
θ_i	Fabric area weight	1	$\rho_A \in [0.10, 0.17]\,\mathrm{kg\,m^{-2}}$
θ_e	Wind velocity	1	$v_w \in [0, 10]\,\mathrm{m\,s^{-1}}$

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 a real-world observation, we pass the sequence of meshes through a 3D render engine [5]. Given render parameters $\boldsymbol{\zeta}$ comprising of camera position, scene geometry, lighting conditions and the cloth's visual texture, the renderer produces a simulated video clip ($\boldsymbol{x}_{\text{sim}}$) which we can compare directly to the real-world observation ($\boldsymbol{x}_{\text{target}}$). We emphasize that our focus is neither on inferring render parameters $\boldsymbol{\zeta}$ from observations nor on attaining visual realism for our renders.

Parameter Search Space ($\boldsymbol{\theta}_i, \boldsymbol{\theta}_e$). 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 $\rho_A \sim \text{Uniform}(0.10, 0.17)\,\mathrm{kg\,m^{-2}}$ after consulting various flag materials at online retailers. And, we restrict the range of the bending stiffness coefficients by multiplying the base material's $\bar{k}_e$ 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-frequent 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:

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

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 $I(\omega_{\max_i})$ and $\omega_{\max_i}$ 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.

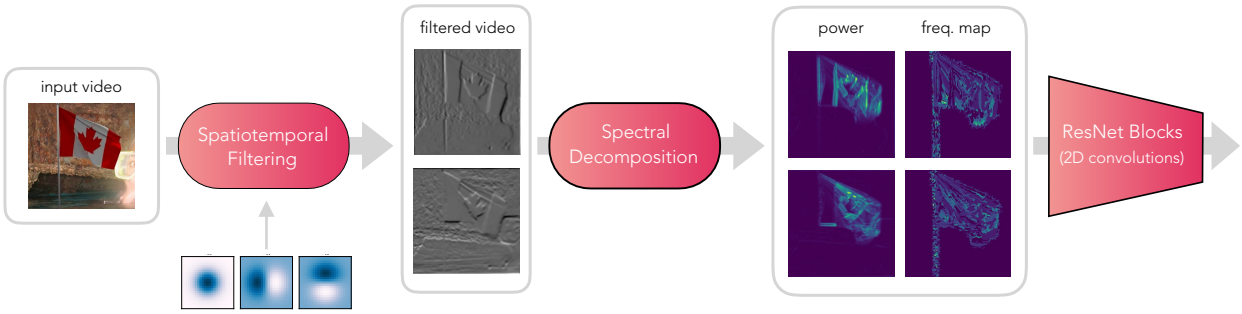

Figure 5. Overview of our SDN architecture $s_\phi(\boldsymbol{x})$ for learning the physical correspondence between the simulation and real-world observation of dynamic flags. Given a 3D video volume as input, we first apply a 0^{th}-order temporal Gaussian filter followed by two directional 1^{st}-order 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 $s_\phi(\boldsymbol{x})$, with the spectral decomposition layer at its core, is illustrated in Figure 5. First, our model convolves the input video $\boldsymbol{x}$ 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 ϕ. We use 3 standard ResNet blocks [16] and a final linear layer that maps to the $\mathbb{R}^{D_e}$ embedding space. We refer to our network as *Spectral Decomposition Network (SDN)*.

Network Details. Our network is implemented in PyTorch [31] and is publicly available[1]. 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 ($D_e = 512$) for the paper. Input video clips of size 224×224 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 $s_\phi(\boldsymbol{x})$ 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 $\boldsymbol{\theta}$ 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 $\boldsymbol{x}^i_{\text{sim}}$ at different render settings $\boldsymbol{\zeta}^i$. 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° 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.

[1] https://tomrunia.github.io/projects/cloth/

Table 2. External wind speed prediction from real-world flag observations on the dataset of Cardona *et al*. [9]. We regress the wind speed ($v_w \in \boldsymbol{\theta}_e$) in the range 0 m s^{-1} to 15.5 m s^{-1} and report numbers on the evaluation set.

Model	Input Modality	RMSE ↓	Acc@0.5 ↑
Cardona *et al*. [9]	$30 \times 224 \times 224$	1.458	0.301
ResNet-18	$1 \times 224 \times 224$	1.390	0.274
ResNet-18	$10 \times 224 \times 224$	1.237	0.314
ResNet-18	$20 \times 224 \times 224$	1.347	0.296
SDN (ours)	$30 \times 224 \times 224$	**1.179**	**0.337**

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 ($\boldsymbol{\theta}_e$). 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^{-1} (Acc@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 ($\boldsymbol{\theta}_i, \boldsymbol{\theta}_e$). 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.

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

Input Frames	10	20	30	40	50
FlagSim Accuracy	89.3	92.1	**96.3**	90.1	92.4

Figure 7. Barnes-Hut t-SNE [44] visualization of the learned flag embedding space. For visualization purpose we only display examples with wind from the left. Top-right examples exhibit flags at low wind speeds while bottom-left corresponds to strong winds.

Real-world Intrinsic Cloth Parameter Recovery ($\boldsymbol{\theta}_i$). 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.

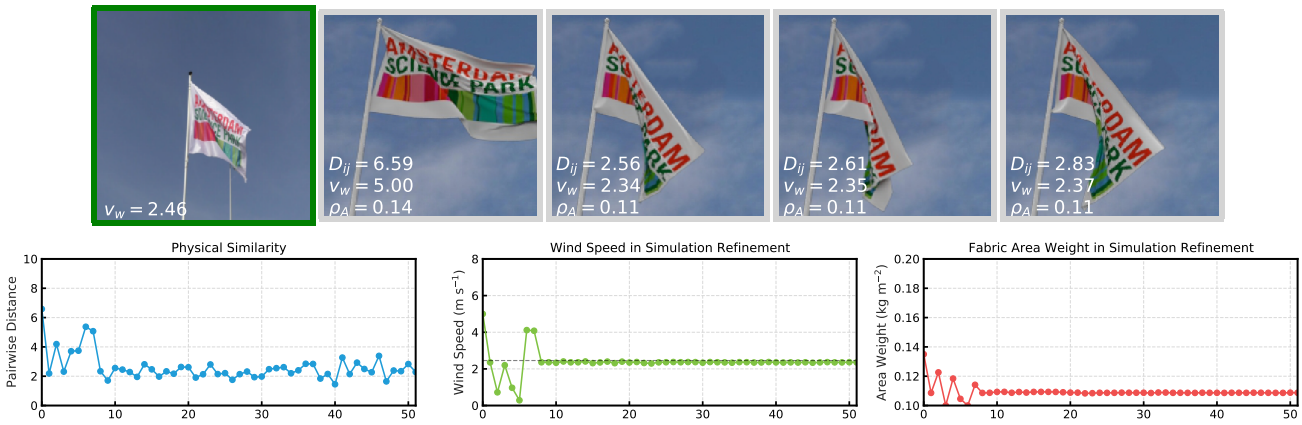

Figure 8. Result of our iterative measurement for a target video capturing a flag in the wind. *Top left:* frame from the real-world target video clip with the ground-truth wind speed measured using an anemometer. *Top remaining:* simulated examples throughout the refinement process with corresponding simulation parameters. *Bottom:* development throughout the refinement process for 50 iteration steps. We plot the distance between simulation and target instance in the embedding space and the estimated wind speed (m s^{-1}). We annotate the ground-truth wind speed with a dashed line. As the plot indicates, the refinement process converges towards the real wind speed.

Real-world Combined Parameter Refinement (θ_i, θ_e). 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 $(\boldsymbol{\theta}_i, \boldsymbol{\theta}_e)$ 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×) and extrinsic (1×) is normalized to the domain $[-1, +1]$ and are all initialized to 0, *i.e.* their center values. We fix the render parameters $\boldsymbol{\zeta}$ 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 $\boldsymbol{\theta}_i, \boldsymbol{\theta}_e$ and render its video clip with fixed render parameters $\boldsymbol{\zeta}$. Both the simulation and real-world video clips are then projected onto the embedding space using $s_\phi(\boldsymbol{x})$, and we compute their pairwise distance (1). Finally, the Bayesian optimization's acquisition function (Section 3.2) determines where to make the next evaluation $\boldsymbol{\theta}_i, \boldsymbol{\theta}_e \in [-1, +1]$ 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.

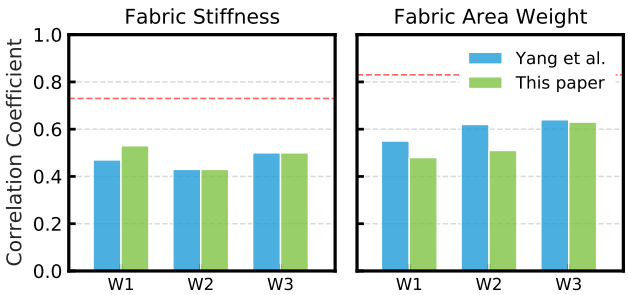

Figure 9. Intrinsic cloth material measurements from real videos. We report the Pearson correlation coefficients (higher is better) between predicted material type and both ground-truth stiffness/-density on the Bouman *et al.* [6] hanging cloth dataset. The dashed red line indicates human performance as determined by [6].

References

[1] David Baraff and Andrew Witkin. Large steps in cloth simulation. In *SIGGRAPH*, 1998. 4

[2] David Baraff, Andrew Witkin, and Michael Kass. Untangling cloth. In *SIGGRAPH*, 2003. 4

[3] James Bergstra and Yoshua Bengio. Random search for hyper-parameter optimization. *JMLR*, 13(Feb):281–305, 2012. 4

[4] Kiran S Bhat, Christopher D Twigg, Jessica K Hodgins, Pradeep K Khosla, Zoran Popović, and Steven M Seitz. Estimating cloth simulation parameters from video. In *SIGGRAPH*, 2003. 2

[5] Blender Online Community. *Blender - a 3D modelling and rendering package*. Blender Foundation, 2018. 5, 6

[6] Katherine L Bouman, Bei Xiao, Peter Battaglia, and William T Freeman. Estimating the material properties of fabric from video. In *ICCV*, 2013. 2, 7, 8

[7] Robert Bridson, Sebastian Marino, and Ronald Fedkiw. Simulation of clothing with folds and wrinkles. In *SIGGRAPH*, 2005. 1, 4

[8] Jane Bromley, Isabelle Guyon, Yann LeCun, Eduard Säckinger, and Roopak Shah. Signature verification using a" siamese" time delay neural network. In *NIPS*, 1994. 3

[9] Jennifer L Cardona, Michael F Howland, and John O Dabiri. Seeing the wind: Visual wind speed prediction with a coupled convolutional and recurrent neural network. In *NeurIPS*, 2019. 2, 7

[10] Kenneth James Williams Craik. *The Nature of Explanation*, volume 445. CUP Archive, 1967. 1

[11] Abe Davis, Katherine L Bouman, Justin G Chen, Michael Rubinstein, Fredo Durand, and William T Freeman. Visual vibrometry: Estimating material properties from small motion in video. In *CVPR*, 2015. 2

[12] Shengyong Ding, Liang Lin, Guangrun Wang, and Hongyang Chao. Deep feature learning with relative distance comparison for person re-identification. *Pattern Recognition*, 48(10):2993–3003, 2015. 6

[13] Christophe Eloy, Romain Lagrange, Claire Souilliez, and Lionel Schouveiler. Aeroelastic instability of cantilevered flexible plates in uniform flow. *Journal of Fluid Mechanics*, 611:97–106, 2008. 5

[14] John Haddon and David Forsyth. Shading primitives: Finding folds and shallow grooves. In *ICCV*, 1998. 2

[15] Raia Hadsell, Sumit Chopra, and Yann LeCun. Dimensionality reduction by learning an invariant mapping. In *CVPR*, 2006. 3

[16] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 6, 7

[17] Mary Hegarty. Mechanical reasoning by mental simulation. *Trends in cognitive sciences*, 8(6):280–285, 2004. 1

[18] Alexander Hermans, Lucas Beyer, and Bastian Leibe. In defense of the triplet loss for person re-identification. *arXiv preprint arXiv:1703.07737*, 2017. 6

[19] Wei-Xi Huang and Hyung Jin Sung. 3d simulation of a flapping flag in a uniform flow. *Journal of Fluid Mechanics*, 653:301–336, 2010. 1

[20] Stephen James, Paul Wohlhart, Mrinal Kalakrishnan, Dmitry Kalashnikov, Alex Irpan, Julian Ibarz, Sergey Levine, Raia Hadsell, and Konstantinos Bousmalis. Sim-to-real via sim-to-sim: Data-efficient robotic grasping via randomized-to-canonical adaptation networks. In *CVPR*, 2019. 5

[21] Andrej Karpathy, George Toderici, Sanketh Shetty, Thomas Leung, Rahul Sukthankar, and Li Fei-Fei. Large-scale video classification with convolutional neural networks. In *CVPR*, 2014. 7

[22] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. 6

[23] Wenbin Li, Seyedmajid Azimi, Aleš Leonardis, and Mario Fritz. To fall or not to fall: A visual approach to physical stability prediction. *arXiv preprint arXiv:1604.00066*, 2016. 2

[24] Junbang Liang, Ming Lin, and Vladlen Koltun. Differentiable cloth simulation for inverse problems. In *NeurIPS*, 2019. 2

[25] Abhimitra Meka, Maxim Maximov, Michael Zollhoefer, Avishek Chatterjee, Hans-Peter Seidel, Christian Richardt, and Christian Theobalt. Lime: Live intrinsic material estimation. In *CVPR*, 2018. 2

[26] Roozbeh Mottaghi, Hessam Bagherinezhad, Mohammad Rastegari, and Ali Farhadi. Newtonian scene understanding: Unfolding the dynamics of objects in static images. In *CVPR*, 2016. 2

[27] 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

[28] Rahul Narain, Armin Samii, and James F O'Brien. Adaptive anisotropic remeshing for cloth simulation. In *SIGGRAPH*, 2012. 1, 4, 5

[29] Changyong Oh, Efstratios Gavves, and Max Welling. Bock: Bayesian optimization with cylindrical kernels. In *ICML*, 2018. 4

[30] Alan V Oppenheim. *Discrete-time signal processing*. Pearson Education India, 1999. 5

[31] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in PyTorch. In *NIPS Autodiff Workshop*, 2017. 6

[32] Xingchao Peng, Ben Usman, Neela Kaushik, Dequan Wang, Judy Hoffman, and Kate Saenko. Visda: A synthetic-to-real benchmark for visual domain adaptation. In *CVPR-W*, 2018. 5

[33] Xavier Provot. Deformation constraints in a mass-spring model to describe rigid cloth behaviour. In *Graphics interface*. Canadian Information Processing Society, 1995. 4

[34] Tom FH Runia, Cees GM Snoek, and Arnold WM Smeulders. Repetition estimation. *IJCV*, 127(9):1361–1383, 2019. 5

[35] Hidetomo Sakaino. Fluid motion estimation method based on physical properties of waves. In *CVPR*, 2008. 2

[36] Camille Schreck, Christian Hafner, and Chris Wojtan. Fundamental solutions for water wave animation. *SIGGRAPH*, 2019. 1

[37] Michael J Shelley and Jun Zhang. Flapping and bending bodies interacting with fluid flows. *Annual Review of Fluid Mechanics*, 43:449–465, 2011. 1, 2, 5

[38] William S Slaughter. *The linearized theory of elasticity*. Springer Science & Business Media, 2012. 4

[39] Jasper Snoek, Hugo Larochelle, and Ryan P Adams. Practical bayesian optimization of machine learning algorithms. In *NIPS*, 2012. 4

[40] Lisa Spencer and Mubarak Shah. Water video analysis. In *ICIP*, 2004. 2

[41] Meng Sun, Allan D Jepson, and Eugene Fiume. Video input driven animation (vida). In *ICCV*, 2003. 2

[42] Sadatoshi Taneda. Waving motions of flags. *Journal of the Physical Society of Japan*, 24(2):392–401, 1968. 2, 5

[43] Fang-Bao Tian. Role of mass on the stability of flag/flags in uniform flow. *Applied Physics Letters*, 103(3):034101, 2013. 2

[44] Laurens Van Der Maaten. Accelerating t-sne using tree-based algorithms. *JMLR*, 15(1):3221–3245, 2014. 7

[45] Andreas Veit, Serge Belongie, and Theofanis Karaletsos. Conditional similarity networks. In *CVPR*, 2017. 7

[46] Huamin Wang, James F O'Brien, and Ravi Ramamoorthi. Data-driven elastic models for cloth: modeling and measurement. In *SIGGRAPH*, 2011. 1, 4, 5

[47] Tuanfeng Y Wang, Duygu Ceylan, Jovan Popović, and Niloy J Mitra. Learning a shared shape space for multimodal garment design. In *SIGGRAPH Asia*, 2019. 2

[48] Jakub Wejchert and David Haumann. Animation aerodynamics. In *SIGGRAPH*, 1991. 1, 4

[49] Ryan White, Keenan Crane, and David A Forsyth. Capturing and animating occluded cloth. In *SIGGRAPH*, 2007. 2

[50] Jiajun Wu, Joseph J Lim, Hongyi Zhang, Joshua B Tenenbaum, and William T Freeman. Physics 101: Learning physical object properties from unlabeled videos. In *BMVC*, 2016. 2

[51] Jiajun Wu, Ilker Yildirim, Joseph J Lim, Bill Freeman, and Josh Tenenbaum. Galileo: Perceiving physical object properties by integrating a physics engine with deep learning. In *NIPS*, 2015. 2

[52] Jianxiong Xiao, James Hays, Krista A Ehinger, Aude Oliva, and Antonio Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *CVPR*, 2010. 7

[53] Tianfan Xue, Jiajun Wu, Zhoutong Zhang, Chengkai Zhang, Joshua B Tenenbaum, and William T Freeman. Seeing tree structure from vibration. In *ECCV*, 2018. 2

[54] Shan Yang, Junbang Liang, and Ming C Lin. Learning-based cloth material recovery from video. In *ICCV*, 2017. 2, 7

[55] Shan Yang, Zherong Pan, Tanya Amert, Ke Wang, Licheng Yu, Tamara Berg, and Ming C Lin. Physics-inspired garment recovery from a single-view image. *ACM Transactions on Graphics (TOG)*, 37(5):170, 2018. 1, 2

Supplementary Material

Cloth in the Wind: A Case Study of Physical Measurement through Simulation

1. Table of Contents

The supplementary material has the following content:

- **Section 2** and **Algorithm 1**: Formalization of the batched version of the spectral decomposition layer.
- **Section 3** and **Figure 1**: Details on the collection of real wind speed measurements and video recordings.
- **Section 4**: Details on the video data augmentation and optimization for the training of all networks.
- **Section 5** and **Figure 4**: Additional experiments on refined measurements for the hanging cloth dataset and details on our ClothSim dataset.
- **Table 1**: Supplement to Table 2 in the main paper with wind speed regression results on our FlagSim dataset.
- **Figure 3**: Supplement to Figure 8 in the main paper to include more refined measurement examples.
- **Listing 1**, **Listing 2** and **Listing 3**: Specification of ArcSim scene and material configuration files for flag and cloth simulations.
- **Table 2**: Exhaustive list of Blender's rendering parameters for generating the FlagSim and ClothSim datasets.

2. Spectral Decomposition Layer

To support Section 4.3 in the main paper, we formalize the batched version of the spectral decomposition layer in Algorithm 1. Given a batch of video clips as input, our spectral layer applies the discrete Fourier transform along the temporal dimension to compute the temporal frequency spectrum. From the periodogram, we can select the top-k strongest frequency responses and their corresponding spectral power. The resulting frequency maps and power maps all have the same dimensions and can, therefore, be stacked as a multi-channel image. These tensors can be further processed by standard 2D convolution layers to learn frequency-based feature representations. The proposed layer is efficiently implemented in PyTorch [6] to run on the GPU using the `torch.irfft` operation. The source code is available through the project website.

Algorithm 1 Spectral Decomposition Layer

Input. Video tensor $\boldsymbol{x}$ of shape $[N_b, C, N_t, H, W]$
Input. Number of frequency peaks to select, k
Output. Decomposition of shape $[N_b, 2kC, H, W]$

procedure SpectralDecompositionLayer($\boldsymbol{x}$)
    Reshape $\boldsymbol{x}$ to $[N_b CHW, N_t]$ to obtain batch of signals
    Apply a Hanning window to signals
    Compute the DFT of signals using Eq. 4 (main paper)
    Compute periodogram of signals $I(\omega)$
    Select top-k peaks of $I(\omega)$ and corresponding ω's
    $P \leftarrow$ top-k peaks of $I(\omega)$ reshaped to $[N_b, kC, H, W]$
    $\Omega \leftarrow$ corresponding ω's reshaped to $[N_b, kC, H, W]$
    return P, Ω
end procedure

3. Real-World Flag Dataset Acquisition

We here describe our data acquisition to obtain real-world wind speed measurements serving as ground-truth for our final experiment. To accurately gauge the wind speed next to the flag, we have obtained two anemometers:

- SkyWatch BL-400: windmill-type anemometer
- Testo 410i: vane-type anemometer

The measurement accuracy of both anemometers is 0.2 m s^{-1}. To verify the correctness of both anemometers, we have checked that both wind meters report the same wind speeds before usage. After that, we use the SkyWatch BL-400 anemometer for our measurements as it measures omnidirectional which is more convenient. We hoisted the anemometer in a flag pole such that the wind speeds are measured at the same height as the flag. Wind speed measurements are recorded at 1 second intervals and interfaced to the computer. In Figure 1 we display an example measurement and report the dataset's wind speed distribution. For the experiments (Section 6, main paper, last section), we randomly sample video clips of 30 consecutive frames from our video recordings and consider the ground-truth wind speed to be the average over the last minute. This procedure ensures that small wind speed deviations and measurement errors are averaged out over time.

To capture the videos, we use a Panasonic HC-V770 video camera. The camera records at 1920 × 1080 at 60 frames

per second. We perform post-processing of the videos in the following ways. Firstly, we temporally subsample the video frames at 25 fps such that the clips are in accordance with the frame step size in the physics simulator. Moreover, we assert that the video recordings are temporally aligned with the wind speed measurements using their timestamps. Secondly, we manually crop the videos such that the curling flag appears in the approximate center of the frame. After this, the frames are spatially subsampled to 300×300, again in agreement with animations obtained from the render engine.

4. Training Details

Data Augmentation. The examples in the FlagSim dataset are stored as a sequence of 60 JPEG frames of size 300×300. During training, when using less than 60 input frames (30 is used in all experiments), we randomly sample N_t successive frames from each video clip. This is achieved by uniform sampling of a temporal offset within the video. After this, for the sampled sequence of frames, we convert images to grayscale, perform multi-scale random cropping and apply random horizontal flipping [8] to obtain a $N_t \times 1 \times 224 \times 224$ input clip. Finally, we subtract the mean and divide by the standard deviation for each video clip.

Optimization Details. We train all networks using stochastic gradient descent with Adam [4]. We initialize training with a learning rate of 10^{-2} and decay the learning rate with a factor 10 after 20 epochs. To prevent overfitting, we utilize weight decay of $2 \cdot 10^{-3}$ for all networks. Training continues until validation loss plateaus – typically around 40 epochs. Total training time for our spectral decomposition network is about 4 hours on a single Nvidia GeForce GTX Titan X. When training the recurrent models [3, 10] we also perform gradient clipping (max norm of 10) to improve training stability.

5. Experiments on Hanging Cloth Video

Our real-world flag dataset enables us to evaluate our method's measurement performance of external parameters ($v_w \in \boldsymbol{\theta}_e$). However, the cloth's internal parameters are unknown and cannot be evaluated beyond visual inspection. Therefore, we also perform experiments on the hanging cloth dataset of Bouman *et al*. [2]. The authors have carefully determined the internal cloth material properties, which we can leverage for quantitative evaluation of our simulated-refined measurements. Specifically, we assess our method's ability to measure the cloth's *area weight* ($\mathrm{kg\,m^{-2}}$). The method is identical to that explained in the main paper with its results presented in the final experiment of Section 6. However, we retrain the embedding function $s_\phi(\boldsymbol{x})$ on a dataset of hanging cloth simulations, which we refer to as ClothSim. In this section, we will briefly discuss the characteristics of this dataset and report experimental results.

Table 1. External wind speed prediction from simulation. We regress the wind speed ($v_w \in \boldsymbol{\theta}_e$) on our **FlagSim dataset**. The metrics are computed over the 3.5K test examples. Target velocities range from 0 $\mathrm{m\,s^{-1}}$ (no wind) to 10 $\mathrm{m\,s^{-1}}$ (strong wind). Experimental setup is identical to Table 2 in the main paper.

Model	Input Modality	RMSE ↓	Acc@0.5 ↑
Yang *et al*. [10]	$10 \times 227 \times 227$	0.380	0.620
Cardona *et al*. [3]	$30 \times 227 \times 227$	0.271	0.580
ResNet-18	$1 \times 224 \times 224$	0.381	0.615
ResNet-18	$10 \times 224 \times 224$	0.264	0.734
ResNet-18	$20 \times 224 \times 224$	0.207	0.775
SDN (ours)	$20 \times 224 \times 224$	0.183	0.813
SDN (ours)	$30 \times 224 \times 224$	**0.180**	**0.838**

ClothSim Dataset. Following the same procedure as for the FlagSim dataset, we additionally generate a dataset of simulated hanging cloth excited by a constant wind force. The main difference between the FlagSim dataset is the wider variety of cloth material. Specifically, we use all the materials presented in [7] available in ArcSim. The increased diversity allows us to model the dynamics in real-world hanging cloth recording [2]. Our dataset shares similarity with the simulated hanging cloth dataset of [10]. However, in their work, the dataset is employed to train a classifier for predicting the material class. In Listing 2 and Table 2 we present an exhaustive overview of the simulation and render parameters that were used for generating the dataset.

Real-world Parameter Refinement ($\boldsymbol{\theta}_i, \boldsymbol{\theta}_e$). Given the embedding function $s_\phi(\boldsymbol{x})$ trained on ClothSim using contrastive loss, we run our refined measurement experiment on the hanging cloth dataset of Bouman *et al*. [2]. Our goal is to measure the cloth's area weight as we have access to its ground-truth measurement. Unlike for our real-world flag dataset, we do not know the true wind speed beyond the setting of the industrial fan that was used for exciting the fabric artificially. In Figure 4 we report the results for 3 randomly sampled real-world videos.

References

[1] Blender Online Community. *Blender - a 3D modelling and rendering package*. Blender Foundation, 2018. 8

[2] Katherine L Bouman, Bei Xiao, Peter Battaglia, and William T Freeman. Estimating the material properties of fabric from video. In *ICCV*, 2013. 2, 4, 7

[3] Jennifer L Cardona, Michael F Howland, and John O Dabiri. Seeing the wind: Visual wind speed prediction with a coupled convolutional and recurrent neural network. In *NeurIPS*, 2019. 2

[4] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015. 2

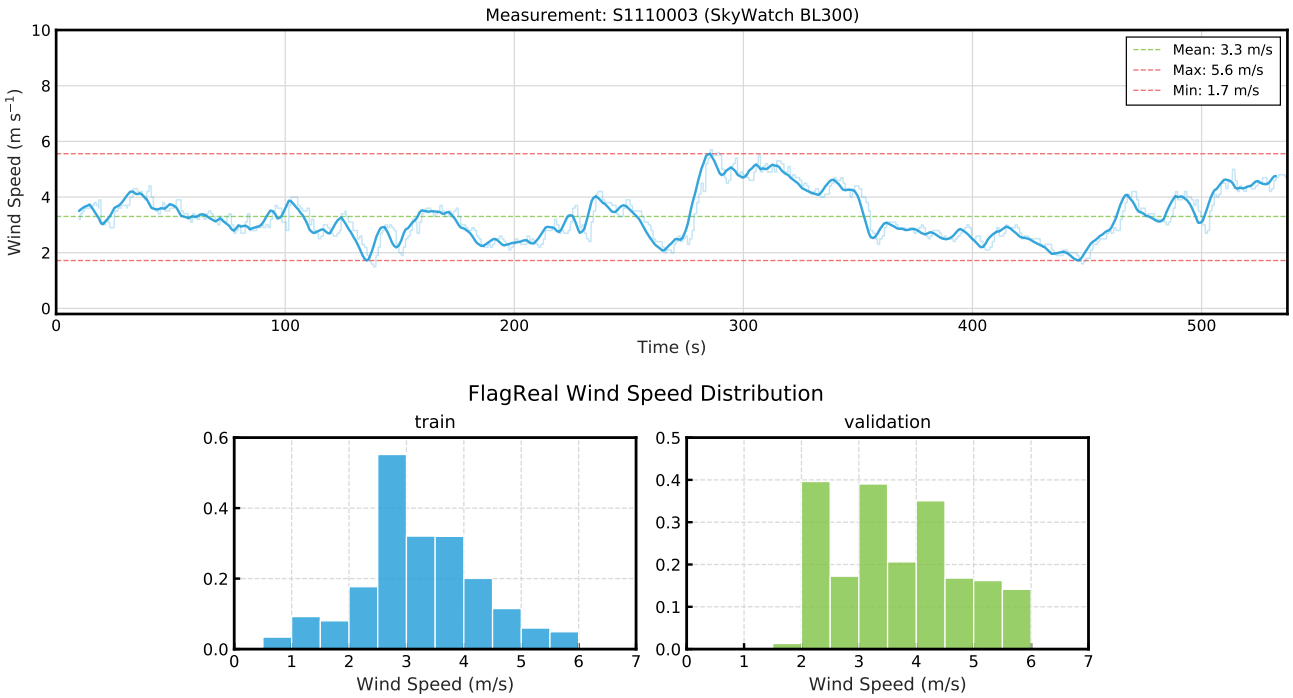

Figure 1. *Top:* Example of the time-varying wind speed as obtained by the SkyWatch BL-400 anemometer positioned directly next to the video-recorded flag. The wind speed is sampled at 1 Hz and interfaced to a computer using bluetooth. For our final experiment, we sample video clips of 30 frames and consider the ground-truth wind speed to be the average wind speed over the last minute. *Bottom:* Distribution statistics of the dataset we collected. Over all 4K non-overlapping videos the average wind speed is 3.2m s^{-1} while the minimum and maximum wind speeds are 0.5m s^{-1} and 6.0m s^{-1} respectively.

[5] Rahul Narain, Armin Samii, and James F O'Brien. Adaptive anisotropic remeshing for cloth simulation. In *SIGGRAPH*, 2012. 6, 8

[6] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in PyTorch. In *NIPS Autodiff Workshop*, 2017. 1

[7] Huamin Wang, James F O'Brien, and Ravi Ramamoorthi. Data-driven elastic models for cloth: modeling and measurement. In *SIGGRAPH*, 2011. 2, 8

[8] Limin Wang, Yuanjun Xiong, Zhe Wang, and Yu Qiao. Towards good practices for very deep two-stream convnets. *arXiv preprint arXiv:1507.02159*, 2015. 2

[9] Jianxiong Xiao, James Hays, Krista A Ehinger, Aude Oliva, and Antonio Torralba. Sun database: Large-scale scene recognition from abbey to zoo. In *CVPR*, 2010. 8

[10] Shan Yang, Junbang Liang, and Ming C Lin. Learning-based cloth material recovery from video. In *ICCV*, 2017. 2, 8

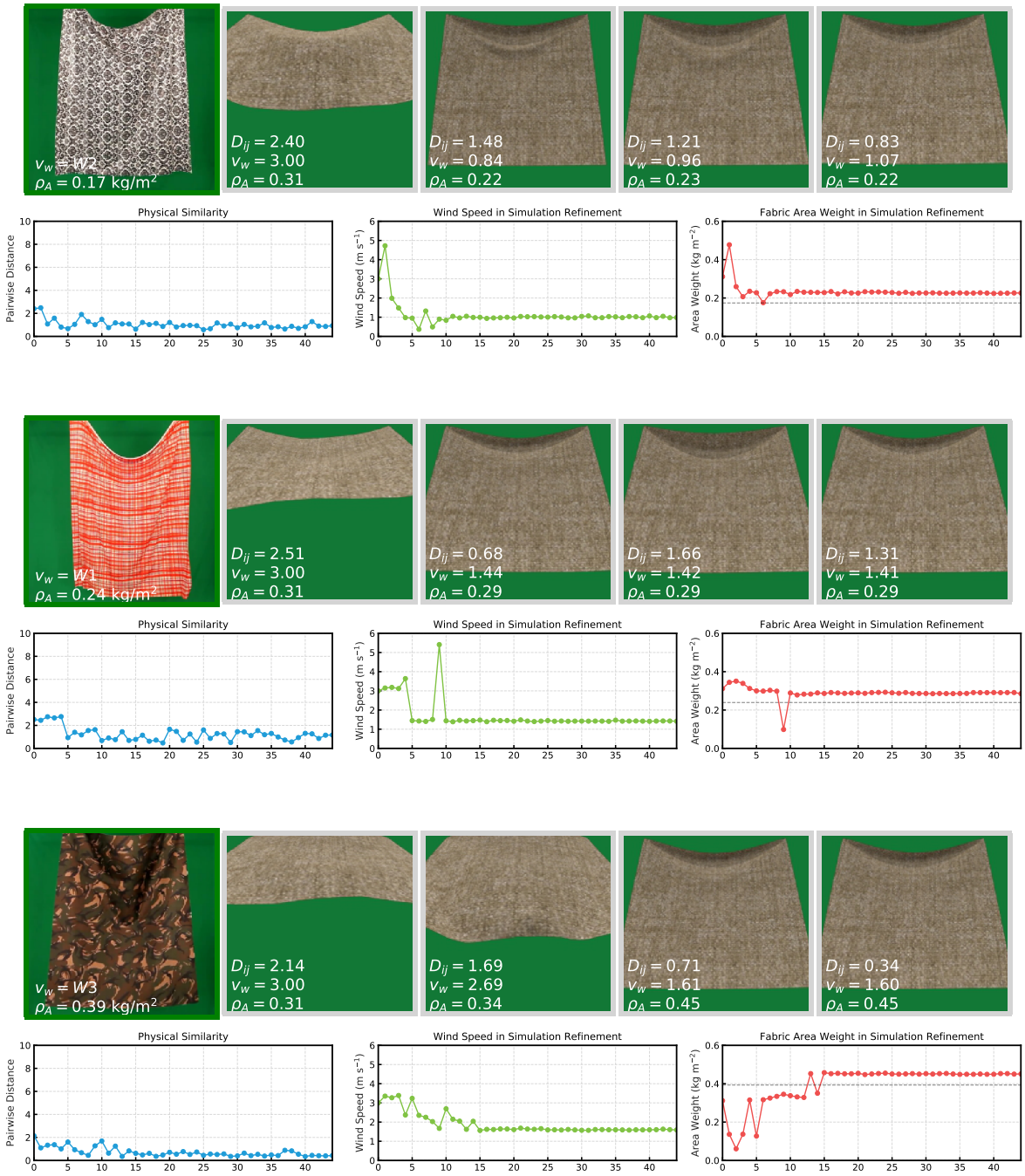

Figure 4. Results for our **hanging cloth refined-measurements** for a random target video capturing the hanging cloth as recorded by Bouman *et al*. [2]. The five images are the center frames of the real-world target video (left) and simulations throughout the refinement process after $t = 0, 10, 20, 40$ optimization steps. Optimization is performed over all 16 intrinsic cloth parameters $\boldsymbol{\theta}_i$ and 1 external wind speed $\boldsymbol{\theta}_e$. We plot the estimated cloth material area weight (kg m^{-2}) and wind speed velocity (m s^{-1}), although we only have access to the true material area weight (dashed horizontal line). For this dataset, the wind speed has three settings of increasing wind speed: W1, W2 and W3. *Top row:* The cloth's true area weight is 0.17 kg m^{-2} while the final measurement attains 0.22 kg m^{-2} after only 10 iterations. *Center row:* The cloth's true area weight is 0.24 kg m^{-2} while the prediction is 0.29 kg m^{-2}. While the ground-truth wind speed is not known, the wind speed in the simulations seems like an underestimate. *Top:* A heavier cloth at 0.39 kg m^{-2} while the simulation measures 0.45 kg m^{-2}.

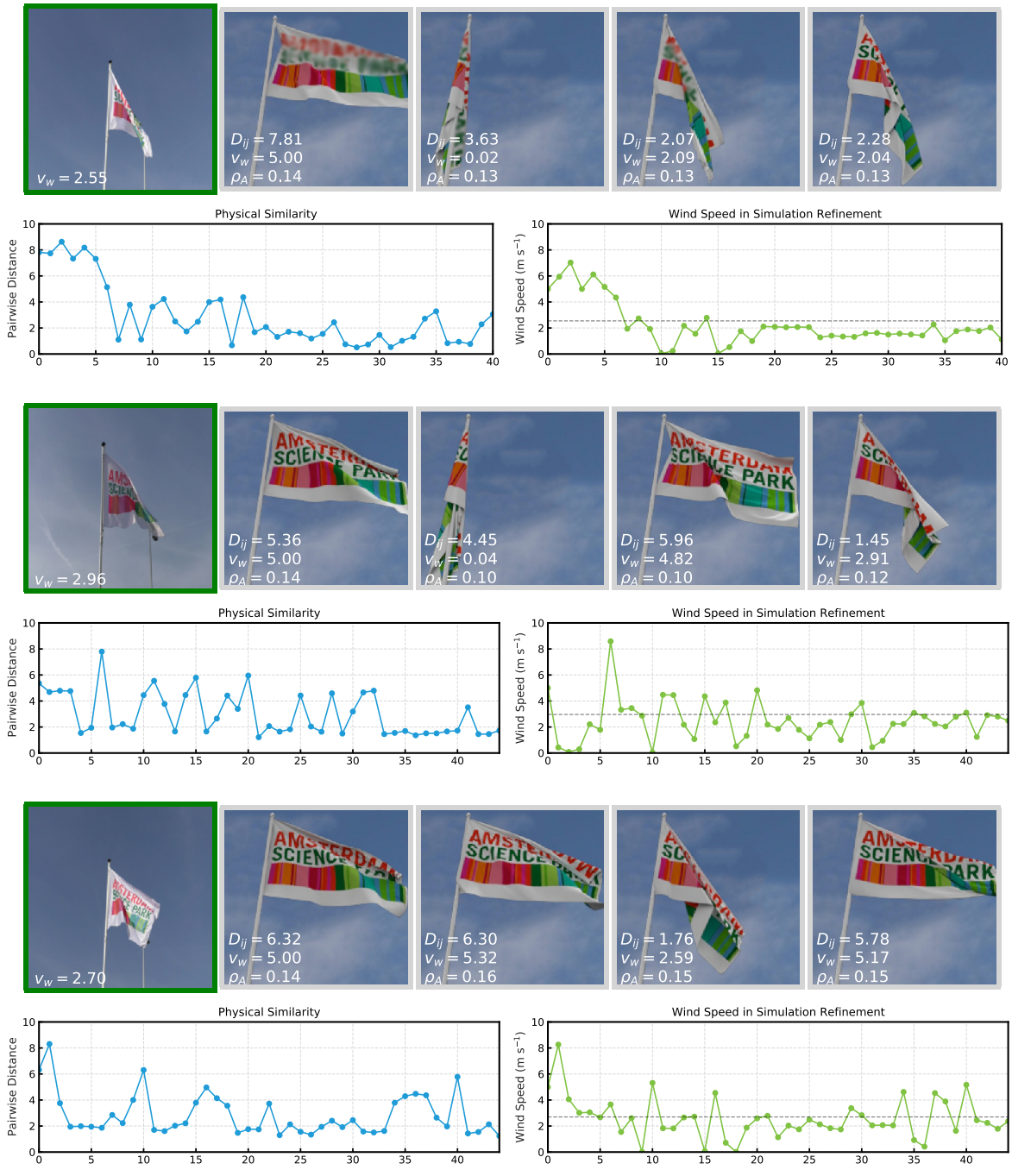

Figure 3. Additional results for our **FlagReal refined-measurements** for a random target video capturing the flag in the wind (corresponding to Figure 8 in the main paper). The five images are the center frames of the real-world target video (left) and simulations throughout the refinement process after $t = 0, 10, 20, 40$ optimization steps. Optimization is performed over all 16 intrinsic cloth parameters $\boldsymbol{\theta}_i$ and 1 external wind speed $\boldsymbol{\theta}_e$. We only visualize the simulated wind speed as it is the only parameter for which we have ground-truth (dashed line). *Top row:* successful optimization example; although the scale between real observation and simulation is different, our method is able to precisely determine the external wind speed. The real-world video has a ground-truth wind speed of 2.46 m s^{-1} while the refinement procedure finds a wind speed of 2.34 m s^{-1} in less than 10 optimization steps. *Center row:* Another successful optimization example. The real-world video has a ground-truth wind speed of 2.96 m s^{-1} while the refinement procedure finds a wind speed of 2.36 m s^{-1} after 45 refinement steps. *Bottom row:* failure case; even though after 45 steps the wind speed is approximately correct, the optimization procedure has not converged.

Listing 1. The ArcSim base **configuration for flags** [5] as JSON file to be read by the simulator. The simulation runs on a flag mesh of 3 : 2 aspect ratio in a constant wind field defined by the wind speed $\boldsymbol{\theta}_e$ parameter. During simulation we only consider a wind field in a single direction, but during rendering we use multiple relative camera orientations creating the appearance of varying wind directions. The intrinsic cloth material parameters $\boldsymbol{\theta}_i$ reside inside the material configuration file (Listing 3).

```
{
    "frame_time": 0.04,
    "frame_steps": 8,
    "duration": 20,
    "cloths": [{
        "mesh": "meshes/flag.obj",
            "transform": {
            "translate": [0, 0, 0],
            "rotate": [120, 1, 1, 1]
        },
        "materials": [{
            "data": "materials/camel-ponte-roma.json",
            "thicken": 2,
            "strain_limits": [0.95, 1.05]
        }],
        "remeshing": {
            "refine_angle": 0.3,
            "refine_compression": 0.01,
            "refine_velocity": 1,
            "size": [20e-3, 500e-3],
            "aspect_min": 0.2
        }
    }],
    "handles": [{
        "nodes": [0,3]
    }],
    "gravity": [0, 0, -9.81],
    "wind": {
        "velocity": [wind_speed, 0, 0]
    },
    "magic": {
        "repulsion_thickness": 10e-3,
        "collision_stiffness": 1e6
    }
}
```

Listing 2. The ArcSim base **configuration for hanging cloth** as JSON file to be read by the simulator. The wind speed is defined on the horizontal plane (x and y components). Again, the starting point of the intrinsic cloth material parameters $\boldsymbol{\theta}_i$ are given in Listing 3. However, in comparison to the flag simulations, we set a much larger variety of fabrics and define the fabric area weight range to correspond to the hanging cloth dataset [2].

```
{
    "frame_time": 0.04,
    "frame_steps": 8,
    "duration": 20,
    "cloths": [{
        "mesh": "meshes/square.obj",
            "transform": {
            "translate": [0, 0, 0],
            "rotate": [120, 1, 1, 1]},
        "materials": [{
            "data": "materials/camel-ponte-roma.json",
            "thicken": 1,
            "strain_limits": [0.95, 1.05]
            }],
        "remeshing": {
            "refine_angle": 0.3,
            "refine_compression": 0.01,
            "refine_velocity": 1,
            "size": [20e-3, 500e-3],
            "aspect_min": 0.2
        }
    }],
    "motions": [],
    "handles": [{"nodes": [2,3]}],
    "gravity": [0, 0, -9.8],
    "wind": {"velocity": [wind_speed_x, wind_speed_x, 0]},
    "magic": {"repulsion_thickness": 10e-3, "collision_stiffness": 1e6}
}
```

Listing 3. The **ArcSim material configuration** [5] as JSON file to be consumed by the simulator. As base material, we use "camel ponte roma" with its properties determined in the mechanical setup by [7]. This file specifies the cloth's area weight, bending stiffness coefficients and stretching coefficients. As flags are of strong, weather-resistant material, we optimize over the area weight (1×) and bending parameters (15×). Together these 16 parameters define $\boldsymbol{\theta}_i$. For hanging cloth, we also keep the bending parameters fixed to constrain the number of free parameters.

```
{
    "density": 0.135,
    "bending": [
        [36.3483e-6, 49.5855e-6, 45.7440e-6, 47.4133e-6, 20.7266e-6],
        [33.0132e-6, 29.7443e-6, 35.1036e-6, 34.0410e-6, 14.4399e-6],
        [37.1575e-6, 34.1074e-6, 33.2294e-6, 34.6855e-6, 10.4399e-6]
    ],
    "stretching": [
        [31.146198,  -12.802702,  44.028667,  31.896357],
        [78.707756,   26.754574, 268.680725,  27.743423],
        [67.368431,   77.767944, 182.273407, -14.661531],
        [113.367035,  54.802021, 175.126572,  44.657330],
        [144.294830, 111.404854, 138.422150, -29.861851],
        [143.933365,  49.654823, 191.777588,  39.491055]
    ]
}
```

Table 2. Exhaustive overview of the **render parameters** ζ for rendering the FlagSim and ClothSim datasets.

Name	**Description**	**Value/Range (Flags)**	**Value/Range (Cloth)**
`background_image`	Background image of scene	Sampled from SUN397 [9]	
`background_offset`	Background image translation	$\sim$ Uniform$(-20, +20)$	
`background_scale`	Background image scale	$\sim$ Uniform$(0.6, 1.0)$	
`sun_height`	The sun's height above the ground plane	$\sim$ Uniform$(4, 10)$	
`sun_radius`	The sun's distance to mesh	$\sim$ Uniform$(0, \ 5)$	
`sun_strength`	The sun's illumination strength	$\sim$ Uniform$(2, 10)$	
`sun_shadow_soft_size`	The sun's shadow hardness	$\sim$ Uniform$(2, 10)$	
`cycles_samples`	Cycles [1] number of render samples	50	
`cycles_bounces`	Cycles [1] light bounces, object dependent	$[0, 6]$	
`camera_height`	The height above the ground plane	$\sim$ Uniform$(0.2, 3)$	$\sim$ Uniform$(0.5, 2)$
`camera_radius`	The distance to the mesh	$\sim$ Uniform$(4, 6)$	$\sim$ Uniform$(1, 2.5)$
`camera_angle`	The orientation w.r.t. wind direction	$\sim$ Uniform$(-15, +15)$	$\sim$ Uniform$(-5, +5)$
`mesh_height`	The flag's height above the ground plane	4.6	2
`mesh_aspect_ratio`	The flag's aspect ratio	3 : 2	1 : 1
`mesh_texture`	The flag/cloth texture	Sampled from 12 countries	Sampled from [10]