Articulated pose estimation with tiny synthetic videos

Dennis Park
UC Irvine
iypark@ics.uci.edu

Deva Ramanan
UC Irvine
dramanan@ics.uci.edu

Abstract

We address the task of articulated pose estimation from video sequences. We consider an interactive setting where the initial pose is annotated in the first frame. Our system synthesizes a large number of hypothetical scenes with different poses and camera positions by applying geometric deformations to the first frame. We use these synthetic images to generate a custom labeled training set for the video in question. This training data is then used to learn a regressor (for future frames) that predicts joint locations from image data. Notably, our training set is so accurate that nearest-neighbor (NN) matching on low-resolution pixel features works well. As such, we name our underlying representation "tiny synthetic videos". We present quantitative results the Friends benchmark dataset that suggests our simple approach matches or exceed state-of-the-art.

1. Introduction

Humans have remarkable abilities to synthesize mental imagery "with the minds eye" [25, 10]. To examine this phenomena, consider the practical problem of tracking human poses in a one-take video clip, say "Phoebe" in the televised show "Friends". Given the first frame of the video, humans can immediately picture in mind what the other frames might look like. Phoebe, who was folding arms in the first frame, may be reaching her arm to grab a cup or be answering the phone in the other frames. In addition, the plate of salad might become empty; the ketchup bottle on the table might be in her hand or moved to the other side of the table. Even though such a generative approach to recognition seems intuitively plausible, most contemporary approaches takes a decidedly discriminative route [17, 34, 11].

Our approach: We describe a radically simple architecture for articulated pose estimation based on scene synthesis. We consider the setting where the articulated pose of the person is provided in the first frame. Our system works by synthesizing a large number of hypothetical scenes with different poses, dynamic objects (e.g., cups) and camera locations using image-based rendering algorithms. We use these synthetic scenes to generate a custom labeled training set for the video in question. This custom training data is then used to learn a classifier (or rather regressor) that predicts joint locations from image features. Notably, our training set is so accurate that *nearest-neighbor (NN) matching on pixel values produce state-of-the-art performance*. Specifically, we find that low-resolution pixel features work well. As such, we name underlying representation **tiny synthetic videos**.

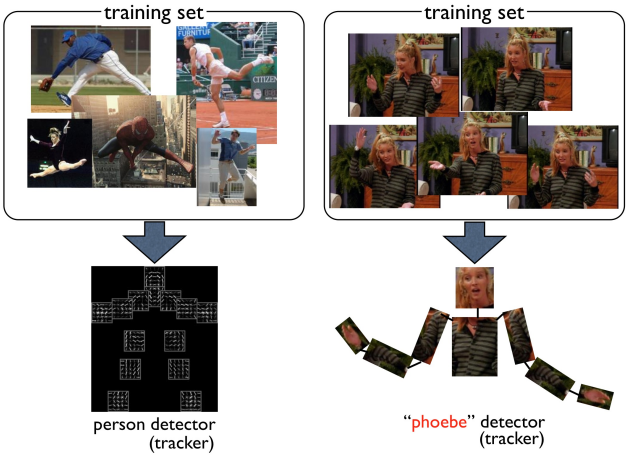

Figure 1: *Overfit the video!* We propose to use synthetic video frames that emulate hypothetical test frames as training data for performing recognition in video. Previous approaches use as base models generic detectors trained using images in the wild (**left**). We show that, by using training data customized to a particular video (**right**), one can achieve state-of-the-art performance on challenging pose estimation problem even with simple models and features.

Tracking by detection: We tackle the problem of articulated tracking using a tracking-by-detection framework, applying a detector/estimator at each frame. Typically, one makes use of a generic pose detector trained using images of arbitrary persons and backgrounds. This seems to be overkill, since it forces the detector to recognize Obama's pose in the Whitehouse even though the detector only needs

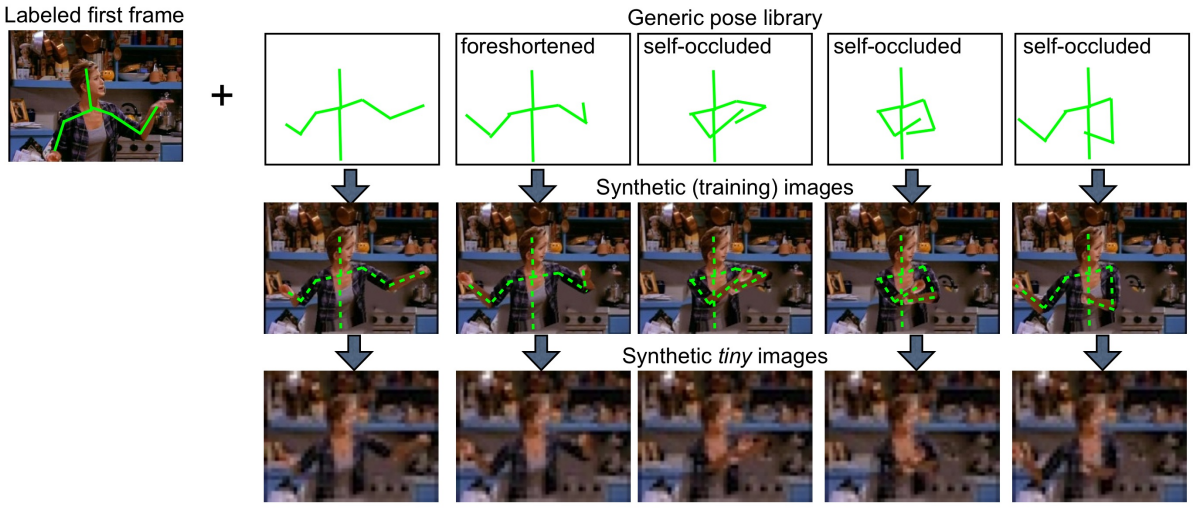

Figure 2: Synthesis overview. We assume the first frame of the video, as well as its pose label, is available upon test time (**left**). We define a generic library of poses that represents all potential poses (**top**). We use an image-based rendering engine to warp body parts from the first frame into target poses from the library (**middle**). Notably, we make use of layered compositing to accurately render self-occlusions between the arm and torso, a notoriously difficult problem for articulated models. We use the collection of rendered frames to construct a custom "Rachel" training data as shown in Fig. 1. This training set is powerful enough to produce state-of-art pose estimation results using low-resolution pixel features (**bottom**).

to recognize Phoebe in her living room. Instead, we propose to train a highly-tuned appearance model that *overfits* the particular video using a large set of synthetic training data.

Data-augmentation: Our approach of image-based synthesis is inspired by learning architectures that apply synthetic perturbations to training data. Such "data-augmentation" appears to be crucial components of state-of-the-art methods like deep learning [27, 17]. However, instead of applying simple perturbations like rotations, we make use of an image-based rendering engine to "cut and paste" regions of image in a layered fashion, generating new scene configurations. With a rich enough synthesis engine, the resulting learning algorithm (a) does not need to generalize to unseen test poses (because they can be directly synthesized) and (b) does not need to be invariant to nuisance factors like color (because only scene-specific colors will be synthesized). Both observations imply that "its all about the data", and that simple learning architectures (such as NN-matching) and pixel features will suffice.

Self-occlusions: Our simple approach addresses a notorious difficulty of articulated pose estimation - self-occlusions. It is difficult to estimate poses where the torso is occluded by the arm. This is because the resulting edge patterns are heavily dependent on clothing. Our approach synthesizes the "right" clothing and edge patterns. To synthesize accurate self-occlusions, our rendering engine makes use of a 2.1D layered depth model. Similarly, our approach synthesize body parts that appear shrunk due to out-of-plane rotation. This is particularly effective for lower arms [29].

Overview: After discussing related work in Sec.2, we describe our synthesis engines in Sec.3. Pixel synthesis is the process of generating a synthetic frame given a target pose and pose-annotated first frame; pose synthesis defines a set of target poses to synthesize. In Sec.4, we describe a simple nearest-neighbor algorithm for estimating pose. In Sec.5, we diagnose and evaluate our approach for the task of estimating upper body pose using *Friends* dataset.

2. Related work

Visual tracking: The problem of visual tracking have been addressed in various settings of inputs and initializations; first-frame labeled [38], online tracking [36], interactive tracking [5], etc. Articulated tracking [12], [39], [29], [3] recently gained attention. See [32] for complete discussion. Much recent work has focused on single-frame pose estimation, as such methods will likely be useful to (re) initialize a tracker. Our work requires the first frame to be labeled (a historically commonplace assumption [8, 31, 15, 22]), but differs from past initialized trackers in that no temporal tracking is performed (making our approach resistant to drift).

Appearance models: Tracking with learned appearance models [26, 14] have proved to be effective. Our work is closest to [15] in that they use labeled first frame to learn an appearance model that is then used to track the articulation of human body in subsequent frames. Our work is

also well aligned with [26], [39] in that they attempt to use consistency existing throughout the video. But our work is unique in that our system synthesizes custom training data to exploit consistency without using temporal cues.

Layered shape models: Since the pioneering work of [35], layered shape model as a weak form of a 3D model has been widely accepted as a useful representation for image formation. Our work is closely related to [37], where the authors introduce generative probabilistic models that formulate layered models for object detection and segmentation.

Synthetic training data: There exists a steady body of work that has examined pose estimation using (partially) synthetic training data. Perhaps the earliest example dates back to [30], who use a large set of rendered poses for nearest-neighbor (pose) regression. [18] generate synthetic rendering of real objects under synthetic backgrounds, using green-screening. The recent work of [13] has generated 3-million frame dataset of synthetic images of 3D articulated models in real backgrounds. Our work differs in that we perform "image-based rendering", cutting and pasting existing images to yield novel ones. From this perspective, our approach is most related to [24], who fit 3D articulated models to real images, and generate synthetic renderings by slightly perturbing joint angles. However, in our case, we do not need to synthesize appearance variations since we want to train a model that "overfits" to the appearance of a particular figure in the video.

3. Synthesis engine

At the heart of our approach is a simple 2.1D synthesis engine that artificially generates hypothetical video frames, given the labeled first frame of the video. Unlike other synthesis-based approaches which produce high-quality synthetic images ([24], [13]) our goal is to produce a large set of *reasonably* photorealistic images which captures most of the variability expected in the future frames. Most importantly it captures various poses, but also locations, scale, camera movement, and other dynamic objects in the scene (assuming they are also labeled in the first frame). The synthesis process consists of two components: pose synthesis and pixel synthesis, which are discussed in the following subsections. Fig. 2 summarizes the overall process.

Pose parameterization: We parameterize the appearance of a person as "cardboard puppet" model [15]. Specifically, each part p is represented by a triple: its relative orientation to its parent part, foreshortening ratio, and depth layer (θ, r, d). A pose $\mathcal{P}$ is defined by its root location and scale (x_0, y_0, s_0) and a collection of $N = 9$ upper body parts: left/right upper arms, lower arms, hands; and torso, neck, face. That is, $\mathcal{P} = (x_0, y_0, s_0, (\theta, r, d)_{1:N})$.

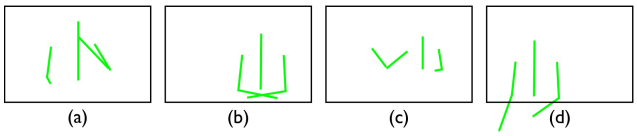

Figure 3: Pose synthesis. The pose pool consists of diverse poses in different scales and locations. It includes challenging cases, such as (from left to right) self-occlusion, interacting parts, various scales, and truncation.

Assumption: Our synthesis process assumes that the first frame of the video is given with a human annotation $\mathcal{P}_1$. The annotation is in the skeleton format shown in Fig. 2, which require users to provide 9 body joint keypoint locations and their depth layers. From this annotation, we derive the pose parameters with a anthropometric-based heuristic to decide scales of parts (e.g., we assume all arms are .5 headlength-wide [19]).

3.1. Pose synthesis

In this section, we describe a generic library of poses P. We use this library to synthesize a custom training set given the first annotated frame of a test video. Ideally, we would synthesize all possible poses, locations and scales of the human, all possible camera translations, and all possible dynamic scene elements. We make simplifying assumptions that the video mostly stabilized (implying we need to only synthesize small camera translations) and that the central figure and interacting objects are the only dynamic parts of the scene. It turns out that such assumptions hold for a large amount of televised footage.

Specifically, we uniformly sample parameters in $\mathcal{P}$ by enumerating over discrete values in the following domain:

$$\begin{aligned} &(x_0, y_0, s_0) \in \text{Near}(x_0', y_0', s_0') \\ &\theta_{shld} \in [0, \pi), \quad \theta_{elb} \in [0, 2\pi) \\ &r \in [0.2, 1], \quad d \in D, \end{aligned} \tag{1}$$

where (x_0', y_0', s_0') is the root location and scale of the figure $\mathcal{P}_1$ in frame 1, θ are joint angles of shoulders and elbows, and D is a small set of depth layer configurations consistent with frontal and side viewpoints (e.g. arms are always in front of torsos for frontal viewpoints). The above synthesizes root locations and scales around a neighborhood of those from the first frame. This simple procedure works well for synthesizing plausible upper-body poses. We show example poses in Fig. 3.

3.2. Pixel synthesis

The pixel synthesis engine takes the labeled first frame, $(I_1, \mathcal{P}_1)$, and target pose from a pose library $\mathcal{P} \in P$ as input, and produces a synthetic image I. We illustrate this process in Fig. 4. It is based on a 2.1D image representation

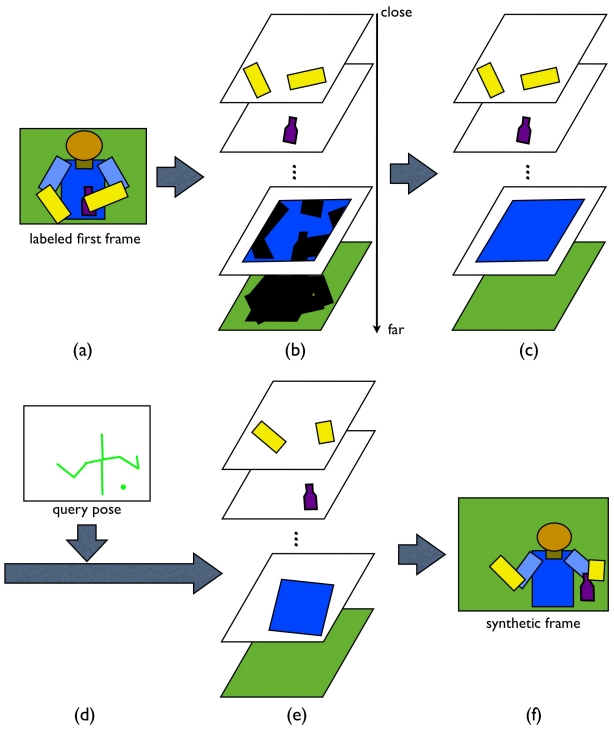

Figure 4: Pixel synthesis. Our rendering engine is based on a 2.1D representations of images. We first decompose the labeled first frame **(a)** into multiple depth layers **(b)**. Occlusion (black) regions are estimated using standard hole-filling algorithms **(c)**. Given a target pose **(d)**, we warp corresponding body parts from (c) into new locations **(e)** by rotating and scaling them. By compositing each layer according to a target depth ordering, we produce a synthetic frame **(f)**.

[37, 35], where each pixel is augmented by its depth layer. To derive a part region from pose parameters (s, θ, r, d), we need to define a forward kinematics model and a shape model.

Forward kinematics. $\mathcal{P}$ represents pose in a *local* coordinate frame because it more natural for the pose synthesis algorithm in Sec. 3.1. To render an associated image, we need to compute part locations in *global* image coordinates. This is straightforward with classic forward kinematics ([23], [21]). Specifically, given $\mathcal{P}$ and a tree-structured graph $T = (V, E)$ specifying part connectivity, we derive two global pose parameters for a part indexed by i; orientation angle θ_i^g and translation vector t_i^g. t_i^g is the pixel position of part i's joint (e.g. elbow for lower arm). For notational simplicity, we drop the dependance on global scale s_0:

$$\theta_i^g = \theta_{par(i)}^g + \theta_i \quad (2)$$

$$t_i^g = t_{par(i)}^g + \text{Rot}(\theta_{par(i)}) l_i r_i \quad (3)$$

$$\theta_{root}^g = \theta_{root} \quad (4)$$

$$t_{root}^g = (x_0, y_0) \quad (5)$$

where $par(i)$ is the part index of parent of i as defined in graph T, Rot is a 2D rotation matrix, and l_i is the location offset of joint with i with respect to the joint of its parent (e.g. default location of elbow in the reference frame of shoulder). We now can write the global location, orientation, and foreshortening of part i as

$$g_i = (t_i^g, \theta_i^g, r_i)$$

Part shapes: Let us write $M_i \in \{0,1\}^{W \times H}$ for a binary support mask for part i transformed to location g_i. This mask is computed by translating, rotating, and scaling a canonical support mask for part i [1, 9, 15]. We use a *mean shape mask* obtained by averaging annotated part support masks from a generic dataset of segmented people (not from the testset). We visualize example shape masks in Fig.5.

Part textures: Let us write $R_i \in \mathbb{R}^{W \times H \times 3}$ for a RGB texture map for part i transformed to location g_i. We learn part-specific texture maps from the first frame I_1. We do this by computing a binary pixel region mask for each part i. Each region is divided into two types of subregions: visible regions and occluded regions (due to overlapping parts in a closer depth layer). Most layers include occluded regions. We estimate occluded pixel values using standard hole-filling algorithms [4, 6]. We experimented with various approaches such as PatchMatch [2], but found simple linear interpolation to look well. We visualize the procedure for part texture extraction in Fig. 4(a)-(c).

Compositing layers: Given a target pose $\mathcal{P}$ with associated global part positions $\{g_i\}$, we wish to create the image by recompositing the layered part textures. Let us write $M_i \in \{0,1\}^{W \times H}$ for the binary support mask for part i at global position g_i. If parts have been ordered from back ($i = N$) to front ($i = 1$), the final rendered image is generated by initializing the back layer $C_N = M_N$ and iterating to the front:

$$I = C_1 \quad \text{where} \quad C_i = (1 - M_i) C_{i+1} + M_i R_i \quad (6)$$

The compositing process is visualized in Fig. 4(d)-(f).

3.3. Low resolution rendering

An ideal synthesis engine needs to generate a training set with two computationally demanding properties; (1) it needs to be *photorealistic* enough so that it matches well

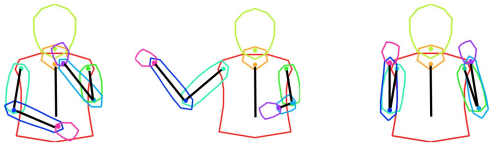

Figure 5: Upper body shape model. The 2D shape of each body part is a polygon parameterized by scale, orientation, and foreshortening ratio, which are mostly represented by pose skeletons (black).

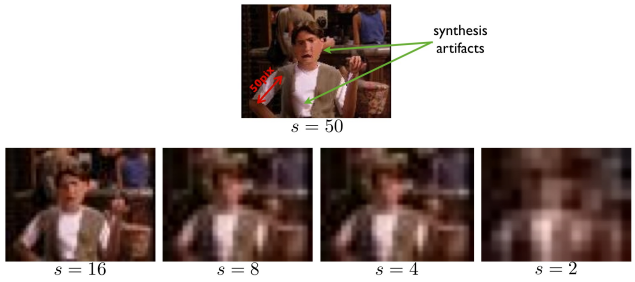

Figure 6: Low resolution color space. Typically, synthesized images are presented with significant artifacts (**top**). In order to mitigate the effect, we render images in low resolution space. Even with quite low resolution ($s = 4$), humans can reasonably estimate upper body poses. The resolution s is represented by the length of full upper arm in pixels.

with the real test images, and (2) it needs to be *comprehensive* enough to cover all expected test frames. In order to enhance photorealism, one can add more parameters to the synthesis engine, e.g. shearing parameters for out-of-plane rotation or parameters for face expression and detailed clothing models. However, since each of those introduces another axis in the joint parameter space, the number of images one need to synthesize grows exponentially.

One way to address both problems is to use low resolution training images. As shown in Fig. 6, the artifacts from our simple warping methods and heuristics for resolving occlusion is significantly reduced in low resolution space. Interestingly, humans still reasonably perform pose estimation with a resolution as low as $s = 4$, where s represents the length of the full upper arm. We quantitatively show in Sec. 5 that our approach exhibits similar behaviour.

Pose discretization: Furthermore, one needs to synthesize only a small set of images that are distinct in appearance in the low resolution space. Recall that we synthesize poses by enumerating over our pose space in (1). When reducing the image resolution by a factor of R, we also reduce the number of enumerable poses by a factor of R^{2N}, where N is the number of parts (assuming joints can only lie at discrete pixel locations). We further analyze performance as a function of image resolution and the amount of training poses in Sec. 5.

Efficiency: As a by-product of using low-resolution training images, one can significantly speed up the synthesis process by directly rendering the images in the low-resolution space. This is achieved in Sec. 3.2 by projecting the 2.1D layered model and labeled/queried poses to smaller scales. In practice, synthesizing a single image of $s = 16$ from Fig.6 takes 0.04s, while $s = 50$ takes 0.28s in a 3.0GHz single-core desktop.

Synthesizing blur: Our layered synthesis engine produces crisp edges across layers, while actual low-resolution images are quite blurred (Fig. 6). We mimic this blur during our synthesis by rendering at $b = 2$ times the target resolution, and then subsampling the rendered image with antialiasing. Such a procedure actually improves performance in two ways. First, generated image features appear more realistic due to blurred edges. Second, we can now represent a larger family of poses, specifically b^{2N} more poses. Perhaps surprisingly, we show that one can still resolve such "sub-pixel" pose configurations in a low-resolution image space.

4. Inference

Intuitively, given highly customized training data for a particular video, training an accurate recognition machine may be greatly simplified. We verify this hypothesis by performing upper body pose estimation using very simple image features and learning/inference algorithms. As image features, we use (low-resolutional) raw pixel values of the entire frame in perceptually uniform color spaces such as LUV or LAB. As a classifier, we use a nearest-neighbor regressor. That is, for each test frame we independently find the training image with the least L_2 distance in given feature space, and report its pose (after converting to skeleton format).

$$(I^*, \mathcal{P}^*) = \arg\min_{(I,P)} \|\Phi(I) - \Phi(I_{test})\|_2 \qquad (7)$$

It is widely accepted that the most crucial property of robust image features is their invariance to affine deformation, luminance, albedo, etc. As a result, modern image features are based on normalized edge-orientations ([7], [20]) or gabor-like filter responses ([28], [16]). However, in the scenario of tracking where there exist large consistency in appearance, *less* invariant features are likely to perform competitively. In Sec. 5, we compare pixel-value features with edge-based features to demonstrate this idea.

5. Experimental results

Dataset: We use the *Friends* dataset [29] to investigate the effect of key parameters (resolution, features, size of

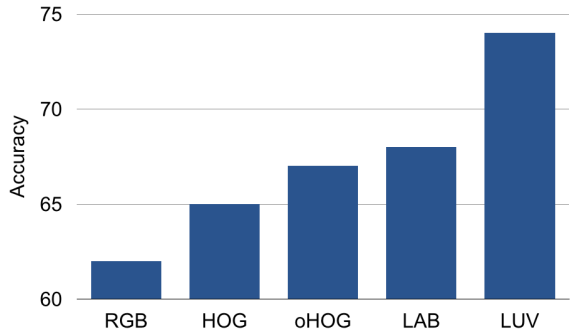

Figure 7: Image features. We show how the choice of image feature affects the pose estimation accuracy. Overall, with the exception of RGB, *less* invariant features performs better than standard image features. Particularly, raw pixel values in perceptually uniform color space such as LUV and LAB significantly outperform standard edge-based features (HOG). Color-augmented HOG (oHOG, [33]) with no contrast normalization performs better than HOG.

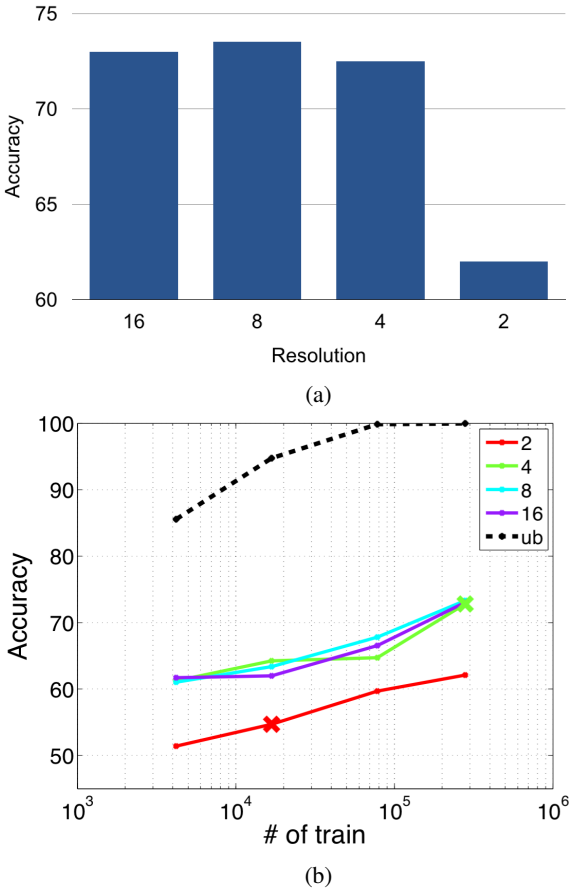

Figure 8: Low resolution color features and training data. In **(a)**, we show the accuracy of pose estimation with respect to the resolution of features. The number of synthesized training images is fixed (∼280k). The x-values are scales represented by the length of full upper arm (See Fig. 6). In **(b)**, we plot performance as a function of the number of training images (i.e sampling rate in pose space) for each resolution of LUV features. "ub" denotes an upper bound obtained by reporting the training pose closest to the ground truth test pose, measured in high-resolution image coordinates. This plot reveals that high accuracy (85%) can be theoretically obtained with a small number of rendered training images (∼4k). The "**x**" denotes the number of unique quantized poses that are resolvable at a fixed resolution (only shown for $s = 2$ and $s = 4$). It may appear strange that one can continue to improve accuracy for $s = 2$ by adding additional poses. This additional performance comes from rendering "subpixel" poses, as discussed in Sec. 3.3.

training data) and to compare our approach with the state-of-the-art. We focus on 13 test video clips (out of a total of 18 test clips) that contain a frontal view of a human. Frontal views simplify our synthesis process as we can assume a fixed depth ordering of body parts (both arms are in front of the torso), but we believe that generalizing to other views would be straightforward. Moreoever, these 13 clips are grouped and concatenated to form 5 longer videos, each of which contains a single character and background scene. (5 *take*s are split into 13 clips in the dataset.) We build a custom training set for each of the 5 takes by synthesizing purturbations of a single initial annotated frame (for each take). The length of the invidual clips range from 50 to 120 frames. Background scenes are mostly stable, but there exists mild motion due to movement of camera and/or objects.

Evaluation: In all diagnostic experiments, we evaluate the percentage of correctly predicted joints with 25-pixel threshold in a normalized scale. This radius roughly corresponds to the width of the fist of given character. We consider 4 joints; two elbows and wrists. When comparing with other approaches, we present the result with full range of thresholds as in [39] and [29].

Feature invariance: We first compare features with various degree of invariance (Fig.7). **HOG** is designed to be invariant to both spatial deformations (by pooling gradient statistics over large spatial and orientation neighborhoods) and illumination (through contrast normalization) [7]. We experimented with a less-invariant version by separately binning gradients computed across the 3 color channels (inspired by OpponentSift in [33]) and removing contrast normalization (**oHOG**). This modification yields a 3% improvement in accuracy. We explored different sizes of spatial/orientation bin, and report the best one.

We also evaluate the simplest and the least invariant type of features, *pixel-values*. Interestingly, these features work better than the best setting of HOG by a significant margin (with the exception of **RGB** color space). We found

that using perceptually uniform color space such as **LUV** or **LAB** is important, presumably because they were designed to make L_2 distance more meaningful.

Resolution: The next question we answer is about the working resolution of color features and its interplay between the number of training frames. Interestingly, as shown in Fig.8a, we achieve competitive accuracy using quite low resolution ($s = 4$), and observe a sharp drop for $s = 2$. This is consistent with visual inspection of the pixel data as well; it is quite hard for a human to see structure at such low resolutions (Fig.6).

In fact, the correlation between feature resolution and accuracy is more subtle, since the accuracy also depends on the number of rendered poses (or the pose space sampling rate mentioned in Sec.3.3). Intuitively, the number of visually distinguishable poses must decrease at low resolutions. This observation suggests that one may need to render only those poses with unique quantized configurations at a given resolution. Fig8b shows that "subpixel" pose configurations further improves accuracy. An upper-bound analysis reveals that a small number of poses ($\sim 4K$) can potentially achieve a quite high accuracy ($\sim 85\%$), but this may require complex image matching function (capable of deforming images while matching). Rather, our approach is to synthesize a set of deformations with consistent depth-layering.

Benchmark comparisons: Lastly, we compare our approaches with the state-of-the-art on the *Friends* dataset. [29] uses an ensemble of tree models, each of them rooted on one of the 6 parts and temporally linked only through roots, to approximate underlying loopy spatiotemporal model. [39] uses optical flows and learned 2D articulated shape models as means to exploit pixel informations of adjacent frames and to propagate part assignments temporally. Both methods use optical flows and designated off-the-shelf hand detector based on assumptions on skin colors. We compare both methods on the same of 13 frontal test clips on which we evaluate our model.

Our nearest-neighbor regressor predicts elbow locations significantly better than other two methods; 93.7% versus 73.2% and 74.2% at 25-pixel threshold. For wrist, our methods is less accurate than [29] and [39]; 54.8% versus 69.9% and 59.8% at 25-pixel threshold (Fig.9). Unlike other two methods, we independently estimate poses in each frame without using temporal models or motion features. Plus, there is no extra effort for detecting hands.

Error analysis: One of the benefits of simple features and learning algorithms are that visualizing and understanding the predictor is straightforward (Fig.10). For instance, a common mistake is that hands are often confused by background objects with similar color (failure of the model to correctly *explain-away* the background). In addition, our approach of using customized synthetic frames facilitate further error analysis. For instance, one can synthesize images with ground-truth test poses to compare the accuracy of our pose-synthesis and pixel-synthesis engines.

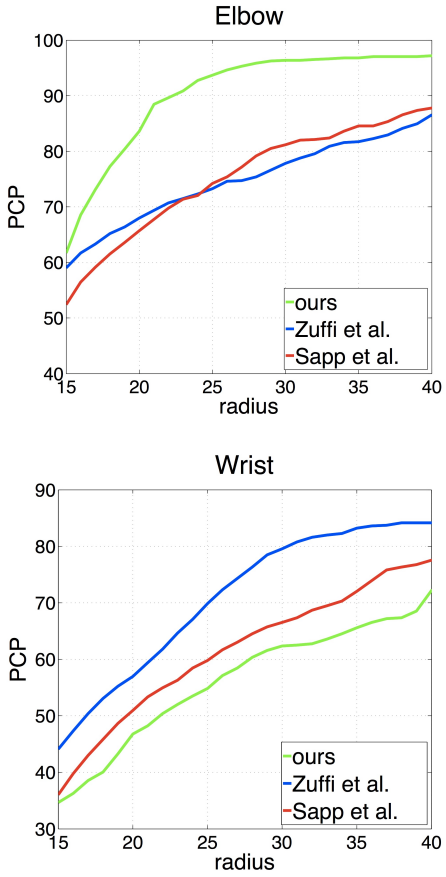

Figure 9: We compare our best result (LUV features, $s = 5.7$, $\sim$ 280k training frames) to two other state-of-the-art methods that report results on the *Friends* dataset, [29] and [39]. We evaluate results on 13 out of the 18 test clips with frontal-facing subjects. We outperform both methods by a large margin (**25%** at 25-pixel threshold) in the task of predicting elbow locations (**left**). We perform competitively in predicting wrist locations (**right**)

6. Conclusion

In this paper, we described an approach of using synthetic training dataset to train models highly customized to the particular video. We show that, with simple image-based rendering algorithms, one can generate reasonably photorealistic training data that captures important modes of variation (human poses) of given video, while maintaining its invariants. We showed that this custom training data greatly simplify learning and inference. We demonstrated our approach on the challenging task of estimating upper body pose of humans in videos.

Figure 10: Pose estimation. We independently estimate upper body pose in each test frame by finding the nearest training frame in low-resolutional color spaces. Although simple, our method is robust against (self) occlusion **(a)** and challenging interaction of parts **(b)**. A common mistake is due to confusing color in the background **(c, d)**, which are also readily confused by human in such low-resolution space.

References

[1] M. Andriluka, S. Roth, and B. Schiele. Pictorial structures revisited: People detection and articulated pose estimation. In *Proc. CVPR*, volume 1, page 4, 2009. 4

[2] C. Barnes, E. Shechtman, A. Finkelstein, and D. B. Goldman. Patchmatch: A randomized correspondence algorithm for structural image editing. In *ACM SIGGRAPH 2009 Papers*, SIGGRAPH '09, pages 24:1–24:11, New York, NY, USA, 2009. ACM. 4

[3] V. Belagiannis, X. Wang, B. Schiele, P. Fua, S. Ilic, and N. Navab. Multiple human pose estimation with temporally consistent 3D pictorial structures. In *ChaLearn Looking at People (ECCV workshop)*, September 2014. 2

[4] M. Bertalmio, G. Sapiro, V. Caselles, and C. Ballester. Image inpainting. In *Proceedings of the 27th annual conference on Computer graphics and interactive techniques*, pages 417–424. ACM Press/Addison-Wesley Publishing Co., 2000. 4

[5] A. Buchanan and A. Fitzgibbon. Interactive feature tracking using kd trees and dynamic programming. In *Computer Vision and Pattern Recognition, 2006 IEEE Computer Society Conference on*, volume 1, pages 626–633. IEEE, 2006. 2

[6] A. Criminisi, P. Pérez, and K. Toyama. Region filling and object removal by exemplar-based image inpainting. *Image Processing, IEEE Transactions on*, 13(9):1200–1212, 2004. 4

[7] N. Dalal and B. Triggs. Histograms of oriented gradients for human detection. In *CVPR*, pages I: 886–893, 2005. 5, 6

[8] J. Deutscher, A. Blake, and I. Reid. Articulated body motion capture by annealed particle filtering. In *Computer Vision and Pattern Recognition, 2000. Proceedings. IEEE Conference on*, volume 2, pages 126–133. IEEE, 2000. 2

[9] M. Eichner, M. Marin-Jimenez, A. Zisserman, and V. Ferrari. 2D articulated human pose estimation and retrieval in (almost) unconstrained still images. *International Journal of Computer Vision*, 99(2):190–214, 2012. 4

[10] M. J. Farah. The neural basis of mental imagery. *Trends in neurosciences*, 12(10):395–399, 1989. 1

[11] P. F. Felzenszwalb, R. B. Girshick, D. McAllester, and D. Ramanan. Object detection with discriminatively trained part-based models. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 32(9):1627–1645, 2010. 1

[12] D. M. Gavrila. The visual analysis of human movement: A survey. *Computer vision and image understanding*, 73(1):82–98, 1999. 2

[13] C. Ionescu, D. Papava, V. Olaru, and C. Sminchisescu. Human3.6m: Large scale datasets and predictive methods for 3d human sensing in natural environments. Technical report, Institute of Mathematics of the Romanian Academy and University of Bonn, September 2012. 3

[14] A. D. Jepson, D. J. Fleet, and T. F. El-Maraghi. Robust online appearance models for visual tracking. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 25(10):1296–1311, 2003. 2

[15] S. X. Ju, M. J. Black, and Y. Yacoob. Cardboard people: A parameterized model of articulated image motion. In *Automatic Face and Gesture Recognition, 1996., Proceedings of the Second International Conference on*, pages 38–44. IEEE, 1996. 2, 3, 4

[16] A. Krizhevsky, I. Sutskever, and G. E. Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012. 5

[17] Y. LeCun, L. Bottou, Y. Bengio, and P. Haffner. Gradient-based learning applied to document recognition. *Proceedings of the IEEE*, 86(11):2278–2324, 1998. 1, 2

[18] Y. LeCun, F. J. Huang, and L. Bottou. Learning methods for generic object recognition with invariance to pose and lighting. In *CVPR*, volume 2, pages II–97. IEEE, 2004. 3

[19] T. G. Lohman, A. F. Roche, and R. Martorell. Anthropometric standardization reference manual. 1988. 3

[20] D. Lowe. Distinctive image features from scale-invariant keypoints. *International journal of computer vision*, 60(2):91–110, 2004. 5

[21] J. M. McCarthy. *Introduction to theoretical kinematics*. MIT press, 1990. 4

[22] T. B. Moeslund, A. Hilton, and V. Krüger. A survey of advances in vision-based human motion capture and analysis. *Computer vision and image understanding*, 104(2):90–126, 2006. 2

[23] R. P. Paul. *Robot manipulators: mathematics, programming, and control: the computer control of robot manipulators*. Richard Paul, 1981. 4

[24] L. Pishchulin, A. Jain, M. Andriluka, T. Thormahlen, and B. Schiele. Articulated people detection and pose estimation: Reshaping the future. In *CVPR*, pages 3178–3185. IEEE, 2012. 3

[25] Z. W. Pylyshyn. What the mind's eye tells the mind's brain: A critique of mental imagery. *Psychological bulletin*, 80(1):1, 1973. 1

[26] D. Ramanan, D. Forsyth, and A. Zisserman. Tracking people by learning their appearance. *IEEE PAMI*, pages 65–81, 2007. 2, 3

[27] A. S. Razavian, H. Azizpour, J. Sullivan, and S. Carlsson. CNN features off-the-shelf: an astounding baseline for recognition. *arXiv preprint arXiv:1403.6382*, 2014. 2

[28] X. Ren and D. Ramanan. Histograms of sparse codes for object detection. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 3246–3253. IEEE, 2013. 5

[29] B. Sapp, D. Weiss, and B. Taskar. Parsing human motion with stretchable models. In *CVPR*, 2011. 2, 5, 6, 7

[30] G. Shakhnarovich, P. Viola, and T. Darrell. Fast pose estimation with parameter-sensitive hashing. In *CVPR*, pages 750–757. IEEE, 2003. 3

[31] H. Sidenbladh, M. J. Black, and L. Sigal. Implicit probabilistic models of human motion for synthesis and tracking. In *Computer VisionECCV 2002*, pages 784–800. Springer, 2002. 2

[32] A. Smeulders, D. Chu, R. Cucchiara, S. Calderara, A. Dehghan, and M. Shah. Visual tracking: An experimental survey. 2013. 2

[33] K. van de Sande, T. Gevers, and C. Snoek. Evaluating color descriptors for object and scene recognition. *IEEE Trans. Pattern Anal. Mach. Intell.*, 32(9):1582–1596, Sept. 2010. 6

[34] P. Viola and M. Jones. Rapid object detection using a boosted cascade of simple features. In *Computer Vision and Pattern Recognition, 2001. CVPR 2001. Proceedings of the 2001 IEEE Computer Society Conference on*, volume 1, pages I–511. IEEE, 2001. 1

[35] J. Y. Wang and E. H. Adelson. Representing moving images with layers. *Image Processing, IEEE Transactions on*, 3(5):625–638, 1994. 3, 4

[36] Y. Wu, J. Lim, and M.-H. Yang. Online object tracking: A benchmark. In *Computer Vision and Pattern Recognition (CVPR), 2013 IEEE Conference on*, pages 2411–2418. IEEE, 2013. 2

[37] Y. Yang, S. Hallman, D. Ramanan, and C. C. Fowlkes. Layered object models for image segmentation. *Pattern Analysis and Machine Intelligence, IEEE Transactions on*, 34(9):1731–1743, 2012. 3, 4

[38] A. Yilmaz, O. Javed, and M. Shah. Object tracking: A survey. *Acm computing surveys (CSUR)*, 38(4):13, 2006. 2

[39] S. Zuffi, J. Romero, C. Schmid, and M. J. Black. Estimating human pose with flowing puppets. In *IEEE International Conference on Computer Vision (ICCV)*, pages 3312–3319, 2013. 2, 3, 6, 7