Procedural Generation of Videos to Train Deep Action Recognition Networks (Supplementary Material)

César Roberto de Souza[1,3], Adrien Gaidon[2], Yohann Cabon[1], Antonio Manuel López[3]
[1]Computer Vision Group, Xerox Research Center Europe, Meylan, France
[2]Toyota Research Institute, Los Altos, CA, USA
[3]Centre de Visió per Computador, Universitat Autònoma de Barcelona, Bellaterra, Spain

1. Introduction

This material provides additional information regarding our publication. In particular, we provide in-depth details about the parametric generative model we used to generate our procedural videos, an extended version of the probabilistic graphical model (whereas the graph shown in the publication had to be simplified due to size considerations), expanded generation statistics, details about additional data modalities we include in our dataset, and results for our Cool-TSN model for the separate RGB and flow streams.

2. Generation details

In this section, we provide more details about the interpretable parametric generative model used in our procedural generation of videos, presenting an extended version of the probabilistic graphical model given in our section 3.5.

2.1. Variables

We start by defining the main random variables used in our generative model. Here we focus only on critical variables that are fundamental in understanding the orchestration of the different parts of our generation, whereas all part-specific variables are shown in Section 2.2. The categorical variables that drive most of the procedural generation are:

$$\begin{aligned}
H : h \quad & \in \{model_1, model_2, \ldots, model_{20}\} \\
A : a \quad & \in \{\text{“}clap\text{”}, \ldots, \text{“}bump\ into\ each\ other\text{”}\} \\
B : b \quad & \in \{motion_1, motion_2, \ldots, motion_{953}\} \\
V : v \quad & \in \{\text{“}none\text{”}, \text{“}random\ perturbation\text{”}, \\
& \quad \text{“}weakening\text{”}, \text{“}objects\text{”}, \text{“}blending\text{”}\} \\
C : c \quad & \in \{\text{“}kite\text{”}, \text{“}indoors\text{”}, \text{“}closeup\text{”}, \text{“}static\text{”}\} \\
E : e \quad & \in \{\text{“}urban\text{”}, \text{“}stadium\text{”}, \text{“}middle\text{”}, \\
& \quad \text{“}green\text{”}, \text{“}house\text{”}, \text{“}lake\text{”}\} \\
D : d \quad & \in \{\text{“}dawn\text{”}, \text{“}day\text{”}, \text{“}dusk\text{”}\} \\
W : w \quad & \in \{\text{“}clear\text{”}, \text{“}overcast\text{”}, \text{“}rain, \text{“}fog\text{”}\}
\end{aligned} \tag{1}$$

where H is the human model to be used by the protagonist, A is the action category to be generated, B is the base motion sequence used for the action, V is the variation to be applied to the base motion, C is the camera behavior, E is the environment of the virtual world where the action will take place, D is the day phase, W is the weather condition.

These categorical variables are in turn controlled by a group of parameters that can be adjusted in order to drive the sample generation. These parameters include the θ_A parameters of a categorical distribution on action categories A, the θ_W for weather conditions W, θ_D for day phases D, θ_H for model models H, θ_V for variation types V, and θ_C for camera behaviors C.

Additional parameters include the conditional probability tables of the dependent variables: a matrix of parameters θ_{AE} where each row contains the parameters for categorical distributions on environments E for each action category A, the matrix of parameters θ_{AC} on camera behaviors C for each action A, the matrix of parameters θ_{EC} on camera behaviors C for each environment E, and the matrix of parameters θ_{AB} on motions B for each action A.

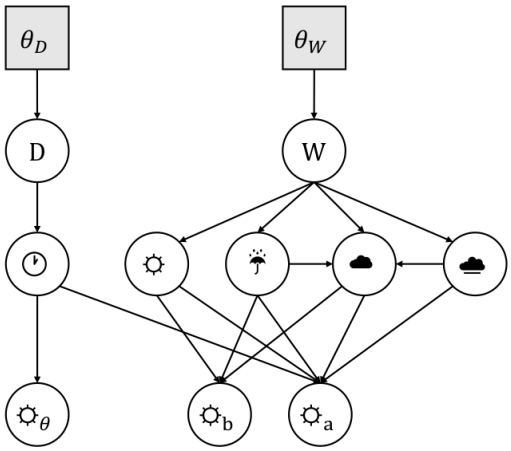
Finally, other relevant parameters include T_{min}, T_{max}, and T_{mod}, the minimum, maximum and most likely durations for the generated video. We denote the set of all parameters in our model by $\boldsymbol{\theta}$.

2.2. Model

The complete interpretable parametric probabilistic model used by our generation process, given our generation parameters $\boldsymbol{\theta}$, can be written as:

$$\begin{aligned}
P(H,&A,L,B,V,C,E,D,W \mid \boldsymbol{\theta}) = \\
& P_1(D,W \mid \boldsymbol{\theta})\ P_2(H \mid \boldsymbol{\theta}) \\
& P_3(A,L,B,V,C,E,W \mid \boldsymbol{\theta})
\end{aligned} \tag{2}$$

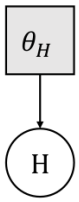
where P_1, P_2 and P_3 are defined by the probabilistic graphical models represented on Figure 1, 2 and 3, respectively. We use extended plate notation [1] to indicate repeating variables, marking parameters (non-variables) using filled rectangles.



$W : w \in \{clear, overcast, rain, fog\}$
$D : d \in \{dawn, day, dusk\}$

θ_W : $\{\omega_i \in \mathbb{R} | \sum_{i \in W} \omega_i = 1\}$ θ_D : $\{\omega_i \in \mathbb{R} | \sum_{i \in D} \omega_i = 1\}$
: $b \in \{0,1\}, sun\ enabled$ θ : $\theta \in \mathbb{R}, earth\ rotation\ angle$
: $r \in \{0,1\}, rain\ enabled$: $t \in \mathbb{R}, world\ clock\ time$
: $c \in \{0,1\}, cloud\ enabled$ b : $b \in \mathbb{R}, sun\ brightness$
: $f \in \{0,1\}, fog\ enabled$ a : $a \in \mathbb{R}, ambient\ brightness$

Figure 1: Probabilistic graphical model for $P_1(D, W \mid \boldsymbol{\theta})$, the first part of our parametric generator (world time and weather).



$H : h \in \{model_1, model_2, \ldots, model_{20}\}$
$\theta_W : \{\omega_i \in \mathbb{R} \,|\, \sum_{i \in H} \omega_i = 1\}$

Figure 2: Probabilistic graphical model for $P_2(H \mid \boldsymbol{\theta})$, the second part of our parametric generator (human models).

2.3. Distributions

The generation process makes use of four main families of distributions: categorical, uniform, Bernoulli and triangular. We adopt the following three-parameter formulation for the triangular distribution:

$$Tr(x \mid a, b, c) = \begin{cases} 0 & \text{for } x < a, \\ \frac{2(x-a)}{(b-a)(c-a)} & \text{for } a \leq x < c, \\ \frac{2}{b-a} & \text{for } x = c, \\ \frac{2(b-x)}{(b-a)(b-c)} & \text{for } c < x \leq b, \\ 0 & \text{for } b < x. \end{cases} \quad (3)$$

All distributions are implemented using the open-source Accord.NET Framework[1]. While we have used mostly uniform distributions to create the dataset used in our experiments, we have the possibility to bias the generation towards values that are closer to real-world dataset statistics.

Day phase. As real-world action recognition datasets are more likely to contain video recordings captured during daylight, we fixed the parameter θ_D such that

$$\begin{aligned} P(D = dawn \mid \theta_D) &= 1/3 \\ P(D = day \mid \theta_D) &= 1/3 \\ P(D = dusk \mid \theta_D) &= 1/3 \\ P(D = night \mid \theta_D) &= 0. \end{aligned} \quad (4)$$

We note that although our system can also generate night samples, we do not include them in PHAV at this moment.

Weather. In order to support a wide range of applications of our dataset, we fixed the parameter θ_W such that

$$\begin{aligned} P(W = clear \mid \theta_W) &= 1/4 \\ P(W = overcast \mid \theta_W) &= 1/4 \\ P(W = rain \mid \theta_W) &= 1/4 \\ P(W = fog \mid \theta_W) &= 1/4, \end{aligned} \quad (5)$$

ensuring all weather conditions are present.

Camera. In addition to the Kite camera, we also included specialized cameras that can be enabled only for certain environments (Indoors), and certain actions (Close-Up). We fixed the parameter θ_C such that

$$\begin{aligned} P(C = kite \mid \theta_C) &= 1/3 \\ P(C = closeup \mid \theta_C) &= 1/3 \\ P(C = indoors \mid \theta_C) &= 1/3. \end{aligned} \quad (6)$$

However, we have also fixed θ_{CE} and θ_{AC} such that the Indoors camera is only available for the house environment, and that the Close-Up camera can also be used for the *BrushHair* action in addition to Kite.

Environment, human model and variations. We fixed the parameters θ_E, θ_H, and θ_V using equal weights, such that the variables E, H, and V can have uniform distributions.

Base motions. All base motions are weighted according to the minimum video length parameter T_{min}, where motions whose duration is less than T_{min} are assigned weight zero, and others are set to uniform, such that

$$P(B = b | T_{min}) \propto \begin{cases} 1 & \text{if } length(b) \geq T_{min} \\ 0 & \text{otherwise} \end{cases} \quad (7)$$

We then perform the selection of a motion B given a category A by introducing a list of regular expressions associated with each of the action categories. We then compute

[1] http://accord-framework.net

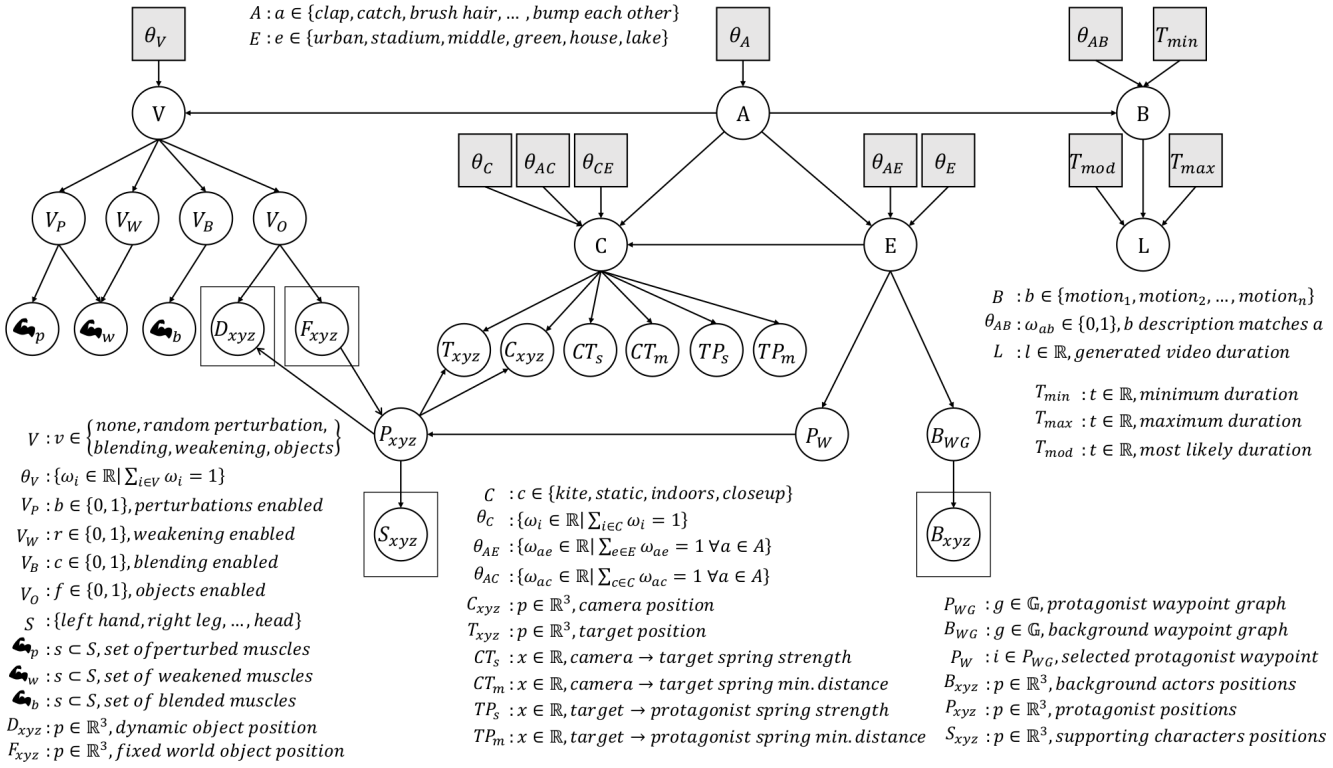


Figure 3: Probabilistic graphical model for $P_3(A, L, B, V, C, E, W \mid \boldsymbol{\theta})$, the third part of our parametric generator (scene and action preparation).

matches between the textual description of the motion in its source (*e.g.,* short text descriptions in [2]) and these expressions, such that

$$(\theta_{AB})_{ab} = \begin{cases} 1 & \text{if } match(\text{regex}_a, \text{desc}_b) \\ 0 & \text{otherwise} \end{cases} \forall a \in A, \forall b \in B. \tag{8}$$

We then use θ_{AB} such that

$$P(B = b \mid A = a, \theta_{AB}) \propto (\theta_{AB})_{a,b}. \tag{9}$$

Weather elements. The selected weather W affects world parameters such as the sun brightness, ambient luminosity, and multiple boolean variables that control different aspects of the world (*cf.* Figure 1). The activation of one of these boolean variables (*e.g.,* fog visibility) can influence the activation of others (*e.g.,* clouds) according to Bernoulli distributions ($p = 0.5$).

World clock time. The world time is controlled depending on D. In order to avoid generating a large number of samples in the borders between two periods of the day, where the distinction between both phases is blurry, we use different triangular distributions associated with each phase, giving a larger probability to hours of interest (sunset, dawn, noon) and smaller probabilities to hours at the transitions. We therefore define the distribution of the world clock times $P(T)$ as:

$$P(T = t \mid D) \propto \sum_{d \in D} P(T = t \mid D = d) \tag{10}$$

where

$$\begin{aligned} P(T = t \mid D = dawn\ ) &= Tr(t \mid 7h,\ \ 10h,\ 9h\ \ ) \\ P(T = t \mid D = day\ ) &= Tr(t \mid 10h,\ \ 16h,\ 13h\ ) \\ P(T = t \mid D = dusk\ ) &= Tr(t \mid 17h,\ \ 20h,\ 18h\ ). \end{aligned} \tag{11}$$

Generated video duration. The selection of the clip duration L given the selected motion b is performed considering the motion length L_b, the maximum video length T_{min} and the desired mode T_{mod}:

$$\begin{aligned} P(L = l \mid B = b) = Tr(a &= T_{min}, \\ b &= min(L_b, T_{max}), \\ c &= min(T_{mod}, L_b)) \end{aligned} \tag{12}$$

Actors placement and environment. Each environment E has at most two associated waypoint graphs. One graph refers to possible positions for the protagonist, while an additional second graph gives possible positions B_{WG} for spawning background actors. Indoor scenes (*cf.* Figure 4)

Figure 4: Example of indoor and outdoors scenes.

do not include background actor graphs. After an environment has been selected, a waypoint P_W is randomly selected from the graph using an uniform distribution. The protagonist position P_{xyz} is then set according to the position of P_W. The S_{xyz} position of each supporting character, if any, is set depending on P_{xyz}. The position and destinations for the background actors are set depending on B_{WG}.

Camera placement and parameters. After a camera has been selected, its position C_{xyz} and the position T_{xyz} of the target are set depending on the position P_{xyz} of the protagonist. The camera parameters are randomly sampled using uniform distributions on sensible ranges according to the observed behavior in Unity. The most relevant secondary variables for the camera are shown in Figure 3. They include Unity-specific parameters for the camera-target (CT_s, CT_m) and target-protagonist springs (TP_s, CT_m) that can be used to control their strength and a minimum distance tolerance zone in which the spring has no effect (remains at rest). In our generator, the minimum distance is set to either 0, 1 or 2 meters with uniform probabilities. This setting is responsible for a "delay" effect that allows the protagonist to not be always in the center of camera focus (and thus avoiding creating such bias in the data).

Action variations. After a variation mode has been selected, the generator needs to select a subset of the ragdoll muscles (*cf.* Figure 5) to be perturbed (random perturbations) or to be replaced with movement from a different motion (action blending). These muscles are selected using a uniform distribution on muscles that have been marked as non-critical depending on the previously selected action category A. When using weakening, a subset of muscles will be chosen to be weakened with varying parameters independent of the action category. When using objects, the choice of objects to be used and how they have to be used is also dependent on the action category.

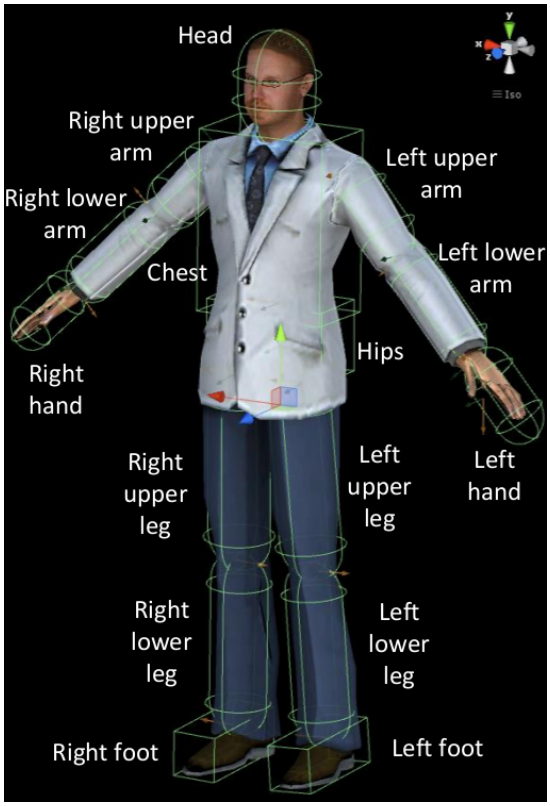


Figure 5: Ragdoll configuration with 15 muscles.

Object placement. Interaction with objects can happen in two forms: dynamic or static. When using objects dynamically, an object of the needed type (*e.g.,* bow, ball) is spawned around (or is attached to) the protagonist at a predetermined position, and is manipulated using 3D joints, inverse kinematics, or both. When using static (fixed) objects, the protagonist is moved to the vicinity of an object already present in the virtual world (*e.g.,* bench, stairs).

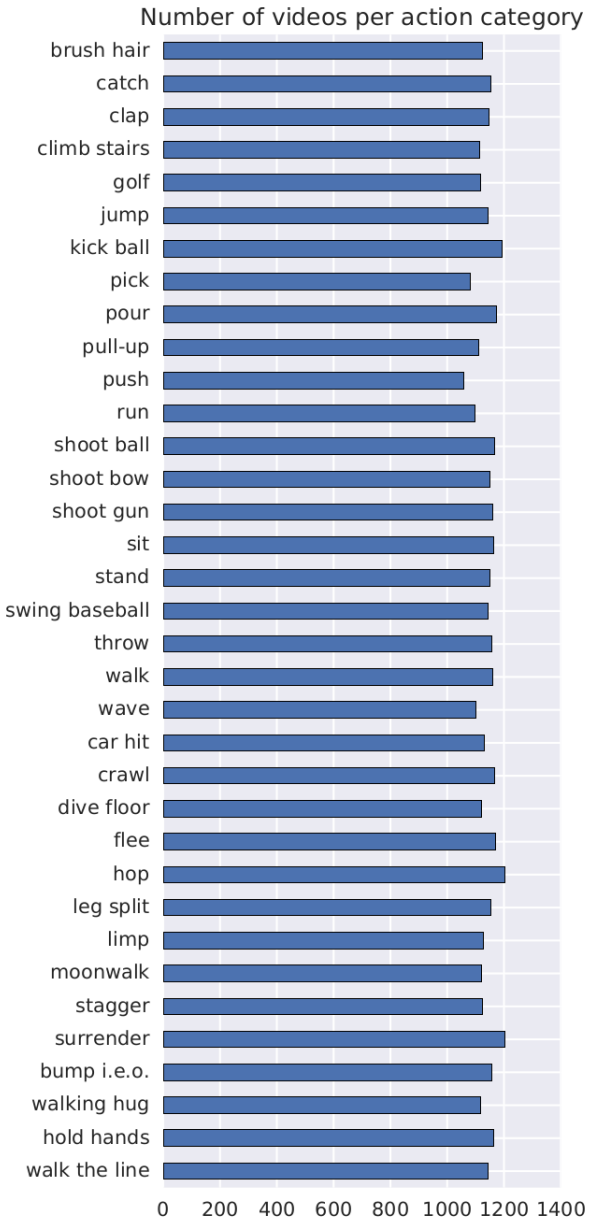


Figure 6: Plot of the number of videos generated for each category in the version of our PHAV dataset used in the publication.

2.4. Statistics

In this section we show statistics about the version of PHAV that has been used in experimental section of our paper. A summary of the key statistics for the generated dataset can be seen in Table 1. Figure 6 shows the number of videos generated after each action category in PHAV. As it can be seen, the number is higher than 1,000 samples for all categories.

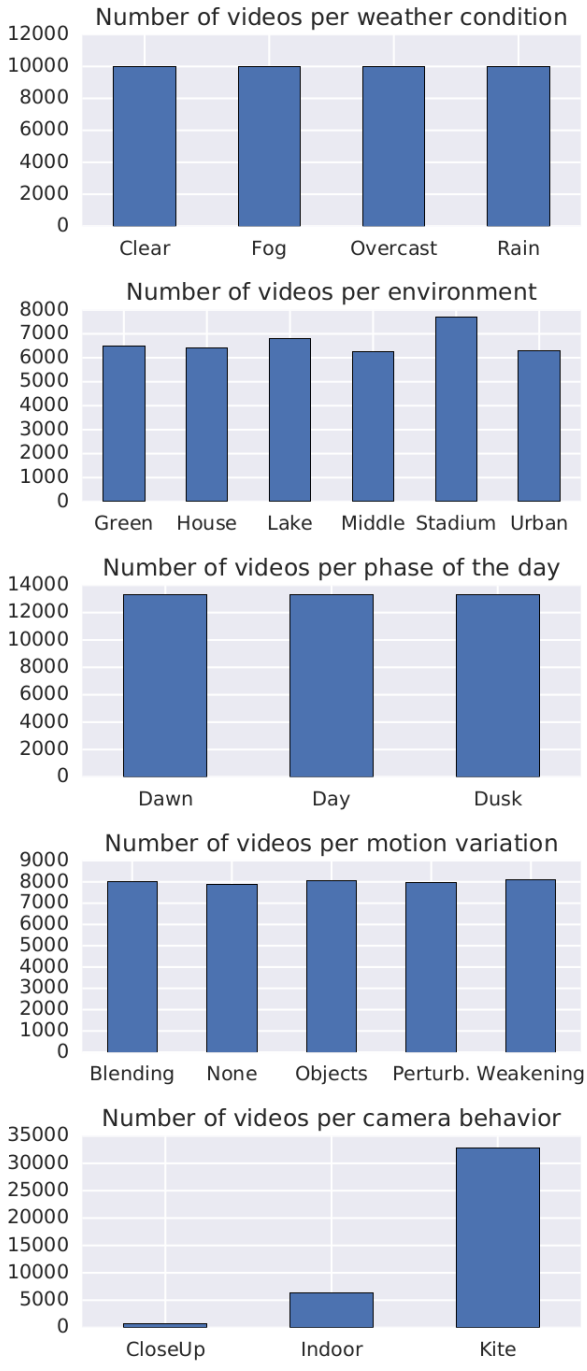


Figure 7: Plot of the number of videos per parameter value.

Figure 7 shows the number of videos generated by value of each main random generation variable. The histograms reflect the probability values presented in Section 2.3. While our parametric model is flexible enough to generate a wide range of world variations, we have focused on generating videos that would be more similar to those in the target datasets.

Statistic	Value
Clips	39,982
Total dataset frames	5,996,286
Total dataset duration	2d07h31m
Average video duration	4.99s
Average number of frames	149.97
Frames per second	30
Video dimensions	340x256
Average clips per category	1,142.3
Image modalities (streams)	6

Table 1: Statistics of the generated dataset instance.

2.5. Data modalities

Although not discussed in the paper, our generator can also output multiple data modalities for a single video, which we include in our public release of PHAV. Those data modalities are rendered roughly at the same time using Multiple Render Targets (MRT), resulting in a superlinear speedup as the number of simultaneous output data modalities grow. The modalities in our public release include:

Rendered RGB Frames. Those are the RGB frames that constitute the action video. They are rendered at 340x256 resolution and 30 FPS such that they can be directly feed to Two-Stream style networks. Those frames have been post-processed with 2x Supersampling Anti-Aliasing (SSAA), motion blur, bloom, ambient occlusion, screen space reflection, color grading, and vignette.

Semantic Segmentation. Those are the per-pixel semantic segmentation ground-truths containing the object class label annotations for every pixel in the RGB frame. They are encoded as sequences of 24-bpp PNG files with the same resolution as the RGB frames. We provide 63 pixel classes, including the same 14 classes used in Virtual KITTI [3], classes specific for indoor scenarios, classes for dynamic objects used in every action, and 27 classes depicting body joints and limbs.

Instance Segmentation. Those are the per-pixel instance segmentation ground-truths containing the person identifier encoded as different colors in a sequence of frames. They are encoded in exactly the same way as the semantic segmentation ground-truth explained above.

Depth Map. Those are depth map ground-truths for each frame. They are represented as a sequence of 16-bit grayscale PNG images with a fixed far plane of 655.35 meters. This encoding ensures that a pixel intensity of 1 can correspond to a 1cm distance from the camera plane.

Optical Flow. Those are the ground-truth (forward) optical flow fields computed from the current frame to the next frame. We provide separate sequences of frames for the horizontal and vertical directions of optical flow represented as sequences of 16-bpp JPEG images with the same resolution as the RGB frames.

Figure 8: Example frames and data modalities for a synthetic action (car hit, left) and MOCAP-based action (sit, right). From top to bottom: Rendered RGB Frames, Semantic Segmentation, Instance Segmentation, Depth Map, Horizontal Optical Flow, and Vertical Optical Flow. Depth image brightness has been adjusted in this figure to ensure visibility on paper.

Fraction	UCF101	UCF101+PHAV	HMDB51	HMDB51+PHAV
1%	25.9	**27.7**	8.1	**12.7**
5%	68.5	**71.5**	30.7	**37.3**
10%	80.9	**84.4**	44.2	**49.7**
25%	89.0	**90.4**	54.8	**60.7**
50%	92.5	**92.7**	62.9	**65.8**
100%	92.8	**93.3**	67.8	**70.1**

Table 2: TSN and Cool-TSN (+PHAV) with different fractions of real-world training data (split 1).

Raw RGB Frames. Those are the raw RGB frames before any of the post-processing effects mentioned above are applied. This modality is mostly included for completeness, and has not been used in experiments shown in the paper.

Pose, location and additional information. Although not an image modality, our generator can also produce textual annotations for every frame. Annotations include camera parameters, 3D and 2D bounding boxes, joint locations in screen coordinates (pose), and muscle information (including muscular strength, body limits and other physical-based annotations) for every person in a frame.

3. Experiments

In this section, we show more details about the experiments shown in the experimental section of our paper.

Table 2 shows the impact of training our Cool-TSN models using only a fraction of the real world data (Figure 7 of original publication) in a tabular format. As it can be seen, mixing real-world and virtual-world data from PHAVis helpful in almost all cases.

Figure 9 shows the performance of each network stream separately. The second image on the row shows the performance on the Spatial (RGB) stream. The last image on the row shows the performance for the Temporal (optical flow) stream. One can see how the optical flow stream is the biggest responsible for the good performance of our Cool-TSN, including when using very low fractions of the real data. This confirms that our generator is indeed producing plausible motions that are being helpful to learn both the virtual and real-world data sources.

4. Video

We have included a video (*cf.* Figure 10) as additional supplementary material to our submission. The video shows random subsamples for a subset of the action categories in PHAV. Each subsample is divided into 5 main variation categories. Each video is marked with a label indicating the variation being used, using the legend shown in Figure 11.

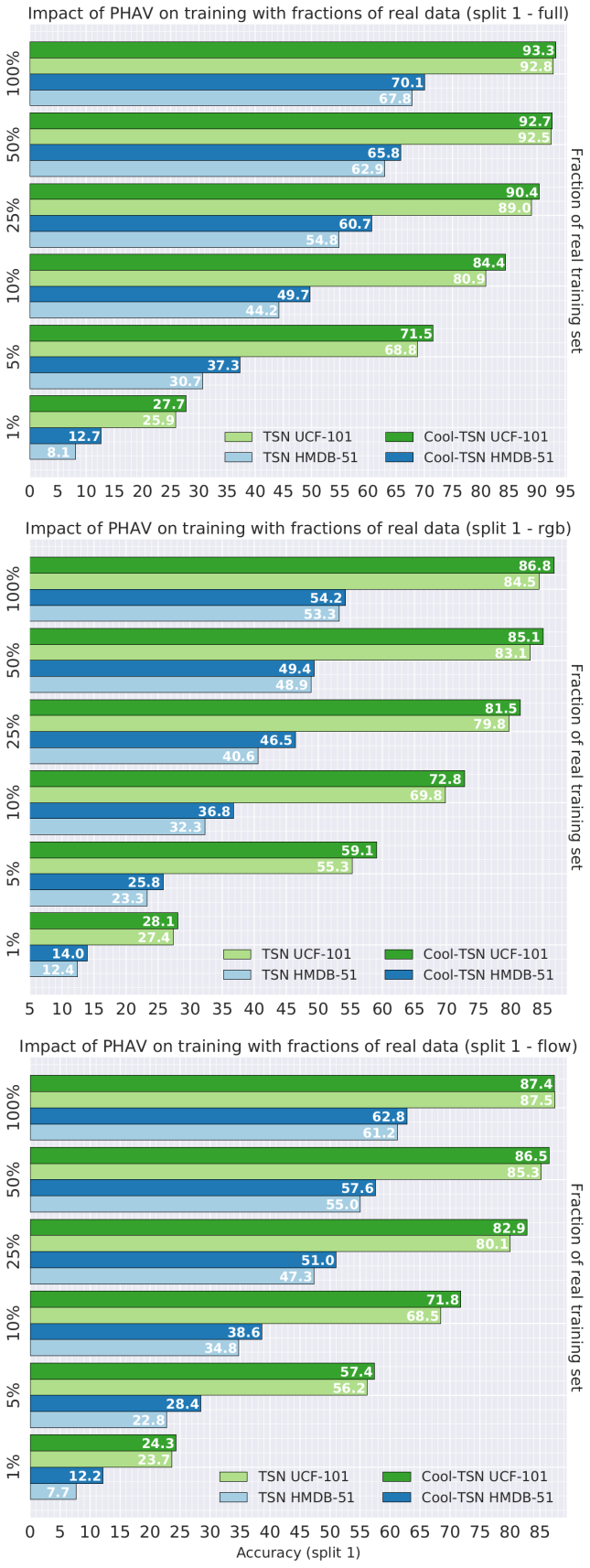


Figure 9: TSN and Cool-TSN results for different amounts of training data for combination and separate streams.

5. Conclusion

Our detailed graphical model shows how a complex video generation can be driven through few, simple parameters. We have also shown that generating action videos while still taking the effect of physics into account is a challenging task. Nevertheless, we have demonstrated that our approach is feasible through experimental evidence on two real-world datasets, disclosing further information about the performance of each RGB and optical flow channels in this supplementary material.

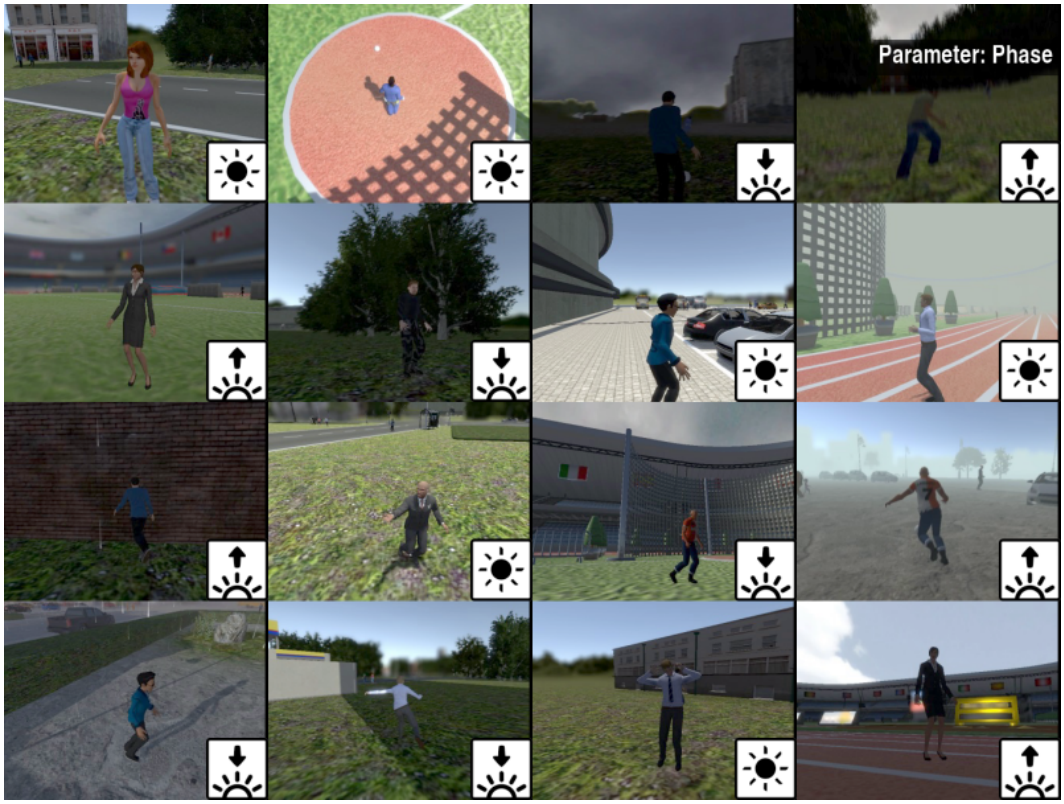


Figure 10: Sample frame from the supplementary video available at http://adas.cvc.uab.es/phav/.

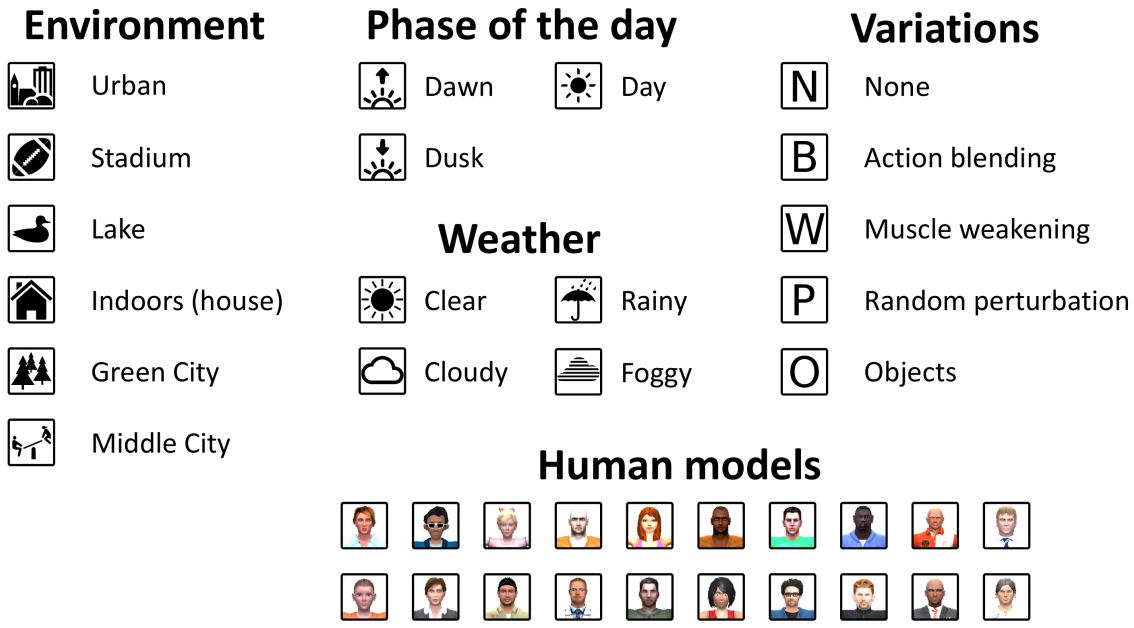


Figure 11: Legend for the variations shown in the video.

References

[1] C. M. Bishop. *Pattern Recognition and Machine Learning*. 2006. 1

[2] Carnegie Mellon Graphics Lab. Carnegie Mellon University Motion Capture Database, 2016. 3

[3] A. Gaidon, Q. Wang, Y. Cabon, and E. Vig. Virtual Worlds as Proxy for Multi-Object Tracking Analysis. *CVPR*, pages 4340–4349, 2016. 6