3DFG-PIFu: 3D Feature Grids for Human Digitization from Sparse Views

Kennard Yanting Chan[1,2], Fayao Liu[2], Guosheng Lin[1], Chuan Sheng Foo[2,3], and Weisi Lin[1]

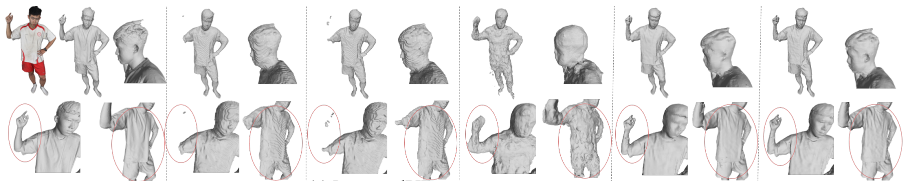
[1] Nanyang Technological University, Singapore
[2] Institute for Infocomm Research, A*STAR
[3] Centre for Frontier AI Research, A*STAR

Abstract. Pixel-aligned implicit models, such as Multi-view PIFu, DeepMultiCap, DoubleField, and SeSDF, are well-established methods for reconstructing a clothed human from sparse views. However, given V images, these models would only combine features from these images in a point-wise and localized manner. In other words, the V images are processed individually and are only combined in a very narrow fashion at the end of the pipeline. To a large extent, this defeats the purpose of having multi-view information since the multi-view task in question is predominantly treated as a single-view task. To resolve this, we introduce 3DFG-PIFu, a pixel-aligned implicit model that exploits multi-view information right from the start and all the way to the end of the pipeline. Our 3DFG-PIFu makes use of 3D Feature Grids to combine features from V images in a global manner (rather than point-wise or localized) and throughout the pipeline. Other than the 3D Feature Grids, 3DFG-PIFu also proposes an iterative mechanism that refines and updates an existing output human mesh using the different views. Moreover, 3DFG-PIFu introduces SDF-based SMPL-X features, which is a new method of incorporating a SMPL-X mesh into a pixel-aligned implicit model. Our experiments show that 3DFG-PIFu significantly outperforms SOTA models. Our code is released at https://github.com/kcyt/3DFG-PIFu.

Keywords: 3D Clothed Human Reconstruction from Sparse Views · 3D Feature Grids · Pixel-aligned Implicit Models

1 Introduction

The field of 3D reconstruction of human bodies has gained considerable interest due to its potential use in various domains such as virtual reality, game production, and 3D printing. Pixel-aligned implicit models, such as Multi-view PIFu [13] DeepMultiCap [22], DoubleField [15], and SeSDF [1] are an influential class of deep learning methods for reconstructing clothed human bodies from sparse views. These models learn an implicit function that represents the surface of a human body. During testing, the learned implicit function is sampled using a grid of uniformly-spaced sample points. For each sample point, the learned implicit function (or the model) will return a predicted occupancy label (i.e.



(a) Groundtruth (b)Multi-view PIFu (c) IntegratedPIFu (Multi-view version) (d) DeepMultiCap (e) Ours (LowRes) (f) Ours (HighRes)

Fig. 1: Our models (last two columns) vs SOTA models.

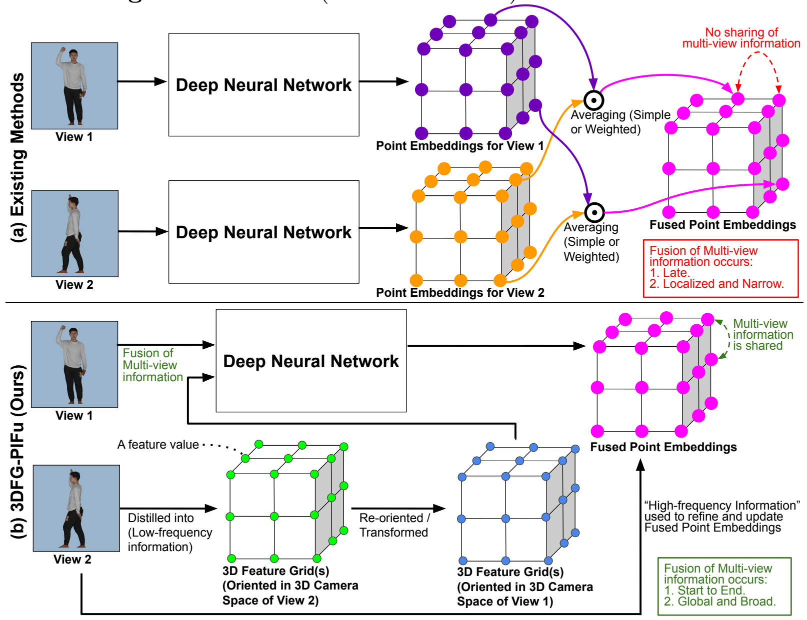


Fig. 2: (a) Existing multi-view pixel-aligned implicit models vs (b) Our 3DFG-PIFu.

whether the sample point is 'inside' or 'outside' of a human body surface). Once a grid of predicted occupancy labels is obtained, a human body mesh can be extracted from this grid using the Marching Cubes algorithm [11].

In order to predict the occupancy labels, existing multi-view pixel-aligned implicit models [1, 13, 15, 22], when given V views or images, would compute V different point embeddings for each sample point. This is illustrated in Fig. 2a for the case where $V = 2$. For each sample point, its V point embeddings would be fused together into a single point embedding via either simple averaging [13] or weighted averaging [1,15,22], as illustrated in the same figure. These fused point embeddings are then converted into predicted occupancy labels, from which a human body mesh can be obtained. It is important to note that the "Point Embeddings for View 1" grid and the "Point Embeddings for View 2" grid in Fig. 2a are in different 3D coordinate spaces. The former is in the 3D camera space of View 1, and the latter is in the 3D camera space of View 2. This means that a point located at the top left corner of a grid may not correspond to the top left corner of another grid. We let the "Fused Point Embeddings" grid follow the 3D camera space of View 1 (It is possible to choose another 3D camera space, but that is trivial).

As shown in Fig. 2a, there are two problems with existing multi-view pixel-aligned implicit models. 1. Firstly, the fusion of View 1 and View 2 are carried out in a point-wise and very localized manner. This is a problem because, as shown in the bidirectional red dashed arrow in Fig. 2a, there is no interaction between fused point embeddings, even if they are located close to each other. So if there is a sample point A that is closely surrounded by ten sample points, the existing multi-view pixel-aligned implicit models may assign those ten points with the same label and yet assign point A with an opposite label, which is an obvious error that would lead to a floating artefact. 2. Secondly, the fusion of View 1 and View 2 occurs at the end of the pipeline in a very simple manner (either simple or weighted averaging (e.g. attention)). To a large extent, the existing multi-view pixel-aligned implicit models are not very different from their single-view counterparts except for the simplistic point-wise fusion of point embeddings at the end of the pipeline.

Hence, we propose 3DFG-PIFu, a pixel-aligned implicit model that rethinks how multi-view information is incorporated in its pipeline. One key feature of 3DFG-PIFu is its use of 3D Feature Grid(s). As seen in Fig. 2b, 3DFG-PIFu makes use of 3D Feature Grid(s) to extract structural information from View 2. The 3D Feature Grid, due to its inherent design, is able to easily orient the extracted information to a different camera space. Thus, we re-orient the 3D Feature Grid from the 3D camera space of View 2 to the 3D camera space of View 1. Now aligned with View 1, the transformed 3D Feature Grid can be concatenated with View 1 and processed by a deep neural network to form 'Fused Point Embeddings'. These fused point embeddings will then be further refined using the fine-grained information from View 2 (Section 3.2).

Crucially, this means that, unlike existing models, the fusion of multi-view information in 3DFG-PIFu occurs from the start to the end of the pipeline. Moreover, the multi-view fusion in 3DFG-PIFu occurs in a global and broad manner (rather than point-wise and localized) as information from View 2 is allowed to influence each and every fused point embedding.

In total, 3DFG-PIFu makes three contributions: 1. The aforementioned 3D Feature Grids that fuse multi-view information (Section 3.1). 2. An iterative mechanism that refines and updates an existing output human mesh using the fine-grained information from the different views (Section 3.2). 3. Introduction of SDF-based SMPL-X features, which is a new method of incorporating a SMPL-X mesh into a pixel-aligned implicit model (Section 3.3).

2 Related Work

2.1 Human Reconstruction from Sparse Views

Methods that reconstruct a human body mesh from a sparse number of images can be broadly classified into two classes: Parametric approaches and non-parametric approaches.

Parametric approaches, such as [7,9,10,20], reconstruct a human body surface by predicting parameters of a human parametric model (e.g. SMPL-X [12]). However, these methods can only produce human body meshes that are clothless.

On the other hand, non-parametric methods do not use a human parametric model. An important subclass of non-parametric methods is pixel-aligned implicit models. There are other subclasses like NERF methods (e.g. [21]), but they have yet to outperform pixel-aligned implicit models.

Pixel-aligned implicit models can be single-view (e.g. [2, 5, 6]) or multi-view (e.g. Multi-view PIFu [13], DeepMultiCap [22], DoubleField [15], and SeSDF [1]).

As a side note, there are also pixel-aligned implicit models that use stereo images to reconstruct a clothed human mesh. However, these models, that include StereoPIFu [8] and DiffuStereo [16], require pairs of images to be taken at two similar viewpoints. This is often infeasible in many real-life applications and is thus not used in our experiments. Instead, our benchmarks are the aforementioned Multi-view PIFu, DeepMultiCap, DoubleField, SeSDF, and a few others.

As mentioned in Section 1 ('Introduction'), these benchmark models suffer from the problems of: 1. Fusing multi-view information in a very narrow or point-wise manner, and 2. Fusing multi-view information only at the very end of the pipeline. To resolve this problem, we introduce 3DFG-PIFu.

3 Method

3DFG-PIFu is a two-staged model that works as long as the number of views $V > 1$ and the camera calibrations are known. One view will be randomly picked as the primary view and the other view(s) will be designated as the secondary view(s). Let us first assume $V = 2$. This means that we have one primary view and one secondary view. As shown in Fig. 3, front and back normal maps, as well as a mask, can be predicted from a RGB image. We use the method outlined in PIFuHD [14] to predict the normal maps. Then, from the predicted normal maps, we can easily extract out the mask. Hereafter, we refer to a view as a collection of a RGB image, a front normal map, a back normal map, and a mask.

1st Stage In the first stage (refer to Fig. 4 and assume $V = 2$), we first generate two 3D feature grids (G_N and G_M) from the secondary view's front normal map and mask. We will elaborate on how G_N and G_M are generated later. In short, the G_N and G_M contain normal pixels and mask pixels, respectively, from the secondary view, but these pixels have been transformed into the 3D camera space of the primary view. Then, G_N and G_M are concatenated with the primary view's RGB image, front normal map, and back normal map. The concatenated output is sent to an encoder, which is a 2D CNN. The encoder will produce a

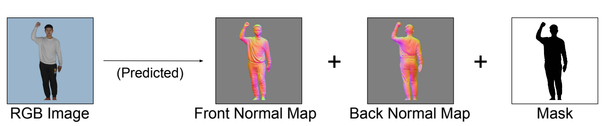


Fig. 3: Predictions from a RGB image.

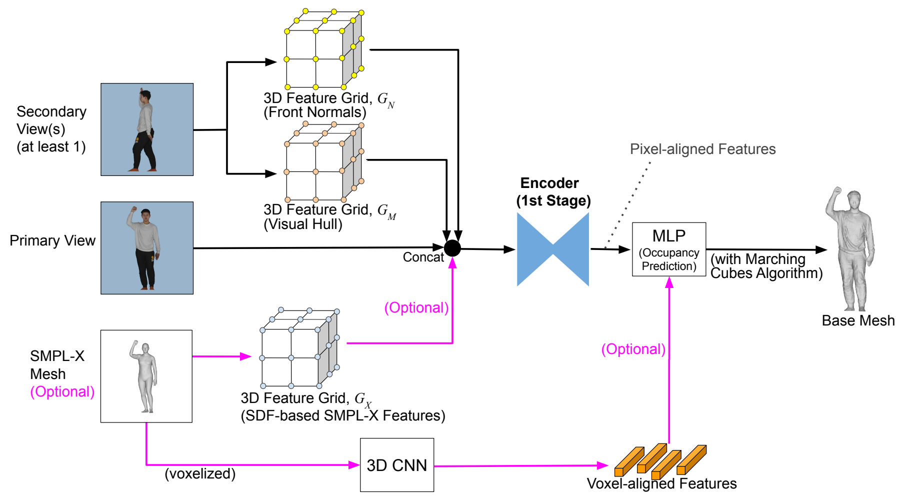


Fig. 4: 1st Stage of 3DFG-PIFu. Each view includes the mask and the predicted front and back normal maps. The primary view and the feature grids extracted from the secondary view(s) are fed into an encoder and MLP to generate a base mesh.

set of feature maps that is used by a Multilayer Perceptron (MLP) to produce a human body mesh, which we refer to as a **base mesh**.

2nd Stage While the **base mesh** from the 1st stage has good structural accuracy, it fails to capture the more fine-grained appearance details (e.g. clothes wrinkles) from all the views. Thus, a 2nd stage is needed. The 2nd stage of 3DFG-PIFu is an iterative mechanism or pipeline to combine appearance details from multiple views. We will briefly describe the flow of the pipeline here, but the details and rationales behind each step in the pipeline will be explained in Section 3.2.

At the start of the 2nd stage, a view is picked from the set of input views. Assume that $V = 2$, and the view that we selected is the secondary view. The secondary view, as well as the base mesh from the 1st stage, will be used as inputs in the 2nd stage (shown in Fig. 5). First, we will rotate (or transform) the base mesh into the 3D camera space of the secondary view. From this **rotated base mesh**, we would generate two additional 3D feature grids (G_V and G'_S). G_V and G'_S will have the visibility information and the SDF values, respectively, of the **rotated base mesh**. We will elaborate on how G_V and G'_S are obtained later. G_V, G'_S and the secondary view will be concatenated together and fed into an encoder, which is a 2D CNN. This encoder will produce a set of feature maps that is used by a MLP to produce a **partial refined mesh**. The partial refined mesh will have the fine-grained appearance details of the secondary view.

Finally, we will obtain a 3D feature grid of SDF values (G_S) from the base mesh. Then, G_S is refined and updated using information from the partial refined mesh via a process that we call visibility-based fusion (to be explained later). Visibility-based fusion will return a final 3D grid of SDF values, G_F. From G_F, we will retrieve the **final mesh** via the Marching Cubes algorithm.

For simplicity, Fig. 5 only shows the scenario where there is only 1 secondary view, and the secondary view (rather than the primary view) is picked at the

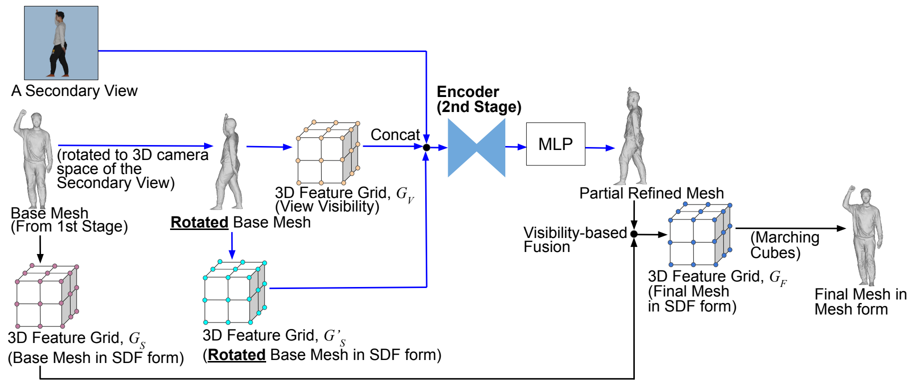


Fig. 5: 2nd Stage of 3DFG-PIFu. The base mesh is first aligned to the secondary view. Once aligned, it is combined with the secondary view to produce a partial refined mesh.

start of the 2nd stage. In reality, the primary view and every secondary view will be separately processed in the 2nd stage (See the blue arrows in Fig. 5), and each view will generate a different partial refined mesh. All these partial refined meshes will be used to refine and update the base mesh during visibility-based fusion before the final mesh is produced.

Optionally, in the 1st stage, if a SMPL-X mesh is given as an input, we will convert the SMPL-X mesh into another 3D feature grid of SDF values (G_X). See illustration in Fig. 4. G_X is what we refer to as **SDF-based SMPL-X Features**. This feature grid will be concatenated with the other inputs in the 1st stage. Concurrently, we use the technique described in PaMIR [23] to obtain voxel-aligned features. The voxel-aligned features will be used by the MLP at the end of the pipeline. We will explain the rationale behind this set-up later.

Now, we will first elaborate on the different 3D feature grids (Sect. 3.1) before moving on to our iterative mechanism described in 3DFG-PIFu's 2nd stage (Sect. 3.2). Finally, we will explain our SDF-based SMPL-X Features (Sect. 3.3).

3.1 3D Feature Grids

In Section 1 ('Introduction'), we thoroughly explained why we need to use 3D feature grids. Indeed, the central theme of our paper revolves around the use of 3D feature grids. We define a 3D feature grid as a D x H x W grid where each element on the grid can be either a scalar value or a vector. D, H, and W are each an integer. A 3D feature grid is useful as it can contain various types of information and can represent these information in different 3D camera spaces. In total, we use **four different types** of 3D feature grids:

1. 3D Feature Grid for Visual Hull (G_M) In Fig. 6, we illustrate how a 3D Feature Grid for Visual Hull is obtained if we only have 2 views - View 1 (Primary view) and View 2 (Secondary view). First, given a 256×256 mask of View 2, we replicate the mask pixels (i.e. the non-empty pixels) 256 times in the z-dimension (camera direction), giving us $M_2 \times 256$ elements in the 3D camera space of View 2, where M_2 represents the number of mask pixels in the mask of View 2. We do

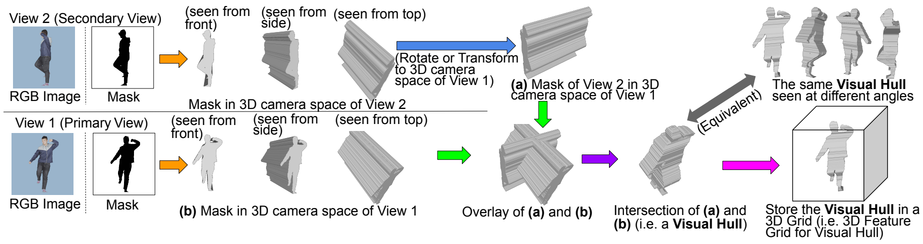


Fig. 6: 3D Feature Grid - Visual Hull. Above shows how G_M is extracted.

the same for View 1 to get $M_1 \times 256$ elements in the 3D camera space of View 1. The elements belonging to View 2 are then rotated or transformed into the 3D camera space of View 1 and placed together with the elements that belong to View 1. Lastly, we take the 3D intersection of the two groups of elements to obtain a visual hull. The visual hull is stored in a 3D grid that corresponds to the 3D camera space of View 1, and this is basically a **3D Feature Grid for Visual Hull** or G_M. We will always store the visual hull in the 3D camera space of the primary view (instead of a secondary view). On a side note, G_M can be generated with more than 2 views too. For example, with 3 views, the visual hull is formed by the 3D intersection of 3 groups of elements.

G_M is useful because it contains the structural information of both View 1 and View 2. Concretely, each element in G_M is a binary value ('0' or '1'). '0' means the element or grid position is unoccupied, and '1' means the element is occupied. Together, these elements represent a possibility space for occupancy. No part of the groundtruth human body mesh can be outside of this possibility space, and this is very useful information for a pixel-aligned implicit model whose task is to predict and reconstruct a human body mesh.

2. 3D Feature Grid for Front Normals (G_N) However, G_M does not fully capture all the relevant information from View 2. Specifically, only mask information from View 2 is captured. If we look at the mask of View 2 in Fig. 6, we cannot actually differentiate the outlines of the arms from that of the torso, or the outlines of the person's left thigh from the right thigh. This information (the outlines) is not captured by the mask but is captured by the front normal map of View 2. Thus, we introduce 3D Feature Grid for Front Normals or G_N as a complement to G_M. G_N is similar to G_M except that each element on its 3D grid is a normal vector rather than a scalar occupancy value. G_N is obtained in a manner similar to the **first row** of Fig. 6 except that the mask of View 2 is replaced by the front normal map of View 2. First, given a $3 \times 256 \times 256$ front normal map of View 2, we replicate the normal pixels (i.e. only the non-empty pixels) 256 times in the z-dimension (camera direction), giving us $N_2 \times 256$ elements (i.e. vectors) in the 3D camera space of View 2, where N_2 represents the number of normal pixels in the front normal map of View 2. The elements belonging to View 2 are then rotated or transformed into the 3D camera space of View 1 and then stored in a 3D feature grid that corresponds to View 1's 3D camera space. This grid is the **3D Feature Grid for Front Normals** or G_N. Like G_M, G_N is also used as an input in the 1st stage of our 3DFG-PIFu.

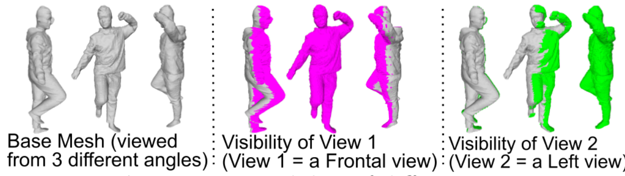


Fig. 7: View visibility of different views

3. 3D Feature Grid for SDF values (G_S, G'_S, G_F) Given an input view, a single-view pixel-aligned implicit model produces a human body mesh that is oriented in the 3D camera space of that input view. But once we have the mesh, we can transform or rotate the mesh into the 3D camera space of any view. This means that a mesh produced using View 1 can be transformed from its initial 3D camera space of View 1 to the 3D camera space of View 2. Once oriented to the 3D camera space of View 2, the mesh becomes a useful prior for a pixel-aligned implicit model that is trying to use View 2 to predict a human body mesh.

But we cannot feed a mesh, which consists of vertices and faces, into a pixel-aligned implicit model. To resolve this, we propose converting the mesh into a 3D feature grid of SDF values. This 3D grid will correspond to the 3D camera space of View 2, and each element in the 3D grid is a truncated SDF value that ranges from -1 to 1, where the value of 0 represents a mesh surface.

This grid, which is **3D Feature Grid for SDF values**, is a simple and effective way to condition a pixel-aligned implicit model on a mesh (as a prior). We will elaborate more on the usefulness of such a prior in Section 3.2. In our 3DFG-PIFu, 3D Feature Grid for SDF values is used as a prior in the 2nd stage (see Fig. 5). In Fig. 5, we see three variants of 3D Feature Grid for SDF values: G_S, G'_S, and G_F. G_S and G'_S represent SDF values from an unrotated and rotated base mesh respectively. G_F represents SDF values from the final mesh.

4. 3D Feature Grid for View Visibility (G_V) In the 2nd stage (see Fig. 5), we use G'_S as an input to the encoder. Since G'_S contains a rotated base mesh in SDF form, we are essentially using the rotated base mesh as a prior in the encoder.

This rotated base mesh already has an accurate structure and shape of a human body. Thus, given a view (e.g. View 2) in the 2nd stage, we only want to modify the rotated base mesh in regions where we are confident of editing. The regions that we are most confident of editing are the regions that are visible from that given view. Examples of such regions are shown in Fig. 7. If we are given View 2, for example, then we only want to edit the green regions of the base mesh, as shown in the rightmost column of Fig. 7.

If the selected view in the 2nd stage is indeed View 2, then we want to have a 3D feature grid that contains all those green regions. Such a 3D feature grid would serve as a complement to the G'_S by telling the pixel-aligned implicit model which part of the rotated base mesh should (and should not) be edited. This 3D feature grid is our **3D Feature Grid for View Visibility** or G_V. Each element in G_V is a binary value (0 or 1). A value of 1 indicates that, at that grid position, there is a mesh surface **and** this mesh surface is visible from the view that is selected in the 2nd stage (as illustrated in Fig. 7).

3.2 An Iterative Mechanism to Combine Appearance Details from Multiple Views

Our iterative mechanism or pipeline to combine appearance details from multiple views is the 2nd stage of our 3DFG-PIFu. It refines and updates the base mesh.

While a base mesh has a highly accurate structure, we observed that it often lacks fine-grained appearance details from all the given input views (primary view and secondary views(s)). This is illustrated in Fig. 9a and b.

To resolve this, we designed a 2nd stage that focuses on capturing the fine-grained appearance details of each view. Our 2nd stage is outlined in Fig. 5. Firstly, we select a view v from the set of input views. Then, as seen in the figure, we condition our encoder on the rotated base mesh (G'_S), which already captured the coarse but accurate structural information from all the input views. Given such a conditioning, we allow the encoder to now focus on capturing fine-grained appearance details from selected view v.

To further ensure that appearance details are captured, we have two additional features in the 2nd stage. The first feature, which was just explained, is the use of G_V (as shown in Fig. 5). By complementing G'_S with G_V, the encoder is able to identify which regions on the rotated base mesh are visible from view v. Relative to invisible regions, the encoder will make less error modifying the visible regions. Thus, knowing where the visible regions are encourages the encoder to make more decisive and sharper modifications to these regions.

The second additional feature in the 2nd stage is the use of Depth Oriented Sampling (DOS) from IntegratedPIFu [4]. As shown in Fig. 5, for a given view, we will generate a partial refined mesh. However, for a partial refined mesh, we are actually only interested in the regions on the mesh that are visible from that given view. For this reason, it makes sense to use DOS to train the encoder and MLP that are used in the 2nd stage. This is because DOS works best when reconstructing mesh surfaces that are directly facing the camera direction (i.e. mesh regions that are visible from the given view). We briefly explain DOS now.

Our 1st stage model predicts coarse-grained occupancy (in or out) of sample points in a 3D space to produce the base mesh. In contrast, our 2nd stage model, with use of DOS, predicts fine-grained displacement values of the sample points in the camera direction to produce a partial refined mesh.

Intuitively, given the base mesh as prior and the use of DOS, our 2nd stage model is trying to shift and adjust the base mesh's surface in the camera direction such that the resulting partial refined mesh better reflects the appearance details of the given views (see Fig. 8).

Visibility-based Fusion Each given view is used to generate a partial refined mesh. We aim to use these partial refined meshes to update the original base mesh. To do so, we transform the partial refined meshes to the primary view's 3D camera space so that they are physically aligned with the base mesh. Then, we will use these partial refined meshes to update the values in G_S, which is a $256 \times 256 \times 256$ 3D feature grid containing the SDF values of the base mesh.

If a partial refined mesh is created from view v, then this mesh will have the most accurate shape and geometry at regions that are visible from view v. For

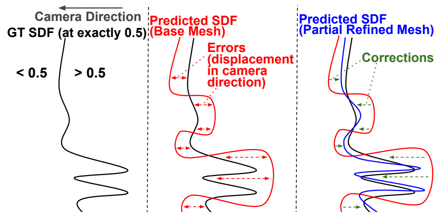


Fig. 8: Illustration of our Iterative Mechanism

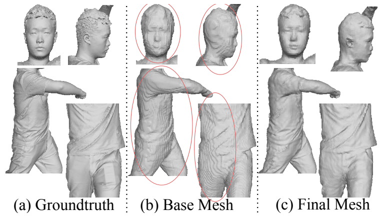


Fig. 9: Evaluation of 3DFG-PIFu's 2nd Stage.

this reason, we will identify locations on a partial refined mesh that are visible from its corresponding view and then extract the SDF values at these locations. So, for each partial refined mesh, these 'visible' SDF values are extracted and used to overwrite the G_S grid. In the end, the updated G_S, which is also referred to as our final mesh in SDF form (G_F), will be a mix of SDF values from the base mesh and the partial refined mesh(es). To convert the G_F to mesh form, we use the Marching Cube algorithm.

3.3 SDF-based SMPL-X features

In multi-view settings, it is possible to use methods, such as [18] and [9], to predict a SMPL-X mesh that is fairly close to the ground truth. Thus, some multi-view pixel-aligned implicit models, like DeepMultiCap [22] and SeSDF [1], use a SMPL-X mesh as a prior before predicting a human body mesh. In 3DFG-PIFu, we also offer an option to use SMPL-X meshes as a prior.

A well-known approach to incorporate a SMPL-X mesh as a prior in a pixel-aligned implicit model is via the use of voxel-aligned features introduced by PaMIR [23]. To obtain the voxel-aligned features, the SMPL-X mesh is first voxelized and then fed as an input to a 3D CNN, as shown in bottom of Fig. 4. Voxel-aligned features are produced by this 3D CNN. The voxel-aligned features are then used as an input to a MLP, which will produce a human body mesh. Voxel-aligned features are used in DeepMultiCap and SeSDF (with a PointNet). We can use voxel-aligned features in 3DFG-PIFu as well, as seen in Fig. 4.

But, as Fig. 4 shows, the features produced by the Encoder ('Pixel-aligned Features') are only fused with voxel-aligned features at the end of the pipeline. Moreover, the fusion is point-wise and localized. This means the pixel-aligned feature that corresponds to a sample point is fused only with the specific voxel-aligned feature that corresponds to the same sample point. In other words, there is no global interaction between voxel-aligned features and pixel-aligned features.

We aim to design a method to fuse a SMPL-X mesh earlier in the pipeline and in a global manner. A recent method that does this is S-PIFu [3]. S-PIFu extracts a set of handcrafted 2D feature maps from a SMPL-X mesh. These maps are concatenated with the input image and then used as inputs at the start of the pipeline. However, useful 3D information is lost when S-PIFu reduces a SMPL-X mesh into a set of 2D handcrafted features. Thus, we propose our **SDF-based SMPL-X features** to directly replace the 2D handcrafted features. SDF-based

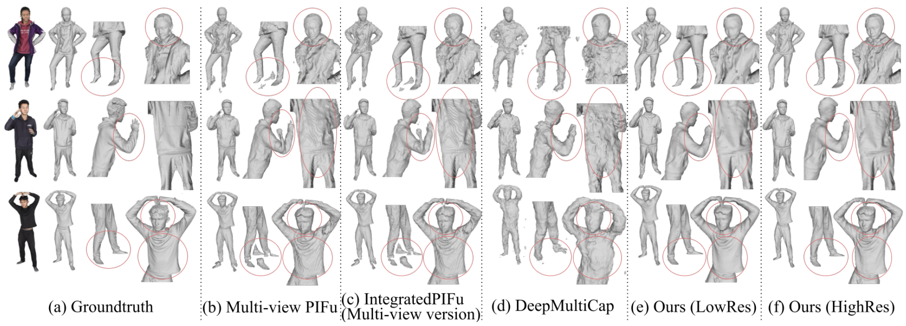


Fig. 10: Qualitative evaluation with SOTA models

SMPL-X features retain 3D information by directly converting a SMPL-X mesh into a 3D feature grid of SDF values.

SDF-based SMPL-X features (G_X) is a 3D grid of SDF values (as seen in Fig. 4). G_X is similar to G_S, G'_S, and G_F except that G_X involves a SMPL-X mesh. To get G_X, we first transform the SMPL-X mesh to the 3D camera space of the primary view. From the transformed SMPL-X mesh, we sample a 3D grid of SDF values. Each SDF value ranges from -1 to 1, where the value of 0 represents a surface on the SMPL-X mesh.

As shown in Fig. 4, G_X can be used together with PaMIR's voxel-aligned features, and we show later on that this combination yields the best results.

4 Experiments

As this is a sparse views set-up, we set the number of views V=2 and set the angle between the two views as 90 degree. It is feasible to use other angles as well. Later in Sect. 4.3, we experiment with V>2.

4.1 Datasets

In our experimental setup, we utilize the THuman2.0 dataset [17] as the training set for our models as well as other competing models. The THuman2.0 dataset comprises 526 high-quality, full-body scans (or meshes) of ethnic Chinese human subjects. A 80-20 train-test split of the dataset is used. For each training mesh, we render 36 RGB images (each spaced 10 degree apart) using a weak-perspective camera. For each training iteration, two views that are 90 degree apart are randomly selected.

Furthermore, we use BUFF dataset [19] and MultiHuman dataset [22] for the evaluation of all models. No model is trained using these datasets. For BUFF dataset, we followed IntegratedPIFu [4] and performed systematic sampling (based on sequence number) on the dataset. This resulted in 101 human meshes that were used for evaluating the models. Utilizing systematic sampling allowed us to avoid meshes that have both the same human subject and the same pose. For MultiHuman dataset, all single human scans are used.

Table 1: SOTA vs Ours. The IntegratedPIFu [4] used is its multi-view version. 'SM' indicates if a groundtruth SMPL-X mesh is used. 'HR' indicates if 1024x1024 RGB images are used. By default, 512x512 RGB images are used.

			THuman2.0			BUFF			MultiHuman		
Methods	SM	HR	CD (10^{-5})	P2S (10^{-5})	Normal	CD (10^{2})	P2S (10^{2})	Normal	CD (10^{-5})	P2S (10^{-5})	Normal
Multi-view PIFu	×	×	10.79	17.03	5471	4.357	7.005	4839	8.197	9.843	6046
IntegratedPIFu	×	✓	10.05	15.75	5324	4.576	7.497	4738	8.481	9.970	5961
DeepMultiCap	✓	×	8.208	7.506	9589	12.45	14.78	12081	32.83	29.16	11518
SeSDF	✓	×	6.302	9.181	5388	3.848	5.779	5259	7.167	9.276	6157
Ours (No HR, No SM)	×	×	5.796	5.811	5386	2.509	2.286	4797	6.320	5.737	5352
Ours (HR, No SM)	×	✓	5.133	5.028	5317	**2.508**	**2.121**	4694	**5.315**	**4.866**	5116
Ours (No HR, w SM)	✓	×	3.560	3.139	5285	3.375	2.694	4758	5.633	5.070	5428
Ours (HR, w SM)	✓	✓	**3.555**	**3.129**	**5212**	3.412	2.700	**4560**	5.391	4.934	**5003**

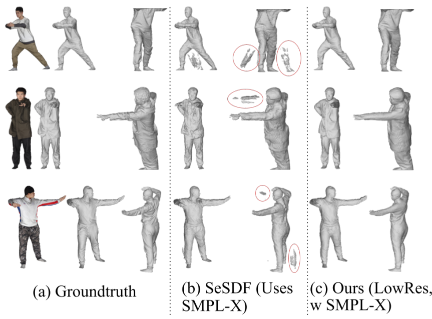


Fig. 11: SeSDF vs our 3DFG-PIFu.

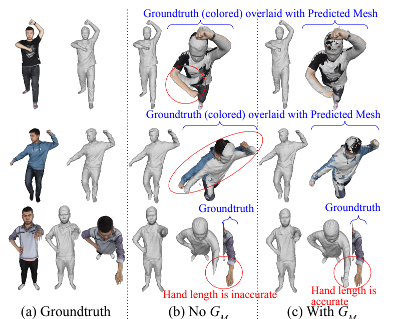


Fig. 12: Qualitative evaluation of G_M

4.2 Comparison with State-of-the-art

We compared our models against other existing models on multi-view clothed human reconstruction. The models we compared with include Multi-view PIFu [13], IntegratedPIFu (multi-view version) [4], DeepMultiCap [22], and SeSDF [1]. We also compared with DoubleField [15] and Data-Driven 3D Reconstruction method [24] in our Supp. Mat. In our quantitative evaluation, we use metrics that include Chamfer distance (CD), Point-to-Surface (P2S), and Normal reprojection error (Normal). These metrics are also used in [1, 4, 13, 22].

Qualitative Evaluation We evaluate the methods qualitatively in Fig. 1 and Fig. 10. In these figures, we show the meshes produced by two of our models. Our first model (in column (e)) uses neither a SMPL-X mesh nor 1024x1024 high-res images. Our second model (in column (f)) does not use a SMPL-X mesh but uses 1024x1024 high-res images. Among the SOTA models, IntegratedPIFu uses high-res images, while DeepMultiCap and SeSDF use a groundtruth SMPL-X mesh. Comparison with SeSDF is shown in Fig. 11. We find that our models outperformed SOTA models in both structural accuracy and appearance details.

Quantitative Evaluation In Tab. 1, we compared our models with existing methods quantitatively. Because different SOTA methods require different types of inputs (i.e. groundtruth SMPL-X or high-res images), and these different inputs may give additional advantage to a method, we decided to train four different versions of our model, with each version using a different combination of inputs as shown in the table. The table shows that our methods significantly outperform the existing models in all three datasets. See Supp. Mat. for more analysis.

Table 2: Quantitative evaluation of G_M

Methods	THuman2.0		BUFF	
	CD (10^{-5})	P2S (10^{-5})	CD (10^{2})	P2S (10^{2})
PIFu	26.97	25.10	9.651	9.247
PIFu + G_M (G_N not used)	6.626	7.212	2.530	3.005
PIFu + G_M (G_N is used)	**6.007**	**6.634**	**2.386**	**2.999**

Table 3: Quantitative evaluation of G_V **at visible regions**

Methods	THuman2.0		BUFF	
	CD (10^{-5})	P2S (10^{-5})	CD (10^{-4})	P2S (10^{-4})
No G_V	4.036	2.964	1.315	1.096
With G_V	**3.891**	**2.840**	**1.284**	**1.056**

4.3 Ablation Studies

Evaluation of the Different 3D Feature Grids Firstly, in order to assess the effectiveness of G_M, we train and compare a single-view PIFu that is either not given or given G_M as an additional input. The comparison is shown quantitatively in the first two rows of Tab. 2 and qualitatively in Fig. 12. Notably, with G_M, the single-view PIFu can also outperform a Multi-view PIFu (1st row of Tab. 1).

Next, as aforementioned, G_M can be complemented with G_N. Thus, we also show the results when G_M is used with G_N in a single-view PIFu (see last row of Tab. 2). The results clearly demonstrated the benefit of including G_N.

We also evaluated G_V by training the 2nd stage of 3DFG-PIFu with or without G_V. The results in Tab. 3 and Fig. 13 show that G_V improves the partial refined meshes obtained in the 2nd stage. Aside: As only visible regions of partial refined meshes are used to form the final mesh, Tab. 3 must consider only visible regions.

Evaluating our Iterative Mechanism (i.e. Our 2nd Stage) We also show that 3DFG-PIFu's 2nd stage indeed improves the base meshes from the 1st stage. See Fig. 9 and Tab. 4. The improved meshes show sharper appearance details.

When more views are made available (i.e. $V{>}2$), the 3DFG-PIFu can incrementally update and improve the current mesh without the need for additional training. We simply replace the base mesh with the current mesh and re-run the 2nd stage again. Results are shown in Fig 14.

Evaluation of SDF-based SMPL-X features In order to evaluate the effectiveness of our SDF-based SMPL-X features (G_X), we train and compare a single-view PIFu that is given either (i) S-PIFu features, (ii) PaMIR's voxel-aligned features, (iii) our G_X, or (iv) PaMIR's voxel-aligned features + our G_X.

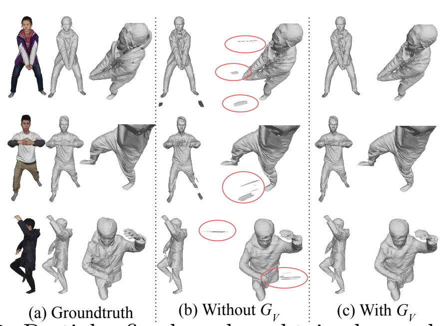


Fig. 13: Partial refined meshes obtained w and w/o G_V

A quantitative comparison is shown in Tab. 5. The table shows our G_X outperformed S-PIFu features. Whether our G_X is combined with voxel-aligned features or not, it clearly improves the performance of a model when in use. Qualitatively, Fig. 15 shows that combining PaMIR's voxel-aligned features with our G_X yields the most robust results.

5 Limitations and Conclusion

In our Supp. Mat., we address concerns on 3DFG-PIFu's efficiency. In short, via a series of implementation tricks, we show 3DFG-PIFu is actually more efficient than roughly half of the existing SOTA methods.

We have introduced 3DFG-PIFu, a multi-view pixel-aligned implicit model that uses 3D Feature Grids to fuse multi-view information. 3DFG-PIFu also proposed an iterative pipeline that combines appearance details from multiple views into a single mesh. Lastly, 3DFG-PIFu introduced SDF-based SMPL-X features, which is a new way of incorporating a SMPL-X mesh into a pixel-aligned implicit model.

Table 4: Quantitative evaluation of 3DFG-PIFu's 2nd Stage

Methods	THuman2.0 CD (10^{-5})	THuman2.0 P2S (10^{-5})	BUFF CD (10^{2})	BUFF P2S (10^{2})
Base meshes (1st Stage)	6.007	6.634	**2.386**	2.999
Final meshes (2nd Stage)	**5.796**	**5.811**	2.509	**2.286**

Table 5: Quantitative evaluation of G_X

Methods	THuman2.0 CD (10^{-5})	THuman2.0 P2S (10^{-5})	BUFF CD (10^{2})	BUFF P2S (10^{2})
S-PIFu Features	4.488	4.030	6.812	6.880
Voxel-aligned Features	4.104	3.740	8.037	9.225
G_X (Ours)	4.352	3.734	**6.351**	**6.641**
Voxel-aligned Features + G_X (Ours)	**3.970**	**3.483**	8.012	8.827

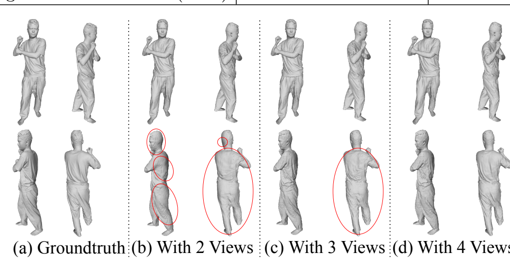


Fig. 14: Effect of using more views in 3DFG-PIFu.

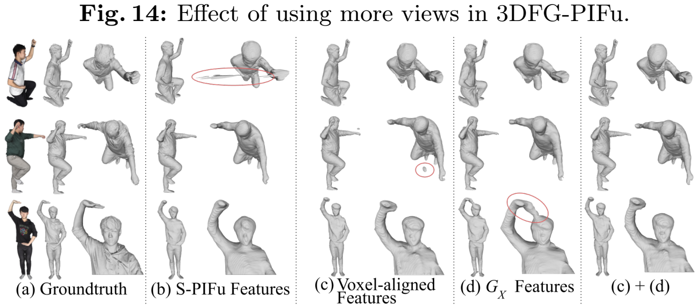


Fig. 15: Qualitative evaluation of G_X

Acknowledgements This research work is supported by the Agency for Science, Technology and Research (A*STAR) under its MTC Programmatic Funds (Grant No. M23L7b0021).

References

1. Cao, Y., Han, K., Wong, K.Y.K.: Sesdf: Self-evolved signed distance field for implicit 3d clothed human reconstruction. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 4647–4657 (2023)
2. Chan, K., Lin, G., Zhao, H., Lin, W.: S-pifu: Integrating parametric human models with pifu for single-view clothed human reconstruction. Advances in Neural Information Processing Systems **35**, 17373–17385 (2022)
3. Chan, K., Lin, G., Zhao, H., Lin, W.: S-pifu: Integrating parametric human models with pifu for single-view clothed human reconstruction. In: Advances in Neural Information Processing Systems (2022)
4. Chan, K.Y., Lin, G., Zhao, H., Lin, W.: Integratedpifu: Integrated pixel aligned implicit function for single-view human reconstruction. In: Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part II. pp. 328–344. Springer (2022)
5. Chan, K.Y., Liu, F., Lin, G., Foo, C.S., Lin, W.: Fine structure-aware sampling: A new sampling training scheme for pixel-aligned implicit models in single-view human reconstruction. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 38, pp. 964–971 (2024)
6. Chan, K.Y., Liu, F., Lin, G., Foo, C.S., Lin, W.: R-cyclic diffuser: Reductive and cyclic latent diffusion for 3d clothed human digitalization. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 10304–10313 (2024)
7. Gong, X., Song, L., Zheng, M., Planche, B., Chen, T., Yuan, J., Doermann, D., Wu, Z.: Progressive multi-view human mesh recovery with self-supervision. In: Proceedings of the AAAI Conference on Artificial Intelligence. vol. 37, pp. 676–684 (2023)
8. Hong, Y., Zhang, J., Jiang, B., Guo, Y., Liu, L., Bao, H.: Stereopifu: Depth aware clothed human digitization via stereo vision. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 535–545 (2021)
9. Kolotouros, N., Pavlakos, G., Jayaraman, D., Daniilidis, K.: Probabilistic modeling for human mesh recovery. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 11605–11614 (2021)
10. Liang, J., Lin, M.C.: Shape-aware human pose and shape reconstruction using multi-view images. In: Proceedings of the IEEE/CVF international conference on computer vision. pp. 4352–4362 (2019)
11. Lorensen, W.E., Cline, H.E.: Marching cubes: A high resolution 3d surface construction algorithm. ACM siggraph computer graphics **21**(4), 163–169 (1987)
12. Pavlakos, G., Choutas, V., Ghorbani, N., Bolkart, T., Osman, A.A.A., Tzionas, D., Black, M.J.: Expressive body capture: 3D hands, face, and body from a single image. In: Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR). pp. 10975–10985 (2019)
13. Saito, S., Huang, Z., Natsume, R., Morishima, S., Kanazawa, A., Li, H.: Pifu: Pixel-aligned implicit function for high-resolution clothed human digitization. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 2304–2314 (2019)

14. Saito, S., Simon, T., Saragih, J., Joo, H.: Pifuhd: Multi-level pixel-aligned implicit function for high-resolution 3d human digitization. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 84–93 (2020)
15. Shao, R., Zhang, H., Zhang, H., Chen, M., Cao, Y.P., Yu, T., Liu, Y.: Doublefield: Bridging the neural surface and radiance fields for high-fidelity human reconstruction and rendering. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 15872–15882 (2022)
16. Shao, R., Zheng, Z., Zhang, H., Sun, J., Liu, Y.: Diffustereo: High quality human reconstruction via diffusion-based stereo using sparse cameras. In: Computer Vision–ECCV 2022: 17th European Conference, Tel Aviv, Israel, October 23–27, 2022, Proceedings, Part XXXII. pp. 702–720. Springer (2022)
17. Yu, T., Zheng, Z., Guo, K., Liu, P., Dai, Q., Liu, Y.: Function4d: Real-time human volumetric capture from very sparse consumer rgbd sensors. In: IEEE Conference on Computer Vision and Pattern Recognition (CVPR2021) (June 2021)
18. Yu, Z., Zhang, L., Xu, Y., Tang, C., Tran, L., Keskin, C., Park, H.S.: Multiview human body reconstruction from uncalibrated cameras. In: Advances in Neural Information Processing Systems (2022)
19. Zhang, C., Pujades, S., Black, M.J., Pons-Moll, G.: Detailed, accurate, human shape estimation from clothed 3d scan sequences. In: The IEEE Conference on Computer Vision and Pattern Recognition (CVPR) (July 2017)
20. Zhang, H., Tian, Y., Zhou, X., Ouyang, W., Liu, Y., Wang, L., Sun, Z.: Pymaf: 3d human pose and shape regression with pyramidal mesh alignment feedback loop. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 11446–11456 (2021)
21. Zhao, F., Yang, W., Zhang, J., Lin, P., Zhang, Y., Yu, J., Xu, L.: Humannerf: Efficiently generated human radiance field from sparse inputs. In: Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. pp. 7743–7753 (2022)
22. Zheng, Y., Shao, R., Zhang, Y., Yu, T., Zheng, Z., Dai, Q., Liu, Y.: Deepmulticap: Performance capture of multiple characters using sparse multiview cameras. In: Proceedings of the IEEE/CVF International Conference on Computer Vision. pp. 6239–6249 (2021)
23. Zheng, Z., Yu, T., Liu, Y., Dai, Q.: Pamir: Parametric model-conditioned implicit representation for image-based human reconstruction. IEEE transactions on pattern analysis and machine intelligence **44**(6), 3170–3184 (2021)
24. Zins, P., Xu, Y., Boyer, E., Wuhrer, S., Tung, T.: Data-driven 3d reconstruction of dressed humans from sparse views. In: 2021 International Conference on 3D Vision (3DV). pp. 494–504. IEEE (2021)