ReFit: Recurrent Fitting Network for 3D Human Recovery

Yufu Wang    Kostas Daniilidis
University of Pennsylvania

Abstract

We present Recurrent Fitting (ReFit), a neural network architecture for single-image, parametric 3D human reconstruction. ReFit learns a feedback-update loop that mirrors the strategy of solving an inverse problem through optimization. At each iterative step, it reprojects keypoints from the human model to feature maps to query feedback, and uses a recurrent-based updater to adjust the model to fit the image better. Because ReFit encodes strong knowledge of the inverse problem, it is faster to train than previous regression models. At the same time, ReFit improves state-of-the-art performance on standard benchmarks. Moreover, ReFit applies to other optimization settings, such as multi-view fitting and single-view shape fitting. Project website: https://yufu-wang.github.io/refit_humans/

1. Introduction

Single-view 3D human reconstruction has transformed the way we analyze and create virtual content. Progress has primarily been driven by the use of parametric human models [43], powerful neural networks [30], and high-quality annotated data [61]. However, the best systems to date still struggle with various difficulties, including occlusion, uncommon poses, and diverse shape variations.

The traditional approach fits a parametric human model to an image using handcrafted objectives and energy minimization techniques [9]. However, the objective function computes the L2 distance between the reprojected and the detected keypoints [9]. In ReFit, each reprojected keypoint queries a window on a feature map. A window encodes first-order information, such as 2D flow, which is the derivative of the L2 objective. The queried features are fed to the update module to compute updates for the human model.

Another key element of ReFit is the disentangled and recurrent update streams. Iterative regression methods commonly map a feature vector to parameter updates with an MLP [30,67]. Therefore, the update rules for the parameters are entangled until the last linear layer. In ReFit, each reprojected keypoint queries a window on a feature map. A window encodes first-order information, such as 2D flow, which is the derivative of the L2 objective. The queried features are fed to the update module to compute updates for the human model.

ReFit has three main steps. (1) A backbone network extracts pixel-aligned image features. (2) A feedback module queries feedback features from keypoint reprojection. (3) A recurrent update module uses a set of disentangled GRUs to update the body. Feedback and updates are repeated until the body mesh is well-aligned with the image (Fig. 1).

The design of ReFit mirrors traditional model fitting. In model fitting, the objective function computes the L2 distance between the reprojected and the detected keypoints [9]. In ReFit, each reprojected keypoint queries a window on a feature map. A window encodes first-order information, such as 2D flow, which is the derivative of the L2 objective. The queried features are fed to the update module to compute updates for the human model.

In this paper, we propose Recurrent Fitting (ReFit), an architecture for 3D human reconstruction. ReFit mimics the structure of model fitting, reducing the regression problem to a learning-to-optimize problem. This allows ReFit to learn faster than other architectures and improves state-of-the-art accuracy.
the update function, we hypothesize that some degree of disentanglement is beneficial.

To this end, ReFit’s update module uses one GRU per parameter group to disentangle the parametric update. Each GRU learns the prior and the update rule for one parameter group, e.g., the rotation of a single joint. GRUs utilize memory from the previous iterations, analogous to a first-order method with momentum [16]. At each iteration, the GRUs can take different step sizes. In the example of right-hand misalignment, the GRUs for the right side of the body can output adjustments, while the GRUs for the left side can return little or no updates. This disentanglement leads to better performance. The disentangled GRUs can be implemented as batch matrix operations in PyTorch, allowing for efficient training and inference.

Moreover, ReFit acts as a learned optimizer that applies to other settings of model fitting. Using ReFit as a learned optimizer, we propose a multi-view fitting procedure that significantly outperforms regression on separate views. We also demonstrate using ReFit to register a pre-scanned shape to images of the same subject, a helpful setting for outdoor motion capture.

The ReFit architecture is substantially different from prior works. It uses a recurrent-based updater instead of a coarse-to-fine pyramid that limits the number of updates [67]. ReFit applies reprojection to query feedback instead of using a static global feature vector [65]. Compared to learning-to-optimize approaches [16,55], ReFit does not detect keypoints but instead uses learned features and is end-to-end trainable.

ReFit improves state-of-the-art results on standard benchmarks [27,61]. The learning-to-optimize design accelerates training: ReFit converges with only 50K training iterations. We conduct extensive ablation studies to analyze the core components and applications, and contribute insights for future works.

2. Related Work

Human Mesh Model Fitting. Reconstructing human pose and shape from images has a long history [5,8,10,25]. The reconstruction often is formulated as an energy minimization problem by fitting a parametric model with optimization in various settings: single view [9,19,47,51,54], multiple views [1,18,26], video [6,52], and other sensing inputs [61,62]. These optimization procedures involve multiple stages and sophisticated designs to avoid local minima.

The optimization objective commonly consists of a data term and a prior term. The data term measures the deviation between the estimation and the detected features, while the prior term imposes constraints on the pose and shape space. In practice, optimization encounters many difficulties, including noisy keypoint detection [11,57], complicated priors [51,53,59], and the trade-off between the two terms. In the single-view setting where such difficulties compound, recent research shifts to two directions: learning to regress and learning to optimize.

Human Mesh Regression. The recognition power of deep neural network [22], paired with the representation capability of parametric human models [43,64], has fueled the recent progress in single-view human mesh regression [14,15,28,30,31,33–35,39,49,63].

Iterative refinement is an essential strategy used by many regression works [30,48,65]. Carreira et al. [12] motivate it from a neural science standpoint, which states that the human brain relies on feedback signals for various visual localization tasks. The de facto implementation introduced by HMR [30] is to concatenate the prediction with the image feature vector to make new predictions recurrently. This alone is not an effective strategy, as the network has to learn error feedback solely in the latent space.

PyMAF [66,67] proposes a refinement strategy based on a coarse-to-fine feature pyramid. The parametric mesh predicted from the coarse level is reprojected to feature maps at the finer levels to gather spatial features for refinement updates. This strategy lifts the error feedback to the image space, but the pyramid limits the number of update steps.

ReFit reprojects mesh keypoints to the feature maps at a single resolution but constructs a recurrent loop with GRUs that is not limited to the number of pyramid levels. Moreover, ReFit’s feedback and update modules have novel designs that further boost the accuracy.

Learning to Optimize. Many vision problems are inverse problems traditionally solved by optimization. Learning to optimize aims to train a neural network to propose a descend direction that replaces or supplements the optimizer update [2]. This paradigm inspires network designs that mimic optimizers in various vision tasks [3,20,41,44,58].

For the human model, LGD [55] and LFMM [16] improve traditional model fitting by using a neural network to predict residues for the optimizer updates. This approach is not end-to-end differentiable because the optimizer uses keypoint detection to compute the gradients. Neural Descent [65] replaces the optimizer with an LSTM-based updater [24] and reduces the detections into a feature vector. However, reducing spatial representation into a feature-based updater prevents it from having direct error feedback in the image space. ReFit is an evolution of these methods. It uses gated recurrent units [13,17] as updaters but utilizes reprojection on the learned feature maps as the feedback to infer an update.

ReFit can be regarded as a learned optimizer. Beside single-view inference, we demonstrate that ReFit can be a drop-in replacement for traditional optimizers in multi-view model fitting. We also apply ReFit to register a pre-fitted shape to images of the same subject.
3. Recurrent Fitting

Given an image of a person, our goal is to predict the parameters of the SMPL human model. The ReFit network extracts image features (Sec. 3.1), compares the human mesh with the pixel-aligned features (Sec. 3.2), and iteratively updates the SMPL parameters (Sec. 3.3). An overview of the method is given in Figure 2.

Preliminaries. The SMPL model [43] is parametrized with \( \Theta = \{\theta, \beta, \pi\} \), where \( \theta \in \mathbb{R}^{24 \times 3} \) are the relative rotations of the 24 body joints, \( \beta \in \mathbb{R}^{10} \) is the shape parameter, and \( \pi \in \mathbb{R}^{3} \) is the root translation w.r.t the camera. Given \( \Theta \), SMPL outputs the 3D mesh \( M(\Theta) \in \mathbb{R}^{6890 \times 3} \).

3.1. Feature Extraction

We use High-Resolution Net (HRNet) [57] as the feature extractor. It produces spatially precise feature maps that are beneficial for the feedback step. Given an image \( I \in \mathbb{R}^{H \times W \times 3} \), it outputs feature maps \( F \in \mathbb{R}^{H/4 \times W/4 \times K} \), where \( K \) is the number of channels. We set \( K \) to equal the number of keypoints.

In addition, we average pool the low-resolution branch of HRNet to produce a global feature vector and add a linear layer to predict an initialization \( \Theta_0 \) for the parameters.

3.2. Feedback

Given an initial estimate, we reproject keypoints from the SMPL model to the feature maps \( F \) to retrieve spatial features. Each keypoint is only reprojected to one channel, yielding \( K \) channels for \( K \) keypoints. This design is motivated by model fitting, where the keypoint detector outputs one channel per keypoint.

For a reprojection \( x_k = (u, v) \) where \( u \) and \( v \) are the pixel coordinates, we take the feature values inside a window centered at \( x_k \) as

\[
  f_k = \{ f(x) \in F_k \mid \| x - x_k \| \leq r \}
\]

where we set \( r = 3 \) pixels as the radius of the window. We concatenate feedback from all the keypoints, along with the current estimate \( \Theta_t \) and the bounding box center \( c_{bbox} \) and scale \( s_{bbox} \), to form the final feedback vector as

\[
  f = [f_1, ..., f_K, \Theta_t, c_{bbox}^{bbox}, s_{bbox}]
\]

Types of Keypoints. The per-keypoint feature map does not directly detect a keypoint. Instead, each channel learns the features associated with a keypoint. This process does not require direct supervision, which allows us to test different types of keypoints where 2D annotations are unavailable. We test three types (Figure 3): semantic keypoints (\( K = 24 \)); mocap markers (\( K = 67 \)); and evenly sampled mesh vertices (\( K = 231 \)). We examine the three types separately and do not combine them.

Of the three, mocap markers provide better pose and shape information than semantic keypoints, and are less redundant than mesh vertices. We use the same mocap markers as in AMASS [45], and the markers are defined on the mesh surface by selecting the closest vertices.

Full-frame Adjusted Reprojection. Human regression methods take a square-cropped image of a person as input, assuming the optical axis going through the crop center. But the input is usually cropped from a full-frame image with a different optical axis. This deviation incurs an error in the global rotation estimation, and by implication, the body pose estimation.

CLIFF [38] proposes to supervise the 2D reprojection loss in the original full-frame image, by using the full-frame reprojection:

\[
  x_{2D}^{full} = \Pi(X_{3D}^{full}) = \Pi(X_{3D} + t_{full})
\]
where $\Pi$ is the perspective projection using the original camera intrinsics, $X_{3D}$ are the body keypoints in the canonical body coordinate, and $f^{full}$ is the translation with respect to the optical center of the original image. The camera intrinsics, if unknown, can be estimated from the dimensions of the full image [32]. This reprojection is more faithful to the image formation process and leads to a better global rotation estimation.

We propose to adjust the points back to the cropped image after full-frame reprojection:

$$x_{2D}^{\text{crop}} = (x_{2D}^{f^{full}} - c_{bbox}) / s_{bbox}$$  \hspace{1cm} (4)

where $c_{bbox}$ and $s_{bbox}$ are the location and the size of the bounding box from which the crop is obtained. We call this full-frame adjusted reprojection. This seemingly trivial operation grants two advantages. First, the scale of the reprojection is normalized. The reprojection error is unaffected by the size of the person in the original image.

But more importantly, it extends full-frame reprojection to the feedback step. We reproject keypoints to the cropped image feature maps, but with this camera model, the locations are properly adjusted to be consistent with full-frame reprojection. We use this model to retrieve $f_k$ for each keypoint as in Eq 1.

**Feedback Dropout.** Each keypoint reprojection produces a feedback signal. To combine signals from all the keypoints, we formulate it as ensemble learning using dropout [56]. Specifically, we add a dropout layer, where there is a $p = 0.25$ chance that feedback $f_k$ will be zeroed out during training.

During training, the network learns to infer the pose and shape using subsets of keypoints. This prevents co-adaptations [23] of feedback signals, and makes the network robust at test time when some keypoints are occluded.

This design also has ties to Spatial Dropout [60], which randomly drops feature map channels as opposed to pixel-wise activations. Similarly, we drop a keypoint’s feedback completely instead of dropping some values.

### 3.3. Update

The update module takes the feedback signal $f$ as input (Eq. 2), and predicts an update step $\Delta \Theta_t$, which is added to produce the next estimate: $\Theta_t + \Delta \Theta_t \rightarrow \Theta_{t+1}$. The update is split into $N = 26$ parallel streams. Each stream is responsible for updating one SMPL parameter. There are 26 streams: 24 for the joint rotations, one for the shape parameter, and one for the translation.

Each stream has an identical structure, consisting of a GRU and a 2-layer MLP. The GRU updates its hidden state: $(f, h_{n-1}^s) \rightarrow h_n^s$, and the MLP maps the hidden state to the parametric update: $(h_n^s) \rightarrow \Delta \Theta_t^p$. The parallel streams can be implemented as batch matrix operations to run efficiently.

The update module acts as a first-order optimizer. Each stream learns the update rule of a parameter. Multiple streams effectively disentangle the update rules. The feedback signal is similar to the data term that informs the error. The prior, which is hand-crafted in optimization, is now learned by the update module.

Having 26 update streams does not increase the complexity because we also reduce the hidden units. The complexity per layer is $O(NM^2)$, where $N$ is the number of streams and $M$ is the size of the hidden layer. Previous works use $N = 1$ and $M = 1024$, leading to 1M parameters per layer. We use $N = 26$ and $M = 32$, which are 27K parameters per layer.

### 3.4. Supervision

The iterative update steps produce a sequence of estimates $\{\Theta_0, ..., \Theta_T\}$. At inference time, $\Theta_T$ is the final prediction. During training, we supervise all the iterations.

At each iteration $t$, the loss is made up of three terms,

$$\mathcal{L}_t = \lambda_2D \mathcal{L}_{2D}^t + \lambda_3D \mathcal{L}_{3D}^t + \lambda_{\text{SMPL}} \mathcal{L}_{\text{SMPL}}^t$$  \hspace{1cm} (5)

where each term is calculated as

$$\mathcal{L}_{2D}^t = \| \hat{J}_2D - \Pi(J_{3D}) \|_F$$

$$\mathcal{L}_{3D}^t = \| \hat{J}_{3D} - J_{3D} \|_F^2$$

$$\mathcal{L}_{\text{SMPL}}^t = \| \hat{\Theta} - \Theta_t \|_2^2$$

$J_{3D}$ are the 3D joints obtained from the SMPL model, the hat operator denotes the ground truth of that variable, and $\Pi$ is the full-frame adjusted reprojection.

The final loss is a weighted sum of the loss at each iterative update

$$\mathcal{L} = \sum_{t=0}^{T} \gamma^{T-t} \mathcal{L}_t$$  \hspace{1cm} (6)

where we set $\gamma = 0.85$ and $T = 5$ for all experiments. Prior works supervise only the last iteration to prevent the overshoot behavior, but it requires the gradient to flow through a long sequence of estimates, which slows down training convergence.

Our supervision is inspired by RAFT [58], an iterative optical flow method. First, the gradient is backpropagated through $\Delta \Theta_t$ but not through $\Theta_t$. In other words, at each
iteration, we only supervise the update but not the prediction from the previous step. Second, we use the proposed weighted sum that downscales the importance of earlier iterations.

### 3.5. Applications

ReFit is trained with single-view images for single-view inference. But since ReFit operates similarly to an optimizer, we demonstrate two applications in which traditionally an optimizer is used: multi-view model fitting, and fitting a pre-acquired shape to images of the same subject.

**Multi-view ReFit.** Motion capture with multiple calibrated cameras offers the highest accuracy [18, 27]. In a markerless setting, an optimization procedure detects keypoints in each view and fits the model by minimizing the reprojection error [1]. We replace this procedure using ReFit as shown in Figure 4.

ReFit operates on each view independently to produce updates. The updates are averaged across views with a multi-view averaging procedure. We average the shapes by taking the mean of the predicted PCA coefficients across views. Because the body pose consists of the relative rotation of each joint, we directly average the body poses across views with rotation averaging, which averages the rotation matrices and uses SVD to project the result back to $\text{SO}(3)$. The predicted global rotations are with respect to each view, so we transform them to a global coordinate with the calibrated extrinsic before performing rotation averaging. This procedure results in a single copy of the updated model. The model is then reprojected to all views to repeat the feedback-update step.

The difference between multi-view ReFit versus simply averaging the final predictions is that the averaging happens during fitting. So at each iteration, the update is in the direction that explains multiple views.

**Shape ReFit.** In this setting, we aim to fit a known body shape to images of the same subject with different poses. This is relevant for applications where the shape of a subject can be pre-scanned or pre-fitted and then used for outdoor motion capture (mocap) [21, 61]. Registering the shape to its images is commonly done with optimization, but we demonstrate that ReFit can perform a similar function.

We call this procedure Shape ReFit. At each iteration, we ignore the shape prediction and use the known shape to render the SMPL model. Therefore, the reprojection reflects the alignment status of the ground truth shape with the image, and ReFit will adjust the pose and position of the model for better alignment.

We perform experiments to verify that Shape ReFit can indeed fit a known and fixed shape to its images.

### 4. Experiments

**Datasets.** We train the ReFit model with 3DPW [61], Human3.6M [27], MPI-INF-3DHP [46], COCO [40] and MPII [4]. We use the pseudo ground truth SMPL annotations from EFT [29] for COCO and MPII. We evaluate ReFit on 3DPW and Human3.6M, with the MPJPE (mean per-joint error) and PA-MPJPE (Procrustes-aligned) metrics.

**Implementation.** We train ReFit for 50k iterations with all the datasets and evaluate on 3DPW. We then finetune the model with only Human3.6M for another 50k iterations and evaluate on Human3.6M. We use the Adam optimizer with a learning rate of 1e-4 and a batch size of 64. The input image is resized to 256×256. Standard data augmentations are applied. We include additional details in the supplementary.

#### 4.1. Quantitative Evaluation

We present quantitative evaluation in Table 1. Our method outperforms all single-view methods. We organize the methods by their training data to present a moderate view of progress: the quality of training data plays a major role in improving test accuracy, as is pointed out by recent studies [50]. We follow this practice and train CLIFF [38] as our baseline with the same data and procedure. Overall, ReFit outperforms other architectures.

We examine the generalization by training with synthetic data from BEDLAM [7] following their proposed losses. Table 2 shows that ReFit generalizes better, and by using both synthetic and real data for training, it achieves the best results.

#### 4.2. Ablations

We conduct ablation experiments to examine the core components of ReFit. All models are trained with the same data as the proposed main model, and tested on 3DPW.

**Type of Keypoints.** We train three models with the three types of keypoints in the feedback step. They are used for...
Feedback Dropout. We test the effect of dropout in the feedback step. We see that feedback dropout significantly boosts accuracy, likely because it prevents co-adaptations of keypoint signals, and makes the model robust when keypoints are occluded.

Full-frame Adjusted Reprojection. Our result confirms that using full-frame reprojection for supervision improves accuracy [38]. The proposed full-frame adjusted model extends to the feedback step. The best result is achieved when the camera model is faithful to the full image formation in all stages of the network.

Feedback Radius. For each keypoint, we query a window at its reprojection location as the feedback feature. The radius of the window affects the local context. The motivation is that the window encodes first-order information to indicate where the keypoint should move on the 2D image plane. Overall, we find that \( r = 3 \) works well.

Inference Iterations. We train the model with \( T = 5 \) update steps during training. At inference time, we test the model accuracy with various update steps. \( T = 0 \) indicates the initial guess from the backbone. Most previous methods use 3 steps, corresponding to \( T = 2 \) in our case. We see the benefit of using more update steps. We also observe that increasing to \( T = 10 \) does not cause the estimation to diverge.

Update Module. We use 26 parallel GRU streams to predict updates. To test the alternative, we swap out the 26 GRU with one larger GRU with 516 hidden units. This model has lower accuracy, confirming our hypothesis that it is beneficial to have separate update operators to learn separate update rules.

Supervision. We test supervising all iterations against supervising only the last iteration. Supervising all iterations achieves higher accuracy, but stopping the gradient across separate update rules.

### 4.3. Qualitative Evaluation

We show qualitative results on 3DPW in Figure 5. We organize the results by MPJPE percentile, with \( X^{th} \) percentile indicating higher error than \( X\% \) of the samples.

Overall, we observe accurate reconstructions with good alignment to images throughout different percentiles. We carefully inspect examples at the 99\(^{th}\) percentile and find that most examples have severe occlusions. Occlusion by another human produces a second level of complexity. We include more examples in the supplementary.

We provide examples from COCO in Figure 9 that highlights the difference between the initial estimation without refinement (T=0) and the final results from ReFit (T=5).
30th percentile  60th percentile  90th percentile  99th percentile
Figure 5. **Qualitative results from ReFit** on 3DPW. Examples are organized by MPJPE percentile. Higher percentile indicates higher error. For example, samples at the 99th percentile have higher error than 99% of the examples. The MPJPE at the four percentiles are 50.5mm, 65.4mm, 99.3mm and 158.8mm respectively. Samples at the 99th percentile often have severe occlusions or cropping.

Figure 6. **Shape ReFit**. From left to right are image, ReFit, ReFit with ground truth shape substitute, and Shape ReFit. White boxes highlight misalignment if shape is substituted in place.

### 4.4. Application Evaluation

We evaluate how ReFit performs in the two proposed applications: Shape ReFit and Multi-view ReFit.

Shape ReFit is more accurate than ReFit, as ground truth shape is used during the fitting process (Tab 4). Replacing the final prediction from ReFit with the ground truth shape (ReFit + gt shape) yields similar pose accuracy. However, substituting the shape in place produces misalignment to the image when the predicted shape is noticeably different from the ground truth, as shown in Figure 6. From the qualitative examples, we see that Shape ReFit can indeed fit a pre-acquired shape to its image.

In the multi-view experiments, we run ReFit and Multi-view ReFit on S9 and S11 from Human3.6M (Tab 5). In the first baseline, we run ReFit on each view separately and report the average reconstruction errors. In the second baseline, we run ReFit on each view separately but average the final predictions. Averaging the predictions from multiple views improves the results, particularly in MPJPE, due to a better global rotation estimation. The proposed multi-view ReFit produces the best results. The improvement is more evident in terms of the per-vertex error (PVE), where we see a 27% improvement over the baseline. In this setting, we also see the benefit of running for more update iterations, with 10 iterations slightly better than 5 iterations.

We show qualitative results for Multi-view ReFit in Figure 7. We compare the results against the mesh recovered using MoSH [42]. Multi-view ReFit uses multi-view information during fitting, and is more accurate than simply averaging the predictions (ReFit + avg).

### Tables

<table>
<thead>
<tr>
<th>Method</th>
<th>3DPW</th>
<th>Human3.6M</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MPJPE</td>
<td>PA-MPJPE</td>
</tr>
<tr>
<td>ReFit</td>
<td>65.8</td>
<td>41.0</td>
</tr>
<tr>
<td>ReFit + gt shape</td>
<td><strong>61.7</strong></td>
<td><strong>40.6</strong></td>
</tr>
<tr>
<td>Shape ReFit</td>
<td>61.7</td>
<td>40.7</td>
</tr>
</tbody>
</table>

Table 4. **Shape ReFit** on 3DPW and Human3.6M. Shape ReFit (Sec. 3.5) recovers more accurate poses than ReFit. Substituting shape in place (ReFit + gt shape) has similar accuracy, but produces misalignment (Fig. 6).

<table>
<thead>
<tr>
<th>Method</th>
<th>Human3.6M</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>MPJPE</td>
</tr>
<tr>
<td>ReFit</td>
<td>52.6</td>
</tr>
<tr>
<td>ReFit + avg</td>
<td>41.7</td>
</tr>
<tr>
<td>Multi-view ReFit (5 iters)</td>
<td>38.4</td>
</tr>
<tr>
<td>Multi-view ReFit (10 iters)</td>
<td><strong>37.5</strong></td>
</tr>
</tbody>
</table>

Table 5. **Multi-view ReFit** on Human3.6M. Per-vertex error (PVE) is computed with mesh recovered from MoSH [42]. Multi-view ReFit uses multi-view information during fitting, and is more accurate than simply averaging the predictions (ReFit + avg). We further test combining Multi-view with Shape ReFit. The recovered mesh is very close to MoSH results. MoSH is a proprietary optimization procedure that uses multi-view video and localized 3D markers, while our method only assumes multi-view images and optionally a pre-fitted shape. This result points to an alternative mocap procedure, where one can pre-fit the shape to an A-posed subject with optimization and use Multi-view + Shape ReFit for motion capture. The result can be used on its own or as initialization for optimization. A detailed comparison with MoSH is left for future work.
Two Views  Multi-view ReFit  Multi-view + Shape ReFit

Figure 7. Multi-view ReFit. We show results for 2 of 4 views. In the middle (blue meshes), multi-view ReFit reconstructs accurate poses compare to the ground truth grey meshes from MoSH [42]. On the right (green meshes), assuming ground truth shapes are pre-fitted and available, multi-view ReFit produces results that are very close to MoSH.

Figure 8. Per-keypoint feature map. Each channel corresponds to the learned features of a keypoint. Here we show feature maps from the semantic keypoint model. The network learns a peak (blue) or a valley (yellow surrounded by blue) response around the keypoint, without direct supervision.

4.5. Visualization of Feature Maps

Feature maps from the proposed model are slightly blurry and harder to visualize, as they have positive and negative values. We train an alternative model, where we add a ReLU operator on the feature maps. This modification decreases the performance slightly, as the feature maps become less expressive (strictly positive), but makes them easier to visualize. Figure 8 shows examples from the model with ReLU, and we include more examples from both models in the supplementary.

We show feature maps from the semantic keypoint model, as semantic keypoints are more interpretable than markers. The model learns meaningful features around the corresponding keypoints, often as peaks or valleys. When a keypoint is reprojected onto the feature map, the peak or valley provides directional information that helps the next network layer infer where the keypoint should move on the 2D plane. This signal, combined with signals from other keypoints, can then be converted to updates for the 3D pose.

Figure 9. Refinement in-the-wild. Examples from the coco validation set. From top to bottom are images, predictions without refinement update (T=0), and predictions from ReFit (T=5).

5. Conclusion

We have presented ReFit, the Recurrent Fitting Network that iteratively fits the parametric human model to images to recover the pose and shape of people. ReFit mirrors traditional model fitting, but learns the objective and update rules from data end-to-end. ReFit utilizes parallel GRU units that disentangle the learned update rules. Moreover, we demonstrate ReFit as a learned optimizer for multi-view fitting and shape fitting. ReFit is efficient to train and achieves state-of-the-art accuracy in challenging datasets.

Acknowledgements: We gratefully acknowledge support by the NSF grants FRR-2220868 and IIS-RI-2212433.
References


