

# FLEX: Extrinsic Parameters-free Multi-view 3D Human Motion Reconstruction

Brian Gordon\* , Sigal Raab\* , Guy Azov ,  
Raja Giryes , and Daniel Cohen-Or

Tel Aviv University  
**briangordon@mail.tau.ac.il, sigalraab@tauex.tau.ac.il,**  
**guyazov@mail.tau.ac.il, {raja,dcor}@tauex.tau.ac.il**

**Abstract.** The increasing availability of video recordings made by multiple cameras has offered new means for mitigating occlusion and depth ambiguities in pose and motion reconstruction methods. Yet, multi-view algorithms strongly depend on camera parameters, particularly on relative transformations between the cameras. Such a dependency becomes a hurdle once shifting to dynamic capture in uncontrolled settings. We introduce FLEX (Free muLti-view rEconstruXion), an end-to-end extrinsic parameter-free multi-view model. FLEX is *extrinsic parameter-free* (dubbed *ep-free*) in the sense that it does not require extrinsic camera parameters. Our key idea is that the 3D angles between skeletal parts, as well as bone lengths, are invariant to the camera position. Hence, learning 3D rotations and bone lengths rather than locations allows for predicting common values for all camera views. Our network takes multiple video streams, learns fused deep features through a novel multi-view fusion layer, and reconstructs a single consistent skeleton with temporally coherent joint rotations. We demonstrate quantitative and qualitative results on three public data sets, and on multi-person synthetic video streams captured by dynamic cameras. We compare our model to state-of-the-art methods that are not ep-free and show that in the absence of camera parameters, we outperform them by a large margin while obtaining comparable results when camera parameters are available. Code, trained models, and other materials are available on <https://briang13.github.io/FLEX>.

**Keywords:** Motion reconstruction, Character animation, Pose estimation, Camera parameters, Deep learning.

## 1 Introduction

Human motion reconstruction is the task of associating a skeleton with temporally coherent joint locations and rotations. Acquiring accurate human motion in a controlled setting, using motion capture systems with adequate sensors is a tedious and expensive procedure that cannot be applied for capturing spontaneous activities, such as sporting events. Motion reconstruction from RGB cameras is low-cost and non-intrusive, but is an uncontrolled setup. Thus, while being simple, it has technical challenges that are worsened by occlusion and depth ambiguity. Using multiple cameras may alleviate these difficulties as different views may compensate for occlusion and be used for mutual consistency.

---

\* equal contribution.

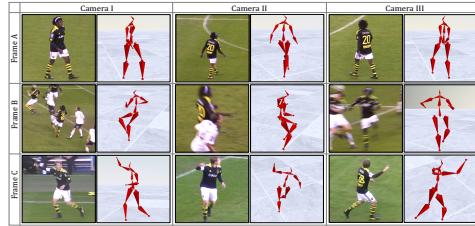


Fig. 1: Results on the KTH Multi-view Football II dataset [34], in occluded and blurry scenes with dynamic cameras.

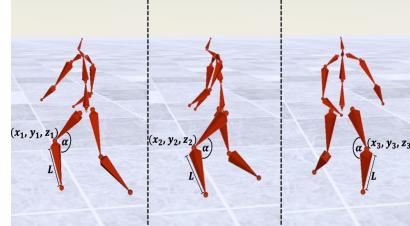


Fig. 2: 3D locations vary across axis systems while 3D rotation angles and bone lengths remain identical.

Recently, there has been a significant progress in using deep learning for pose and motion reconstruction [55,43,64,67,57,37,52]. Most of these methods work in a monocular setting, but a growing number of works learn a multi-view setting [30,75,59,25,20,61]. However, these approaches depend on the relative position between the cameras, derived from extrinsic camera parameters, and assume they are given. In the lack of extrinsic parameters, several works estimate them [16,38], but at the cost of innate inaccuracy of estimated values. While camera parameters are often given in multi-view datasets, they are rarely given in dynamic capture environments. We refer to cameras as *dynamic* if they occasionally move during video capture, such that their extrinsic parameters and their inter-camera relative positions are not fixed. An example of such a camera is the SkyCam [69], commonly used in sports events.

This work introduces an extrinsic parameter-free (dubbed *ep-free*) multi-view motion reconstruction method, whose setting is illustrated in the inset to the right. Our method builds upon a new conceptual observation that uses the well-known joint rotations and bone lengths, to free us from the burdening dependency on extrinsic camera parameters. Our approach relies on a key insight that joint rotations and bone lengths are identical for all views. That is, the 3D angle between skeletal parts is invariant to the camera position. We train a neural network to predict 3D joint angles and bone lengths *without* using the extrinsic camera parameters, neither in training nor in test time. Predicting motion rather than locations is not a novel idea by itself. The innovation of our work is in the way we use motion to bypass the need for camera parameters. The input from multiple cameras is integrated by a novel fusion layer that implicitly promotes joints detected by some cameras and demotes joints detected by others, hence mitigating occlusion and depth ambiguities.



Our model, named FLEX, is an end-to-end deep convolutional network. Its input is multi-view 2D joints that are either given or extracted using a 2D pose estimation technique. FLEX employs multi-view blocks with cross-view attention on top of a monocular baseline [67], and uses temporal information over a video of arbitrary length, thus obtaining temporal consistency.

We evaluate FLEX qualitatively and quantitatively using the Human3.6M [29,9], the KTH Multi-view Football II [34] and the Ski-Pose PTZ-Camera [62] datasets.

Figure 1 demonstrates qualitative results, and more are depicted in Section 4 and in the appendix. FLEX is also applied on synthetic videos. We have generated these videos using Mixamo [1] and Blender [18], to mitigate the lack of a multi-person video dataset that is captured by dynamic cameras, and created them such that they contain severe inter-person occlusions.

We compare performance with state-of-the-art methods that are not ep-free and show comparable results. To simulate an ep-free setting, we perturb ground-truth camera parameters or use works that estimate them. We show that in an ep-free setting, our model outperforms state-of-the-art by a large margin.

Our main contributions are twofold: (i) a network that reconstructs motion and pose in a multi-view setting with unknown extrinsic camera parameters, and (ii) a novel fusion layer with a multi-view convolutional layer combined with a multi-head attention mechanism over a number of views.

## 2 Related work

**Pose Estimation using a Single View** Pose estimation receives significant interest in computer vision. Before the deep era, it was approached using heuristics such as physical priors [63]. The emergence of deep learning and large datasets [29,17,50,34], have led to significant advances. Pose estimation methods can generally be divided into two groups. The first infers 3D locations directly from images or videos [40,56,87,73,71,13,24]. The second, aka *lifting*, applies two steps: (i) estimating 2D poses and (ii) lifting them to 3D space [51,57,22,70,43,26,66]. The first group benefits from directly using images, which are more descriptive compared to 2D joint locations. The second gains from using intermediate supervision. Recently, transformers and convolutional graph based methods were shown to improve performance [45,42,44,41,82,27,54].

**Pose Estimation using Multiple Views** The growing availability of synchronized video streams taken by multiple cameras has contributed to the emergence of multi-view algorithms. Such algorithms exploit the diversity in camera views to predict more accurate 3D poses. All works described below predict pose and many of them analyze each frame individually. On the other hand, our model, FLEX, reconstructs motion and exploits temporal information.

Most works in the multi-view setting rely on lifting from 2D to 3D space. Early works [4,5,7] estimate the input 2D pose from single images, while later works [20,59,30,25,38,10,16,61,31,15] obtain the 2D pose by running a CNN over 2D poses given in multiple views; resulting in an increase in 2D pose prediction accuracy. After estimating the 2D poses, most works apply heuristics such as triangulation or pictorial structure model (PSM). FLEX is one of the few works [30,75] that present an end-to-end model.

Several methods use multi-view data to improve the 2D pose estimation. Some use the camera parameters to find the matching epipolar lines such that features gathered from several cameras are aggregated [59,25]. Chen *et al.* [10] learn a geometric representation in latent space with an encoder-decoder.

Several works [38,16,80,74] use self-supervision, hence need no 3D ground-truth. Their main idea is to project the predicted 3D joints (using real or esti-

mated camera parameters) and expect consistency with 2D input joints. Recent techniques [28,85] exploit more sensors, such as IMU, during data capturing.

Current state-of-the-art results are attained by Iskakov *et al.* [30], Tu *et al.* [75] and Reddy *et al.* [60]. They use end-to-end networks, and present a volumetric approach, where 2D features are un-projected from individual views to a common 3D space, using camera parameters. Sun *et al.* [70] show that synthetic generation of additional views helps produce more accurate lifting.

At inference time, some of the aforementioned works expect monocular inputs [70,25,10,16] and some, including FLEX, get multi-view inputs [30,75,59]. The advantage of the first is the use of monocular data that is more common, and of the second is better results on multi-view settings.

Epipolar Transformers [25] attend to spatial locations on an epipolar line in a *single* view and query it using one joint in a query view. A concurrent work, TransFusion [48], applies a transformer on inter and intra-view features.

In the absence of camera parameters, most of the methods cannot be used. Some estimate rotation assuming the translation is given [38,2] or engage an extra effort to estimate the camera parameters [16,80,11,76,72]. Such an effort is not required by FLEX as it uses no camera parameters whatsoever.

**Rotation and Motion Reconstruction** Pose estimation may suffice for many applications; however, pose alone does not fully describe the motion and the rotations associated with the joints. *Rotation reconstruction* relates to the prediction of joint rotation angles, while *motion reconstruction* requires the prediction of bone lengths associated with them. Many works explore the task of *3D shape recovery* [46,32,39,37,36,84,33,23,47,14], focusing on human mesh prediction along with joint rotations. Most of them do not guarantee temporal coherence, *e.g.*, bone length may vary across time frames.

Other works [58,49] focus on motion generation. Given a series of human motions, they predict future motions, using various techniques such as temporal supervision and graph convolutional networks (GCN). Similar to us, human motion reconstruction methods [86,52,67,53] focus on the temporal coherence of the body, where the bone lengths are fixed over time and rotations are smooth.

### 3 Extrinsic Parameter-free multi-view model

The premise of our work is that 3D joint rotations and bone lengths are view-independent values. For example, the 3D angle between, say, the thigh and the shin, as well as the length of these bones, are fixed, no matter which camera transformation is used. On the other hand, joint locations differ for each camera transformation, as seen in Figure 2. Our key idea is to directly predict joint 3D angles and bone lengths without using the extrinsic camera parameters, during both training and test time. *Extrinsic* parameters correspond to the rotation and translation (aka transformation) from 3D real world axes into 3D camera axes. A formal definition of the camera parameters can be found in Appendix F.

Our method takes multi-view sequences of 2D poses and estimates the motion of the observed human. The 2D poses are either given or extracted using a prediction technique. Having multi-view data compensates for the inherent inaccuracy

of 2D pose estimation algorithms. Many methods estimate view-dependent 2D joint positions and then lift them to 3D by transforming them into a shared space. Such transformations require acquaintance of the relative position (rotation and translation) between the cameras, which is derived from the extrinsic camera parameters. Our model directly predicts 3D rotations and bone lengths, which are agnostic to camera transformation. The predicted values are shared by all views, so there is no need for extrinsic parameters information.

Pose estimation methods may mitigate the lack of extrinsic parameters by estimating them [16,38]. Yet, this has two drawbacks: (i) most approaches perform the estimation in a prepossessing step that breaks the end-to-end computation, and (ii) the estimated parameters are never exact and typically lead to a performance drop, as we show in Section 4.

Our architecture leverages Shi *et al.* [67] and is illustrated in high-level terms in Figure 3. FLEX is an end-to-end network that maps 2D joint positions, extracted from multiple synchronized input videos, into two separate components: (i) a sequence of 3D joint rotations, global root positions and foot contact labels (upper branch in the figure); this sequence is skeleton-independent and varies per frame; and (ii) a single, symmetric, 3D skeleton, represented by its bone lengths (lower branch in the figure). We can combine these two components into a complete description of a motion and use it for 3D animation tasks without further processing or inverse kinematics (IK).

In addition to being free of extrinsic parameters, our model does not use intrinsic parameters at all, at the cost of an up-to-scale global skeleton position. While FLEX removes the need for extrinsics, it uses the common weak perspective assumption [36] for intrinsics; in particular for mitigating the lack of focal length. Indeed, some works seek to mitigate the lack of intrinsic parameters [68,23,36] whereas this is not the focus of our work. In Section 4 we show that using a customary weak perspective we attain an accurate global position.

The terms *motion*, *pose*, *reconstruction* and *estimation* are used in various contexts in the literature. To avoid confusion, we define *motion* as one set of bone lengths associated with temporally coherent 3D joint rotations, and *pose* as a temporal sequence of 3D joint locations. We use the term *reconstruction* rather than *estimation*, as the latter often describes 2D spatial motion. A weakly related term, *pose tracking*, associates poses to identities in a multi-person setting.

Motion data, and in particular rotations rather than positions, are required in animation platforms and game engines. FLEX directly outputs a kinematic skeleton, which is a complete, commonly used, motion representation. On the other hand, methods that predict joint positions, rely on IK to associate a skeleton with joint rotations. IK is slow, non-unique, and prone to temporal inconsistencies and unnatural postures. Moreover, methods that only predict pose cannot guarantee the consistency of bone lengths across frames.

### 3.1 Architecture

We start with a high-level description of the architecture (see Figure 3). The inputs are  $K$  synchronized video streams of  $T$  frames each. For each video

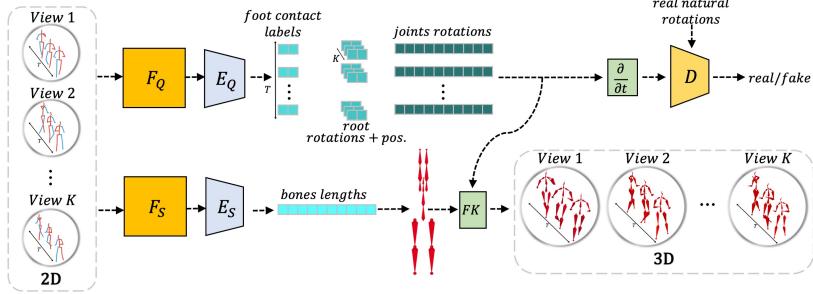


Fig. 3: FLEX takes multi-view temporal sequences of 2D poses and their confidence values. It uses two encoders,  $E_Q$  and  $E_S$ , to extract per-frame 3D rotations and foot contact labels, per-view and per-frame 3D root transformations, and one static skeleton. A discriminator  $D$  monitors the temporal differences of rotation angles, and a forward kinematic layer,  $FK$ , combines encoders' outputs into 3D joint locations. These outputs depict one human, transformed into the axis systems of  $K$  cameras, to be compared with  $K$  sets of ground-truth values.

stream, we obtain 2D joints, which are either the ground-truth of a dataset or the output of a 2D pose estimation algorithm. Our network is agnostic to the way those 2D joints were obtained. In addition, each estimated joint is associated with a confidence value. The confidence value plays an important role in balancing between visible and occluded joints.

Our model takes input from all views, aggregates it, and streams it into two independent fusion layers  $F_S$  and  $F_Q$ , followed by encoders  $E_S$  and  $E_Q$ , respectively. The two fusion layers differ in some architectural details, but share the same concept. Both aggregate data of all views and frames and fuse it to exploit characteristics that recur in views and/or frames. Each fusion layer outputs view-agnostic features that represent the target human.

The fusion layers consist of two innovative elements, a multi-view convolutional layer and a *cross-view attention* mechanism, which encodes information from all views. Our use of attention is unique, as typically attention in other works is applied mostly over pixels [35] and sometimes over time [45, 42, 44]. Attention over views is a novel approach, which we find only in concurrent works for other tasks, assessing human shape [88] and rigid objects [81]. The fusion layers are described in detail in the supplementary material.

The encoder  $E_S$  predicts the length of each bone. As the same human is analyzed along all frames and views, the output is a single set of bone lengths.

The encoder  $E_Q$  predicts joint rotations, global root positions, and foot contact labels. Since 3D joint rotations and foot contact labels are identical to all views,  $E_Q$  predicts a single set of rotations and contact labels per frame, shared by all views. One exception is the root (pelvis) joint, whose rotation angle and position depend on the camera view and not on the human itself. Thus, for the root joint we predict the rotation angle and position for each frame and view. Root rotation and position are relative to the camera; hence visualizing the reconstructed object depicts the filmed person from the camera view, as expected.

Notice that root rotation and position carry the knowledge of the relative transformation between the cameras. This insight suggests that our algorithm has the potential to output additional valuable information, *e.g.*, cameras relative location to each other (left to future work).

At train time, the output of both encoders is combined in  $K$  identical forward kinematic (*FK*) layers. Each *FK* layer computes the estimated 3D joint positions related to one view, which in turn are compared to the ground-truth for loss computation. In addition, temporal differences of the rotations extracted out of  $E_Q$ , are fed to a discriminator  $D$  [32], so they get near the manifold of true rotations in an adversarial way.

Formally, let  $\mathbf{L}$  denote the number of bones,  $\mathbf{T}$  the temporal length of the sequence,  $\mathbf{J}$  the number of joints,  $\mathbf{Q}$  the size of the rotations representation vector, and  $\mathbf{K}$  the number of cameras. Let  $\mathbf{P}_{s,q,r} \in \mathbb{R}^{T \times 3J \times K}$  denote  $K$  temporal sequences of 3D joint positions generated by a skeleton  $\mathbf{s} \in \mathbb{R}^L$  with joint rotations  $\mathbf{q} \in \mathbb{R}^{T \times Q \times (J-1)}$ , and global root position and rotation  $\mathbf{r} \in \mathbb{R}^{T \times (3+Q) \times K}$ . Note that  $q$  is related to all joints except for the root joint. The rotation of the root joint, as well as its position, are related to  $r$ .

Our approach expects an input  $\mathbf{V}_{s,q,r} \in \mathbb{R}^{T \times 3J \times K}$  denoting  $K$  temporal sequences of 2D joints and a confidence value per joint, related to a skeleton  $s$ , joint rotations  $q$ , and global root position and rotation  $r$ . Each input  $V$  is fed into our deep neural network, which in turn predicts  $\tilde{\mathbf{q}} \in \mathbb{R}^{T \times Q \times (J-1)}$ , that captures the dynamic, rotational information of the motion,  $\tilde{\mathbf{s}} \in \mathbb{R}^L$ , that describes a single, consistent, skeleton,  $\tilde{\mathbf{r}} \in \mathbb{R}^{T \times (3+Q) \times K}$  that estimates the global position and rotation of the root along time and along views, and  $\tilde{\mathbf{f}} \in \{0, 1\}^{T \times 2}$  that predicts whether each of the two feet touches the ground in each frame:

$$\tilde{\mathbf{s}} = E_S(F_S(\mathbf{V}_{s,q,r})), \quad \tilde{\mathbf{q}}, \tilde{\mathbf{r}}, \tilde{\mathbf{f}} = E_Q(F_Q(\mathbf{V}_{s,q,r})). \quad (1)$$

These attributes can be then combined via forward kinematics to estimate  $K$  global 3D pose sequences,  $\tilde{\mathbf{P}}_{\tilde{s}, \tilde{q}, \tilde{r}} \in \mathbb{R}^{T \times 3J \times K}$ , specified by joint positions:

$$\tilde{\mathbf{P}}_{\tilde{s}, \tilde{q}, \tilde{r}} = FK(\tilde{\mathbf{s}}, \tilde{\mathbf{q}}, \tilde{\mathbf{r}}). \quad (2)$$

We employ five loss functions. Our losses are inspired by Shi *et al.* [67] and are enhanced to encompass the multitude of views.

**Joint Position Loss** (the main loss)  $\mathcal{L}_P$  ensures that joints in the extracted positions are in their correct 3D positions:

$$\mathcal{L}_P = \mathbb{E}_{\mathbf{P}_{s,q,r} \sim \mathcal{P}} [\|FK(\tilde{\mathbf{s}}, \tilde{\mathbf{q}}, \tilde{\mathbf{r}}_{pos_0}) - \mathbf{P}_{s,q,r_{pos_0}}\|^2], \quad (3)$$

where  $\mathbf{P}_{s,q,r} \in \mathbb{R}^{T \times 3J \times K}$  denotes a 3D motion sequence,  $\mathcal{P}$  represents the distribution of 3D motion sequences in our dataset, and  $\tilde{r}_{pos_0}, r_{pos_0}$  stand for global position and rotation of the predicted and given root respectively, where the location is set to  $(0, 0, 0)$ , but the rotation is unchanged.

**Skeleton Loss**  $\mathcal{L}_S$  stimulates the skeleton branch of the network,  $F_S$  and  $E_S$ , to correctly extract the skeleton  $\mathbf{s}$ :

$$\mathcal{L}_S = \mathbb{E}_{\mathbf{P}_{s,q,r} \sim \mathcal{P}} [\|E_S(F_S(\mathbf{V}_{s,q,r})) - \mathbf{s}\|^2]. \quad (4)$$

**Adversarial Rotation Loss** Our network learns to output rotations with natural velocity distribution using adversarial training. To achieve this, instead of focusing on rotation absolute values, like Kanazawa *et al.* [32] we focus on the temporal differences of joint rotations. We create a discriminator  $D_j$  for each joint. Note that the loss involving  $D_{j \neq 0}$  takes the rotation values from  $\tilde{q}$  while the loss involving  $D_0$  takes the rotation values from  $\tilde{r}$ . It reads as

$$\begin{aligned}\mathcal{L}_{Q\text{-}GAN_{j \neq 0}} &= \mathbb{E}_{q \sim \mathcal{Q}} [\|D_j(\Delta_t q_j)\|^2] + \mathbb{E}_{\mathbf{P}_{s,q,r} \sim \mathcal{P}} [\|1 - D_j(\Delta_t E_Q(F_Q((\mathbf{V}_{s,q,r}))_{q_j})\|^2] \\ \mathcal{L}_{Q\text{-}GAN_{j=0,k}} &= \mathbb{E}_{q \sim \mathcal{Q}} [\|D_j(\Delta_t q_j)\|^2] \\ &\quad + \mathbb{E}_{\mathbf{P}_{s,q,r} \sim \mathcal{P}} [\|1 - D_j(\Delta_t E_Q(F_Q((\mathbf{V}_{s,q,r}))_{r_{rot_k}})\|^2],\end{aligned}\tag{5}$$

where  $\mathcal{Q}$  stands for the distribution of natural joint angles in the dataset,  $E_Q(F_Q(\cdot))_{q_j}$  denotes the predicted rotations of the  $j$ th joint,  $E_Q(F_Q(\cdot))_{r_{rot_k}}$  represents the predicted rotation of the pelvis joint relative to camera  $k$ , and  $\Delta_t$  denotes temporal differences.

**Global Root Position Loss** We estimate the depth parameter,  $Z_r \in \mathbb{R}^{T \times K}$ , by minimizing:

$$\mathcal{L}_R = \mathbb{E}_{\mathbf{P}_{s,q,r} \sim \mathcal{P}} [\|E_Q(F_Q(\mathbf{V}_{s,q,r}))_{r_{pos_z}} - Z_r\|^2],\tag{6}$$

where  $Z_r$  is the depth of the ground-truth root, and  $E_Q(F_Q(\cdot))_{r_{pos_z}}$  is the depth of the predicted root. Note that  $Z_r$  consists of values for all views and all frames.

**Foot Contact Loss** We predict whether each foot contacts the ground in each frame and train the network via

$$\mathcal{L}_F = \mathbb{E}_{\mathbf{P}_{s,q,r} \sim \mathcal{P}} [\|E_Q(F_Q(\mathbf{V}_{s,q,r}))_f - \mathbf{f}\|^2],\tag{7}$$

where  $E_Q(F_Q(\cdot))_f$  denotes the predicted foot contact label part ( $\mathbf{f} \in \{0, 1\}^{T \times 2}$ ). We encourage the velocity of foot positions to be zero during contact frames, by

$$\mathcal{L}_{FC} = \mathbb{E}_{\mathbf{P}_{s,q,r} \sim \mathcal{P}} \left[ \|\mathbf{f}_i \sum_j \Delta_t FK(\tilde{\mathbf{s}}, \tilde{\mathbf{q}}, \tilde{\mathbf{r}})_{f_i}\|^2 \right],\tag{8}$$

where  $FK(\cdot, \cdot, \cdot)_{f_i} \in \mathbb{R}^{T \times 3}$  and  $\mathbf{f}_i$  denote the positions and the contact labels of one of the feet joints ( $i \in \text{left, right}$ ), and  $\sum_j$  sums the components for all axes.

Altogether, we obtain a total loss of:

$$\begin{aligned}\mathcal{L} &= \mathcal{L}_P + \lambda_S \mathcal{L}_S \lambda_Q \left( \sum_{j \neq 0} \mathcal{L}_{Q\text{-}GAN_j} + \sum_{j=0,k} \mathcal{L}_{Q\text{-}GAN_{j,k}} \right) \\ &\quad + \lambda_R \mathcal{L}_R + \lambda_F \mathcal{L}_{P_F} + \lambda_{FC} \mathcal{L}_{P_{FC}}.\end{aligned}\tag{9}$$

In most experiments we use  $\lambda_S = 0.1$ ,  $\lambda_Q = 1$ ,  $\lambda_R = 1.3$ ,  $\lambda_F = 0.5$  and  $\lambda_{FC} = 0.5$ .

In the appendix we provide more implementation details, such as the description of each architectural block; in particular the novel fusion layers  $F_S$  and  $F_Q$ . We discuss the advantages of early vs. middle and late fusion, and describe how we improve skeleton topology comparing to our single-view baseline. We also provide a detailed description of the datasets, a discussion of 2D pose estimators, and a description of the ground-truth we use.

Table 1: Protocol #1 MPJPE error on Human3.6M. Legend: (\*) is a **non** ep-free algorithm. In case parameters are not given, we imitate their computation by perturbing the GT params by an unrealistically small perturbation amount; (†) exploit temporal information; (+) extra training data. In **blue** - best result when camera parameters are not given, in **bold** - best result per method group.

Method	Dir.	Disc.	Eat	Greet	Phone	Photo	Pose	Purch.	Sit	SitD.	Smoke	Wait	WalkD.	Walk	WalkT.	Mean
<b>Monocular methods</b>																
Shi <i>et al.</i> [67](†)	47.3	53.1	50.3	53.9	53.5	52.8	52.0	55.4	64.2	54.8	66.8	55.0	50.3	59.1	50.3	54.6
Llopert [45](†)	42.2	44.5	42.6	43.0	46.9	53.9	42.5	41.7	55.2	62.3	44.9	42.9	45.3	31.8	31.8	44.8
Reddy <i>et al.</i> [60](†)	38.4	46.2	44.3	43.2	44.8	48.3	52.9	<b>36.7</b>	<b>45.3</b>	54.5	63.4	44.4	41.9	46.2	39.9	44.6
Li <i>et al.</i> [41](†)	39.9	43.4	40.0	40.9	46.4	50.6	42.1	39.8	55.8	61.6	44.9	43.3	44.9	29.9	30.3	43.6
Hu <i>et al.</i> [27](†)	<b>35.5</b>	41.3	<b>36.6</b>	39.1	42.4	49.0	39.9	37.0	51.9	63.3	<b>40.9</b>	<b>41.4</b>	<b>40.3</b>	<b>29.8</b>	28.9	41.1
Cheng <i>et al.</i> [13] (†)	36.2	<b>38.1</b>	42.7	<b>35.9</b>	<b>38.2</b>	<b>45.7</b>	<b>36.8</b>	42.0	45.9	<b>51.3</b>	41.8	41.5	43.8	33.1	<b>28.6</b>	<b>40.1</b>
<b>Multi-view methods, extrinsic camera parameters are given</b>																
Tome <i>et al.</i> [74] (+)	43.3	49.6	42.0	48.8	51.1	64.3	40.3	43.3	66.0	95.2	50.2	52.2	51.1	43.9	45.3	52.8
Kadkhodamohammadi and Padoy [31]	39.4	46.9	41.0	42.7	53.6	54.8	41.4	50.0	59.9	78.8	49.8	46.2	51.1	40.5	41.0	49.1
He <i>et al.</i> [25]	25.7	27.7	23.7	24.8	26.9	31.4	24.9	26.5	28.8	31.7	28.2	26.4	23.6	28.3	23.5	26.9
Qiu <i>et al.</i> [59] (+)	24.0	26.7	23.2	24.3	24.8	22.8	24.1	28.6	32.1	26.9	31.0	25.6	25.0	28.0	24.4	26.2
Ma <i>et al.</i> [48](†)	24.4	26.4	23.4	21.1	25.2	23.2	24.7	33.8	29.8	26.4	26.8	24.2	23.2	26.1	23.3	25.8
Iskakov <i>et al.</i> [30]	19.9	20.0	18.9	18.5	20.5	19.4	18.4	22.1	22.5	28.7	21.2	20.8	19.7	22.1	20.2	20.8
Reddy <i>et al.</i> [60](†)	<b>17.5</b>	<b>19.6</b>	<b>17.2</b>	<b>18.3</b>	<b>18.2</b>	<b>17.7</b>	<b>18.0</b>	<b>18.0</b>	<b>20.5</b>	<b>20.3</b>	<b>19.4</b>	<b>17.2</b>	<b>18.9</b>	<b>19.0</b>	<b>17.8</b>	<b>18.7</b>
<b>Multi-view methods, extrinsic camera parameters are not given</b>																
Chu and Pan [16](†)	49.1	63.6	48.6	56.0	57.4	69.6	50.4	62.0	75.4	77.4	57.2	53.5	57.7	37.6	38.1	56.9
Iskakov <i>et al.</i> [30](*)	30.2	37.2	32.7	33.2	38.8	43.7	29.7	43.0	49.4	67.6	38.0	33.1	42.1	27.2	29.3	38.4
param. perturb by 4%																
Iskakov <i>et al.</i> [30](*)	27.6	30.3	29.0	29.4	33.1	36.5	27.4	34.8	39.1	54.0	34.4	30.7	36.2	26.2	28.4	33.1
param. perturb by 3%																
<b>Ours</b> (†)	<b>22.0</b>	<b>23.6</b>	<b>24.9</b>	<b>26.7</b>	<b>30.6</b>	<b>35.7</b>	<b>25.1</b>	<b>32.9</b>	<b>29.5</b>	<b>32.5</b>	<b>32.6</b>	<b>26.5</b>	<b>34.7</b>	<b>26.0</b>	<b>27.7</b>	<b>30.2</b>

## 4 Experiments and evaluation

We present quantitative results on the Human3.6M [29,9] and Ski-Pose PTZ-Camera [62] datasets. We present qualitative results on the Human3.6M, KTH Multi-view Football II [34] and Ski-Pose PTZ-Camera [62] datasets, and on synthetic videos captured by dynamic cameras. Detailed description of these datasets can be found in the supplementary material.

**Quantitative results** We show quantitative results using the Mean Per Joint Position Error (MPJPE) [29,9], and report standard protocol #1 MPJPE (that is, error relative to the pelvis), in millimeters.

Table 1 presents a quantitative comparison of the MPJPE metric on the Human3.6M [29] dataset. We present monocular methods, followed by multi-view ones that are split into ones that are acquainted with camera parameters and ones that are not. We show that in the absence of camera parameters, our model outperforms state-of-the-art methods by a large margin, and that even when camera parameters are available, FLEX is among the top methods. Note that these achievements are although FLEX aims at a slightly different task, which is motion reconstruction rather than pose estimation.

Being the only ep-free algorithm, we have no methods to compare to directly. However, algorithms can mitigate the lack of extrinsic camera parameters by estimating them. In the following comparisons, we show that when extrinsic parameters are not given, using estimated ones induces larger prediction errors, due to the innate inaccuracy of predicted values. On the other hand, FLEX

is not affected by the lack of extrinsic parameters, since it does not use them whatsoever. We compare FLEX with two models:

- (1) There are two methods that do not use given camera parameters [16,38]. They are not ep-free since they use estimated camera parameters, but we can still use them in settings where camera parameters are not given. Only one of them [16] publishes MPJPE protocol #1 results, and we significantly outperform it (See Table 1). This gap is mostly because of the inaccuracy of parameter prediction and partially because their model is semi-supervised.
- (2) For comparing with the best available method, we have chosen the current state-of-the-art multi-view algorithm of Iskakov *et al.* [30] (TesseTrack [60] is marginally better, but it does not provide code). Since their algorithm is *not* ep-free, we imitate parameter estimation by running a controlled perturbation of the camera parameters. We re-train their method with distorted data to simulate an environment where camera distortion parameters are unknown. In addition, we perturb the extrinsic parameters by Gaussian noise with an extremely small standard deviation of 3% of each parameter’s value. That is, for a parameter  $p$ , we sample  $\tilde{p} \sim \mathcal{N}(p, (0.03p)^2)$  and use  $\tilde{p}$  as the input extrinsic parameter. We show that increasing the standard deviation from 3% to 4% yields a significant increase in the error, reflecting the sensitivity of non ep-free methods to inaccuracy in camera parameters. To obtain an equivalent environment, we compare FLEX to the method of Iskakov *et al.* [30] after using their own 2D pose estimation. The lower part of Table 1 shows that FLEX outperforms the non ep-free state-of-the-art, even when perturbation percentage is extremely small. Their results in that lower part are grayed out, to emphasize that we simulate an unrealistic setting. Next, we show that a 3% perturbation, rather than estimation, is fairer toward the compared method, as estimation induces larger inaccuracy. We estimate the extrinsic camera parameters with two leading frameworks, COLMAP [65] and OpenCV-SFM [6], and obtain errors of 5.5% and 8.6%, respectively. The error is the mean value of  $\frac{|p - \tilde{p}|}{p}$  for all extrinsic values  $p$  and their estimation  $\tilde{p}$ . Moreover, the estimation process involves friction: OpenCV-SFM strongly depends on an initial guess, and COLMAP requires that each pair of cameras observes partially overlapping images, a limiting factor that prevents its usage in settings where the cameras face each other.

In addition to the comprehensive comparison on the Human3.6 dataset, in Table 2 we show a quantitative comparison on the Ski-Pose PTZ-Camera [62] dataset, for methods that are trained when camera parameters are *not* given. These methods are comparable in settings that lack extrinsic parameters because they estimate them. However, since they still use (estimated) parameters, they are not ep-free. FLEX leads the table with a large gap. This gap is mostly because parameter

Table 2: MPJPE on the Ski-PTZ dataset, measured for methods trained when extrinsic parameters are *not* given. ( $\dagger$ ) is self/weakly-supervised.

Method	MPJPE
CanonPose [80] ( $\dagger$ )	128.1
Chen <i>et al.</i> [11] ( $\dagger$ )	99.4
Ours	<b>65.5</b>

Table 3: Smoothness, measured by acceleration error ( $mm/s^2$ ), on Human3.6M. ( $\star$ ): 2D pose from [30]. ( $\bullet$ ): ground-truth 2D poses.

Method	Acc. Err. ↓
VIBE[37]	18.3
MEVA[47]	15.3
HMMR[33]	9.1
TCMR[14]	5.3
Iskakov[30]	3.9
Shi[67]	3.6( $\star$ ) / 2.0( $\bullet$ )
FLEX	<b>1.6(<math>\star</math>) / 0.9(<math>\bullet</math>)</b>

Table 4: Attention impact.  
TE: Transformer Encoder.  
MHA: Multi-head Attention.  
 $l$ : no. of stacked layers.  
 $h$ : no. of attention heads.

Method	MPJPE
Conv. layer	31.9
TE - $1l$ , $64h$	30.9
TE - $2l$ , $64h$	37.8
MHA - $128h$	30.5
MHA - $64h$	<b>30.2</b>
MHA - $32h$	30.6
MHA - $16h$	30.9

estimation induces an inevitable inaccuracy, and partially because the compared models are self/semi-supervised.

A known strength of predicting rotation angles rather than locations, is the *smoothness* of predicted motion. In Table 3 we show that FLEX’s smoothness result outperforms others by a large margin. Following Kanazawa *et al.* [33], we measure smoothness using the acceleration error of each joint.

**Qualitative results** In the following figures we show rigs, that is, bone structure from reconstructed animation videos, selecting challenging scenes. Videos of the reconstructed motions are available on our project page, presenting the smoothness of motion and the naturalness of rotations. Figures 1, 4 and 5 show scenes from the KTH Multi-view Football II [34], the Human3.6M [29,9] and the Ski-Pose PTZ-Camera [62] datasets, respectively. Each row depicts three views of one time frame. To the right of each image, we place a reconstructed rig, which is sometimes zoomed in for better visualization. Notice the occluded and blurry scenes in the football figure (1). The KTH Football dataset is filmed using dynamic (moving) cameras, a setting where extrinsic parameters are rarely given, thus disqualifying methods that require camera parameters. Our algorithm is agnostic to the lack of camera parameters and attains good qualitative results.

In Figure 6 we show qualitative results of FLEX, compared to current non ep-free multi-view state-of-the-art [30], and to our monocular baseline [67]. Note that the method in [30] produces unnatural poses such as a huge leg in the first row and a backward-bent elbow in the last row.

**Multi-person captured by dynamic cameras** We evaluate our algorithm on a setting with dynamic cameras, with multi-person scenes introducing severe inter-person occlusions. Recall that the term *dynamic* refers to moving cameras that occasionally change their location and rotation. There are several multi-view datasets. Most of them are not fully dynamic: Human3.6M [29,9], CMU Panoptic [17] and TUM Shelf & Campus [3] contain static scenes only, while Tagging [79] and Ski-Pose PTZ-Camera [62] contain rotating cameras whose locations are fixed. KTH [34] is fully dynamic, but it is too blurry and does not provide ground-truth for all subjects. Despite its limitations, we use the KTH

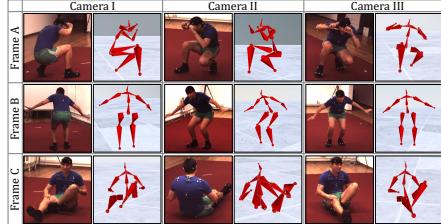


Fig. 4: Our results on videos from the Human3.6M dataset.

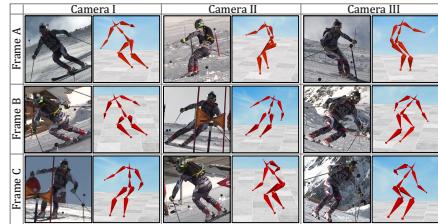


Fig. 5: Our results on videos from the Ski-Pose PTZ-Camera dataset.

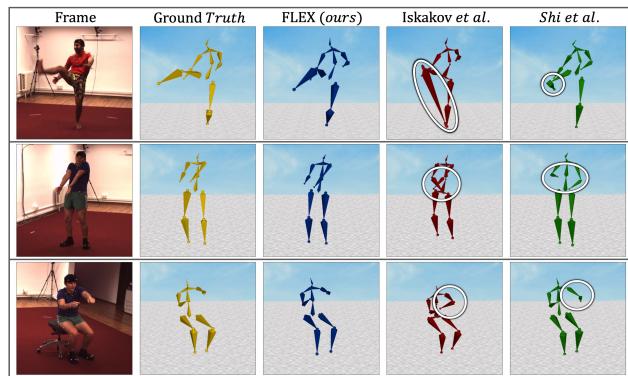


Fig. 6: Qualitative comparison of our work vs. non ep-free state-of-the-art (Iskakov *et al.* [30]) and vs. our single-view baseline (Shi *et al.* [67]).

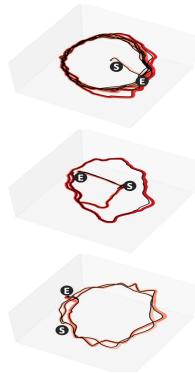


Fig. 7: Global root position. Ground-truth is in thin black.

dataset for qualitative analysis, but we cannot use it for thorough research. To mitigate the lack of a dynamic dataset, we generate synthetic videos using animated characters downloaded from Mixamo [1], an online dataset of character animation. Then, we generate video sequences of two interacting characters using Blender [18], which is a 3D creation suite. The newly created data is available on our project page. Our "synthetic studio" is illustrated at the appendix, where two interacting figures are video-filmed by multiple dynamic cameras. Using Blender, we obtain a rendered video stream from the view angle of each synthetic camera. Recall that the input to our algorithm is 2D joint locations, hence it is agnostic to the video appearance, and to whether the input image is real or synthetic.

The 2D backbone we use over the rendered video sequences is Alphapose [21], a state-of-the-art multi-person 2D pose estimator. Once obtaining the 2D joint locations, we use a naïve heuristic, which is not part of the suggested algorithm, to associate each detected person with its ID: for each frame, we associate the detected 2D pose with the one that is geometrically closest to it in the previous frame. In Figure 8 we depict qualitative results of two boxers. We emphasize several viewpoints where the 2D estimator attains large errors. Yet, FLEX com-

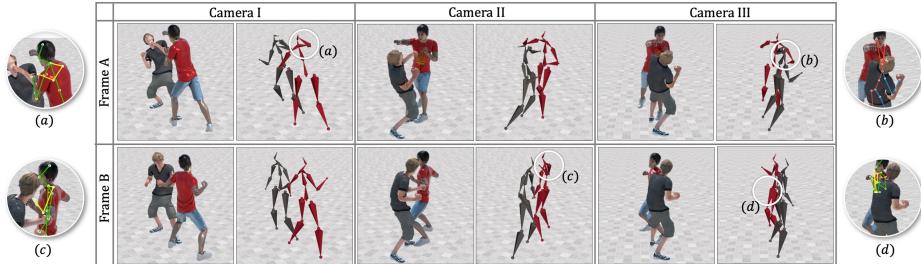


Fig. 8: Results on multi-person synthetic videos. In the zoomed-in circular images we depict 2D pose estimations, which are erroneous due to occlusion. A matching circle in the center rectangular image shows that our method reconstructs correct 3D motion although it takes inaccurate 2D joints for input.

Table 5: Ablation studies: The impact of (a) Number of views; (b) 2D backbone, and (c) Fusion method (refer to the sup. mat. for details regarding fusion).

(a)		(b)		(c)	
#Views	2D backbone	2D backbone	MPJPE	Method	MPJPE
GT	[30]	[8]	38.6	Averaged $K$ views	36.4
1	47.7	[12]	31.7	Late fusion	31.0
2	33.9	[30]	30.2	FLEX	<b>22.9</b>
3	26.3	GT	<b>22.9</b>		
4	<b>22.9</b>				
	<b>30.2</b>				

penses for these errors by fusing multi-view information. In the appendix we show additional characters and the predicted 2D pose for all the viewpoints.

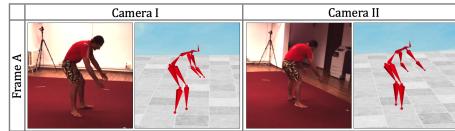
**Global position** In Figure 7 we draw the global position of the scaled predicted root joint along time. Ground-truth is depicted using a thin black curve, and our prediction is an overlay on top of it, changing from light to dark as time progresses. The start and the end of each trajectory are signaled by the letters S and E, respectively. Depicted motions are evaluated on the test set of Human3.6M, on the motions of walking, talking on the phone, and eating. Note that our predictions almost completely overlap the ground-truth curve. Recall we use weak perspective to bypass dependency on intrinsic parameters, resulting in up-to-scale global position accuracy. Quantitatively, our MPJPE on the H36M validation set is 118mm, outperforming Iskakov *et al.* [30] (perturbed by 3%) that attain 123mm. The other ep-free work [16] does not solve global locations.

**Ablation study** We evaluate the impact of different settings on the performance of FLEX using various ablation tests. Table 5 compares different multi-view fusion architectures. Note that using attention rather than convolution yields a 2mm improvement. The performance degrades with the transformer encoder due to its large number of parameters, which require more data for training than what is available in our case.

Table 5 measures MPJPE on Human3.6M in several studies. Table 5(a) studies a varying number of views, where the 2D pose is once given and once estimated. It confirms that a larger number of views induces more accurate results.

Note that the gap between the two columns decreases once the number of views increases. It shows that using several views compensates for the inaccuracy of estimated 2D poses. Table 5(b) compares 2D pose estimation backbones, and justifies our use of Iskakov *et al.* [30]. Finally, in Table 5(c) we explore two variations, both with ground-truth 2D inputs. The first variation runs FLEX as a monocular method ( $K=1$ ) and averages the monocular predictions. The second changes the fusion layers,  $F_S$  and  $F_Q$ , to use late fusion instead of an early one. We conclude that the configuration used by FLEX is better than both variations.

**Generalization** We exhibit generalization by training on one dataset and evaluating on a different, more challenging one. The train dataset is Human3.6M, and the evaluation ones are the KTH Football dataset, and the synthetic videos. For quantitative measurement, we train our model on two of the four cameras of the Human3.6M dataset. We test it using the other two cameras, on which the model has not been trained. We repeat this process for all possible camera pairs and obtain an average MPJPE of 148mm. Note that this error is not large compared to the human body size, and indeed we attain pleasing visual results as shown in the inset on the right.



## 5 Conclusions and limitations

We have presented FLEX, a multi-view method for motion reconstruction. It relies on a key understanding that 3D rotation angles and bone lengths are invariant to camera view, and their direct reconstruction spares the need for camera parameters. On a technical viewpoint, we presented a novel fusion layer with a multi-view convolutional layer and a multi-head attention mechanism that attends views.

One limitation of our approach is the dependency on 3D joint location ground-truth, and in particular, the requirement that it is given at the axis system of the train cameras. Another limitation is the dependency on the 2D backbone quality, and on the accuracy value associated with each joint. Lastly, being ep-free, the output 3D joint positions are only relative to the camera, lacking the transformation with respect to a global axis system.

In summary, FLEX is unique in fusing multi-view information to reconstruct motion and pose in dynamic photography environments. It is unaffected by settings in which the relative rotations between the cameras are unknown, and can maintain a high level of accuracy regardless. FLEX offers a simpler setting, where the correspondence and compatibility among the different views are rather lean, and thus more resilient to input errors and innate inaccuracies.

**Acknowledgments** This research would not have been possible without the exceptional support of Mingyi Shi. We are grateful to Kfir Aberman and Yuval Alaluf for reviewing earlier versions of the manuscript, and to Yuval Alaluf and

Shahaf Goren for contributing to FLEX’s video clip. This work was supported in part by the Israel Science Foundation (grants no. 2366/16 and 2492/20).

## References

1. Adobe Systems Inc.: Mixamo (2018), <http://www.mixamo.com>
2. Bachmann, R., Spörri, J., Fua, P., Rhodin, H.: Motion capture from pan-tilt cameras with unknown orientation. In: 2019 International Conference on 3D Vision (3DV). pp. 308–317. IEEE, IEEE Computer Society, Washington, DC, USA (2019)
3. Belagiannis, V., Amin, S., Andriluka, M., Schiele, B., Navab, N., Ilic, S.: 3d pictorial structures for multiple human pose estimation. In: 2014 IEEE Conference on Computer Vision and Pattern Recognition. pp. 1669–1676 (2014). <https://doi.org/10.1109/CVPR.2014.216>
4. Belagiannis, V., Amin, S., Andriluka, M., Schiele, B., Navab, N., Ilic, S.: 3d pictorial structures revisited: Multiple human pose estimation. *IEEE Transactions on Pattern Analysis and Machine Intelligence* **38**, 1929–1942 (10 2016). <https://doi.org/10.1109/TPAMI.2015.2509986>
5. Bergtholdt, M., Kappes, J., Schmidt, S., Schnörr, C.: A study of parts-based object class detection using complete graphs. *International Journal of Computer Vision* **87**, 93–117 (03 2010). <https://doi.org/10.1007/s11263-009-0209-1>
6. Bradski, G.: The OpenCV Library. Dr. Dobb’s Journal of Software Tools (2000)
7. Burenius, M., Sullivan, J., Carlsson, S.: 3d pictorial structures for multiple view articulated pose estimation. In: Proceedings / CVPR, IEEE Computer Society Conference on Computer Vision and Pattern Recognition. pp. 3618–3625. IEEE Computer Society, Washington, DC, USA (06 2013). <https://doi.org/10.1109/CVPR.2013.464>
8. Cao, Z., Hidalgo, G., Simon, T., Wei, S., Sheikh, Y.: Openpose: Realtime multi-person 2d pose estimation using part affinity fields. In: Proceedings of the 2018 IEEE Conference on Computer Vision and Pattern Recognition. CVPR ’18, vol. 43, pp. 172–186. IEEE Computer Society, Washington, DC, USA (2018)
9. Catalin Ionescu, Fuxin Li, C.S.: Latent structured models for human pose estimation. In: International Conference on Computer Vision (2011)
10. Chen, X., Lin, K.Y., Liu, W., Qian, C., Wang, X., Lin, L.: Weakly-supervised discovery of geometry-aware representation for 3d human pose estimation. 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) pp. 10887–10896 (2019)
11. Chen, X., Wei, P., Lin, L.: Deductive learning for weakly-supervised 3d human pose estimation via uncalibrated cameras. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 35, pp. 1089–1096 (2021)
12. Chen, Y., Wang, Z., Peng, Y., Zhang, Z., Yu, G., Sun, J.: Cascaded pyramid network for multi-person pose estimation. In: Proceedings of the IEEE conference on computer vision and pattern recognition. pp. 7103–7112. IEEE Computer Society, Washington, DC, USA (2018)
13. Cheng, Y., Yang, B., Wang, B., Tan, R.T.: 3d human pose estimation using spatio-temporal networks with explicit occlusion training. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 34, pp. 10631–10638 (2020)
14. Choi, H., Moon, G., Lee, K.M.: Beyond static features for temporally consistent 3d human pose and shape from a video. In: Conference on Computer Vision and Pattern Recognition (CVPR) (2021)

15. Chu, H., Lee, J.H., Lee, Y.C., Hsu, C.H., Li, J.D., Chen, C.S.: Part-aware measurement for robust multi-view multi-human 3d pose estimation and tracking. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) Workshops. pp. 1472–1481 (June 2021)
16. Chu, W.T., Pan, Z.W.: Semi-supervised 3d human pose estimation by jointly considering temporal and multiview information. *IEEE Access* **8**, 226974–226981 (2020). <https://doi.org/10.1109/ACCESS.2020.3045794>
17. CMU: Cmu graphics lab motion capture database (May 2019), <http://mocap.cs.cmu.edu>
18. Community, B.O.: Blender - a 3D modelling and rendering package. Blender Foundation, Stichting Blender Foundation, Amsterdam (2018), <http://www.blender.org>
19. Devlin, J., Chang, M.W., Lee, K., Toutanova, K.: Bert: Pre-training of deep bidirectional transformers for language understanding (2018). <https://doi.org/10.48550/ARXIV.1810.04805>, <https://arxiv.org/abs/1810.04805>
20. Dong, J., Jiang, W., Huang, Q., Bao, H., Zhou, X.: Fast and robust multi-person 3d pose estimation from multiple views. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 7792–7801 (2019)
21. Fang, H.S., Xie, S., Tai, Y.W., Lu, C.: Rmpe: Regional multi-person pose estimation. In: ICCV. p. 2334–2343. IEEE Computer Society, Washington, DC, USA (2017)
22. Fang, H.S., Xu, Y., Wang, W., Liu, X., Zhu, S.C.: Learning pose grammar to encode human body configuration for 3d pose estimation. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 32 (2018)
23. Habermann, M., Xu, W., Zollhofer, M., Pons-Moll, G., Theobalt, C.: Deepcap: Monocular human performance capture using weak supervision. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5052–5063 (2020)
24. Habibie, I., Xu, W., Mehta, D., Pons-Moll, G., Theobalt, C.: In the wild human pose estimation using explicit 2d features and intermediate 3d representations. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 10905–10914. IEEE Computer Society, Washington, DC, USA (2019)
25. He, Y., Yan, R., Fragkiadaki, K., Yu, S.I.: Epipolar transformers. 2020 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) pp. 7776–7785 (2020)
26. Hossain, M.R.I., Little, J.J.: Exploiting temporal information for 3d human pose estimation. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 68–84. Springer International Publishing, Berlin/Heidelberg, Germany (2018)
27. Hu, W., Zhang, C., Zhan, F., Zhang, L., Wong, T.T.: Conditional Directed Graph Convolution for 3D Human Pose Estimation, p. 602–611. Association for Computing Machinery, New York, NY, USA (2021), <https://doi.org/10.1145/3474085.3475219>
28. Huang, F., Zeng, A., Liu, M., Lai, Q., Xu, Q.: Deepfuse: An imu-aware network for real-time 3d human pose estimation from multi-view image. In: 2020 IEEE Winter Conference on Applications of Computer Vision (WACV). pp. 418–427. IEEE Computer Society, Los Alamitos, CA, USA (mar 2020). <https://doi.org/10.1109/WACV45572.2020.9093526>, <https://doi.ieeecomputersociety.org/10.1109/WACV45572.2020.9093526>

29. Ionescu, C., Papava, D., Olaru, V., Sminchisescu, C.: Human3.6m: Large scale datasets and predictive methods for 3d human sensing in natural environments. *IEEE Transactions on Pattern Analysis and Machine Intelligence* (2014)
30. Iskakov, K., Burkov, E., Lempitsky, V.S., Malkov, Y.: Learnable triangulation of human pose. 2019 IEEE/CVF International Conference on Computer Vision (ICCV) pp. 7717–7726 (2019)
31. Kadkhodamohammadi, A., Padoy, N.: A generalizable approach for multi-view 3d human pose regression. *Machine Vision and Applications* **32**(1), 1–14 (2021)
32. Kanazawa, A., Black, M.J., Jacobs, D.W., Malik, J.: End-to-end recovery of human shape and pose. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 7122–7131. CVPR ’18, IEEE Computer Society, Washington, DC, USA (2018). <https://doi.org/10.1109/CVPR.2018.00744>
33. Kanazawa, A., Zhang, J.Y., Felsen, P., Malik, J.: Learning 3d human dynamics from video. In: Computer Vision and Pattern Recognition (CVPR) (2019)
34. Kazemi, V., Burenius, M., Azizpour, H., Sullivan, J.: Multi-view body part recognition with random forests. In: BMVC 2013 - Electronic Proceedings of the British Machine Vision Conference 2013. BMVA, UK (09 2013). <https://doi.org/10.5244/C.27.48>
35. Khan, S., Naseer, M., Hayat, M., Zamir, S.W., Khan, F.S., Shah, M.: Transformers in vision: A survey. *ACM Computing Surveys* (2021). <https://doi.org/10.1145/3505244>
36. Kissos, I., Fritz, L., Goldman, M., Meir, O., Oks, E., Kliger, M.: Beyond weak perspective for monocular 3d human pose estimation. In: European Conference on Computer Vision. pp. 541–554. Springer (2020)
37. Kocabas, M., Athanasiou, N., Black, M.J.: Vibe: Video inference for human body pose and shape estimation. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5253–5263 (2020)
38. Kocabas, M., Karagoz, S., Akbas, E.: Self-supervised learning of 3d human pose using multi-view geometry. 2019 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) pp. 1077–1086 (2019)
39. Kolotouros, N., Pavlakos, G., Black, M.J., Daniilidis, K.: Learning to reconstruct 3d human pose and shape via model-fitting in the loop. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 2252–2261. ICCV’19, IEEE Computer Society, Washington, DC, USA (2019)
40. Li, S., Chan, A.: 3d human pose estimation from monocular images with deep convolutional neural network (11 2014). [https://doi.org/10.1007/978-3-319-16808-1\\_23](https://doi.org/10.1007/978-3-319-16808-1_23)
41. Li, W., Liu, H., Ding, R., Liu, M., Wang, P., Yang, W.: Exploiting temporal contexts with strided transformer for 3d human pose estimation (2021)
42. Lin, K., Wang, L., Liu, Z.: End-to-end human pose and mesh reconstruction with transformers. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 1954–1963 (2021)
43. Liu, D., Zhao, Z., Wang, X., Hu, Y., Zhang, L., Huang, T.: Improving 3d human pose estimation via 3d part affinity fields. In: 2019 IEEE Winter Conference on Applications of Computer Vision (WACV). pp. 1004–1013. IEEE, IEEE Computer Society, Washington, DC, USA (2019)
44. Liu, R., Shen, J., Wang, H., Chen, C., Cheung, S.c., Asari, V.: Attention mechanism exploits temporal contexts: Real-time 3d human pose reconstruction. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 5064–5073. IEEE Computer Society, Washington, DC, USA (2020)

45. Llopart, A.: Liftformer: 3d human pose estimation using attention models. CoRR **abs/2009.00348** (2020), <https://arxiv.org/abs/2009.00348>
46. Loper, M., Mahmood, N., Romero, J., Pons-Moll, G., Black, M.J.: SMPL: A skinned multi-person linear model. ACM Trans. Graphics (Proc. SIGGRAPH Asia) **34**(6), 248:1–248:16 (Oct 2015)
47. Luo, Z., Golestaneh, S.A., Kitani, K.M.: 3d human motion estimation via motion compression and refinement. In: Proceedings of the Asian Conference on Computer Vision (ACCV) (November 2020)
48. Ma, H., Chen, L., Kong, D., Wang, Z., Liu, X., Tang, H., Yan, X., Xie, Y., Lin, S.Y., Xie, X.: Transfusion: Cross-view fusion with transformer for 3d human pose estimation. In: British Machine Vision Conference (2021)
49. Mao, W., Liu, M., Salzmann, M., Li, H.: Learning trajectory dependencies for human motion prediction. In: Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV) (October 2019)
50. von Marcard, T., Henschel, R., Black, M., Rosenhahn, B., Pons-Moll, G.: Recovering accurate 3d human pose in the wild using imus and a moving camera. In: European Conference on Computer Vision (ECCV). pp. 601–617. Springer International Publishing, Berlin/Heidelberg, Germany (Sep 2018)
51. Martinez, J., Hossain, R., Romero, J., Little, J.J.: A simple yet effective baseline for 3d human pose estimation. In: Proceedings of the IEEE International Conference on Computer Vision. pp. 2640–2649 (2017)
52. Mehta, D., Sotnychenko, O., Mueller, F., Xu, W., Elgarib, M., Fua, P., Seidel, H.P., Rhodin, H., Pons-Moll, G., Theobalt, C.: Xnect: Real-time multi-person 3d motion capture with a single rgb camera. ACM Transactions on Graphics (TOG) **39**(4), 82–1 (2020)
53. Ohashi, T., Ikegami, Y., Yamamoto, K., Takano, W., Nakamura, Y.: Video motion capture from the part confidence maps of multi-camera images by spatiotemporal filtering using the human skeletal model. pp. 4226–4231 (10 2018). <https://doi.org/10.1109/IROS.2018.8593867>
54. Pavlakos, G., Malik, J., Kanazawa, A.: Human mesh recovery from multiple shots. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 1485–1495 (June 2022)
55. Pavlakos, G., Zhou, X., Daniilidis, K.: Ordinal depth supervision for 3d human pose estimation. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 7307–7316. IEEE Computer Society, Washington, DC, USA (2018)
56. Pavlakos, G., Zhou, X., Derpanis, K.G., Daniilidis, K.: Coarse-to-fine volumetric prediction for single-image 3d human pose. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 1263–1272. CVPR ’17, IEEE Computer Society, Washington, DC, USA (2017)
57. Pavllo, D., Feichtenhofer, C., Grangier, D., Auli, M.: 3d human pose estimation in video with temporal convolutions and semi-supervised training. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 7753–7762 (2019)
58. Pavllo, D., Grangier, D., Auli, M.: Quaternet: A quaternion-based recurrent model for human motion. In: British Machine Vision Conference (BMVC) (2018)
59. Qiu, H., Wang, C., Wang, J., Wang, N., Zeng, W.: Cross view fusion for 3d human pose estimation. 2019 IEEE/CVF International Conference on Computer Vision (ICCV) pp. 4341–4350 (2019)

60. Reddy, N., Guigues, L., Pischulini, L., Eledath, J., Narasimhan, S.G.: Tesseltrack: End-to-end learnable multi-person articulated 3d pose tracking. 2021 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) pp. 15185–15195 (2021)
61. Rhodin, H., Spörri, J., Katircioglu, I., Constantin, V., Meyer, F., Müller, E., Salzmann, M., Fua, P.V.: Learning monocular 3d human pose estimation from multi-view images. 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition pp. 8437–8446 (2018)
62. Rhodin, H., Spörri, J., Katircioglu, I., Constantin, V., Meyer, F., Müller, E., Salzmann, M., Fua, P.V.: Learning monocular 3d human pose estimation from multi-view images. 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition pp. 8437–8446 (2018)
63. Sarafianos, N., Boteanu, B., Ionescu, B., Kakadiaris, I.A.: 3d human pose estimation: A review of the literature and analysis of covariates. *Comput. Vis. Image Underst.* **152**(C), 1–20 (Nov 2016). <https://doi.org/10.1016/j.cviu.2016.09.002>
64. Sárándi, I., Linder, T., Arras, K.O., Leibe, B.: Metric-scale truncation-robust heatmaps for 3d human pose estimation. 2020 15th IEEE International Conference on Automatic Face and Gesture Recognition (FG 2020) pp. 407–414 (2020)
65. Schönberger, J.L., Frahm, J.M.: Structure-from-motion revisited. In: Conference on Computer Vision and Pattern Recognition (CVPR) (2016)
66. Shan, W., Lu, H., Wang, S., Zhang, X., Gao, W.: Improving robustness and accuracy via relative information encoding in 3d human pose estimation. Proceedings of the 29th ACM International Conference on Multimedia (2021)
67. Shi, M., Aberman, K., Aristidou, A., Komura, T., Lischinski, D., Cohen-Or, D., Chen, B.: Motionet: 3d human motion reconstruction from monocular video with skeleton consistency. *ACM Transactions on Graphics (TOG)* **40**(1), 1–15 (2020)
68. Shimada, S., Golyanik, V., Xu, W., Pérez, P., Theobalt, C.: Neural monocular 3d human motion capture with physical awareness. *ACM Trans. Graph.* **40**(4) (jul 2021). <https://doi.org/10.1145/3450626.3459825>, <https://doi.org/10.1145/3450626.3459825>
69. Skycam: <http://www.skycam.tv>
70. Sun, J., Wang, M., Zhao, X., Zhang, D.: Multi-view pose generator based on deep learning for monocular 3d human pose estimation (07 2020)
71. Sun, X., Xiao, B., Wei, F., Liang, S., Wei, Y.: Integral human pose regression. In: Proceedings of the European Conference on Computer Vision (ECCV). pp. 529–545 (2018)
72. Takahashi, K., Mikami, D., Isogawa, M., Kimata, H.: Human pose as calibration pattern: 3d human pose estimation with multiple unsynchronized and uncalibrated cameras. In: 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW). pp. 1856–18567 (2018). <https://doi.org/10.1109/CVPRW.2018.00230>
73. Tekin, B., Katircioglu, I., Salzmann, M., Lepetit, V., Fua, P.: Structured prediction of 3d human pose with deep neural networks (2016)
74. Tome, D., Toso, M., Agapito, L., Russell, C.: Rethinking pose in 3d: Multi-stage refinement and recovery for markerless motion capture. In: 2018 international conference on 3D vision (3DV). pp. 474–483. IEEE, IEEE Computer Society, Washington, DC, USA (2018)
75. Tu, H., Wang, C., Zeng, W.: Voxelpose: Towards multi-camera 3d human pose estimation in wild environment. In: ECCV (2020)

76. Usman, B., Tagliasacchi, A., Saenko, K., Sud, A.: Metapose: Fast 3d pose from multiple views without 3d supervision. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). pp. 6759–6770 (June 2022)
77. Vaswani, A., Shazeer, N., Parmar, N., Uszkoreit, J., Jones, L., Gomez, A.N., Kaiser, L., Polosukhin, I.: Attention is all you need (2017). <https://doi.org/10.48550/ARXIV.1706.03762>, <https://arxiv.org/abs/1706.03762>
78. Villegas, R., Yang, J., Ceylan, D., Lee, H.: Neural kinematic networks for unsupervised motion retargetting. In: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition. pp. 8639–8648. CVPR ’18, IEEE Computer Society, Washington, DC, USA (2018)
79. Vo, M.P., Yumer, E., Sunkavalli, K., Hadap, S., Sheikh, Y., Narasimhan, S.G.: Self-supervised multi-view person association and its applications. IEEE Transactions on Pattern Analysis and Machine Intelligence **43**, 2794–2808 (2021)
80. Wandt, B., Rudolph, M., Zell, P., Rhodin, H., Rosenhahn, B.: Canonpose: Self-supervised monocular 3d human pose estimation in the wild. In: Computer Vision and Pattern Recognition (CVPR) (Jun 2021)
81. Wang, D., Cui, X., Chen, X., Zou, Z., Shi, T., Salcudean, S., Wang, Z.J., Ward, R.: Multi-view 3d reconstruction with transformer. In: Proceeding of the IEEE International Conference on Computer Vision. pp. 5722–5731. ICCV ’21 (2021)
82. Wang, J., Yan, S., Xiong, Y., Lin, D.: Motion guided 3d pose estimation from videos. In: ECCV (2020)
83. Xiao, B., Wu, H., Wei, Y.: Simple baselines for human pose estimation and tracking. In: Proceedings of the European conference on computer vision (ECCV). pp. 466–481. Springer International Publishing, Berlin/Heidelberg, Germany (2018)
84. Yoshiyasu, Y., Sagawa, R., Ayusawa, K., Murai, A.: Skeleton transformer networks: 3d human pose and skinned mesh from single rgb image. In: Asian Conference on Computer Vision. pp. 485–500. Springer (2018)
85. Zhang, Z., Wang, C., Qin, W., Zeng, W.: Fusing wearable imus with multi-view images for human pose estimation: A geometric approach. In: CVPR (2020)
86. Zhou, X., Sun, X., Zhang, W., Liang, S., Wei, Y.: Deep kinematic pose regression. In: European Conference on Computer Vision. pp. 186–201. Springer (2016)
87. Zhu, L., Rematas, K., Curless, B., Seitz, S.M., Kemelmacher-Shlizerman, I.: Reconstructing nba players. In: European Conference on Computer Vision. pp. 177–194. Springer (2020)
88. Zins, P., Xu, Y., Boyer, E., Wuhrer, S., Tung, T.: Data-driven 3d reconstruction of dressed humans from sparse views. In: 3DV (2021)

## Appendix

This appendix provides further details for the main paper and is constructed in the following way. Appendix A describes how to access our model’s code, and Appendix B provides additional media, namely figures that have not been included in the main paper due to page limitation, and a description of video files that are available at our project page<sup>1</sup>. In Appendix C we describe an improvement in the skeleton structure, and in Appendix D we detail the internals of our architecture. Appendix E thoroughly describes the datasets and various data aspects of our work, and finally, Appendix F presents technical details related to camera parameters.

### A Code

Our code, together with usage instructions, is available on our project page<sup>1</sup>. The reader is encouraged to run the code and witness the reproducibility of our model.

### B Additional visualizations

On our project page<sup>1</sup>, the reader can find attached video files. The reader is encouraged to browse the video files in full-screen size. Here is their description:

- A clip describing our work: clip.mp4
- Video files showing our results on the Human3.6M dataset: Human36M\*.mp4
- Video files showing our results on the KTH multi-view Football II dataset: KTH\_football.mp4
- Video files comparing MotioNet (single-view) and Iskakov *et al.* [30] results versus ours: MotioNet\_comparison.mp4 and Iskakov\_comparison.mp4, respectively.

Note that we present only results that use input obtained by 2D estimation (as opposed to ground truth). Thus, our input is affected by occlusion and blur. Yet, we are able to mitigate the noisy input by exploiting multi-view data, in an ep-free fashion.

In Figure 9 we show how our algorithm is able to grasp fine details. The player’s left hand cannot be seen in the center view and is blurred in the left views. Yet, our model accurately reconstructs it.

In Figures 16 and 17, we show additional results on the Human3.6M and KTH Football multi-view II datasets. Each row depicts three views of one time frame. To the right of each image we place a reconstructed rig. Figures 18 to 20 are enlarged versions of the figures shown in the main paper.

Figure 10 shows our recording setup for creation of synthetic data. Note the depicted cameras, that dynamically move in the scene.

---

<sup>1</sup> Project page: <https://briang13.github.io/FLEX>



Fig. 9: Our algorithm is able to grasp fine details. The player’s left hand cannot be seen in the center view and is blurred in the left views. Yet, our model accurately reconstructs it.

In Figure 11 we depict qualitative results for a scene with two macarena dancers. We emphasize several viewpoints where the 2D backbone attains large errors. Yet, FLEX is able to compensate for these errors by fusing multi-view information. Figures 21 and 22 depict 2D joint locations estimated by the AlphaPose [21] backbone. A close look at these figures shows that many of the estimated locations are inaccurate, e.g., a hand of one subject is confused with the hand of the other subject. Even though the number of 2D errors is large, our algorithm is able to reconstruct the characters correctly.

## C Skeleton

To better reconstruct the motion in a given video stream, we modify the skeleton connectivity used in our baseline [67] (Figure 12). The root and neck joints of the baseline skeleton are both rigidly attached to the three bones neighboring each of them. This rigid connectivity constrains the skeleton, e.g., a motion where each shoulder moves forward and the neck moves to the right is impossible. In order to remove this constraint we add joints that overlap the root and the neck, hence enabling the neighboring bones to move independently of each other.

The new skeleton connectivity better matches the Human3.6M rotation angles ground-truth, thus, it better matches the way the dataset motions were captured. The new skeleton improves the mean per joint position error (MPJPE) both in the multi-view setting and the monocular case. The improvements are by  $\sim 4\text{mm}$  and  $\sim 6\text{mm}$  for monocular and four cameras setting, respectively.



Fig. 10: Our "synthetic studio" created using Blender [18] software with Mixamo [1] 3D characters. Two interacting characters are captured by multiple dynamic cameras and rendered into multiple video streams.

## D Architecture details

The architectural blocks in our implementation are the multi-view feature fusion layers  $F_S$  and  $F_Q$ , the two encoders,  $E_S$  and  $E_Q$ , a forward kinematics layer  $FK$  and a discriminator  $D$ . Our discriminator  $D$  is a linear component that contains two convolution layers and one fully connected layer. We base it on Kanazawa *et al.* [32]). The  $FK$  layer is based on Villegas *et al.* [78].

There are two novel building blocks contained in the new fusion layers,  $F_S$  and  $F_Q$ . The first is a multi-view convolutional layer; that is, a convolution that is aware of features stemming from multiple views as well as multiple frames. This convolutional layer is used in  $F_Q$  only. The second is a multi-head attention layer, used in both  $F_S$  and  $F_Q$ . A standard attention mechanism looks at the data as a *sequence of embeddings*. In our mechanism, the *views* form the sequence, and the *channels* form the embeddings. Our attention layer is based on the LiftFormer [45]. While the LiftFormer attends to time, we attend to views. The embedding size is 1024, and we use 64 heads (see Table 6), so for the Query, Key and Value (each), we have 64 learned linear filters of size  $K \times 16$ , where  $K$  is the number of views and 16 is the result of 1024/64.

A key architectural choice in our fusion layers,  $F_S$  and  $F_Q$ , is at which stage to fuse the input views. In Figure 13 we depict the conceptual idea of fusing. Each fusing scheme has its own advantages and disadvantages. Following the insight that early convolutional layers yield coarse features and late ones yield semantic features, we observe early fusion as generating all features (coarse and semantic) when a network is aware to all input branches, and observe middle fusion as first generating coarse features that are distinct for each branch, and only then

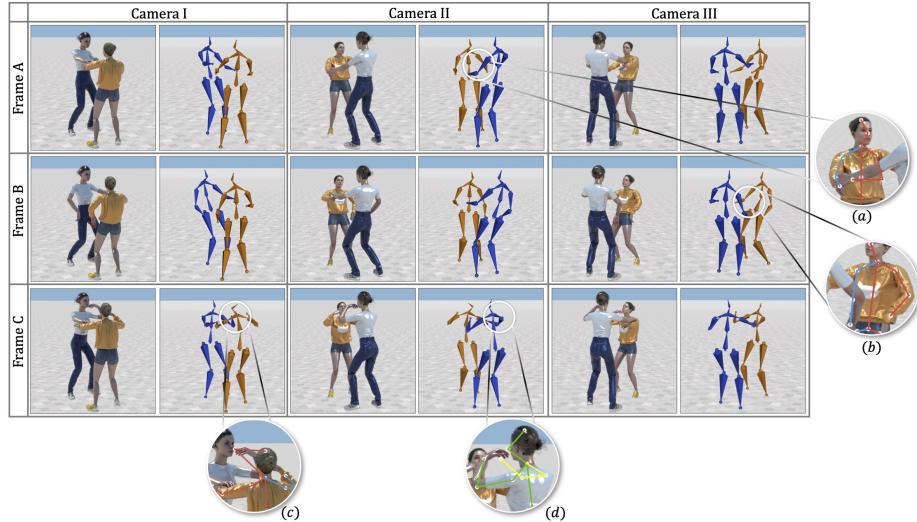


Fig. 11: Our results on multi-person synthetic videos, picturing two Macarena dancers. Some of the 2D joints, used as input to our method, are severely inaccurate. However, our method is able to reconstruct correct 3D motion. In the following examples, let *white dancer* and *orange dancer* denote the dancer wearing a white and an orange shirt respectively. Several 2D based skeleton error examples are depicted in the zoomed-in circular insets: (a) Wrong pose estimation of the left arm of the orange dancer; (b) The right arm of the orange dancer is occluded hence detected erroneously; (c) The nose tip of the orange dancer is erroneously detected as the nose tip of the white dancer; (d) Erroneous 2D pose estimation of the white dancer’s right hand.

fuse the coarse features together to generate common semantic features. When applying late fusion, the network creates distinct coarse and semantic features for each branch and only then fuses them together. During the development of our model, we have experimented with different fusion schemes, and found out that for our setting the early fusion works best.

Figure 14 depicts diagrams of the multi-view fusion layers and the encoders. The input to both fusion layers is  $\mathbf{V}_{s,q,r} \in \mathbb{R}^{T \times 3J \times K}$  (described in Section 3.1). In order to make the diagram more intuitive, we sketch  $V$  as  $K$  temporal sequences. Each temporal sequence is a 2D tensor, where channels are formed by the joints. The fusion layer first streams these temporal sequences through an expansion layer, increasing their channel size. Next, our fusion layer concatenates the expanded data and obtains a 3D tensor, on which it applies multi-view convolutional filters. These filters consider the data from all views. At the next stage we apply a multi-head attention layer that attends to views. Our network uses the attention layer output to create a 2D tensor representing one ‘fused’ view. The features are then passed to the encoder. The encoder block  $E_Q$  consists of three parallel 1D convolutional layers of different kernel sizes, followed by a final

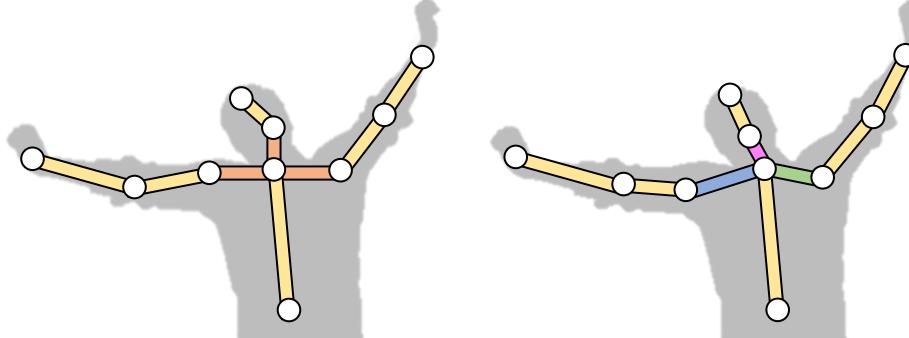


Fig. 12: Skeletal connectivity changes, demonstrated on the neck joint. Left: original connectivity, where shoulders and head are rigidly connected, yielding poor reconstruction. Right: new connectivity, with extra degrees of freedom.

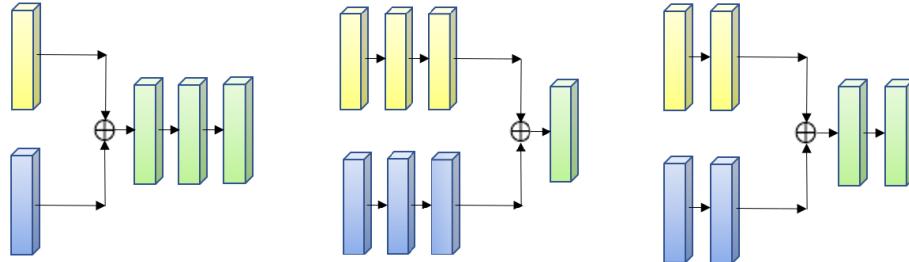


Fig. 13: Fusion schemes. Left: early fusion; Middle: late fusion; Right: middle fusion. Yellow and blue blocks symbolize features from distinct input branches, and green blocks represent fused data. Each block stands for a tensor of features.

additional 1D convolution. The encoder  $E_S$  starts with an adaptive max pooling to collapse the time dimension and then runs a final 1D convolution. After each convolution block, we apply batch normalization, a leaky rectified linear unit and dropout. Finally, we run a shrinking filter to decrease the number of channels to the desired output size. Table 6 describes the weight parameters of each layer.

In Figure 15 we zoom into the attention block (item (d) in Figure 14). Each slice of one temporal frame is separately forwarded through this block. Such a slice contains features from all the views. Within the attention block, we concatenate an additional view, which we call the *fusion view*. This additional view is a learned token [19], in which the attention mechanism combines significant information from all views. Our model attends to all views, including the added one. After exiting the attention block we omit the data related to the given views and keep the learned fusion view only. This fusion view is then concatenated with the outputs of the other temporal frames.

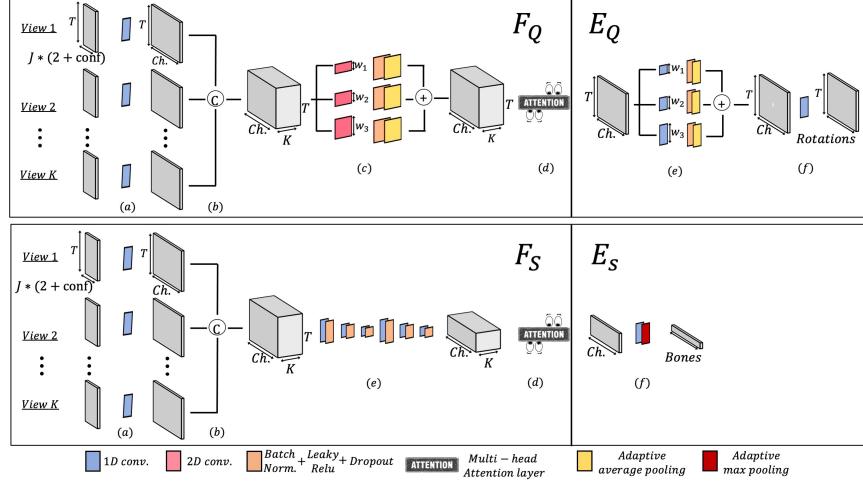


Fig. 14: Architecture in detail. The upper and lower parts are the rotations and bones branches, respectively. (a) Channel-wise expansion layer; (b) View concatenation; (c) Multi-view convolutional filters; (d) Cross-view attention layer (detailed in Figure 15); (e) Single-view convolutional filters; (f) Channel-wise shrinkage layer.

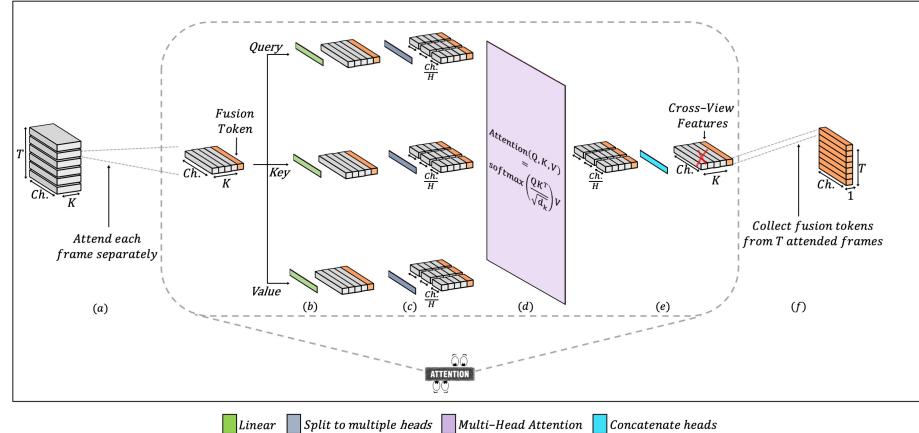


Fig. 15: Cross-view attention layer (Figure 14(d)) in detail. Our attention mechanism processes each frame separately, attending the multi-view features and fusing them to a single output per frame. (a) Process each temporal frame independently. Add a learned token [19] that forms a *fusion view*; (b) Linear layer; (c) Split the channels to  $H$  attention heads; (d) Multi-Head attention [77]; (e) Concatenate the attention heads; (f) Drop features from the original views. Collect fusion view features from all the frames in the temporal sequence.

Table 6: FLEX structure. **J** denotes the number of joints, **K** the number of views, and **L** denotes the number of limbs.  $k$  and  $s$  denote kernel width and the stride, respectively.  $\rightarrow$  denotes parallel convolutions while  $\downarrow$  denotes sequential ones.

Name	Layers	$k$	$s$	in / out
$F_Q$	1D-Conv + BatchNorm + LReLU + Dropout	1	1	$3J / 1024$
	$\rightarrow$ 2D-Conv + BatchNorm + LReLU + Dropout + Adap AP	5	3	$1024 / 1024$
	$\rightarrow$ 2D-Conv + BatchNorm + LReLU + Dropout + Adap AP	3	1	$1024 / 1024$
	$\rightarrow$ 2D-Conv + BatchNorm + LReLU + Dropout + Adap AP	1	1	$1024 / 1024$
	Multi-head Attention layer (64 heads)	—	—	$1024 / 1024$
$E_Q$	$\rightarrow$ 1D-Conv + BatchNorm + LReLU + Dropout + Adap AP	5	3	$1024 / 1024$
	$\rightarrow$ 1D-Conv + BatchNorm + LReLU + Dropout + Adap AP	3	1	$1024 / 1024$
	$\rightarrow$ 1D-Conv + BatchNorm + LReLU + Dropout + Adap AP	1	1	$1024 / 1024$
	1D-Conv	1	1	$1024/4(J-1)+4K$
$D$	1D-Conv + ReLU	1	1	$4J / 1024$
	$\downarrow$ 1D-Conv + ReLU + Adap AP	1	1	$1024 / 24J$
	Linear	—	—	$24J / J$
$F_S$	1D-Conv + BatchNorm + LReLU + Dropout	1	1	$J3 / 1024$
	1D-Conv + BatchNorm + LReLU + Dropout	5	1	$1024 / 1024$
	$\downarrow$ 1D-Conv + BatchNorm + LReLU + Dropout	3	1	$1024 / 1024$
	1D-Conv + BatchNorm + LReLU + Dropout	1	1	$1024 / 1024$
	1D-Conv + BatchNorm + LReLU + Dropout	5	1	$1024 / 1024$
	$\downarrow$ 1D-Conv + BatchNorm + LReLU + Dropout	3	1	$1024 / 1024$
	1D-Conv + BatchNorm + LReLU + Dropout	1	1	$1024 / 1024$
$E_S$	Multi-head Attention layer (64 heads)	—	—	$1024 / 1024$
	Adaptive MP	—	—	—
	1D-Conv	1	1	$1024 / L$

## E Data

### E.1 Train and Evaluation

We train our model on the Human3.6M dataset [29,9]. We evaluate FLEX on the Human3.6M and the KTH Multi-view Football II [34] datasets, and on synthetic multi-person video streams captured by dynamic cameras.

Human3.6M [29,9] is a dataset of 3.6 Million accurate 3D Human poses, acquired by recording the performance of 5 female and 6 male subjects, under 4 different viewpoints. This dataset holds a diverse set of motions and poses encountered as part of 17 typical human activities such as talking on the phone, walking, and eating. As recommended on the Human3.6M dataset page, we use subjects S1, S5, S6, S7, and S8 for training and subjects S9 and S11 for testing. This dataset grants licenses free of charge that are limited to academic use only. More information and access to raw data are provided on the dataset webpage<sup>1</sup>.

KTH Multi-view Football II [34] is a dataset of video streams from three synchronized cameras with 800-time frames per camera. The streams depict two different players (in separate streams), where each player has two sequences in varying levels of scene complexity. This dataset is unique in the sense that the cameras are dynamic, hence the approximation of camera extrinsic parameters is

<sup>1</sup> <https://vision.imar.ro/human3.6m/>

very challenging. We adjust the skeleton topology of the KTH dataset to match the topology of Human3.6M in the following way. KTH extracts 14 joints (top-head, mid-head, shoulders, hips, knees, feet, elbows, and hands). We create root (pelvis) and neck joints by averaging the hips and the shoulders respectively and then create a spine joint by averaging the root and the neck. Then we draw bones according to the Human3.6M skeleton topology. The raw data can be accessed and downloaded from the dataset webpage<sup>2</sup>. This dataset may only be used for academic research and not for commercial use.

Ski-Pose PTZ-Camera [62] is a 6 cameras' multi-view pant-tilt-zoom-camera (PTZ) dataset. It features competitive alpine skiers performing giant slalom runs. It holds 20K images, with a single skier in each. The cameras can rotate, but their locations are fixed. This dataset provides labels for the skiers' 3D locations in each frame, their projected 2D locations, and per-frame calibration of the PTZ cameras. In the dataset webpage<sup>3</sup> there are more descriptions of the dataset as well as download instructions.

Our synthetic videos are generated using Mixamo [1] and Blender [18]. We maintain two scenes with two interacting subjects in each. One scene illustrates a boxing arena, and one shows Macarena dancers. We create as many cameras as we wish, with full control on each camera's dynamic motion trajectory. Each synthetic camera outputs a video of the scene, taken from its view. To evaluate FLEX on these videos, we use a network that has been trained on the Human3.6M dataset, introducing satisfactory generalization abilities of our model.

## E.2 Input data

The input to our network is 2D joint locations per frame, accompanied by a confidence value. We train our network with several variations of input data.

**Ground truth 2D pose** Obviously, training with ground truth input data yields the best possible results. We use the 2D labeling provided by the Human3.6M dataset.

**Estimated 2D pose** To simulate dynamic capture environments, where 2D labels are not available, we use several state-of-the-art 2D pose estimators as 2D backbones. In our ablation studies we demonstrate the dependency on a good estimator. The estimators that we use are OpenPose [8], CPN [12], and the one used by Iskakov *et al.* [30] (who base their 2D estimation on the "simple baselines" architecture [83]). OpenPose and Iskakov *et al.* provide confidence values that we add to the network input. CPN does not provide these values, hence we assign identical confidence values for all joints when using it. While OpenPose and CPN are dedicated 2D pose estimators, Iskakov *et al.*'s 2D estimator is part of a 3D pose estimator. To extract the 2D pose we retrain their model using its given code and save intermediate values. The 2D pose estimation computed by Iskakov *et al.* [30] uses camera distortion parameters. In addition to being free of extrinsic camera parameters, we are strict about not using the intrinsic ones

---

<sup>2</sup> <https://www.csc.kth.se/cvap/cvg/?page=footballdataset2>

<sup>3</sup> <https://www.epfl.ch/labs/cvlab/data/ski-poseptz-dataset/>

as well (see Section 3 in the main paper); hence, we retrain Iskakov *et al.* [30] without those parameters.

Skeleton topology may vary between the aforementioned 3D datasets and 2D poses predicted by backbone algorithms. To mitigate this, we make adjustments to the predicted 2D joints. Openpose [8] extract 16 joints (root, neck, mid-head, top-head, shoulders, hips, knees, feet, elbows, and hands). These joints exist in the aforementioned datasets as well. In addition, a spine joint, which exists only in the 3D datasets, is artificially added (calculated as the 2D spatial average between the root and the neck joint). For the CPN [12] 2D prediction, we simply use the values computed by Pavllo *et al.* [57] and provided in their project page, which already possess the requested topology. Lastly, Iskakov *et al.* [30] predict the exact joints required by the aforementioned 3D datasets.

At inference time, when videos from the wild are used, we use a network that was trained using an estimated 2D pose and make sure that during inference, the exact 2D backbone that was used for training, is applied.

### E.3 Ground truth

During train time we use 3D joint location ground truth per view, plus rotation ground truth for the discriminator. In contrast to location ground truth, rotation ground truth is required only once, no matter how many views we have. During test time we need none of the above.

## F Camera Parameters

We next formulate the notion of camera parameters. Consider a pinhole camera model. Such a model possesses two types of parameters, extrinsic and intrinsic. *Extrinsic* parameters correspond to

- A rotation matrix  $R$ : a matrix of size  $3 \times 3$  characterizing the rotation from 3D real world axes into 3D camera axes.
- A translation vector  $T$ : a vector of size 3 representing the translational offset of the camera in the 3D scene.

*Intrinsic* parameters, stored in a  $3 \times 3$  matrix  $K$ , are specific to a camera.  $K$  consists of the focal length  $f_x, f_y$ , the camera optical center  $c_x, c_y$  and a skew coefficient  $s_k$ :

$$K = \begin{bmatrix} f_x & s_k & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix}. \quad (10)$$

We denote the mapping from 3D world coordinates into a 2D image plane by a  $3 \times 4$  matrix  $P$ .  $P$  is sometimes called *camera matrix* or *projection matrix*. To calculate  $P$ , both camera extrinsic and intrinsic parameters are used:

$$P = K \times [R \mid T]. \quad (11)$$

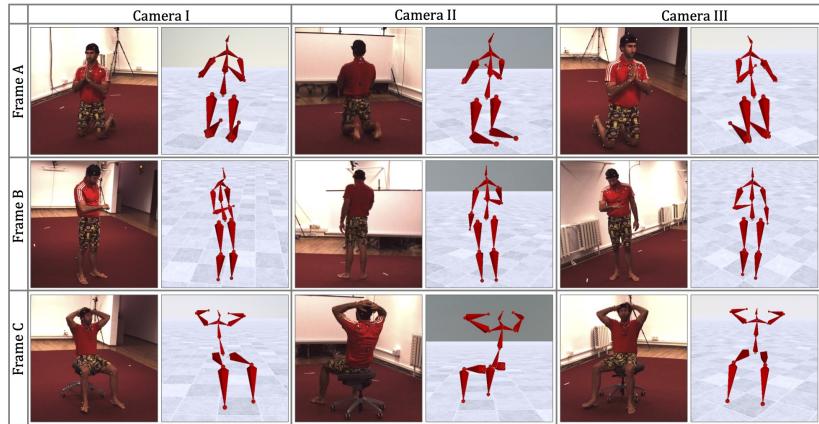


Fig. 16: Additional results on videos from the Human3.6M dataset.

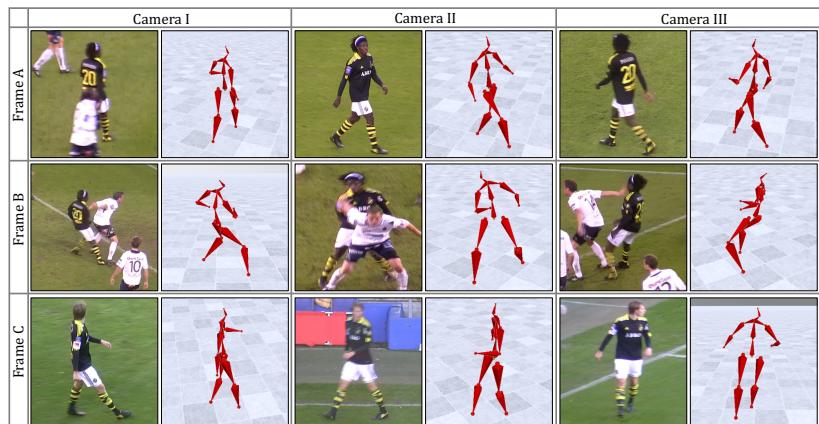


Fig. 17: Additional results on videos from the KTH Multi-view Football II dataset.

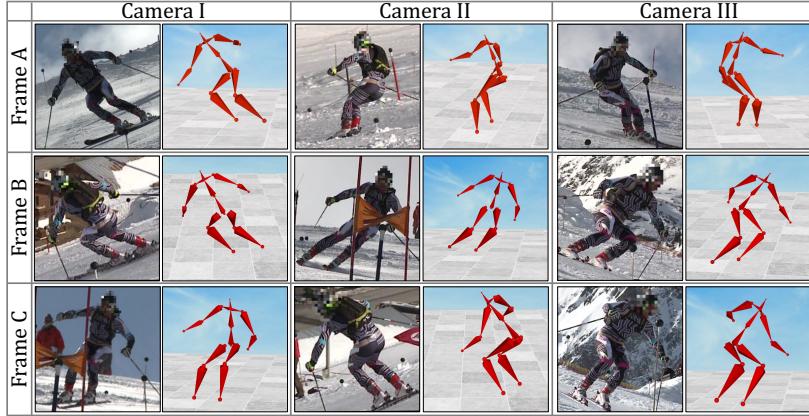


Fig. 18: Enlarged results on the Ski-Pose PTZ-Camera dataset (from main paper).

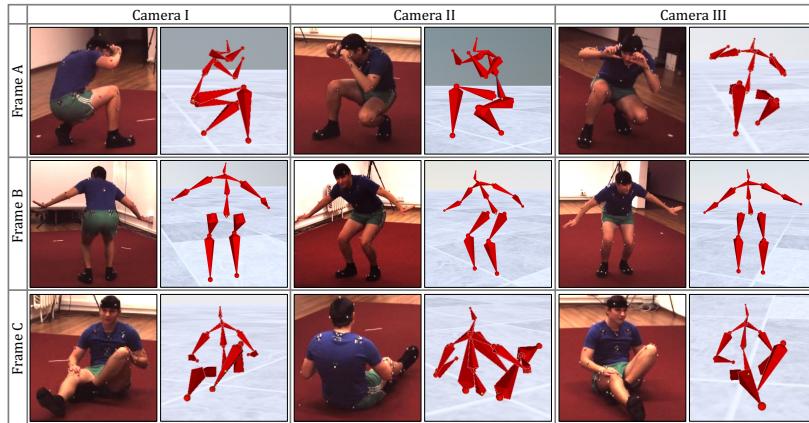


Fig. 19: Enlarged results on the Human3.6M dataset (from main paper).

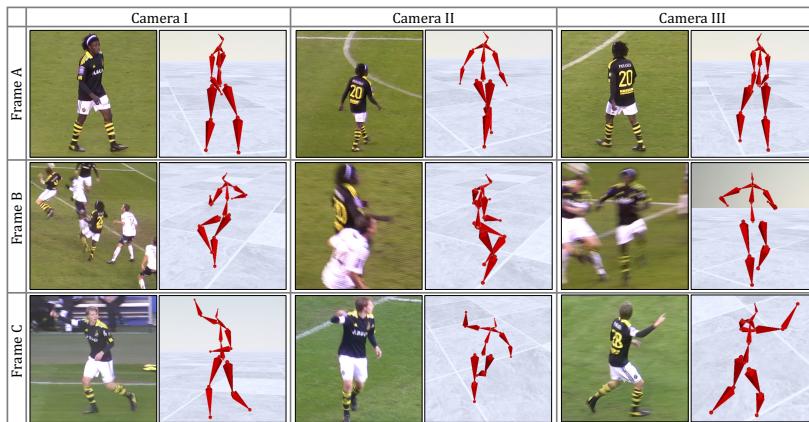


Fig. 20: Enlarged results on the KTH Football II dataset (from main paper).



Fig. 21: 2D joint locations estimated on a multi-person synthetic video of boxers.



Fig. 22: 2D joint locations estimated on a multi-person synthetic video of dancers.