JUMPS: Joints Upsampling Method for Pose Sequences

Lucas Mourot, Franois Le Clerc, Cdric Thbault and Pierre Hellier

{Lucas.Mourot, Francois.LeClerc, Cedric.Thebault, Pierre.Hellier}@InterDigital.com

InterDigital, Rennes, France

Fig. 1. Illustration of our joints upsampling method (JUMPS) applied to the task of Human Pose Estimation (HPE): the yellow 12-joint skeletons (top image) depicts poses estimated by *AlphaPose* [25] at different timestamps of a long jump. From this set of poses, we compute the green 28-joint skeletons (bottom image). Our approach upsamples the number of joints in order to enrich the representation, in a spatially and temporally consistent way. In both images, the background has been stitched from a video with a smaller field of view and five images of the jumper at different frames have been overlaid as foreground.

Abstract—Human Pose Estimation is a low-level task useful for surveillance, human action recognition, and scene understanding at large. It also offers promising perspectives for the animation of synthetic characters. For all these applications, and especially the latter, estimating the positions of many joints is desirable for improved performance and realism. To this purpose, we propose a novel method called JUMPS for increasing the number of joints in 2D pose estimates and recovering occluded or missing joints. We believe this is the first attempt to address the issue. We build on a deep generative model that combines a GAN and an encoder. The GAN learns the distribution of high-resolution human pose sequences, the encoder maps the input low-resolution sequences to its latent space. Inpainting is obtained by computing the latent representation whose decoding by the GAN generator optimally matches the joints locations at the input. Post-processing a 2D pose sequence using our method provides a richer representation of the character motion. We show experimentally that the localization accuracy of the additional joints is on average on par with the original pose estimates.

I. INTRODUCTION

Human Pose Estimation (HPE) refers to the problem of predicting joints position (either in $2D$ or $3D$) of a person in an image or video. It has been a topic of active research for several decades, and all state-of-the-art solutions rely on deep learning [1]–[5]. Even then, the best approaches extract skeletons with a limited number of joints, usually from 12 to 16, which is too rough for the movie industry or video games applications. This issue concerns both the $2D$ and $3D$ cases since $3D$ joint extraction almost always relies on $2D$ pose estimation. Moreover, these approaches still fail in the presence of strong foreshortening, left-right ambiguities, (self)-occlusions, or on previously unseen complex poses.

In this paper we improve on state-of-the-art HPE solutions by upsampling human joints and inpainting occluded ones, thereby paving the way for downstream applications that
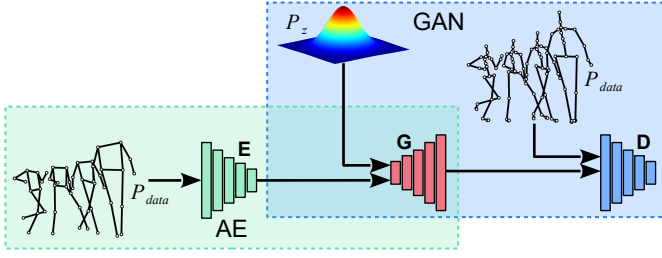
1

Fig. 2. Coarse representation of our deep generative network; the upper right part (blue) is the basis of our model: a Generative Adversarial Network (GAN) with generator G and discriminator D; the bottom left part (green) depicts how the encoder E and the generator G together yield an autoencoder (AE) scheme; P_z and P_{data} denote the prior and data distribution respectively.
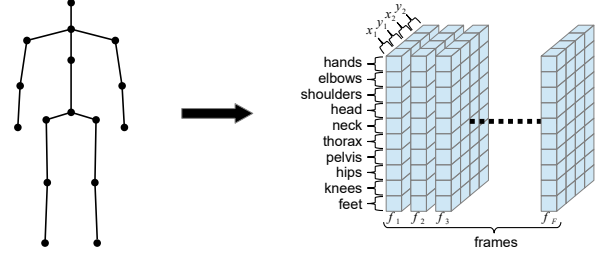

Fig. 3. Illustration of how our subnetworks internally represent a pose sequence whose topology is depicted on the left part. The right part shows how joints coordinates are arranged, with body parts ordered following the human skeleton from hands to feet.

require higher skeleton resolution. Starting with a temporal sequence of partially occluded poses, we recover missing joint locations and improve the resolution of the skeleton animation by estimating the positions of additional joints. To the best of our knowledge, no work has been previously proposed to recover missing joints or increase joints resolution of animated skeletons. We believe that enriching the representation helps in many cases, especially for extremities such as feet/toes and hands. A better extraction of the former provides a better visualisation and understanding of the motion. For instance, extracting the toe in addition to the ankle provides a better sense of feet contacts.

To this purpose, we draw inspiration from past research on human pose, motion modeling and image inpainting based on deep generative models: we leverage a deep generative network that provides an effective prior on spatio-temporal biomechanics. Our model builds on a Generative Adversarial Network (GAN), which we complement by a front-end encoder to learn a mapping from the human pose space to the GAN latent space. The encoder helps selecting better samples in latent space and stabilizes the training of the GAN.

In summary, our paper proposes the following contributions:

- a novel method based on deep generative modeling for inpainting $2D$ pose sequences and enriching the joints representation. The method relies on temporal sequence analysis since motion is key to recover missing joints.
- a hybrid GAN/autoencoder architecture; we show that the autoencoder is crucial for a better convergence and accuracy.
- We show that optimization in latent space is greatly improved by adding a Procrustes alignment at each iteration.
- We provide qualitative and quantitative assessments of the effectiveness of our method on the *MPI-INF-3DHP* human pose dataset.

II. RELATED WORK

Human pose modeling: Autoencoder architectures have been leveraged to learn models of human pose in the context of human pose estimation [6] and character animation synthesis and editing [7]. In [6] the latent space of the autoencoder encodes a structural prior on human pose. Mapping input

images to this latent space provides guarantees as to the validity of the generated poses. Independently, [7] applies the same concept to temporal chunks of human poses, thereby capturing in the latent space a model of human motion. This model is mapped to semantic parameters for intuitive control by creative artists.

Deep generative inpainting: Deep generative models have demonstrated impressive performance on image inpainting [8]–[11]. For this task the need to faithfully reproduce the visible surroundings of missing image regions adds an additional constraint to the generative synthesis process and requires a mapping from the data space to the latent space of the generative model. Yeh *et al.* [9] compute the latent code from the corrupted image in the inference stage by backpropagating the gradients of a GAN generator network. In their seminal paper [12], Pathak *et al.* take a different approach that builds on the combination of a GAN and an autoencoder. The encoder provides the mapping from the input images to the latent space while the decoder acts as the generator network. [10] enriches this architecture with two discriminators to separately capture the small-scale and large-scale image texture, and [11] further adds a self-attention module to better take advantage of distant image patches to fill the missing regions. [8] replaces the autoencoder with a Variational Autoencoder (VAE) [13] and incorporates an image classifier to specialize the generative process to sub-categories. Our work leverages a deep network architecture combining a GAN and an autoencoder in the spirit of the latter approaches and adapts it to human pose data. We optimize our upsampling and inpainting process for temporal chunks of data and develop a generative model that captures both the static and dynamic aspects of human biomechanics.

III. METHOD

A. Overview

We propose a method to upsample and inpaint an animated skeleton to infer the locations of missing or unseen joints and provide a higher-resolution representation of the body pose. To this purpose we leverage a deep generative network that we train with moving skeleton sequences, rather than static poses, in order to better disambiguate the estimation of missing joint locations.

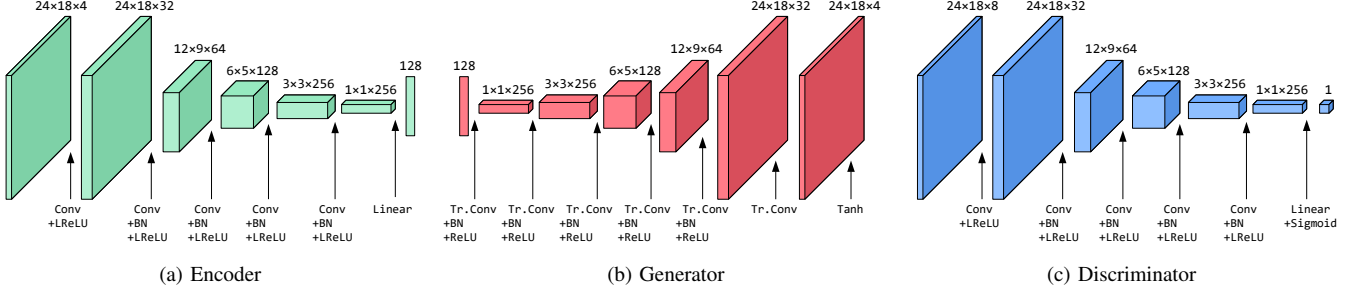As illustrated in Fig. 2, our model consists of a GAN coupled

Fig. 4. Detailed description of our network architecture. Notation: Conv, Tr.Conv, BN, ReLU and LReLU respectively stand for convolution, transposed convolution, Batch Normalization, Rectified Linear Unit and Leaky ReLU .

with an encoder, both forming an autoencoder where the generator plays the role of the decoder. It benefits from the generative power of GANs and mitigates instability during training by introducing supervision from the encoder.

B. Detailed Architecture

Our network conforms to the architecture of Deep Convolutional GANs (DCGANs) [19], using fractionally-strided transposed convolutions in the generator and strided convolutions in the discriminator (see Fig. 4). DCGANs also use Rectified Linear Units (ReLU) as activation functions in the generator and Leaky ReLU (LReLU) in the discriminator. Moreover, batch Normalization (BN) is also applied after almost each convolutional layer. Except for the output size of its final layer, the encoder has the same architecture as the discriminator.

C. Training

In this section we describe the representation used for joint position data, the loss functions and the optimization procedure.

Data Representation: A $2D$ pose sequence is usually represented as a 3-dimensional tensor containing the $2D$ coordinates of each joint at each frame. To obtain meaningful and efficient convolutions, we rearrange the joints as shown in Fig. 3. In this representation, each entry holds the $2D$ coordinates for two joints (*i.e.*, four channels). Symmetric joints (*e.g.*, feet, knees, *etc.*) are paired to form an entry, while joints in the axial skeleton (*e.g.*, pelvis, thorax, *etc.*) are duplicated[1] in order to obtain consistent four-channel entries. This reformatting of data to a rectangular $2D$ grid allows to use regular $2D$ convolutions in our deep network.

Notation: In the following, we note E, G and D the encoder, the generator and the discriminator networks, respectively. In addition P_z and P_{data} denote the latent and the data distributions respectively. Finally, $P_{\tilde{x}}$ stands for the distribution of uniformly sampled points along straight lines between pairs of points sampled from the data distribution P_{data} and the generator distribution, *i.e.* P_z mapped through G, as defined in [17].

---

[1]Both discriminator and encoder duplicate axial input joints while the generator produces duplicated axial joints in a first step and then outputs the average of the two versions.

Adversarial Loss: Traditionally, a GAN consists of a generator and a discriminator. The former is trained to produce realistic samples while the latter aims at distinguishing those from real samples, both competing against each other. The ability to generate realistic samples can be expressed more formally as the similarity between two probability distributions that are the data distribution and the distribution of samples produced by the generator. The original formulation of GANs [15] measures the similarity with the JensenShannon divergence. However, this divergence fails to provide meaningful values when the overlap between the two distributions is not significant which often makes GANs quickly diverge during training. Arjovsky et al. [16] introduced Wasserstein GANs (WGANs), showing that, under the hypothesis that the discriminator is 1-Lipschitz, the JensenShannon divergence can be replaced by the Wasserstein distance that have better properties for convergence. Then, Gulrajani et al. [17] propose a gradient penalty term in the WGAN loss function to enforce the 1-Lipschitz hypothesis on the discriminator.

Therefore, we opt for the gradient-penalized WGAN and have the following loss functions for the generator and the discriminator, respectively:

$$\mathcal{L}_G = \mathop{\mathbb{E}}_{z \sim P_z} \Big[- D\big(G(z)\big) \Big] \tag{1}$$

$$\mathcal{L}_D = \mathop{\mathbb{E}}_{z \sim P_z} \Big[D\big(G(z)\big) \Big] - \mathop{\mathbb{E}}_{x \sim P_{data}} \Big[D(x) \Big]$$
$$+ \lambda_{gp} \mathop{\mathbb{E}}_{\tilde{x} \sim P_{\tilde{x}}} \Big[\Big(\big\| \nabla_{\tilde{x}} D(\tilde{x}) \big\|_2 - 1 \Big)^2 \Big] \tag{2}$$

where λ_{gp} is the gradient penalty coefficient.

Reconstruction Losses: Like autoencoders, our model is encouraged to reconstruct inputs that are encoded and then decoded through a reconstruction loss minimizing differences between inputs and outputs. We also incite our model to be consistent when generating and then encoding from latent codes sampled from the prior distribution with a backward reconstruction loss, as in cycle-consistent VAEs [14]. Such backward reconstruction loss facilitates the convergence but more importantly enforces the distribution of the encoder outputs to match the prior distribution P_z imposed on our GAN. As a result, the total loss in the autoencoding scheme

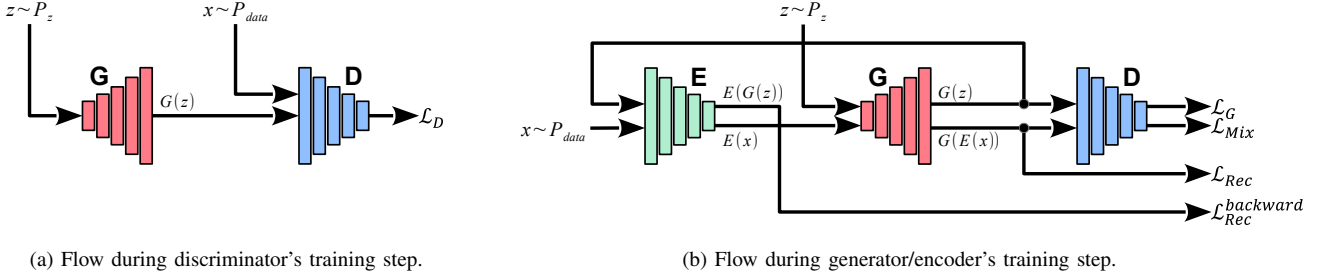(a) Flow during discriminator's training step.   (b) Flow during generator/encoder's training step.

Fig. 5. Flows through our combined AE-GAN model during training.

is

$$\mathcal{L}_{AE} = \mathcal{L}_{Rec} + \mathcal{L}_{Rec}^{backward} \qquad (3)$$

Computational flows conducting to this loss are illustrated in Fig. 5b.

$\mathcal{L}_{Rec}$ is itself made up of two terms penalizing respectively the joints position and velocity errors of the reconstructed sample $\hat{x} = G(E(x))$ with respect to the ground truth x. More formally, we use the *mean per joint position error* (MPJPE) [22] to quantify joint position errors:

$$MPJPE(x, \hat{x}) = \frac{1}{J}\frac{1}{F}\sum_{f,j}\left\|(x - \hat{x})_{j,f}\right\|_2 \qquad (4)$$

where j and f denote the joint and frame considered; J and F are the numbers of joints and frames respectively.

In analogy to the MPJPE, we define the *mean per joint velocity error* (MPJVE) as

$$MPJVE(x, \hat{x}) = \frac{1}{J}\frac{1}{F}\sum_{f,j}\left\|\big(v(x) - v(\hat{x})\big)_{j,f}\right\|_2 \qquad (5)$$

where $v(\cdot)$ computes the velocity of each joint at each frame as the position difference between the current and previous frame. This secondary term penalizing velocity errors acts as a powerful regularizer that accelerates the convergence in early iterations and also reduces temporal jitter in the joint locations of the generated pose sequences. Hence, $\mathcal{L}_{Rec}$ is the weighted sum of Eq. (4) and Eq. (5):

$$\begin{aligned}\mathcal{L}_{Rec} = \quad &\lambda_p \mathop{\mathbb{E}}_{x \sim P_{data}}\Big[MPJPE(x, \hat{x})\Big] \\ + \; &\lambda_s \mathop{\mathbb{E}}_{x \sim P_{data}}\Big[MPJVE(x, \hat{x})\Big]\end{aligned} \qquad (6)$$

where λ_p and λ_s are the weights. The second component $\mathcal{L}_{Rec}^{backward}$ of our autoencoder's objective focuses on the reconstruction of the latent code z sampled from the prior distribution P_z. It minimizes the Mean Squared Error (MSE) between z and its reconstructed version $\hat{z} = E(G(z))$:

$$\mathcal{L}_{Rec}^{backward} = \lambda_z \mathop{\mathbb{E}}_{z \sim P_z}\Big[MSE(z, \hat{z})\Big] \qquad (7)$$

Mixed Loss: We further encourage the generation of realistic sequences by adding a loss term to penalize unrealistic reconstructed pose sequences. Here we make use of the discriminator to tell both the generator and the encoder whether the reconstructed pose sequence $\hat{x} = G(E(x))$ is realistic or not. We use the same formulation as for the generator adversarial loss (see in Eq. 1) but applied to $\hat{x}$ instead of $G(z)$:

$$\mathcal{L}_{Mix} = \lambda_m \mathop{\mathbb{E}}_{x \sim P_{data}}\Big[-D(\hat{x})\Big] \qquad (8)$$

Optimization: In summary, the encoder, the generator and the discriminator are optimized w.r.t. the loss functions $\mathcal{L}_{AE} + \mathcal{L}_{Mix}$, $\mathcal{L}_G + \mathcal{L}_{AE} + \mathcal{L}_{Mix}$ and $\mathcal{L}_D$, respectively. Similarly to a GAN, during the training we optimize at each iteration the discriminator in a first step and then the generator and the encoder. Fig. 5 illustrates the computational flows through the network during both training steps.

Spatio-Temporal Variance Regularization: GANs are known to produce sharp samples, but for the considered task this can lead to perceptually disturbing temporal jitter in the output pose sequences. To optimize the tradeoff between sharpness and temporal consistency, we feed the discriminator with stacked joint positions and velocities (computed for each joint at each frame as the position difference between the current and previous frame). The velocities favour the rejection of generated samples that are either temporally too smooth or too sharp. This idea is conceptually inspired from [18], where the variation of generated samples is increased by concatenating minibatch standard deviations at some point of the discriminator.

D. Inference

We leverage the human motion model learnt by the generator to recover missing joints in an input pose sequence x. Given x, we optimize z using gradient backpropagation across the generator network of a contextual loss that minimizes the discrepancy between $G(z)$ and x on available joints. To this contextual loss we add a prior term that maximizes the discriminator score on the generated pose sequence. This process is closely related to the semantic image inpainting approach in [9]; however we take advantage of our encoder to compute a starting latent code $z = E(x)$ as in [8]. This approach also applies to upsampling, considering that the added joints are missing in the input.

Formally, we first solve

$$z^* = \arg\min_z \mathcal{L}_{Inp}(x, z) \qquad (9)$$

TABLE I
RESULTS OF THE JOINT UPSAMPLING EXPERIMENTS. WE UPSAMPLE BACK TO 28 JOINTS A GROUND TRUTH 2D POSE SEQUENCE PURPOSEDFULLY DOWNSAMPLED TO 12 JOINTS. REMOVING PROCRUSTES ALIGNMENT (W/O P.A.) AND THE ENCODER (W/O ENC.) SUBSTANTIALLY DEGRADES PERFORMANCE. SEE THE TEXT FOR THE DEFINITION OF THE PERFORMANCE METRICS.

method	PCKh@0.1	PCKh@0.5	PCKh@1.0	AUC
JUMPS w/o P.A.	0.0368	0.4384	0.6814	0.3912
JUMPS w/o ENC.	0.1701	0.8259	0.9678	0.7005
JUMPS w/o overlap	0.5821	0.9648	0.9962	0.8727
JUMPS	**0.6096**	**0.9674**	**0.9965**	**0.8803**

TABLE II
RESULTS OF THE ALPHA POSE POST-PROCESSING EXPERIMENTS. WE PERFORM INPAINTING AND UPSAMPLING TO 28 JOINTS OF 2D POSE ESTIMATES OBTAINED BY RUNNING ALPHA POSE ON VIDEO SEQUENCES. THE ABLATION STUDIES CONFIRM THE CONCLUSIONS DRAWN FROM THE JOINT UPSAMPLING EXPERIMENTS (SEE TABLE I).

method	PCKh@0.1	PCKh@0.5	PCKh@1.0	AUC
AlphaPose	**0.0941**	0.7659	0.9157	0.6310
JUMPS w/o P.A.	0.0207	0.3423	0.6304	0.3249
JUMPS w/o ENC.	0.0537	0.6801	0.9059	0.5692
JUMPS w/o overlap	0.0831	0.7701	**0.9277**	0.6326
JUMPS	0.0842	**0.7723**	0.9276	**0.6341**

by gradient descent where $\mathcal{L}_{Inp}$ is our inpainting objective function composed of a contextual loss and a prior loss. Then, we generate $x^* = G(z^*)$ that best reconstructs x w.r.t $\mathcal{L}_{Inp}$.

Inpainting Loss Function: Our contextual loss minimizes the weighted sum of MPJPE and MPJVE between the input pose sequence x and the generated pose sequence $\hat{x} = G(z)$. Additionally, the prior loss maximizes the discriminator score $D(\hat{x})$ on the generated pose sequence:

$$\mathcal{L}_{Inp} = \underbrace{\gamma_p \ MPJPE(x,\hat{x}) + \gamma_s \ MPJVE(x,\hat{x})}_{\text{contextual loss}}$$
$$\underbrace{- \gamma_d \ D(\hat{x})}_{\text{prior loss}} \quad (10)$$

Post-Processing: At each gradient descent step, we generate the pose sequence $\hat{x} = G(z)$. At this point, we additionally use the fact that we are given a pose sequence x to be inpainted by optimally translating, scaling and rotating $\hat{x}$ to match x. This process (known as Ordinary Procrustes Analysis) has a low overhead but makes the gradient descent convergence several times faster and improves inpainting results.

Pose Sequence Length: Our deep network requires pose sequences to have a constant number of frames F. Here we describe a simple mechanism to handle longer variable-length pose sequences. Given a pose sequence x longer than F frames, the idea is to independently inpaint fixed-length subsequences of x and then concatenate the results into a single inpainted pose sequence having the same length as x. Using this process there is no guarantee that two consecutive subsequences will be smoothly concatenated. To prevent such discontinuities in the generated sequences we use half overlapping subsequences. At each temporal sample where an overlap is present we select among the candidate inpainted frames the one closest to the input, in the sense of the minimal contextual loss term in $\mathcal{L}_{Inp}$.

IV. EXPERIMENTS

A. Datasets and Metrics

Training and test sets: We rely on *MPI-INF-3DHP* [21] for our experiments. This dataset contains image sequences in which 8 actors perform various activities with different sets of clothing. This dataset is well suited for our task of joints upsampling since it is one of the public databases having the highest skeleton resolution, i.e. skeletons with 28

joints. Since our method focuses on fixed length $2D$ pose sequences, we generated a set of around $835K$ $2D$ pose sequences of 24 frames (*i.e.*, F=24) each using projections of the original $3D$ pose data from randomized camera viewpoints. We also selected around $166K$ images annotated with $2D$ poses directly from *MPI-INF-3DHP* with no preprocessing for testing.

Evaluation Metrics: We report our experiments results with the Percentage of Correct Keypoints normalized with Head size (PCKh) [26] and the Area Under the Curve (AUC) [27] metrics. PCKh metric consider a joint as correct if its distance to the ground truth normalized by head size is less than a fixed threshold and the AUC aggregates PCKh over an entire range of thresholds. We use the common notation PCKh@α to refer to PCKh with threshold α and we compute the AUC over the range $[0, 1]$ of thresholds.

B. Implementation Details

Our deep network (see Fig. 4 for detailed architecture) has about 3 millions learnable parameters almost equally distributed over the encoder ($1'148'096$), the generator ($1'148'480$) and the discriminator ($1'115'393$). Our implementation is in Python and deeply relies on PyTorch library. Training and experiments have been executed on a NVIDIA Tesla P100 PCIe 16GB.

Training: We trained our model for 60 epochs (about 11 hours) with a minibatch size of 256 using the Adam algorithm [20] with optimization hyperparameters $\alpha = 0.0001$, $\beta_1 = 0.5$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$. We followed the suggestions for DCGANs from [19] to reduce (w.r.t. [20] suggestions) α and β_1. As in [19], we observed that $\beta_1 = 0.5$ helped to stabilize the training.

We set the Wasserstein gradient penalty weight λ_{gp} to 10 as proposed in [17], and our loss weights λ_p, λ_s, λ_z and λ_m to 200, 100, 2 and 1 respectively. We empirically found these values to work well.

Inference: We compute the latent code again using the Adam optimization algorithm with $\alpha = 1$, $\beta_1 = 0.8$, $\beta_2 = 0.999$ and $\epsilon = 10^{-8}$. The weights of the inpainting loss are set to $\gamma_p = 10$, $\gamma_s = 5$ and $\gamma_d = 15$. We stop the optimization after 200 iterations. These hyperparameter values has been chosen to make the optimization in a limited number of iterations and avoid matching noise or imperfections in inputs.
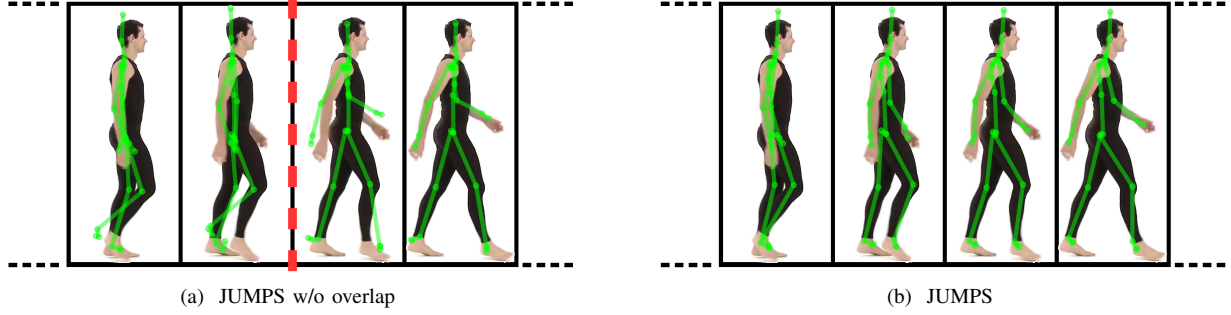
Fig. 6. Qualitative example of the influence of overlapping temporal chunks. This subset of four consecutive frames in a longer sequence is inpainted with no overlap (left) and half overlap (right). The frames are located at the end (first two) and the beginning (last two) of two consecutive chunks. Note the temporal discontinuity of joint locations (head top, forearms, feet) in the inpainted sequence at the chunk boundary with no overlap (dashed red line, left). The temporal consistency is much better using an overlap (right).

To improve inference results we perform several optimizations of the latent code in parallel for a single input, starting from different initializations. One of these starting points is computed as the output of the encoder fed by the input pose sequence, the others are randomly sampled from the prior distribution. We keep the one closest to the input, in the sense of the inpainting loss $\mathcal{L}_{Inp}$.

C. Joints Upsampling

Our first experiment focuses on the upsampling task. We downsample ground truth 28-joint pose sequences to 12 joints that are common to the *MPI-INF-3DHP* dataset and *AlphaPose* skeletons (see fig. 7 left), upsample them back to 28 joints using our method, and compare the result to the original sequence. Table I provides PCKh and AUC values for this experiment. Assuming a typical human head size, the positioning error is less than 2.25 *cm* for half of the upsampled joints (PCKh threshold = 0.1) and less than 11.25 *cm* for 95% of them (PCKh threshold = 0.5).

D. 2D Human Pose Estimation

Our second experiment deals with the concrete use case of inpainting and upsampling joints on a pose sequence obtained using 2D Human Pose Estimation. We rely on *AlphaPose*[2] to preprocess videos in our test set. *AlphaPose* provides 12-joint pose estimates that we post-process using our method to recover missing (*e.g.*, occluded) joints and upsample to 28 joints. Table II summarizes the results for this experiment. The positioning accuracy is roughly the same for the inpainted / upsampled joints and for the joints obtained by Human Pose Estimation. Thus, our method enriches the pose information without sacrificing accuracy. Fig. 7 illustrates how our method is able to correct the right wrist position mispredicted by *AlphaPose* based on the temporal consistency of the right forearm movement.

[2]Implementation based on [23]–[25] available at *https://github.com/MVIG-SJTU/AlphaPose*

E. Ablation Studies

1) Procrustes Analysis: line *JUMPS w/o P.A.* in tables I and II gives the joint positioning accuracy when the Procrustes Analysis post-processing of our method (see section III-D) is removed. Instead we map all pose sequences to the image frame using the same affine transform. Rigidly aligning the generated poses during the gradient descent optimization of the latent code is critical to the performance of our approach.

2) Encoder: as shown by the accuracy estimates in lines *JUMPS w/o ENC.* of the same tables, removing the encoder in front of the GAN in our architecture, during both training and inference stages, substantially degrades performance. The encoder regularizes the generative process and improves the initialization of the latent code at inference time, yielding poses that better match the available part of the input skeleton.

3) Overlapping subsequences: Processing input sequences with an overlap yields only a slight improvement of performance over no overlap, the gain being stronger at high accuracy levels. Indeed, since the optimization of the latent code in our method matches the upsampled pose to the input, an additional selection of the result closest to the input among the several candidate poses at each frame when using an overlap brings little gain in accuracy.

However, as illustrated on Fig. 6, we found that processing overlapping chunks of frames noticeably improves the temporal consistency of the output pose sequence. We observed that the per-frame joint positioning accuracy drops at the extremities of the processed chunks, probably because of the reduced temporal context information there. Without overlap this introduces an increased temporal jitter at the chunk boundaries of the generated pose sequence, which is likely to incur perceptually disturbing artifacts when applying our method to, *e.g.*, character animation.

V. CONCLUSION

In this paper we presented a novel method for human pose inpainting focused on joints upsampling. Our approach relies on a hybrid adversarial generative model to improve the resolution of the skeletons (*i.e.*, the number of joints) with no loss of accuracy. To the best of our knowledge, this
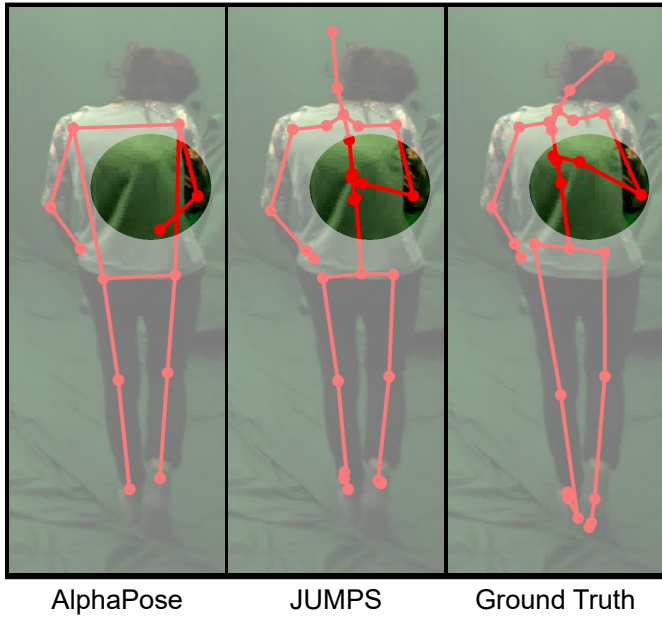
AlphaPose   **JUMPS**   **Ground Truth**

Fig. 7. Example of a limb (right forearm) occluded by subject's body inaccurately estimated by *AlphaPose* but recovered by our method based on human motion priors. Note that these images have been intentionally whitened except for the area around the occlusion for clarity purposes.

is the first attempt to solve this problem with a machine learning technique. We have also shown its applicability and effectiveness to Human Pose Estimation.

Our framework considers a 12-joint $2D$ pose sequence as input and produces a valuable 28-joint $2D$ pose sequence by inpainting the input. The proposed model consists of the fusion of a deep convolutional generative adversarial network and an autoencoder. Ablation studies have shown the strong benefit of the autoencoder, since it provides some supervision that greatly helps the convergence and accuracy of the combined model. Given an input sequence, inpainting is performed by optimizing the latent representation that best reconstructs the low-resolution input. The encoder provides the initialization and a prior loss based on the discriminator is used to improve the plausibility of the generated output.

The obtained results are encouraging and open up future research opportunities. Better consistency of the inpainted pose sequences with true human motion could be obtained either by explicitly enforcing biomechanical constraints, or by extending the method to $3D$ joints, in order to benefit from richer positional information on the joints. Additionally, a potentially fruitful line of research would be to tackle as a whole, from a monocular image input, the extraction of $3D$ human pose and the upsampling of skeleton joints. Finally, we plan to study more genuine temporal analysis by using a different network architecture handling either longer or variable-length pose sequences (*e.g.*, based on recurrent neural networks or fully convolutional networks).

REFERENCES

[1] Z. Cao, G. Hidalgo, T. Simon, S. Wei, Y. Sheikh, *OpenPose: Realtime Multi-Person 2D Pose Estimation using Part Affinity Fields*, IEEE TPAMI, 2019.

[2] Z. Su, M. Ye, G. Zhang, L. Dai, J. Sheng, *Cascade Feature Aggregation for Human Pose Estimation*, arXiv:1902.07837, 2019.

[3] F. Zhang, X. Zhu, H. Dai, M. Ye, C. Zhu, *Distribution-Aware Coordinate Representation for Human Pose Estimation*, arXiv:1910.06278, 2019.

[4] K. Sun, B. Xiao, D. Liu, J. Wang, *Deep High-Resolution Representation Learning for Human Pose Estimation*, arXiv:1902.09212, 2019.

[5] B. Artacho, A. Savakis, *UniPose: Unified Human Pose Estimation in Single Images and Videos*, arXiv:2001.08095, 2020.

[6] B. Tekin, I. Katircioglu, M. Salzmann, V. Lepetit, P. Fua *Structured Prediction of 3D Human Pose with Deep Neural Networks*, In BMVC, 2016.

[7] D. Holden, J. Saito, T. Komura, *A Deep Learning Framework for Character Motion Synthesis and Editing*, ACM ToG vol. 35 no. 4, 2016.

[8] J. Bao, D. Chen, F. Wen, H. Li, G. Hua, *CVAE-GAN: Fine-Grained Image Generation through Asymmetric Training*, In ICCV, 2017.

[9] R. Yeh, C. Chen, T. Yian Lim, A. Schwing, M. Hasegawa-Johnson, M. Do, *Semantic Image Inpainting with Deep Generative Models*, In CVPR, 2017.

[10] S. Iizuka, E. Simo-Serra, H. Ishikawa, *Globally and Locally Consistent Image Completion*, ACM ToG, vol. 36 no. 4, 2017.

[11] J. Yu, Z. Lin, J. Yang, X. Shen, X. Lu, T.S. Huang, *Generative Image Inpainting with Contextual Attention*, In CVPR, 2018.

[12] D. Pathak, P. Krhenbhl, J. Donahue, T. Darrell, A. A. Efros, *Context Encoders: Feature Learning by Inpainting*, In CVPR, 2016.

[13] D. Kingma, M. Welling, *Auto-Encoding Variational Bayes*, In ICLR, 2014.

[14] A. H. Jha, S. Anand, M. Singh, and V. Veeravasarapu, *Disentangling Factors of Variation with Cycle-Consistent Variational Auto-Encoders*, In ECCV, 2018

[15] I. Goodfellow, J. Pouget-Abadie, M. Mirza, B. Xu, D. Warde-Farley, S. Ozair, A. Courville, Y. Bengio, *Generative Adversarial Nets*, In NIPS, 2014.

[16] M. Arjovsky, S. Chintala, L. Bottou, *Wasserstein GAN*, arXiv preprint arXiv:1701.07875, 2017.

[17] I. Gulrajani, F. Ahmed, M. Arjovsky, V. Dumoulin, A. Courville, *Improved Training of Wasserstein GANs*, In NIPS, 2017.

[18] T. Karras, T. Aila, S. Laine, J. Lehtinen, *Progressive Growing of GANs for Improved Quality, Stability, and Variation*, In ICLR, 2018.

[19] A. Radford, L. Metz, S. Chintala, *Unsupervised Representation Learning with Deep Convolutional Generative Adversarial Networks*, In ICLR, 2016.

[20] D. Kingma, J. Ba, *Adam: A method for stochastic optimization*, arXiv preprint arXiv:1412.6980, 2014.

[21] D. Mehta, H. Rhodin, D. Casas, P. Fua, O. Sotnychenko, W. Xu, C. Theobalt, *Monocular 3D Human Pose Estimation In The Wild Using Improved CNN Supervision*, In 3DV, 2017.

[22] C. Ionescu, D. Papava, V. Olaru, C. Sminchisescu, *Human3.6M: Large Scale Datasets and Predictive Methods for 3D Human Sensing in Natural Environments*, IEEE TPAMI, 2014.

[23] H. Fang, S. Xie, Y. Tai, C. Lu, *RMPE: Regional Multi-Person Pose Estimation*, In ICCV, 2017.

[24] J. Li, C. Wang, H. Zhu, Y. Mao, H. Fang, C. Lu, *CrowdPose: Efficient Crowded Scenes Pose Estimation and A New Benchmark*, In arXiv preprint arXiv:1812.00324, 2018

[25] Y. Xiu, J. Li, H. Wang, Y. Fang, C. Lu, *Pose Flow: Efficient Online Pose Tracking*, In BMVC, 2018.

[26] M. Andriluka, L. Pishchulin, P. Gehler, B. Schiele, *2D Human Pose Estimation: New Benchmark and State of the Art Analysis*, In CVPR, 2014.

[27] E. Insafutdinov, L. Pishchulin, B. Andres, M. Andriluka, B. Schiele, *DeeperCut: A Deeper, Stronger, and Faster Multi-Person Pose Estimation Model*, In ECCV, 2016