FLEX: Parameter-free Multi-view 3D Human Motion Reconstruction

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

Tel-Aviv University, Israel

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, in particular, the relative positions among the cameras. Such dependency becomes a hurdle once shifting to dynamic capture in uncontrolled settings. We introduce FLEX (Free multi-view rEconstruction), an end-to-end parameter-free multi-view model. FLEX is parameter-free in the sense that it does not require any camera parameters, neither intrinsic nor extrinsic. 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 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 the Human3.6M and KTH Multi-view Football II datasets. We compare our model to state-of-the-art methods that are not parameter-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, video demonstration, and additional materials will be available on our project page.

1. Introduction

Human motion reconstruction is the fundamental task of associating a skeleton with temporally coherent joint locations and rotations. Acquiring accurate human motion in a controlled setting, using motion capture (mocap) 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 two or more video cameras may alleviate these problems since different views may compensate each other for occlusion and provide means for mutual consistency. Yet, exploiting the multi-view setup to improve reconstruction accuracy remains challenging.

In recent years, there has been a significant progress in utilizing data-driven and deep neural approaches for pose and motion reconstruction [36, 29, 44, 45, 39, 22, 35]. Most of these methods work in a monocular setting, but a progressively growing number of works learn a multi-view setting [16, 50, 41, 13, 10, 42]. These approaches, however, assume that the camera parameters are known, and in particular, their relative (extrinsic) positions.

In this paper, we introduce a parameter-free multi-view motion reconstruction algorithm. While camera parameters are often given in multi-view datasets, they are rarely given in dynamic capture environments.

Our approach is based on a key insight that joint rotations and bone lengths are identical for all views. That is, the 3D angle between the skeletal parts is invariant to the camera position. Thus, our method learns to reconstruct joint rotations rather than joint locations. To achieve this, we train a neural network to predict joint angles and bone lengths without using any of the camera parameters, neither

*equal contribution.

1Project page: https://briang13.github.io/FLEX/
in training nor test time. The information coming from multiple cameras is fused by a novel fusion layer that implicitly promotes joints detected by some cameras and demotes joints detected by others. Hence, increasing the confidence of each joint detection and mitigating occlusions.

Our model is an end-to-end deep convolutional network that leverages temporal information. The input is multi-view 2D joints that are either given or extracted using a 2D pose reconstruction technique. Specifically, we build upon MotioNet [45] as a monocular motion reconstruction backbone to learn bone lengths, global root location, foot contact labels, and a sequence of 3D joint rotations. Our network works over a whole video, embedding sequential information in its data features. This leads to temporal consistency and being agnostic to video lengths.

We evaluate our method qualitatively and quantitatively using the Human3.6M human pose [15] and the KTH Multi-view Football II [19] datasets. We compare our performance with state-of-the-art methods that are not parameter-free and show comparable results. To simulate a parameter-free setting, we perturb ground-truth camera parameters or alternately use prepossessing to estimate them. We show that in such a parameter-free setting, our model outperforms state-of-the-art methods by a large margin. We also show the advantage of our end-to-end solution in a setting where cameras are dynamic during data capture.

Our main contributions are twofold: (i) a network that reconstructs motion and pose in a multi-view setting with unknown camera parameters, and (ii) a novel fusion layer that contains two innovative elements: a multi-view convolutional layer and a transformer model that utilizes the multi-head attention mechanism over a number of views.

2. Related work

We divide this section into three parts, describing single view pose estimation, multi-view pose estimation, and motion reconstruction.

Pose Estimation using a Single View  Pose estimation has received significant interest in computer vision. Prior to the deep learning era, this task was approached using heuristics such as physical priors. Sarafianos et al. [43] provide a comprehensive review of algorithms of the pre-deep era. The emergence of deep learning and the availability of large datasets [15, 9, 52, 19], have led to significant advances in pose estimation. These methods can generally be divided into two groups. The first one infers 3D locations directly from images or videos [27, 37, 56, 48, 47, 7, 12]. The second group, aka lifting, applies two independent stages: (i) estimating 2D poses and (ii) lifting them into 3D space [34, 39, 11, 46, 29, 14]. Most methods in this group use state-of-the-art 2D pose predictors [6, 4] to obtain joint locations in 2D. The first group benefits from directly using the input images, which are more descriptive compared to 2D joint locations. The second group has the advantage of using an intermediate supervision.

Martinez et al. [34] show that given ground-truth 2D key-points, it is possible to achieve a satisfactory 3D pose prediction using simple convolutional networks. Pavllo et al. [39] extend this to show that using temporal information improves performance. More recently, transformers based techniques have been presented [31, 28, 30].

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 most 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 [1, 2, 3] estimate the input 2D pose from single images, while later works [10, 41, 16, 13, 23, 5, 8, 42, 17] 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 [16, 50] that present an end-to-end model.

To improve 2D pose estimation in a multi-view setting, several techniques have been explored. Qiu et al. [41] and He et al. [13] use camera parameters to find the matching epipolar lines between different views such that features gathered from several cameras are aggregated. In addition, He et al. [13] use camera parameters to generalize to new camera setups that are not part of the training data. Chen et al. [5] use an encoder-decoder method and learn a geometric representation in latent space.

Kocabas et al. [23] and Chu and Pan [8] apply self-supervision and show that reasonable results can be achieved even when 3D ground-truth is scarce. They project the predicted 3D joints (using real or estimated camera parameters) and expect consistency with 2D input joints. Kocabas et al. show a variation of their algorithm where camera parameters are estimated, and Chu and Pan introduce a network that estimates the camera parameters.

Current state-of-the-art results are attained by Iskakov et al. [16] and Tu et al. [50]. Both works use end-to-end networks, and both present a volumetric approach, where 2D features are un-projected from individual views to a common 3D space, using camera parameters. Lastly, Sun et al. [46] show that synthetic generation of additional views helps produce a more accurate lift to 3D space.

At inference time, some of the aforementioned works expect monocular inputs [46, 13, 5, 8] and some, including FLEX, expect multi-view inputs [16, 50, 41]. Both ap-
proaches have advantages. The first is applied on monocular data that is more common, and the second can achieve better results on multi-view settings.

**Rotation and Motion Reconstruction** Pose estimation may suffice for many applications, however, pose alone does not completely 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 temporally coherent 3D joint rotations. Many works explore the task of 3D shape recovery [32, 18, 24, 22, 21, 54], focusing on human mesh prediction along with joint rotations. They typically do not guarantee temporal coherence, e.g., bone length may vary across time frames.

Other works [40, 33] 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 our work, techniques that reconstruct human motion [55, 35, 45] focus on the temporal coherence of the predicted body, where the bone lengths are fixed over time and rotations are smooth.

### 3. 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 3D 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 camera parameters, during both training and test-time. More details on the camera parameters can be found in Appendix C.

Pose estimation methods can mitigate the lack of camera parameters by estimating them [8, 23]. Nevertheless, this leads to two drawbacks: (i) most algorithms need to do the estimation in a prepossessing step that breaks the end-to-end computation, and (ii) the estimated camera parameters are never exact and typically lead to a significant reduction in performance, as shall be shown in Section 4.

We take multi-view sequences of poses and estimate the motion of the observed human. Common methods estimate view-dependent 2D joint positions and then lift them to 3D by transforming them into a common space. Such lifting transformations require acquaintance of camera parameters. Our model directly predicts 3D rotations and bone lengths, where both values are agnostic to camera transformation. Thus, the predicted values are common to all camera views, which avoids the need for camera parameters information.

The terms pose estimation and motion estimation/reconstruction are used in various contexts in literature. To avoid confusion, we define 3D motion reconstruction as predicting the bone lengths associated with temporally coherent 3D joint rotations, and use the term reconstruction instead of estimation, which is commonly used to describe a transformation from one 2D image to another.

Our architecture, FLEX, builds upon MotioNet [45], a state-of-the-art single-view motion reconstruction method. FLEX 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 single, symmetric, 3D skeleton, represented by its bone lengths (lower branch in the figure) and (ii) a dynamic, skeleton-independent, sequence of: 3D joint rotations, global root positions and foot contact labels (upper branch in the figure). These two parts constitute a complete global description of the motion, requiring no further processing, or the use of inverse kinematics (IK), to obtain a full 3D animation.

Motion data, and in particular rotations rather than positions, are required in animation platforms and game engines. Our proposed algorithm directly outputs a kinematic skeleton, which is a complete, commonly used, motion representation. On the other hand, methods that predict pose only, 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 predict pose only 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). Below, we point out where it diverges from MotioNet. The inputs are $K$ synchronized video streams of $T$ frames each. For each video 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 agnos-
Our model takes multi-view temporal sequences of 2D poses accompanied by a confidence value per joint. Using two encoders, \( E_Q \) and \( E_S \), it extracts per-frame 3D joint rotations and foot contact labels, per-view and per-frame 3D root transformations, and a 3D static skeleton. Employing a discriminator, \( D \), it brings the temporal differences of rotation angles near the manifold of true rotations. Applying a forward kinematic layer, \( FK \), it extracts 3D joint positions from rotations and static features, which in turn are compared to the ground truth.

At train time, the output of both encoders is used in \( K \) identical forward kinematic (FK) layers. Each FK layer computes the estimated 3D joint positions related to one view. These 3D positions are compared to the ground truth during loss computation. In addition, we compute temporal differences over the rotations extracted out of \( E_Q \) and feed them to a discriminator \( D \), so they get near the manifold of true rotations in an adversarial way.

In addition to introducing fusion layers, FLEX changes the building blocks of MotioNet in the following way. Our input is an aggregation of all views, whereas MotioNet takes each view separately. The encoders, \( E_S \) and \( E_Q \), now take their input from the fusion layers, hence their architecture is slightly changed. Lastly, the overall output is changed such that it contains a root transformation per view in addition to the original outputs.

We now proceed into a more formal description. Throughout the description, we make an effort to match the notation used in MotioNet, up to points where the models diverge. Let \( L \) denote the number of limbs, \( T \) the temporal length of the sequence, \( J \) the number of joints, \( Q \) the size of the rotations representation vector, and \( K \) the number of cameras. Let \( P_{s,q,r,k} \in \mathbb{R}^{T\times 3J\times K} \) denote a temporal sequence of 3D joint positions generated by a skeleton \( s \in \mathbb{R}^J \) with joint rotations \( q \in \mathbb{R}^{T\times Q(J-1)} \), global root position and rotation \( r \in \mathbb{R}^{T\times (3+Q)} \) and a camera index \( k \in [1, \ldots, K] \). Note that \( q \) is related to all joints except for the root joint. The rotation of the root joint, as well its position, are related to \( r \). \( P \) can be thought of as \( K \) temporal sequences, one for each camera view.

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

$$\hat{s} = E_S(F_S(V_{s,q,r,k}))$$

and

$$\hat{q}, \hat{r}, \hat{f} = E_Q(F_Q(V_{s,q,r,k})).$$

These attributes can be then combined via forward kinematics to estimate the global 3D pose sequence for each view, $P_{s,q,r,k} \in \mathbb{R}^{T \times 3J}$, specified by joint positions.

The losses used in our network follow the ones in MioNet [45], hence we describe them only briefly and emphasize the modifications we made in order to accommodate a multi-view setting. One main difference is that all loss terms now hold an additional index for the view.

**Joint Position Loss** This is the main loss. It ensures that joints in the extracted poses are in their correct 3D positions:

$$L_p = \mathbb{E}_{P_{q,s,r,k} \sim P} \left[ \parallel FK(\hat{s}, \hat{q}) - P_{s,q,r=0,k} \parallel^2 \right].$$

**Skeleton Loss.** Let $P_{s,q,r,k} \in \mathcal{P}$ denote a 3D motion sequence in our dataset $\mathcal{P}$, and $V_{s,q,r,k}$ denote its 2D ground truth for all views. The skeleton loss $L_s$ encourages the encoder $E_S$ to correctly extracts the skeleton $s$:

$$L_s = \mathbb{E}_{P_{s,q,r,k} \sim \mathcal{P}} \left[ \parallel E_S(V_{s,q,r,k}) - s \parallel^2 \right].$$

**Adversarial Rotation Loss** Our network is trained to output rotations with natural velocities distribution using adversarial training, focusing on the temporal differences of joint rotations rather than their absolute values. We create $J$ discriminators, $D_j$, where each aims at the rotations of a specific joint:

$$L_{QGAN} = E_{Q \sim Q} \left[ \parallel D_j(\Delta q_j) \parallel^2 \right]$$

$$+ \mathbb{E}_{P_{q,s,r,k} \sim \mathcal{P}} \left[ \parallel 1 - D_j(\Delta t E_Q(V_{s,q,r,k})_{q,j}) \parallel^2 \right],$$

where $E_Q(\cdot)_{q,j}$ denotes the rotations of the $j$th joint of the output of $E_Q$, $Q$ represents the distribution of natural joint angles that are taken from the dataset, and $\Delta t$ stands for temporal differences.

**Global Root Position Loss** We estimate the depth parameter $Z_r$ by minimizing:

$$L_r = \mathbb{E}_{P_{q,s,r,k} \sim \mathcal{P}} \left[ \parallel E_Q(V_{s,q,r,k})_r - Z_r \parallel^2 \right],$$

where $E_Q(\cdot)_r$ denotes the root position part of the output of $E_Q$. Notice that $Z_{r,k}$ varies also per view and not only per frame.

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

$$L_F = \mathbb{E}_{P_{q,s,r,k} \sim \mathcal{P}} \left[ \parallel E_Q(V_{s,q,r,k})_f - f \parallel^2 \right],$$

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

$$L_{FC} = \mathbb{E}_{P_{q,s,r,k} \sim \mathcal{P}} \left[ \parallel f \sum_j \Delta t FK(\hat{s}, \hat{q})_f \parallel^2 \right],$$

where $FK(\cdot)_f \in \mathbb{R}^{T \times 3}$ and $f_j$ denote the positions and the contact labels of one of the feet joints ($i \in \{left, right\}$), and $\sum_j$ sums the components along the coordinate axis.

**Skeleton** To better reconstruct the motion in a given video stream, we modify the skeleton connectivity used in MioNet (see Figure 4).

In the original skeleton, the root and neck joints 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 extra 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 $\sim 4$mm and $\sim 6$mm in the monocular and four cameras setting, respectively.

**4. Experiments and evaluation**

We evaluate FLEX qualitatively and quantitatively using the Human3.6M human pose [15] and the KTH Multi-view Football II [19] datasets.
Implementation 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 $F_K$ and a discriminator $D$. The $F_K$ layer is based on Villeagas et al. [51], and the discriminator is as described in [45]. The encoders are based on the ones in [45]. 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 transformer encoder module, based on the LiftFormer [31] transformer encoder. While Llopart et al. [31] attend to time, we attend to views, and use only two encoder layers with two attention heads in each. The transformer is used in both $F_S$ and $F_Q$.

A key architectural choice in our fusion layers, $F_S$ and $F_Q$, is at which stage to fuse. In Figure 6 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 convolutional layers 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 fuse the coarse features together to generate common semantic features. As for late fusion, it lets the network create distinct coarse and semantic features for each branch and only then fuse 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 5 depicts diagrams of the multi-view fusion layers and the encoders. The input to both fusion layers is $V_{s,q,r,k} \in \mathbb{R}^{T \times 3J \times K}$ (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 together. At the next stage we apply a view transformer to shrink the data view-wise, to obtain a 2D tensor representing one ‘fused’ view.
The features are then passed to the encoder, that in turn runs convolutional filters, this time over the 2D fused tensor. Finally, we run a shrinking filter to decrease the number of channels to the desired output size. Preprocessing and augmentation procedures are identical to the ones in [45].

A detailed description of the datasets we use, a discussion about the 2D pose estimators, and a description of the ground truth we use, appears in Appendix B.

Quantitative results We show quantitative results using the mean per joint position error (MPJPE):

\[ E(\tilde{P}, P) = \frac{1}{J} \sum_{j=1}^{J} ||\tilde{P}_j - P_j||. \]  

(9)

We report protocol #1 MPJPE (that is, error relative to the pelvis), in millimeters.

Being the only parameter-free algorithm, we have no methods to compare to directly. However, algorithms can mitigate the lack of camera parameters by applying a prepossessing step to estimate them. In order to compare with the best, we have chosen the current state-of-the-art multi-view algorithm, Iskakov et al. [16]. In their work, Iskakov et al. use data that had undergone undistortion. We re-train their method with distorted data to imitate an environment where camera distortion parameters are unknown, and perturb intrinsic and extrinsic camera parameters by Gaussian noise with an extremely small standard deviation of 3% of each parameter’s absolute value. That is, for a camera parameter \( p \), we set \( \tilde{p} = X \) where \( X \sim \mathcal{N}(p, (0.03p)^2) \), and use \( \tilde{p} \) as the input camera parameters. We also report the equivalent for a standard deviation of 4%. In order to generate an equivalent comparison environment, we compare to [16] after using their own 2D pose estimation.

Table 1 presents a quantitative comparison of the MPJPE metric on the Human3.6M dataset. We start with results of monocular methods, proceed to result of multi-view methods that require camera parameters, and conclude with results of multi-view methods where camera parameters are not provided. We show that in such a parameter-free setting, 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 despite the fact that our method aims at a slightly different task, that is, motion reconstruction rather than pose estimation.

Qualitative results In the following figures we extract rigs, that is, frames from reconstructed animation videos, selecting challenging postures and scenes. Videos of the reconstructed motions are available on our project page., where the smoothness of motion and the naturalness of rotations can be observed. Our algorithm relies on 2D pose values estimated by the backbone in Iskakov et al. [16].

Figures 7 and 8 show scenes from the Human3.6M and KTH Multi-view Football II 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 enlarged for better visualization. Results on KTH are computed by using the model that was trained using the Human3.6M dataset, without re-training it over this one. Notice the occluded and blurry scenes. The KTH dataset is filmed using moving cameras, hence it is extremely challenging for other multi-view algorithms to reconstruct pose/motion over it without using the given camera parameters. Our algorithm is agnostic to the lack of camera parameters and attains good qualitative results.

Ablation study To check the impact of different settings on FLEX performance, we perform various ablation tests. In Table 2 we compare mean MPJPE over a varying number of views, where the 2D pose is once given and once estimated. Notice that the gap between these two cases decreases when the number of views increases.

Table 3 compares mean MPJPE over varying 2D estimation backbones. It justifies using [16] as 2D pose backbone. Finally, we measure the performance of our network using two variations. In the first, we run FLEX as a monocular method (\( K = 1 \)), average the monocular predictions, and show that it yields results that are inferior to fusing the views by our network. In the second variation, we change

![Figure 7. Our results on videos from the Human3.6M dataset.](image)

![Figure 8. Our results on videos from the more challenging KTH Multi-view Football II dataset. Notice how we mitigate occlusion and blurriness in the middle row.](image)
### Table 1. Protocol #1 MPJPE error on Human3.6M. Legend: (†) exploit temporal information, (+) extra training data. In blue - best result when camera parameters are not provided, in bold black - best result per method group.

| Method                        | Dir. | Disc. | Eat | Greet | Phone | Photo | Pose | Purch. | Sit | SitD. | Smoke | Wait | WalkD. | Walk | WalkT. | Mean |
|-------------------------------|------|-------|-----|-------|-------|-------|------|--------|-----|-------|--------|------|---------|------|--------|------|
| **Monocular methods**         |      |       |     |       |       |       |      |        |     |       |        |      |         |      |         |      |
| Lee et al. [25]†             | 43.8 | 51.7  | 48.8| 53.1  | 52.2  | 74.9  | 52.7 | 44.6   | 56.9| 74.3  | 56.7   | 66.4| 47.5    | 68.4 | 45.6   | 55.8 |
| Shi et al. [45]†             | 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 |
| Chen et al. [26]†            | 43.8 | 48.6  | 49.1| 49.8  | 57.6  | 61.5  | 45.9 | 48.3   | 62.0| 73.4  | 54.8   | 50.6| 56.0    | 43.4 | 45.5   | 52.7 |
| Pavllo et al. [35]†          | 45.2 | 46.7  | 43.3| 45.6  | 48.1  | 55.1  | 44.6 | 44.3   | 57.3| 65.8  | 47.1   | 44.0| 49.0    | 32.8 | 33.9   | 46.8 |
| Liu et al. [30]†             | 41.8 | 44.8  | 41.1| 44.9  | 47.4  | 54.1  | 43.4 | 42.2   | 56.2| 63.6  | 45.3   | 43.5| 45.3    | 31.3 | 32.2   | 45.1 |
| Llopart [31]†                | 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 |
| Cheng et al. [7]†            | 36.2 | 38.1  | 35.9| 38.2  | 35.9  | 47.4  | 36.8 | 42.0   | 45.9| 51.3  | 41.8   | 41.5| 43.8    | 33.1 | 28.6   | 40.1 |
| **Multi-view methods cam. param. provided** |      |       |     |       |       |       |      |        |     |       |        |      |         |      |         |      |
| Multi-View Martinez [49]      | 46.5 | 48.6  | 54.0| 51.5  | 67.5  | 70.7  | 48.5 | 49.1   | 69.8| 79.4  | 57.8   | 53.1| 56.7    | 42.2 | 45.4   | 57.0 |
| Pavlakos et al. [38]         | 41.2 | 49.2  | 42.8| 43.4  | 55.6  | 46.9  | 40.3 | 63.7   | 97.6| 119.0 | 52.1   | 42.7| 51.9    | 41.8 | 39.4   | 56.9 |
| Tome et al. [49] (+)          | 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  | 39.4 | 46.9  | 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 et al. [13]               | 25.7 | 27.7  | 23.7| 24.3  | 26.9  | 31.4  | 24.9 | 26.5   | 28.8| 31.7  | 28.2   | 26.3| 23.6    | 28.3 | 23.5   | 26.9 |
| Qiu et al. [41] (+)           | 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 |
| Iskakov et al. [16]          | 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 |

| **Multi-view methods cam. param. not provided** |      |       |     |       |       |       |      |        |     |       |        |      |         |      |         |      |
| Chu and Pan [8]†             | 49.1 | 65.6  | 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 et al. [16]          | 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 |
| cam. param. perturbed by 4%  | 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    | 28.4 | 33.1   |      |
| Iskakov et al. [16]          | 23.1 | 28.8  | 26.8| 28.1  | 31.6  | 37.1  | 25.7 | 31.4   | 36.5| 39.6  | 35.0   | 29.5| 35.6    | 26.8 | 26.4   | 30.9 |

Table 5. Protocol #1 MPJPE error on Human3.6M. Legend: (†) exploit temporal information, (+) extra training data. In blue - best result when camera parameters are not provided, in bold black - best result per method group.

### Table 2. Varying number of views.

| 2D from #Views ↓ | GT [16] |
|------------------|---------|
| 1                | 47.7    |
| 2                | 33.9    |
| 3                | 27.8    |
| 4                | 24.8    |

Table 2. Varying number of views.

### Table 3. Varying 2D pose backbones.

| 2D backbone       | MPJPE |
|-------------------|-------|
| Ground Truth      | 24.8  |
| OpenPose [4]      | 38.55 |
| CPN [6]           | 31.7  |
| Iskakov et al. [16] (based on [53]) | 30.9 |

Table 3. Varying 2D pose backbones.

### Table 4. The impact of various fusion methods. Note that processing all views together with an early fusion as performed in FLEX outperforms the other variations by a large margin.

| Method                       | MPJPE |
|------------------------------|-------|
| FLEX                         | 24.8  |
| Averaged K views             | 36.4  |
| Late fusion                  | 31.0  |

Table 4. The impact of various fusion methods. Note that processing all views together with an early fusion as performed in FLEX outperforms the other variations by a large margin.
6. Acknowledgments

This paper and the research behind it would not have been possible without the exceptional support of Mingyi Shi. His technical support, knowledge and insights were critical for this paper. We are immensely grateful to Kfir Aberman and Yuval Alauf for their comments on an earlier version of the manuscript. Yuval Alauf also contributed to the video clip that describes our paper. This work was supported in part by the Israel Science Foundation (grants no. 2366/16 and 2492/20).

References

[1] Vasileios Belagiannis, Sikandar Amin, Mykhaylo Andriluka, Bernt Schiele, Nassir Navab, and Slobodan Ilic. 3d pictorial structures revisited: Multiple human pose estimation. IEEE Transactions on Pattern Analysis and Machine Intelligence, 38:1929–1942, 10 2016. 2
[2] Martin Bergtholdt, Joerg Kappes, Stefan Schmidt, and Christoph Schnörr. A study of parts-based object class detection using complete graphs. International Journal of Computer Vision, 87:93–117, 03 2010. 2
[3] Magnus Burenius, Josephine Sullivan, and Stefan Carlsson. 3d pictorial structures for multiple view articulated pose estimation. In Proceedings / CVPR, IEEE Computer Society Conference on Computer Vision and Pattern Recognition, IEEE Computer Society Conference on Computer Vision and Pattern Recognition, pages 3618–3625, 06 2013. 2
[4] Zhe Cao, Gines Hidalgo, Tomas Simon, Shih-En Wei, and Yaser Sheikh. 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, Washington, DC, USA, 2018. IEEE Computer Society. 2, 8, 11
[5] Xipeng Chen, Kwan-Yee Lin, Wentao Liu, Chen Qian, Xi-aogang Wang, and Liang Lin. Weakly-supervised discovery of geometry-aware representation for 3d human pose estimation, 2019. 2
[6] Yilun Chen, Zhicheng Wang, Yuxiang Peng, Zhiqiang Zhang, Gang Yu, and Jian Sun. Cascaded pyramid network for multi-person pose estimation. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 7103–7112, 2018. 2, 8, 11
[7] Yu Cheng, Bo Yang, Bo Wang, and Robby T. Tan. 3d human pose estimation using spatio-temporal networks with explicit occlusion training. 2020. 2, 8
[8] W. T. Chu and Z. W. Pan. Semi-supervised 3d human pose estimation by jointly considering temporal and multiview information. IEEE Access, 8:226974–226981, 2020. 2, 3, 8
[9] CMU. Cmu graphics lab motion capture database, May 2019. 2
[10] Junting Dong, Wen Jiang, Qixing Huang, Hujun Bao, and Xiaowei Zhou. Fast and robust multi-person 3d pose estimation from multiple views, 2019. 1, 2
[11] Haoshu Fang, Yuanlu Xu, Wenguan Wang, Xiaobai Liu, and Song-Chun Zhu. Learning pose grammar to encode human body configuration for 3d pose estimation, 2018. 2
[12] Ilkhsanul Habibie, Weipeng Xu, Dushyant Mehta, Gerard Pons-Moll, and Christian Theobalt. In the wild human pose estimation using explicit 2d features and intermediate 3d representations. arXiv preprint arXiv:1904.03289, 2019. 2
[13] Yihui He, Rui Yan, Katerina Fragkiadaki, and Shou-I Yu. Epipolar transformers, 2020. 1, 2, 8
[14] Mir Rayat Imtiaz Hossain and James J. Little. Exploiting temporal information for 3d human pose estimation. Lecture Notes in Computer Science, page 69–86, 2018. 2
[15] Catalin Ionescu, Dragos Papava, Vlad Olaru, and Cristian Sminchisescu. Human3.6m: Large scale datasets and predictive methods for 3d human sensing in natural environments. IEEE Transactions on Pattern Analysis and Machine Intelligence, 2014. 2, 5, 11
[16] Karim Iskakov, Egor Burkov, Victor Lempiäski, and Yury Malkov. Learnable triangulation of human pose, 2019. 1, 2, 7, 8, 11
[17] Abdolrahim Kadkhodamohammadi and Nicolas Padoy. A generalizable approach for multi-view 3d human pose regression, 2019. 2, 8
[18] Angjoo Kanazawa, Michael J. Black, David W. Jacobs, and Jitendra Malik. End-to-end recovery of human shape and pose. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, CVPR ’18, pages 7122–7131, Washington, DC, USA, 2018. IEEE Computer Society. 3
[19] Vahid Kazemi, Magnus Burenius, Hossein Azizpour, and Josephine Sullivan. Multi-view body part recognition with random forests. In BMVC 2013 - Electronic Proceedings of the British Machine Vision Conference 2013, 09 2013. 2, 5, 11
[20] Salman Khan, Muzammal Naseer, Munawar Hayat, Syed Waqas Amir, Fahad Shahbaz Khan, and Mubarak Shah. Transformers in vision: A survey. arXiv preprint arXiv:2101.01169, 2021. 4
[21] Imry Kissos, Lior Fritz, Matan Goldman, Eguard Oks, and Mark Kliger. Beyond weak perspective for monocular 3d human pose estimation, 2020. 3
[22] Muhammed Kocabas, Nikos Athanasiou, and Michael J. Black. Vibe: Video inference for human body pose and shape estimation, 2020. 1, 3
[23] Muhammed Kocabas, Salih Karagoz, and Emre Akbas. Self-supervised learning of 3d human pose using multi-view geometry, 2020. 1, 3
[24] Nikos Kolotouros, Georgios Pavlakos, Michael J. Black, and Kostas Daniilidis. Learning to reconstruct 3d human pose and shape via model-fitting in the loop. In Proceedings of the IEEE International Conference on Computer Vision, ICCV’19, 2019. 3
[25] Kyoungho Lee, Inwoong Lee, and Sanghoon Lee. Propagating lstm: 3d pose estimation based on joint interdependency. In Proceedings of the European Conference on Computer Vision, ECCV ’18, pages 123–141. Springer International Publishing, 2018. 8
[26] Chen Li and Gim Hee Lee. Generating multiple hypotheses for 3d human pose estimation with mixture density network. arXiv preprint arXiv:1904.05347, 2019. 8
[27] Sijin Li and Antoni Chan. 3d human pose estimation from monocular images with deep convolutional neural network, 11 2014. 2
[28] Kevin Lin, Lijuan Wang, and Zicheng Liu. End-to-end human pose and mesh reconstruction with transformers, 2020.

[29] Ding Liu, Zixu Zhao, Xinchao Wang, Yuxiao Hu, Lei Zhang, and Thomas Huang. Improving 3d human pose estimation via 3d part affinity fields. In 2019 IEEE Winter Conference on Applications of Computer Vision (WACV), pages 1004–1013. IEEE, 2019.

[30] Ruixu Liu, Ju Shen, He Wang, Chen Chen, Sen-ching Cheung, and Vijayan Asari. Attention mechanism exploits temporal contexts: Real-time 3d human pose reconstruction. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition, pages 5064–5073, 2020.

[31] Adrian Llopart. Liftformer: 3d human pose estimation using attention models, 2020.

[32] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J. Black. SMPL: A skinned multi-person linear model. ACM Trans. Graphics (Proc. SIGGRAPH Asia), 34(6):248:1–248:16, Oct. 2015.

[33] Wei Mao, Miaomiao Liu, Mathieu Salzmann, and Hongdong Li. Learning trajectory dependencies for human motion prediction, 2020.

[34] Julieta Martinez, Rayat Hossain, Javier Romero, and Matthew Loper. Neural kinematic networks for unsupervised motion retargetting. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, CVPR ’17, pages 1263–1272, Washington, DC, USA, 2017. IEEE Computer Society.

[35] Georgios Pavlakos, Xiaowei Zhou, and Konstantinos Daniilidis. Harvesting multiple views for marker-less 3d human pose annotations. In Proceedings of the IEEE conference on computer vision and pattern recognition, pages 6988–6997, 2017.

[36] Dario Pavlo, Christoph Feichtenhofer, David Grangier, and Michael Auli. 3d human pose estimation in video with temporal convolutions and semi-supervised training, 2019.

[37] Dario Pavlo, David Grangier, and Michael Auli. Quaternionet: A quaternion-based recurrent model for human motion, 2018.

[38] Haibo Qiu, Chunyu Wang, Jingdong Wang, Naiyan Wang, and Wenjun Zeng. Cross view fusion for 3d human pose estimation, 2019.

[39] Helge Rhodin, Jörg Spörri, Isinsu Katircioglu, Victor Constantin, Frédéric Meyer, Erich Müller, Mathieu Salzmann, and Pascal Fua. Learning monocular 3d human pose estimation from multi-view images, 2018.

[40] Nikolaos Sarafianos, Bogdan Boteanu, Bogdan Ionescu, and Ioannis A. Kakadiaris. 3d human pose estimation: A review of the literature and analysis of covariates. Comput. Vis. Image Underst., 152(C):1–20, Nov. 2016.

[41] Ivstván Sárándi, Timm Linder, Kai O Arras, and Bastian Leibe. Metric-scale truncation-robust heatmaps for 3d human pose estimation. arXiv preprint arXiv:2003.02953, 2020.

[42] Mingyi Shi, Kfir Aberman, Andreas Arisidou, Taku Kornura, Dani Lischinski, Daniel Cohen-Or, and Baoquan Chen. Motionet: 3d human motion reconstruction from monocular video with skeleton consistency. ACM Transactions on Graphics (TOG), 40(1):1–15, 2020.

[43] Jun Sun, Mantao Wang, Xin Zhao, and Dejun Zhang. Multi-view pose generator based on deep learning for monocular 3d human pose estimation, 2020.

[44] Xiao Sun, Bin Xiao, Fangyi Wei, Shuang Liang, and Yichen Wei. Integral human pose regression, 2018.

[45] Bagra Tekin, Isinsu Katircioglu, Mathieu Salzmann, Vincent Lepetit, and Pascal Fua. Structured prediction of 3d human pose with deep neural networks, 2016.

[46] Denis Tome, Matteo Tos, Lourdes Agapito, and Chris Russell. Rethinking pose in 3d: Multi-stage refinement and recovery for markerless motion capture. In 2018 international conference on 3d vision (3DV), pages 474–483. IEEE, 2018.

[47] Hanyue Tu, Chunyu Wang, and Wenjun Zeng. Voxelpose: Towards multi-camera 3d human pose estimation in wild environment, 2020.

[48] Ruben Villegas, Jimei Yang, Duygu Ceylan, and Honglak Lee. Neural kinematic networks for unsupervised motion retargetting. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, CVPR ’18, pages 8639–8648, Washington, DC, USA, 2018. IEEE Computer Society.

[49] Timo von Marcard, Roberto Henschel, Michael Black, Bodo Rosenhahn, and Gerard Pons-Moll. Recovering accurate 3d human pose in the wild using imus and a moving camera. In European Conference on Computer Vision (ECCV), sep 2018.

[50] Bing Xiao, Haiping Wu, and Yichen Wei. Simple baselines for human pose estimation and tracking. In Proceedings of the European conference on computer vision (ECCV), pages 466–481, 2018.

[51] Yusuke Yoshiyasu, Ryusuke Sagawa, Ko Ayusawa, and Akihiko Murai. Skeleton transformer networks: 3d human pose estimation, 2020.

[52] Luay Zhu, Konstantinos Rematas, Brian Curless, Steve Seitz, and Ira Kemelmacher-Shlizerman. Reconstructing nba players, 2020.
A. Media

On our project page\(^2\), the reader can find attached video files. The reader is encouraged to enlarge them to full screen size. Here is their description:

- A clip describing our work: clip.mov
- Video files showing our results on the Human3.6M dataset: Human36M*.mov
- Video files showing our results on the KTH multi-view Football II dataset: KTH_football.mov
- Video files comparing MotioNet (single-view) results versus ours: MotioNet_comparation.mov. Notice that while MotioNet is occasionally wrong, our model is accurate and smooth.

Notice that we introduce 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 a parameter-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 10 and 11, 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 12 and 13 are enlarged versions of the figures shown in the main document.

B. More implementation details

B.1. Datasets

We train our model on the Human3.6M human pose dataset [15]. Our evaluation is done on the Human3.6M and the KTH Multi-view Football II [19] datasets. Human3.6M 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.

KTH Multi-view Football II 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 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 by averaging the hips and the shoulders accordingly and then create a spine by averaging the root and the neck. Then we draw bones according to the Human3.6M skeleton topology.

B.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 the main paper, we demonstrate the dependency on a good estimator. The estimators that we use are OpenPose [4], CPN [6], and Iskakov et al. [16] (who base their 2D estimation on the ”simple baselines” architecture [53]). OpenPose and Iskakov et al. provide confidence values that we add to the network input. CPN does not provide this value and thus, we assign identical confidence values for all joints when using it. While OpenPose and CPN are dedicated 2D pose estimators, Iskakov et al.’s is part of a 3D pose estimator. To extract the 2D pose we retrain it using its given code and save intermediate values. The 2D labeling computed by Iskakov et al. [16] uses camera distortion parameters; hence, we retrain it without those parameters to adhere to our parameter-free approach.

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 [4] 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 [6] 2D prediction, we simply use the values computed by [39] and provided in their project page, which already possesses the requested topology. Lastly, Iskakov et al. [16] predicts 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

---

\(^2\)Project page: https://briang13.github.io/FLEX/
pose and make sure that during inference, the exact 2D backbone that was used for training, is applied.

B.3. Ground truth

During train time we use 3D joint location ground truth per view, plus rotation ground truth if we choose to activate 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.

C. Camera Parameters

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

- Rotation matrix - $R$: a matrix of size $3 \times 3$ characterizing the rotation from 3D real world axes into 3D camera axes.

- Translation vector - $T$: a vector of size $3 \times 1$ 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. $K$ possesses 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:

$$P = K \times [R \mid T]. \quad (11)$$
Figure 10. Additional results on videos from the Human3.6M dataset.

Figure 11. Additional results on videos from the KTH Multi-view Football II dataset.
Figure 12. Results on the Human3.6M dataset. Also shown (in small scale) in the main paper.

Figure 13. Results on the KTH Multi-view Football II dataset. Also shown (in small scale) in the main paper.