MetaFuse: A Pre-trained Fusion Model for Human Pose Estimation

Rongchang Xie[1,6], Chunyu Wang[5], Yizhou Wang[2,3,4]
[1]Center for Data Science, Peking University [2] Adv. Inst. of Info. Tech., Peking University
[3]Center on Frontiers of Computing Studies, Peking University [4]CS Dept., Peking University
[5]Microsoft Research Asia [6]Deepwise AI Lab
{rongchangxie, yizhou.wang}@pku.edu.cn, chnuwa@microsoft.com

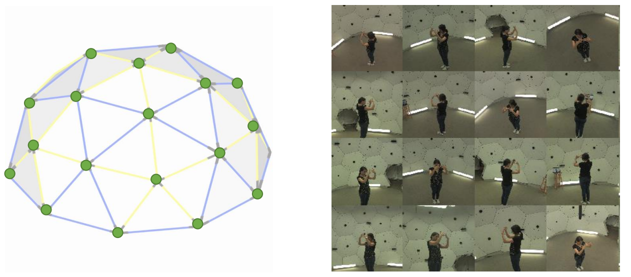
Abstract

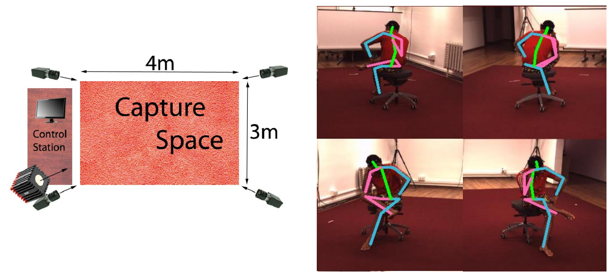
Cross view feature fusion is the key to address the occlusion problem in human pose estimation. The current fusion methods need to train a separate model for every pair of cameras making them difficult to scale. In this work, we introduce MetaFuse*, a pre-trained fusion model learned from a large number of cameras in the Panoptic dataset. The model can be efficiently adapted or finetuned for a new pair of cameras using a small number of labeled images. The strong adaptation power of MetaFuse is due in large part to the proposed factorization of the original fusion model into two parts— (1) a generic fusion model shared by all cameras, and (2) lightweight camera-dependent transformations. Furthermore, the generic model is learned from many cameras by a meta-learning style algorithm to maximize its adaptation capability to various camera poses. We observe in experiments that* MetaFuse *finetuned on the public datasets outperforms the state-of-the-arts by a large margin which validates its value in practice.*

1. Introduction

Estimating 3D human pose from multi-view images has been a longstanding goal in computer vision. Most works follow the pipeline of first estimating 2D poses in each camera view and then lifting them to 3D space, for example, by triangulation [15] or by pictorial structure model [25]. However, the latter step generally depends on the quality of 2D poses which unfortunately may have large errors in practice especially when occlusion occurs.

Multi-view feature fusion [39, 25] has great potential to solve the occlusion problem because a joint occluded in one view could be visible in other views. The most challenging problem in multi-view fusion is to find the corresponding locations between different cameras. In a recent work [25] , this is successfully solved by learning a fusion network for each pair of cameras (referred to as *NaiveFuse* in this paper). However, the learned correspondence is dependent


(a) Large-Scale Pretraining of *MetaFuse* from many camera views.



(b) Efficient Adaptation of *MetaFuse* to Unseen Camera placement.

Figure 1. Concept of *MetaFuse*. We learn a pre-trained feature fusion model from a large number of cameras, *i.e*. the green dots in (a). Then for a new environment, we finetune the pre-trained model for each camera pair using only a few training data to get a customized 2D pose estimator. The feature fusion allows us to localize the 2D joints even when occlusion occurs as in (b).

on camera poses so they need to retrain the model when camera poses change which is not flexible.

This work aims to address the flexibility issue in multi-view fusion. To that end, we introduce a pre-trained cross view fusion model *MetaFuse*, which is learned from a large number of camera pairs in the CMU Panoptic dataset [17]. The fusion strategies and learning methods allow it to be rapidly adapted to unknown camera poses with only a few labeled training data. See Figure 1 for illustration of the concept. One of the core steps in *MetaFuse* is to factorize *NaiveFuse* [25] into two parts: a generic fusion model shared by all cameras and a number of lightweight affine transformations. We learn the generic fusion model to maximize its adaptation performance to various camera poses by a meta-learning style algorithm. In the testing stage, for each new pair of cameras, only the lightweight affine transformations are finetuned utilizing a small number of training images from the target domain.

We evaluate *MetaFuse* on three public datasets including H36M [14], Total Capture [34] and CMU Panoptic [17]. The pre-training is only performed on the Panoptic dataset which consists of thousands of camera pairs. Then we finetune *MetaFuse* on each of the three target datasets to get customized 2D pose estimators and report results. For example, on the H36M dataset, *MetaFuse* notably outperforms *NaiveFuse* [25] when 50, 100, 200 and 500 images are used for training the fusion networks, respectively. This validates the strong adaptation power of *MetaFuse*. In addition, we find that *MetaFuse* finetuned on 50 images [1] already outperforms the baseline without fusion by a large margin. For example, the joint detection rate for elbow improves from 83.7% to 86.3%.

We also conduct experiments on the downstream 3D pose estimation task. On the H36M dataset, *MetaFuse* gets a notably smaller $3D$ pose error than the state-of-the-art. It also gets the smallest error of 32.4mm on the Total Capture dataset. It is worth noting that in those experiments, our approach actually uses significantly fewer training images from the target domain compared to most of the state-of-the-arts. The results validate the strong adaptation capability of *MetaFuse*.

1.1. Overview of MetaFuse

NaiveFuse learns the spatial correspondence between a pair of cameras in a supervised way as shown in Figure 2. It uses a Fully Connected Layer (FCL) to densely connect the features at different locations in the two views. A weight in FCL, which connects two features (spatial locations) in two views, represents the probability that they correspond to the same 3D point. The weights are learned end-to-end together with the pose estimation network. See Section 3 for more details. One main drawback of *NaiveFuse* is that it has many parameters which requires to label a large number of training data for *every* pair of cameras. This severely limits its applications in practice.

To resolve this problem, we investigate how features in different views are related geometrically as shown in Figure 3. We discover that *NaiveFuse* can be factorized into two parts: a *generic* fusion model shared by all cameras as well as a number of *camera-specific* affine transformations which have only a few learnable parameters (see Section 4). In addition, inspired by the success of meta-learning in the few-shot learning literature [10, 19, 28], we propose a *meta-learning* algorithm to learn the generic model on a large number of cameras to maximize its adaptation performance (see Section 4.2). This approach has practical values in that, given a completely new multi-camera environment and a small number of labeled images, it can significantly boost the pose estimation accuracy.

[1]Labeling human poses for 50 images generally takes several minutes which is practical in many cases.

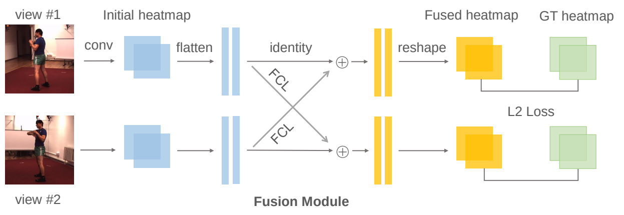


Figure 2. The *NaiveFuse* model. It jointly takes two-view images as input and outputs 2D poses simultaneously for both views in a single CNN. The fusion module consists of multiple FCLs with each connecting an ordered pair of views. The weights, which encode the camera poses, are learned end-to-end from data.

2. Related work

Multi-view Pose Estimation We classify multi-view 3D pose estimators into two classes. The first class is model based approaches such as [21, 5, 11, 27]. They define a body model as simple primitives such as cylinders, and optimize their parameters such that the model projection matches the image features. The main challenge is the difficult non-linear non-convex optimization problems which has limited their performance to some extent.

With the development of 2D pose estimation techniques, some approaches such as [1, 7, 6, 24, 4, 8, 25] adopt a simple two-step framework. They first estimate 2D poses from multi-view images. Then with the aid of camera parameters (assumed known), they recover the corresponding 3D pose by either triangulation or by pictorial structure models. For example in [1], the authors obtain 3D poses by direct triangulation. Later the authors in [6] and in [24] propose to apply a multi-view pictorial structure model to recover 3D poses. This type of approaches have achieved the state-of-the-art performance in recent years.

Some previous works such as [1, 39, 25] have explored multi-view geometry for improving 2D human pose estimation. For example, Amin *et al*. [1] propose to jointly estimate 2D poses from multi-view images by exploring multi-view consistency. It differs from our work in that it does not actually *fuse* features from other views to obtain better 2D heatmaps. Instead, they use the multi-view 3D geometric relation to *select* the joint locations from the "imperfect" heatmaps. In [39], multi-view consistency is used as a source of supervision to train the pose estimation network which does not explore multi-view feature fusion.

NaiveFuse [25] is proposed for the situation where we have sufficient labeled images for the target environment. However, it does not work in a more practical scenario where we can only label a few images for every target camera. To our knowledge, no previous work has attempted to solve the multi-view fusion problem in the context of few-shot learning which has practical values.

Meta Learning Meta-learning refers to the framework which uses one learning system to optimize another learning system [35]. It learns from task distributions rather than a single task [26, 28] with the target of rapid adaptation to new tasks. It has been widely used in few-shot classification [19, 28, 30] and reinforcement learning [9, 23] tasks. Meta learning can be used as an optimizer. For example, Andrychowicz *et al.* [3] use LSTM meta-learner to learn updating base-learner, which outperforms hand-designed optimizers on the training tasks. For classification, Finn *et al.* [10] propose Model-Agnostic Meta-Learning (MAML) to learn good parameter initializations which can be rapidly finetuned for new classification tasks. Sun *et al.* [31] propose Meta-Transfer learning that learns scaling and shifting functions of DNN weights to prevent catastrophic forgetting. The proposed use of meta-learning to solve the adaptation problem in cross view fusion has not been studied previously, and has practical values.

3. Preliminary for Multi-view Fusion

We first present the basics for multi-view feature fusion [12, 39, 25] to lay the groundwork for *MetaFuse*. Let $\boldsymbol{P}$ be a point in 3D space as shown in Figure 3. The projected 2D points in view 1 and 2 are $\boldsymbol{Y}_P^1 \in \mathcal{Z}_1$ and $\boldsymbol{Y}_P^2 \in \mathcal{Z}_2$, respectively. The $\mathcal{Z}_1$ and $\mathcal{Z}_2$ denote the set of pixel coordinates in two views, respectively. The features of view 1 and 2 at different locations are denoted as $\mathcal{F}^1 = \{\boldsymbol{x}_1^1, \cdots, \boldsymbol{x}_{|\mathcal{Z}_1|}^1\}$ and $\mathcal{F}^2 = \{\boldsymbol{x}_1^2, \cdots, \boldsymbol{x}_{|\mathcal{Z}_2|}^2\}$. The core for fusing a feature $\boldsymbol{x}_i^1$ in view one with those in view two is to establish the correspondence between the two views:

$$\boldsymbol{x}_i^1 \leftarrow \boldsymbol{x}_i^1 + \sum_{j=1}^{|\mathcal{Z}_2|} \omega_{j,i} \cdot \boldsymbol{x}_j^2, \quad \forall i \in \mathcal{Z}_1, \tag{1}$$

where $\omega_{j,i}$ is a scalar representing their correspondence relation— $\omega_{j,i}$ is positive when $\boldsymbol{x}_i^1$ and $\boldsymbol{x}_j^2$ correspond to the same 3D point. It is zero when they correspond to different 3D points. The most challenging task is to determine the values of all $\omega_{j,i}$ for each pair of cameras (*i.e.* to find the corresponding points).

Discussion For each point $\boldsymbol{Y}_P^1$ in view 1, we know the corresponding point $\boldsymbol{Y}_P^2$ has to lie on the epipolar line I. But we cannot determine the exact location of $\boldsymbol{Y}_P^2$ on I. Instead of trying to find the exact pixel to pixel correspondence, we fuse $\boldsymbol{x}_i^1$ with all features on line I. Since fusion happens in the *heatmap layer*, ideally, $\boldsymbol{x}_j^2$ has large values near $\boldsymbol{Y}_P^2$ and zeros at other locations on the epipolar line I. It means the non-corresponding locations on the line will not contribute to the fusion. So fusing all pixels on the epipolar line is an appropriate solution.

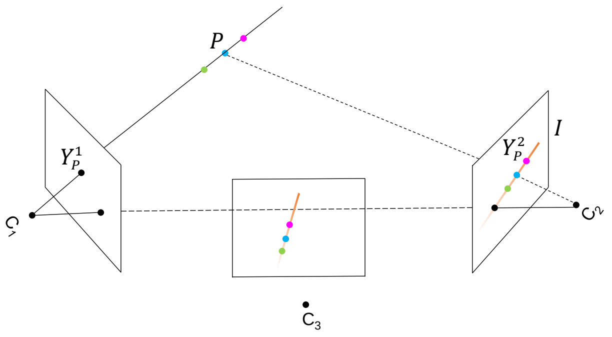


Figure 3. Geometric illustration of multi-view feature fusion. An image point Y_P^1 back-projects to a ray in 3D defined by the first camera center C_1 and Y_P^1. This line is imaged as I in the second view. The 3D point P which projects to Y_P^1 must lie on this ray, so the image of P must lie on I. If the camera poses change, for example, we move the camera 2 to 3, then we can approximately get the corresponding line by applying an appropriate affine transformation to I. See section 4.

Implementation The above fusion strategy is implemented by FCLs (which are appended to the pose estimation network) in *NaiveFuse* as shown in Figure 2. The whole network, together with the FCL parameters, can be trained end-to-end by enforcing supervision on the fused heatmaps. However, FCL naively connects each pixel in one view with all pixels in the other view, whose parameters are position-sensitive and may undergo dramatic changes even when the camera poses change slightly. So it is almost impossible to learn a pre-trained model that can be adapted to various camera poses using small data as our MetaFuse. In addition, the large FCL parameters increase the risk of over-fitting to small datasets and harm its generalization ability.

Note that we do not claim novelty for this *NaiveFuse* approach as similar ideas have been explored previously such as in [39, 25]. Our contributions are two-fold. First, it reformulates *NaiveFuse* by factorizing it into two smaller models which significantly reduces the number of learnable parameters for each pair of cameras in deployment. Second, we present a meta-learning style algorithm to learn the reformulated fusion model such that it can be rapidly adapted to unknown camera poses with small data.

4. MetaFuse

Let $\boldsymbol{\omega}^{\text{base}} \in \mathcal{R}^{H\times W}$ be a *basic* fusion model, *i.e.* the fusion weight matrix discussed in Section 3, which connects ONE pixel in the first view with all $H \times W$ pixels in the second view. See Figure 3 for illustration. For other pixels in the first view, We will construct the corresponding fusion weight matrices by applying appropriate affine transformations to the basic weight matrix $\boldsymbol{\omega}^{\text{base}}$. In addition, we also similarly transform $\boldsymbol{\omega}^{\text{base}}$ to obtain customized fusion matrices for different camera pairs. In summary, this basic fusion weight matrix (*i.e.* the *generic* model we mentioned previously) is shared by all cameras. We will explain this in detail in the following sections.

4.1. Geometric Interpretation

From Figure 3, we know $\boldsymbol{Y}_P^1$ corresponds to the line I in camera 2 which is characterized by $\boldsymbol{\omega}^{\text{base}}$. If camera 2 changes to 3, we can obtain the epipolar line by applying an appropriate affine transformation to I. This is equivalent to applying the transformation to $\boldsymbol{\omega}^{\text{base}}$. Similarly, we can also adapt $\boldsymbol{\omega}^{\text{base}}$ for different pixels in view one. Let $\boldsymbol{\omega_i} \in \mathcal{R}^{H\times W}$ be the fusion model connecting the i_{th} pixel in view 1 with all pixels in view 2. We can compute the corresponding fusion model by applying a dedicated transformation to $\boldsymbol{\omega}^{\text{base}}$

$$\boldsymbol{\omega}_i \leftarrow \mathrm{T}^{\theta_i}(\boldsymbol{\omega}^{\text{base}}), \quad \forall i, \tag{2}$$

where T is the affine transformation and θ_i is a six-dimensional affine transformation parameter for the i_{th} pixel which can be learned from data. See Figure 4 for illustration. We can verify that the total number of parameters in this model is only $\mathcal{Z}_2 + 6 \times \mathcal{Z}_1$. In contrast, the number of parameters in the original naive model is $\mathcal{Z}_1 \times \mathcal{Z}_2$ which is much larger ($\mathcal{Z}_1$ and $\mathcal{Z}_2$ are usually 64^2). The notable reduction of the learnable parameters is critical to improve the adaptation capability of *MetaFuse*. Please refer to the Spatial Transformer Network [16] for more details about the implementation of T.

With sufficient image and pose annotations from a pair of cameras, we can directly learn the generic model $\boldsymbol{\omega}^{\text{base}}$ and the affine transformation parameters θ_i for every pixel by minimizing the following loss function:

$$\mathcal{L}_{\mathcal{D}_{\text{Tr}}}(\boldsymbol{\omega}^{\text{base}}, \theta) = \frac{1}{|\mathcal{D}_{\text{Tr}}|} \sum_{\mathcal{F},\mathcal{F}_{gt}\in\mathcal{D}_{\text{Tr}}} \text{MSE}(f_{[\boldsymbol{\omega}^{\text{base}};\theta]}(\mathcal{F}), \mathcal{F}_{gt}), \tag{3}$$

where $\mathcal{F}$ are the initially estimated heatmaps (before fusion), and $f_{[\boldsymbol{\omega}^{\text{base}};\theta]}$ denotes the fusion function with parameters $\boldsymbol{\omega}^{\text{base}}$ and θ. See Eq.(1) and Eq.(2) for how we construct the fusion function. $\mathcal{F}_{gt}$ denotes the ground-truth pose heatmaps. Intuitively, we optimize $\boldsymbol{\omega}^{\text{base}}$ and θ such as to minimize the difference between the fused heatmaps and the ground truth heatmaps. We learn different θs for different pixels and camera pairs. It is also worth noting that both θ and $\boldsymbol{\omega}^{\text{base}}$ are global variables which do not depend on images. The loss function can be simply minimized by stochastic gradient descent. However, the model trained this way cannot generalize to new cameras with sufficient accuracy when only a few labeled data are available.

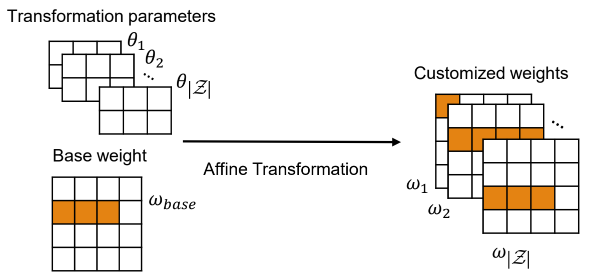


Figure 4. Applying different affine transformations $\mathrm{T}^{\theta_i}(\cdot)$ to the generic base weight ω^{base} to obtain the customized fusion weight ω_i for each pixel in view one.

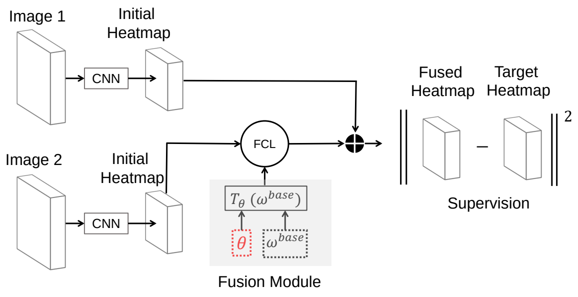


Figure 5. The pipeline for training *MetaFuse*. In the first step, we pre-train the backbone network before fusion on all training images by regular gradient descent. In the second step, we fix the backbone parameters and meta-train $\boldsymbol{\omega}^{\text{base}}$ and θ. In the testing stage, for a new camera configuration, we fix $\boldsymbol{\omega}^{\text{base}}$ and only fine-tune the transformation parameters θ based on small training data from the target camera.

4.2. Learning MetaFuse

We now describe how we learn *MetaFuse* including the generic model (*i.e.* $\boldsymbol{\omega}^{\text{base}}$ and the initializations of θ) from a large number of cameras so that the learned fusion model can rapidly adapt to new cameras using small data. The algorithm is inspired by a meta-learning algorithm proposed in [10]. We describe the main steps for learning MetaFuse in the following subsections.

Warming Up In the first step, we train the backbone network (*i.e.* the layers before the fusion model) to speed up the subsequent meta-training process. All images from the training dataset are used for training the backbone. The backbone parameters are directly optimized by minimizing the MSE loss between the initial heatmaps and the ground truth heatmaps. Note that the backbone network is only trained in this step, and will be fixed in the subsequent meta-training step to notably reduce the training time.

Meta-Training In this step, as shown in Figure 5, we learn the generic fusion model $\boldsymbol{\omega}^{\text{base}}$ and the initializations of θ by a meta-learning style algorithm. Generally speaking, the two parameters are sequentially updated by computing gradients over pairs of cameras (sampled from the dataset) which are referred to as tasks.

Task is an important concept in meta-training. In particular, every task $\mathcal{T}_i$ is associated with a small dataset $\mathcal{D}_i$ which consists of a few images and ground truth 2D pose heatmaps sampled from the same camera pair. For example, the camera pair $(\text{Cam}_1, \text{Cam}_2)$ is used in task $\mathcal{T}_1$ while the camera pair $(\text{Cam}_3, \text{Cam}_4)$ is used in in task $\mathcal{T}_2$. We learn the fusion model from many of such different tasks so that it can get good results when adapted to a new task by only a few gradient updates. Let $\{\mathcal{T}_1, \mathcal{T}_2, \cdots, \mathcal{T}_N\}$ be a number of tasks. Each $\mathcal{T}_i$ is associated with a dataset $\mathcal{D}_i$ consisting of data from a particular camera pair. Specifically, each $\mathcal{D}_i$ consists of two subsets: $\mathcal{D}_i^{train}$ and $\mathcal{D}_i^{test}$. As will be clarified later, both subsets are used for training.

We follow the model-agnostic meta-learning framework [10] to learn the optimal initializations for $\boldsymbol{\omega}^{\text{base}}$ and θ. In the meta-training process, when adapted to a new task $\mathcal{T}_i$, the model parameters $\boldsymbol{\omega}^{\text{base}}$ and θ will become $\boldsymbol{\omega}^{\text{base}\prime}$ and θ', respectively. The core of meta-training is that we learn the optimal $\boldsymbol{\omega}^{\text{base}}$ and θ which will get a small loss on this task if it is updated based on the small dataset of the task. Specifically, $\boldsymbol{\omega}^{\text{base}\prime}$ and θ' can be computed by performing gradient descent on task $\mathcal{T}_i$

$$\theta' = \theta - \alpha \nabla_\theta \mathcal{L}_{\mathcal{D}_i^{\text{train}}}(\boldsymbol{\omega}^{\text{base}}, \theta) \tag{4}$$

$$\boldsymbol{\omega}^{\text{base}\prime} = \boldsymbol{\omega}^{\text{base}} - \alpha \nabla_{\boldsymbol{\omega}^{\text{base}}} \mathcal{L}_{\mathcal{D}_i^{\text{train}}}(\boldsymbol{\omega}^{\text{base}}, \theta). \tag{5}$$

The learning rate α is a hyper-parameter. It is worth noting that we do not actually update the model parameters according to the above equations. $\boldsymbol{\omega}^{\text{base}\prime}$ and θ' are the intermediate variables as will be clarified later. The core idea of meta learning is to learn $\boldsymbol{\omega}^{\text{base}}$ and θ such that after applying the above gradient update, the loss for the current task (evaluated on $\mathcal{D}_i^{test}$) is minimized. The model parameters are trained by optimizing for the performance of $\mathcal{L}_{\mathcal{D}_i^{\text{test}}}(\boldsymbol{\omega}^{\text{base}\prime}, \theta')$ with respect to $\boldsymbol{\omega}_{\text{base}}$ and θ, respectively, across all tasks. Note that, $\boldsymbol{\omega}^{\text{base}\prime}$ and θ' are related to the initial parameters $\boldsymbol{\omega}_{\text{base}}$ and θ because of Eq.(4) and Eq.(4). More formally, the meta-objective is as follows:

$$\min_{\boldsymbol{\omega}^{\text{base}}, \theta} \mathcal{L}_{\mathcal{D}_i^{\text{test}}}(\boldsymbol{\omega}^{\text{base}\prime}, \theta') \tag{6}$$

The optimization is performed over the parameters $\boldsymbol{\omega}_{\text{base}}$ and θ, whereas the objective is computed using the updated model parameters $\boldsymbol{\omega}^{\text{base}\prime}$ and θ'. In effect, our method aims to optimize the model parameters such that one or a small number of gradient steps on a new task will produce maximally effective behavior on that task. We repeat the above steps iteratively on each task $\mathcal{D}_i \in \{\mathcal{D}_1, \mathcal{D}_2, \cdots, \mathcal{D}_N\}$. Recall that each $\mathcal{D}_i$ corresponds to a different camera configuration. So it actually learns a generic $\boldsymbol{\omega}^{\text{base}}$ and θ which can be adapted to many camera configurations with the gradients computed on small data. The $\boldsymbol{\omega}^{\text{base}}$ will be fixed after this meta-training stage.

4.3. Finetuning MetaFuse

For a completely new camera configuration, we adapt the meta-trained model by finetuning θ. This is realized by directly computing the gradients to θ on a small number of labeled training data. Due to the lack of training data, the generic model $\boldsymbol{\omega}^{\text{base}}$ which is shared by all camera configurations will not be updated. The number of learnable parameters in this step is $6 \times H \times W$ which is only several thousands in practice.

5. Experiments

5.1. Datasets, Metrics and Details

CMU Panoptic Dataset This dataset [17] provides images captured by a large number of synchronized cameras. We follow the convention in [37] to split the training and testing data. We select 20 cameras (*i.e.* 380 ordered camera pairs) from the training set to pre-train *MetaFuse*. Note that we only perform pre-training on this large dataset, and directly finetune the learned model on each target dataset to get a customized multi-view fusion based 2D pose estimator. For the sake of evaluation on this dataset, we select six from the rest of the cameras. We run multiple trials and report average results to reduce the randomness caused by camera selections. In each trial, four cameras are chosen from the six for multi-view fusion.

H36M Dataset This dataset [14] provides synchronized four-view images. We use subjects $1, 5, 6, 7, 8$ for finetuning the pre-trained model, and use subjects $9, 11$ for testing purpose. It is worth noting that the camera placement is slightly different for each of the seven subjects.

Total Capture Dataset In this dataset [34], there are five subjects performing four actions including Roaming(**R**), Walking(**W**), Acting(**A**) and Freestyle(**FS**) with each repeating 3 times. We use Roaming 1,2,3, Walking 1,3, Freestyle 1,2 and Acting 1,2 of *Subjects 1,2,3* for finetuning the pre-trained model. We test on Walking 2, Freestyle 3 and Acting 3 of all subjects.

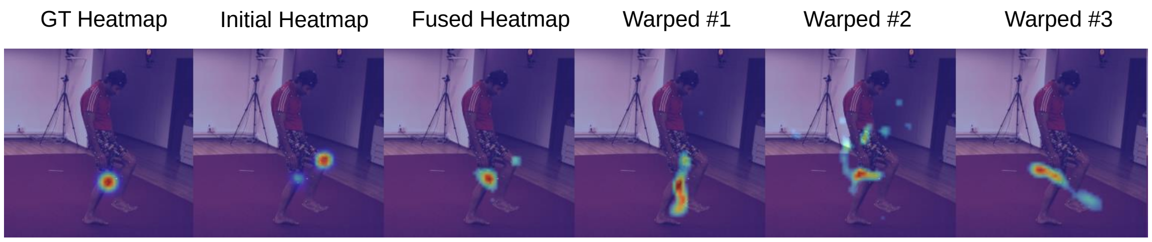


Figure 6. Heatmaps estimated by *MetaFuse*. The first figure shows the ground truth heatmap of left knee. The second shows the initially detected heatmap. The highest response is at the wrong location. The third image shows the fused heatmap which correctly localizes the left knee. The rest images show the heatmaps warped from the other three views.

Table 1. Description of the baselines.

Names	**Description**
No-Fusion	This is a simple baseline which does not perform multi-view fusion. It is equivalent to estimating poses in each camera view independently. This approach has the maximum flexibility since it can be directly applied to new environments without adaptation.
NaiveFuse$^{\text{full}}$	This baseline directly trains the NaiveFuse model using all images from the target camera configuration. This can be regarded as an upper bound for *MetaFuse* when there are sufficient training images. This approach has the LEAST flexibility because it requires to label a large number of images from each target camera configuration.
NaiveFuse^{K}	This baseline pre-trains the *NaiveFuse* model on the Panoptic dataset (using four selected cameras) by regular stochastic gradient descent. Then it finetunes the pre-trained model on K images from the target camera configuration. The approach is flexible when K is small.
AffineFuse^{K}	This baseline first pre-trains our factorized fusion model according to the description in Section 4.1 on the Panoptic dataset (using four selected cameras) by regular stochastic gradient descent. Then it finetunes the model on the target camera configuration.
MetaFuse$^{\text{K}}$	It finetunes the meta-learned model on K images of the target cameras. It differs from *AffineFuse*^{K} in that it uses the meta-learning style algorithm to pre-train the model.

Metrics The 2D pose accuracy is measured by Joint Detection Rate (JDR). If the distance between the estimated and the ground-truth joint location is smaller than a threshold, we regard this joint as successfully detected. The threshold is set to be half of the head size as in [2]. JDR is computed as the percentage of the successfully detected joints. The 3D pose estimation accuracy is measured by the Mean Per Joint Position Error (MPJPE) between the ground-truth 3D pose and the estimation. We do not align the estimated 3D poses to the ground truth as in [22, 32].

Complexity The warming up step takes about 30 hours on a single 2080Ti GPU. The meta-training stage takes about 5 hours. This stage is fast because we use the pre-computed heatmaps. The meta-testing (finetuning) stage takes about 7 minutes. Note that, in real deployment, only meta-testing needs to be performed for a new environment which is very fast. In testing, it takes about 0.015 seconds to estimate a 2D pose from a single image.

Implementation Details We use a recent 2D pose estimator [38] as the basic network to estimate the initial heatmaps. The ResNet50 [13] is used as its backbone. The input image size is 256×256 and the resolution of the heatmap is 64×64. In general, using stronger 2D pose estimators can further improve the final 2D and 3D estimation results but that is beyond the scope of this work. We apply softmax with temperature $T = 0.2$ to every channel of the fused heatmap to highlight the maximum response.

The Adam [18] optimizer is used in all phases. We train the backbone network for 30 epochs on the target dataset in the warming up stage. Note that we do not train the fusion model in this step. The learning rate is initially set to be $1e^{-3}$, and drops to $1e^{-4}$ at 15 epochs and $1e^{-5}$ at 25 epochs, respectively. In meta-training and meta-testing, the learning rates are set to be $1e^{-3}$ and $5e^{-3}$, respectively. We evaluate our approach by comparing it to the five related baselines, which are detailed in Table 1.

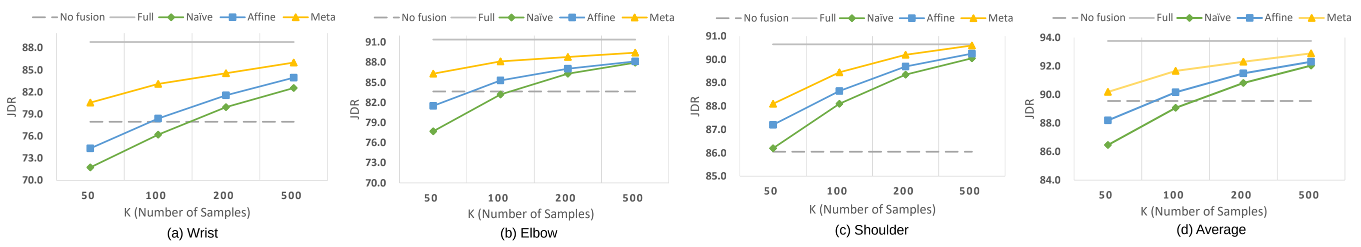


Figure 7. The 2D joint detection rates of different methods on the H36M dataset. The x-axis represents the number of samples for finetuning the fusion model. The y-axis denotes the JDR. We show the average JDR over all joints, as well as the JDRs for several typical joints. The method of "full" denotes *NaiveFuse*$^{\text{full}}$ which can be regarded as an upper bound for all fusion methods.

5.2. Results on the H36M Dataset

2D Results The joint detection rates (JDR) of the baselines and our approach are shown in Figure 7. We present the average JDR over all joints, as well as the JDRs of several typical joints. We can see that the JDR of *No-Fusion* (the grey dashed line) is lower than our *MetaFuse* model regardless of the number of images used for finetuning the fusion model. This validates the importance of multi-view fusion. The improvement is most significant for the wrist and elbow joints because they are frequently occluded by human body in this dataset.

NaiveFuse$^{\text{full}}$ (the grey solid line) gets the highest JDR because it uses all training data from the H36M dataset. However, when we use fewer data, the performance drops significantly (the green line). In particular, *NaiveFuse*^{50} even gets worse results than *No-Fusion*. This is because small training data usually leads to over-fitting for large models. We attempted to use several regularization methods including l_2, l_1 and $L_{2,1}$ (group sparsity) on $\boldsymbol{\omega}$ to alleviate the over-fitting problem of *NaiveFuse*. But none of them gets better performance than vanilla *NaiveFuse*. It means that the use of geometric priors in *MetaFuse* is more effective than the regularization techniques.

Our proposed *AffineFuse*^{K}, which has fewer parameters than *NaiveFuse*^{K}, also gets better result when the number of training data is small (the blue line). However, it is still worse than *MetaFuse*^{K}. This is because the model is not pre-trained on many cameras to improve its adaptation performance by our meta-learning-style algorithm which limits its performance on the H36M dataset.

Our approach *MetaFuse*^{K} outperforms all baselines. In particular, it outperforms *No-Fusion* when only 50 training examples from the H36M dataset are used. Increasing this number consistently improves the performance. The result of *MetaFuse*^{500} is already similar to that of *NaiveFuse*$^{\text{full}}$ which is trained on more than 80K images.

We also evaluate a variant of *NaiveFuse* which is learned by the meta-learning algorithm. The average JDR are 87.7% and 89.3% when 50 and 100 examples are used, which are much worse than *MetaFuse*. The results validate the importance of the geometry inspired decomposition.

Table 2. The 3D MPJPE errors obtained by the state-of-the-art methods on the H36M dataset. *MetaFuse* uses the pictorial model for estimating 3D poses. "Full H36M Training" means whether we use the full H36M dataset for adaptation or training.

Methods	Full H36M Training	MPJPE
PVH-TSP [34]	✓	87.3mm
Pavlakos [24]	✓	56.9mm
Tome [32]	✓	52.8mm
Liang [20]	✓	45.1mm
CrossView [25]	✓	26.2mm
Volume [15]	✓	20.8mm
CrossView [25]	✗	43.0mm
Volume [15]	✗	34.0mm
MetaFuse^{50}	✗	32.7mm
MetaFuse^{100}	✗	31.3mm
MetaFuse^{500}	✗	**29.3**mm

Examples Figure 6 explains how *MetaFuse* improves the 2D pose estimation accuracy. The target joint is the left knee in this example. But the estimated heatmap (before fusion) has the highest response at the incorrect location (near right knee). By leveraging the heatmaps from the other three views, it accurately localizes the left knee joint. The last three images show the warped heatmaps from the other three views. We can see the high response pixels approximately form a line (the epipolar line) in each view.

We visualize some typical poses estimated by the baselines and our approach in Figure 8. First, we can see that when occlusion occurs, *No-Fusion* usually gets inaccurate 2D locations for the occluded joints. For instance, in the first example, the left wrist joint is localized at a wrong location. *NaiveFuse* and *MetaFuse* both help to localize the wrist joint in this example, and *MetaFuse* is more accurate. However, in some cases, *NaiveFuse* may get surprisingly bad results as shown in the third example. The left ankle joint is localized at a weird location even though it is visible. The main reason for this abnormal phenomenon is that the *NaiveFuse* model learned from few data lacks generalization capability. *MetaFuse* approach gets consistently better results than the two baseline methods.

Table 3. 3D pose estimation errors MPJPE (mm) of different methods on the Total Capture dataset.

Approach	Subjects(S1,2,3)			Subjects(S4,5)			Mean
	Walking2	Acting3	FreeStyle3	Walking2	Acting3	FreeStyle3	
Tri-CPM [36]	79.0	106.5	112.1	79.0	73.7	149.3	99.8
PVH [34]	48.3	94.3	122.3	84.3	154.5	168.5	107.3
IMUPVH [34]	30.0	49.0	90.6	36.0	109.2	112.1	70.0
LSTM-AE [33]	**13.0**	**23.0**	47.0	**21.8**	40.9	68.5	34.1
No-Fusion	28.1	30.5	42.9	45.6	46.3	74.3	41.2
MetaFuse^{500}	21.7	23.3	**32.1**	35.2	**34.9**	**57.4**	**32.4**

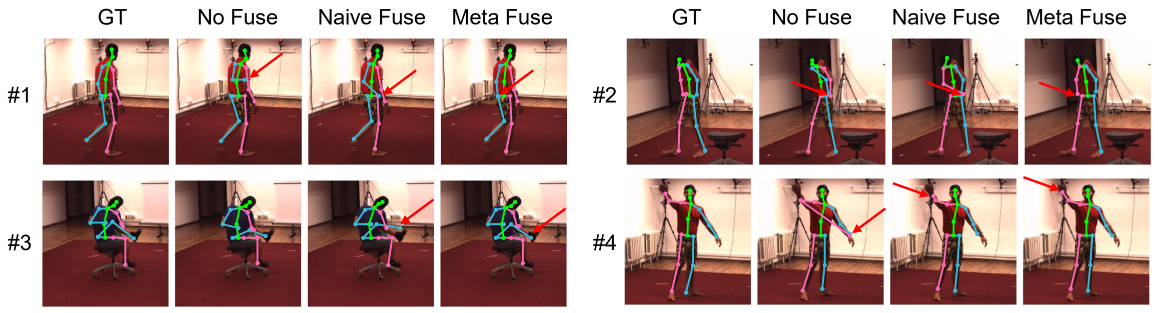


Figure 8. Four groups of sample 2D poses estimated by different methods. Each group has 1x4 sub-figures which correspond to the ground truth(GT) and three methods, respectively. The pink and cyan joints belong to the right and left body parts, respectively. The red arrows highlight the joints whose estimations are different for the three methods.

3D Results We estimate 3D pose from multi-view 2D poses by a pictorial structure model [25]. The results on the H36M dataset are shown in Table 2. Our *MetaFuse* trained on only 50 examples decreases the error to 32.7mm. Adding more training data consistently decreases the error. Note that some approaches in the table which use the full H36M dataset for training are not comparable to our approach.

5.3. Results on Total Capture

The results are shown in Table 3. We can see that *MetaFuse* outperforms *No-Fusion* by a large margin on all categories which validates its strong generalization power. In addition, our approach also outperforms the state-of-the-art ones including a recent work which utilizes temporal information [33]. We notice that LSTM-AE [33] outperforms our approach on the "Walking2" action. This is mainly because LSTM-AE uses temporal information which is very effective for this "Walking2" action. We conduct a simple proof-of-concept experiment where we apply the Savitzky-Golay filter [29] to smooth the 3D poses obtained by our approach. We find the average 3D error for the "Walking" action of our approach decreases by about 5mm. The result of our approach is obtained when *MetaFuse* is finetuned on only 500 images. In contrast, the state-of-the-art methods train their models on the whole dataset.

5.4. Results on Panoptic Dataset

We also conduct experiments on the Panoptic dataset. Note that the cameras selected for testing are different from those selected for pre-training. The 3D error of the *No-Fusion* baseline is 40.47mm. Our *MetaFuse* approach gets a smaller error of 37.27mm when only 50 examples are used for meta-testing. This number further decreases to 31.78mm when we use 200 examples. In contrast. the errors for the *NaiveFuse* approach are 43.39mm and 35.60mm when the training data number is 50 and 200, respectively. The results validate that our proposed fusion model can achieve consistently good results on the three large scale datasets.

6. Conclusion

We present a multi-view feature fusion approach which can be trained on as few as 100 images for a new testing environment. It is very flexible in terms of that it can be integrated with any of the existing 2D pose estimation networks, and it can be adapted to any environment with any camera configuration. The approach achieves the state-of-the-art results on three benchmark datasets. In our future work, we will explore the possibility to apply the fusion model to other tasks such as semantic segmentation. Besides, we can leverage synthetic data of massive cameras to further improve the generalization ability of model.

Acknowledgement This work was supported in part by MOST-2018AAA0102004, NSFC-61625201, 61527804 and DFG TRR169 / NSFC Major International Collaboration Project "Crossmodal Learning".

References

[1] Sikandar Amin, Mykhaylo Andriluka, Marcus Rohrbach, and Bernt Schiele. Multi-view pictorial structures for 3D human pose estimation. In *BMVC*, 2013.

[2] Mykhaylo Andriluka, Leonid Pishchulin, Peter Gehler, and Bernt Schiele. 2D human pose estimation: New benchmark and state of the art analysis. In *CVPR*, pages 3686–3693, 2014.

[3] Marcin Andrychowicz, Misha Denil, Sergio Gomez, Matthew W Hoffman, David Pfau, Tom Schaul, Brendan Shillingford, and Nando De Freitas. Learning to learn by gradient descent by gradient descent. In *NIPS*, pages 3981–3989, 2016.

[4] Vasileios Belagiannis, Sikandar Amin, Mykhaylo Andriluka, Bernt Schiele, Nassir Navab, and Slobodan Ilic. 3d pictorial structures for multiple human pose estimation. In *CVPR*, pages 1669–1676, 2014.

[5] Liefeng Bo and Cristian Sminchisescu. Twin gaussian processes for structured prediction. *IJCV*, 87(1-2):28, 2010.

[6] Magnus Burenius, Josephine Sullivan, and Stefan Carlsson. 3D pictorial structures for multiple view articulated pose estimation. In *CVPR*, pages 3618–3625, 2013.

[7] Xipeng Chen, Kwan-Yee Lin, Wentao Liu, Chen Qian, and Liang Lin. Weakly-supervised discovery of geometry-aware representation for 3d human pose estimation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[8] Junting Dong, Wen Jiang, Qixing Huang, Hujun Bao, and Xiaowei Zhou. Fast and robust multi-person 3d pose estimation from multiple views. In *CVPR*, pages 7792–7801, 2019.

[9] Yan Duan, John Schulman, Xi Chen, Peter L Bartlett, Ilya Sutskever, and Pieter Abbeel. Rl^2: Fast reinforcement learning via slow reinforcement learning. *arXiv preprint arXiv:1611.02779*, 2016.

[10] Chelsea Finn, Pieter Abbeel, and Sergey Levine. Model-agnostic meta-learning for fast adaptation of deep networks. In *ICML*, pages 1126–1135. JMLR. org, 2017.

[11] Juergen Gall, Bodo Rosenhahn, Thomas Brox, and Hans-Peter Seidel. Optimization and filtering for human motion capture. *IJCV*, 87(1-2):75, 2010.

[12] Richard Hartley and Andrew Zisserman. *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[13] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, pages 770–778, 2016.

[14] Catalin Ionescu, Dragos Papava, Vlad Olaru, and Cristian Sminchisescu. Human3. 6m: Large scale datasets and predictive methods for 3D human sensing in natural environments. *T-PAMI*, pages 1325–1339, 2014.

[15] Karim Iskakov, Egor Burkov, Victor Lempitsky, and Yury Malkov. Learnable triangulation of human pose. In *ICCV*, 2019.

[16] Max Jaderberg, Karen Simonyan, Andrew Zisserman, et al. Spatial transformer networks. In *NIPS*, pages 2017–2025, 2015.

[17] Hanbyul Joo, Hao Liu, Lei Tan, Lin Gui, Bart Nabbe, Iain Matthews, Takeo Kanade, Shohei Nobuhara, and Yaser Sheikh. Panoptic studio: A massively multiview system for social motion capture. In *ICCV*, pages 3334–3342, 2015.

[18] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. In *ICLR*, 2015.

[19] Yoonho Lee and Seungjin Choi. Gradient-based meta-learning with learned layerwise metric and subspace. In *ICML*, pages 2933–2942, 2018.

[20] Junbang Liang and Ming C Lin. Shape-aware human pose and shape reconstruction using multi-view images. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 4352–4362, 2019.

[21] Yebin Liu, Carsten Stoll, Juergen Gall, Hans-Peter Seidel, and Christian Theobalt. Markerless motion capture of interacting characters using multi-view image segmentation. In *CVPR*, pages 1249–1256. IEEE, 2011.

[22] Julieta Martinez, Rayat Hossain, Javier Romero, and James J Little. A simple yet effective baseline for 3D human pose estimation. In *ICCV*, page 5, 2017.

[23] Nikhil Mishra, Mostafa Rohaninejad, Xi Chen, and Pieter Abbeel. A simple neural attentive meta-learner. In *ICLR*, 2018.

[24] Georgios Pavlakos, Xiaowei Zhou, Konstantinos G. Derpanis, and Kostas Daniilidis. Harvesting multiple views for marker-less 3D human pose annotations. In *CVPR*, pages 1253–1262, 2017.

[25] Haibo Qiu, Chunyu Wang, Jingdong Wang, Naiyan Wang, and Wenjun Zeng. Cross view fusion for 3d human pose estimation. In *ICCV*, pages 4342–4351, 2019.

[26] Sachin Ravi and Hugo Larochelle. Optimization as a model for few-shot learning. In *ICLR*, 2017.

[27] 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. In *CVPR*, pages 8437–8446, 2018.

[28] Adam Santoro, Sergey Bartunov, Matthew Botvinick, Daan Wierstra, and Timothy Lillicrap. Meta-learning with memory-augmented neural networks. In *ICML*, pages 1842–1850, 2016.

[29] Ronald W Schafer et al. What is a savitzky-golay filter. *IEEE Signal processing magazine*, 28(4):111–117, 2011.

[30] Jake Snell, Kevin Swersky, and Richard Zemel. Prototypical networks for few-shot learning. In *NIPS*, pages 4077–4087, 2017.

[31] Qianru Sun, Yaoyao Liu, Tat-Seng Chua, and Bernt Schiele. Meta-transfer learning for few-shot learning. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019.

[32] Denis Tome, Matteo Toso, Lourdes Agapito, and Chris Russell. Rethinking pose in 3D: Multi-stage refinement and recovery for markerless motion capture. In *3DV*, pages 474–483, 2018.

[33] Matthew Trumble, Andrew Gilbert, Adrian Hilton, and John Collomosse. Deep autoencoder for combined human pose estimation and body model upscaling. In *ECCV*, pages 784–800, 2018.

[34] Matthew Trumble, Andrew Gilbert, Charles Malleson, Adrian Hilton, and John Collomosse. Total capture: 3D human pose estimation fusing video and inertial sensors. In *BMVC*, pages 1–13, 2017.

[35] Ricardo Vilalta and Youssef Drissi. A perspective view and survey of meta-learning. *Artificial intelligence review*, 18(2):77–95, 2002.

[36] Shih-En Wei, Varun Ramakrishna, Takeo Kanade, and Yaser Sheikh. Convolutional pose machines. In *CVPR*, pages 4724–4732, 2016.

[37] Donglai Xiang, Hanbyul Joo, and Yaser Sheikh. Monocular total capture: Posing face, body, and hands in the wild. In *CVPR*, 2019.

[38] Bin Xiao, Haiping Wu, and Yichen Wei. Simple baselines for human pose estimation and tracking. In *ECCV*, pages 466–481, 2018.

[39] Yuan Yao, Yasamin Jafarian, and Hyun Soo Park. Monet: Multiview semi-supervised keypoint detection via epipolar divergence. In *ICCV*, pages 753–762, 2019.

Supplementary Material

7. Additional Results on Panoptic Dataset

Table 4. 2D pose estimation accuracy on the Panoptic dataset. The second column represents the number of samples for finetuning the pre-trained model. We report results for three joints and also the average result over all joints.

Methods	Samples	Shld.	Knee.	Ankle.	Avg
No-Fusion	–	89.9	79.8	89.7	88.9
NaiveFuse^{50}	50	88.1	82.3	87.6	85.1
MetaFuse^{50}		91.2	85.7	90.8	**88.9**
NaiveFuse^{100}	100	88.8	83.7	87.6	86.5
MetaFuse^{100}		91.4	86.3	90.9	**89.5**
NaiveFuse^{200}	200	90.9	85.1	88.1	88.3
MetaFuse^{200}		92.2	86.9	91.6	**90.6**
NaiveFuse^{500}	500	91.6	85.8	89.9	90.2
MetaFuse^{500}		93.2	88.0	91.5	**91.2**

Table 5. The 3D pose MPJPE errors obtained by the baseline methods on the Panoptic dataset.

Methods	Samples	Average MPJPE
No-Fusion	–	40.47mm
NaiveFuse^{50}	50	43.39mm
MetaFuse^{50}		**37.27**mm
NaiveFuse^{100}	100	42.58mm
MetaFuse^{100}		**36.02**mm
NaiveFuse^{200}	200	35.60mm
MetaFuse^{200}		**31.78**mm
NaiveFuse^{500}	500	33.50mm
MetaFuse^{500}		**30.88**mm

8. Algorithm of Meta-Training

Algorithm 1 Meta-Training of *MetaFuse*

```
Input: $\{\mathcal{T}_1, \mathcal{T}_2, \cdots, \mathcal{T}_N\}$ : Each $\mathcal{T}_i$ is associated with a
          small dataset from a particular camera pair.
          $\alpha, \beta$ : Step size, hyper-parameters
Output: $\theta, \omega^{base}$ : Pre-trained fusion model
1: Randomly initialize $\theta, \omega^{base}$
2: for each $\mathcal{T}_i \in \{\mathcal{T}_1, \mathcal{T}_2, \cdots, \mathcal{T}_N\}$ do
3:     Sample $K$ images $\mathcal{D}_i^{train}$ from $\mathcal{T}_i$
4:     Compute $\theta', \omega^{base\prime}$ with gradient descent on $\mathcal{D}_i^{train}$
           $\theta' = \theta - \alpha\nabla_\theta\mathcal{L}_{\mathcal{D}_i^{\text{train}}}(\boldsymbol{\omega}^{\text{base}}, \theta)$
           $\boldsymbol{\omega}^{\text{base}\prime} = \boldsymbol{\omega}^{\text{base}} - \alpha\nabla_{\boldsymbol{\omega}^{\text{base}}}\mathcal{L}_{\mathcal{D}_i^{\text{train}}}(\boldsymbol{\omega}^{\text{base}}, \theta)$
5:     Sample other $K$ images $\mathcal{D}_i^{test}$ from $\mathcal{T}_i$
6:     Update $\theta \leftarrow \theta - \beta\nabla_\theta\mathcal{L}_{\mathcal{D}_i^{\text{test}}}(\boldsymbol{\omega}^{\text{base}\prime}, \theta')$
           $\boldsymbol{\omega}^{\text{base}} \leftarrow \boldsymbol{\omega}^{\text{base}} - \beta\nabla_{\omega^{base}}\mathcal{L}_{\mathcal{D}_i^{\text{test}}}(\boldsymbol{\omega}^{\text{base}\prime}, \theta')$
7: end for
8: return $\theta, \omega^{base}$
```