POEM: Reconstructing Hand in a Point Embedded Multi-view Stereo

Lixin Yang[1,2] Jian Xu[3] Licheng Zhong[1] Xinyu Zhan[1] Zhicheng Wang[3] Kejian Wu[3] Cewu Lu[1,2†]
[1]Shanghai Jiao Tong University [2]Shanghai Qi Zhi Institute [3]Nreal
{siriusyang, zlicheng, kelvin34501, lucewu}@sjtu.edu.cn
{jianxu, kejian}@nreal.ai chgggo@gmail.com

2023 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR) | 979-8-3503-0129-8/23/$31.00 ©2023 IEEE | DOI: 10.1109/CVPR52729.2023.02022

Abstract

*Enable neural networks to capture 3D geometrical-aware features is essential in multi-view based vision tasks. Previous methods usually encode the 3D information of multi-view stereo into the 2D features. In contrast, we present a novel method, named POEM, that directly operates on the 3D **PO**ints **E**mbedded in the **M**ulti-view stereo for reconstructing hand mesh in it. Point is a natural form of 3D information and an ideal medium for fusing features across views, as it has different projections on different views. Our method is thus in light of a simple yet effective idea, that a complex 3D hand mesh can be represented by a set of 3D points that 1) are embedded in the multi-view stereo, 2) carry features from the multi-view images, and 3) encircle the hand. To leverage the power of points, we design two operations: point-based feature fusion and cross-set point attention mechanism. Evaluation on three challenging multi-view datasets shows that POEM outperforms the state-of-the-art in hand mesh reconstruction. Code and models are available for research at* github.com/lixiny/POEM

1. Introduction

Hand mesh reconstruction plays a central role in the field of augmented and mixed reality, as it can not only deliver realistic experiences for the users in gaming but also support applications involving teleoperation, communication, education, and fitness outside of gaming. Many significant efforts have been made for the monocular 3D hand mesh reconstruction [1, 5, 7, 9, 31, 32]. However, it still struggles to produce applicable results, mainly for these three reasons. **(1)** *Depth ambiguity.* Recovery of the absolute position in a monocular camera system is an ill-posed problem. Hence, previous methods [9, 31, 54] only recovered the hand vertices relative to the wrist (*i.e.* root-relative). **(2)** *Unknown perspectives.* The shape of the hand's 2D projection is highly dependent on the camera's perspective model (*i.e.* camera intrinsic matrix). However, the monocular-based methods usually suggest a weak perspective projection [1, 27], which is not accurate enough to recover the hand's 3D structure. **(3)** *Occlusion.* The occlusion between the hand and its interacting objects also challenges the accuracy of the reconstruction [32]. These issues limit monocular-based methods from practical application, in which the absolute and accurate position of the hand surface is required for interacting with our surroundings.

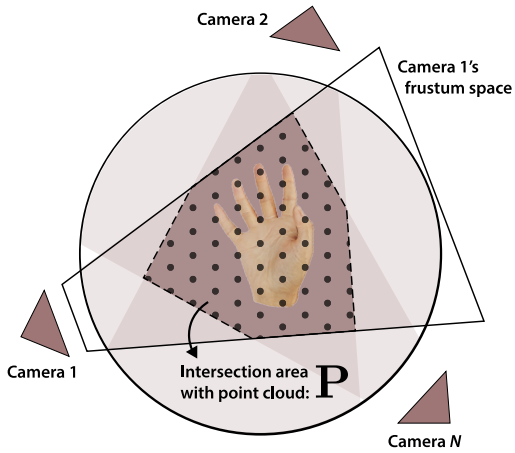


Figure 1. **Intersection area of *N* cameras' frustum spaces.** The gray dots represent the point cloud **P** aggregated from *N* frustums. Our method: POEM, standing for the point embedded multi-view stereo, focuses on the dark area scatted with gray dots.

Our paper is thus focusing on reconstructing hands from multi-view images. Motivation comes from two aspects. First, the issues mentioned above can be alleviated by leveraging the geometrical consistency among multi-view images. Second, the prospered multi-view hand-object tracking setups [2, 4, 49, 55] and VR headsets bring us an urgent demand and direct application of multi-view hand reconstruction in real-time. A common practice of multi-view 3D pose estimation follows a two-stage design. It first estimates the 2D key points of the skeleton in each view and then back-project them to 3D space through several 2D-to-3D lifting methods, *e.g.* algebraic triangulation [17, 18, 39], Pictorial Structures Model (PSM) [33, 38], 3D CNN [18, 43],

†Cewu Lu is the corresponding author, the member of Qing Yuan Research Institute and MoE Key Lab of Artificial Intelligence, AI Institute, Shanghai Jiao Tong University, China and Shanghai Qi Zhi institute.

979-8-3503-0129-8/23/$31.00 ©2023 IEEE
DOI 10.1109/CVPR52729.2023.02022

Authorized licensed use limited to: ShanghaiTech University. Downloaded on October 19,2026 at 14:41:09 UTC from IEEE Xplore. Restrictions apply.

plane sweep [26], etc. However, these two-stage methods are not capable of reconstructing an animatable hand mesh that contains both skeleton and surface. It was not until recently that a one-stage multi-view mesh regression model was proposed [45].

How to effectively fuse the features from different images is a key component in the multi-view setting. Accordingly, previous methods can be categorized into three types. **(1) Fusing in 2D.** The features are directly fused in the 2D domain using explicit epipolar transform [17, 38] or implicit representations that encode the camera transformation (*i.e.* camera intrinsic and extrinsic matrix) into 2D features, *e.g.* feature transform layer (FTL) [14, 39] and 3D position embedding (RayConv) [45]; **(2) Fusing in 3D.** The features are fused in a 3D voxel space via PSM [33, 38] or 3D CNNs [18, 43]; **(3) Fusing via 3D-2D projection.** The features are fused by first projecting the 3D keypoints' initial guess into each 2D plane and then fusing multi-view features near those 2D locations [45];

The fusion mode in type 1 is considered as *holistic*, since it indiscriminately fuses all the features from different views. Consequentially, it ignores the structure of the underlying hand model that we are interested in. On the contrary, the fusion mode in type 3 is considered as *local*. However, only the features around the 2D keypoints are hard to capture the consistent geometrical features from a global view. Besides, the 3D keypoints initial guess may not be accurate enough, resulting in the fusion being unstable. The fusion mode in type 2 is not in our consideration as it tends to be computationally expensive and suffers from quantization error.

Based on the above discussion, we aim to seek a feature representation and a fusion mode between type 1 and type 3 for both holistically and locally fusing the features in multi-views, and to explore a framework for robust and accurate hand mesh reconstruction. Our method is called POEM, standing for ***PO**int **E**mbedded **M**ulti-view Stereo*. We draw inspiration from the Basis Point Set (BPS) [34], which bases on a simple yet effective idea that a complex 3D shape can be represented by a fixed set of points (BPS) that wraps the shape in it. If we consider the intersection of different cameras' frustum spaces as a point cloud, and the hand's vertices as another point cloud, then the intersected space is the basis point set for hand vertices (see Fig. 1). Once we assign the multi-view image features to the point cloud in the intersected space, fusing image features across different views becomes fusing the point features from different camera frustums. The advantages of this representation are two-fold: **(i)** The hand is wrapped in a dense point cloud (set) that carries dense image features collected from different views, which is more holistic and robust than the local fusion mode in type 3. **(ii)** For each vertex on the hand surface, it interacts with basis points in its local neighborhood (*i.e.* k nearest neighbor), which is more selective than the holistic fusion mode in type 1.

Fig. 2 shows our model's architecture. POEM consists of two stages. In the first stage (Sec. 3.2), POEM takes the multi-view images as input and predicts the 2D keypoints of the hand skeleton in each view. Then, the 3D keypoints are recovered by an algebraic triangulation module. In the second stage, POEM fuses the features from different views in a space embedded by points and predicts the hand mesh in this space (Sec. 3.3). The point feature on hand vertices will iteratively interact with the features of the embedded points through a cross-set attention mechanism, and the updated vertex features are further used by POEM to predict the vertex's refined position (Sec. 3.3.3).

We conduct extensive experiments on three multi-view datasets for hand mesh reconstruction under the object's occlusion, namely HO3D [12], DexYCB [4], and OakInk [49]. With the proposed fusion mode and attention mechanism, POEM achieves state-of-the-art on all three datasets.

Our contributions are in three-fold:

- We investigate the multi-view pose and shape reconstruction problem from a new perspective, that is, the interaction between a target point set (*i.e.* mesh vertices) and a basis point set (*i.e.* point cloud in the camera frustum spaces).
- According to that, we propose an end-to-end learning framework: POEM for reconstructing hand mesh from multi-view images through a point embedded multi-view stereo. To encourage interaction between two point sets, POEM introduces two new operations: a point-based feature fusion strategy and a cross-set point attention.
- We conduct extensive experiments to demonstrate the efficacy of the architecture in POEM. As a regression model targeting mesh reconstruction, POEM achieves significant improvement compared to the previous state-of-the-art.

2. Related Work

Multi-view Feature Processing. Representing the observations from different camera systems in a unified way while fusing multi-view features accordingly is a common problem in multi-view stereo (MVS) reconstruction and pose estimation. This literature review focuses on addressing this key challenge. From this perspective, previous methods - in which the camera transformation (extrinsic) is typically encoded differently - can be seen as different types of Position Embedding (PE). For example, the method based on epipolar transform [17] can be classified as a line-formed PE, as pixels in one camera are encoded as epipolar lines in others. Additionally, there are point-formed position embeddings such as FTL in [14, 39], RayConv in [45], and 3D Position Encoder in PETR [29], which apply camera extrinsic directly to point-shaped features (FTL) or add

Authorized licensed use limited to: ShanghaiTech University. Downloaded on October 19,2026 at 14:41:09 UTC from IEEE Xplore. Restrictions apply.

camera ray vectors channel-wise to the features (RayConv, PETR). These two point-formed PEs use point-formed features solely in 2D format because 2D convolution or image-based self-attention cannot capture 3D structure. Therefore they are considered implicit. In contrast, 3D CNN and point cloud network can preserve 3D structure. SurfaceNet [19] and LSM [20] associate features from different views by forming a cost volume and rely on 3D CNNs to perform voxel-wise reconstruction. To address the drawback of the final volumetric output, MVSNet [51] predicts the depth-map instead of voxels. These methods are classified as voxel-formed PEs. Finally, the explicit point-formed PE directly uses a set of 3D points in the MVS scene. For instance, PointMVS [6] unprojects the predicted depth-map to a point cloud and aggregates features from different views using project-and-fetch. Our method also belongs to this type. In our task, preserving the topology of the hand vertex points is crucial, but this can be challenging with PointMVS, which indiscriminately treats points. Instead, our approach, POEM, represents the common-view scene as an unstructured point cloud for feature aggregation and employs a structure-aware vertex query to initialize and update hand vertices. POEM's structured vertices interact with unstructured frustum points through cross-attention, which effectively removes object occlusion and accurately reconstructs the hand mesh.

Monocular Hand Reconstruction. Monocular hand reconstruction has been a long-studied topic. A series of works [1, 16, 23, 54] were built upon the deformable hand mesh with a differentiable skinning function, *e.g.* MANO [40]. However, the difficulty of regressing the non-Euclidean rotations hinders the performance of these methods. There have been emerging works exploring the direct reconstruction of the hand surface. As vertices naturally lie in 3D Euclidean space, [7, 9, 22, 24] leveraged the mesh structure of MANO with the graph-based convolution networks (GCN). Besides, the voxels [31], UV positional maps [5], and signed distance function [8, 21] were also competent choices. Recently, Transformers [27, 28] have been deployed to fuse the features on the hand surface by self-attention mechanism. In this work, we follow the path of direct mesh reconstruction with Transformer and propose to model hand mesh as a **point set** in the multi-view stereo.

Point Cloud Processing. Qi *et al*. [35] proposed PointNet, the first deep model utilizing the permutation-invariant structure of point cloud data. A subsequent work, PointNet++, was proposed in [36] thereafter. PointNet++ introduced ball query and hierarchical grouping, enabling the model to reason form local structures of point clouds. There were a number of successive works [25,42,48] trying to define local convolution operators on point clouds to extract local information. The recent application of transformers on point cloud data has been proven a success. Zhao *et al*. proposed Point Transformer [53], which adopts a vector attention mechanism to perform attention in the local neighborhood. Guo *et al*. proposed the Point Cloud Transformer [10], which used an analogy of the Laplacian matrix on point clouds to fuse long-range relationships in the point cloud. Our method follows the design in Point Transformer, but selectively fuses the points from different cameras to a hand vertex through a **cross-set** vector attention.

3. Method

3.1. Formulation

The general purpose of this paper is to model the joint distribution of hand skeleton and surface under multi-view observations. Given N cameras with different positions and orientations, we first define the 3D coordinates of the hand's skeleton and surface inside a shared *3D world space*: $\mathcal{W}$. Accordingly, their coordinates inside each camera frame can be retrieved by applying extrinsic transformation. Let $\mathbf{X} \in \mathbb{R}^{21\times 3}$ denotes the 3D keypoints on the hand skeleton, and let $\mathbf{V} \in \mathbb{R}^{778\times 3}$ denotes the 3D vertices on the hand surface mesh, given a image set $\boldsymbol{\mathcal{I}} = \{\mathbf{I}_v\}_{v=1}^{N}$ from total N views, the proposed model predicts a distribution $P(\mathbf{X}, \mathbf{V}|\boldsymbol{\mathcal{I}})$. However, directly model this joint distribution is hard and suffers from inferior performance (see Exp. Sec. 4.2-B). Hence, based on the chain rule, we decompose the joint distribution as follows:

$$P(\mathbf{X}, \mathbf{V}|\boldsymbol{\mathcal{I}}) = P_{\boldsymbol{\omega}}(\mathbf{V}|\mathbf{X}, \boldsymbol{\mathcal{I}})P_{\boldsymbol{\phi}}(\mathbf{X}|\boldsymbol{\mathcal{I}}). \tag{1}$$

This equation expresses POEM's architecture (shown in Fig. 2), which consists of two stages. The training process is to fit the learnable parameters $\boldsymbol{\phi}, \boldsymbol{\omega}$ on the training data, *i.e.* for each input $\boldsymbol{\mathcal{I}}$, the model maximizes its probability at the ground-truth $\widehat{\mathbf{X}}, \widehat{\mathbf{V}}$. We represent these two stages: $P_{\boldsymbol{\phi}}$ and $P_{\boldsymbol{\omega}}$ as two neural networks: $f_{\boldsymbol{\phi}}(\boldsymbol{\mathcal{I}})$ and $f_{\boldsymbol{\omega}}(\mathbf{X}, \boldsymbol{\mathcal{I}})$. The fitting objective are set to the standard $l1$ loss:

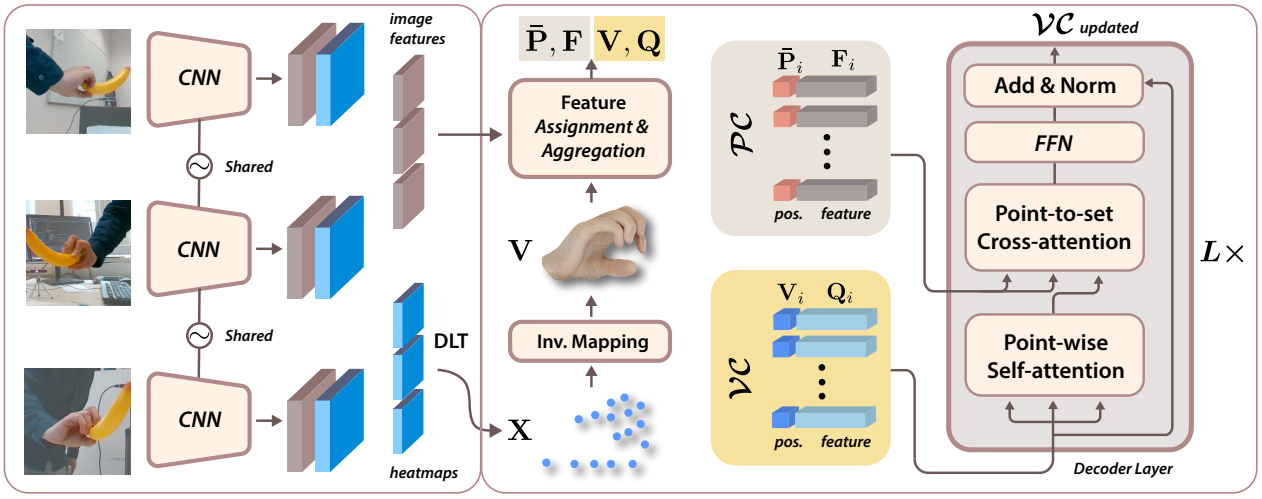
$$\mathcal{L} = \left|\widehat{\mathbf{V}} - f_{\boldsymbol{\omega}}(\mathbf{X}, \boldsymbol{\mathcal{I}})\right| + \left|\widehat{\mathbf{X}} - f_{\boldsymbol{\phi}}(\boldsymbol{\mathcal{I}})\right|. \tag{2}$$

Having outlined the problem decomposition and learning objectives, we now move on to discuss details of the network design.

3.2. Keypoints from M.v. Images

The first stage of POEM is to predict the 3D keypoints $\mathbf{X}$ conditioned on the multi-view (*m.v.*) images $\boldsymbol{\mathcal{I}}$. To achieve this, we first estimate the $\mathbf{x}$'s 2D location in each view $\mathbf{I}_i$ and then lift them to the 3D space through algebraic triangulation. To increase 2D keypoints' robustness against occlusion, we leverage 2D likelihood heatmap [47] and *soft-argmax* operation [41] for $\mathbf{x}$'s 2D location. Inside the first stage, we suppose that the 2D keypoints from each view's prediction have independent contribution to $\mathbf{X}$. Hence we can apply the direct linear transformation (DLT, [15, p.312]) for fast triangulation. Formally, POEM's first stage $f_{\boldsymbol{\phi}}(\boldsymbol{\mathcal{I}})$

Authorized licensed use limited to: ShanghaiTech University. Downloaded on October 19,2026 at 14:41:09 UTC from IEEE Xplore. Restrictions apply.



(a) Stage 1: $\mathbf{X} = f_{\phi}(\mathcal{I})$ (b) Stage 2: $\mathbf{V} = f_{\omega}(\mathbf{X}, \mathcal{I})$

Figure 2. POEM's architecture contains two stages: (a) hand's keypoints prediction conditioned on multi-view images, and (b) hand's vertices prediction conditioned on the predicted keypoints and multi-view images features.

can be expressed as:

$$f_{\phi} : \mathbf{X} = \mathrm{DLT}(\mathbf{h}_{1\sim N}, \mathbf{K}_{1\sim N}, \mathbf{T}_{1\sim N}), \quad \text{where } \mathbf{h}_i = \textit{soft-argmax}\,(\mathcal{F}_{\phi}(\mathbf{I}_i)), \tag{3}$$

in which $\mathcal{F}_{\phi}$ is a backbone network with weights ϕ, $\mathbf{K}_i$, $\mathbf{T}_i$ is the camera intrinsic and extrinsic matrix of i-th camera, respectively.

3.3. Vertices from Keypoints and M.v. Images

In the previous stage, we built a model that can robustly retrieve the hand's keypoints $\mathbf{X}$ from multi-view images. However, only the keypoints $\mathbf{X}$ is deficient for our task in two aspects: **(1)** keypoints are not on the surface, which can hardly reflect the shape of hand; **(2)** the keypoints are retrieved independently from the backbone model, lacking the information fused from different views; Based on (2), the keypoints may not be accurate enough.

Given such deficiencies, inside the second stage, we want the latent embeddings of hand vertices $\mathbf{V}$ can fully interact with the features from all views before reaching its outputs: $\mathbf{V}$. We start by reviewing the problem setting that each image observes a frustum space in front of a camera. As shown in Fig. 1, the common views among all N cameras reflect the *intersection* area of N frustum spaces. The hand that we are interested in lies in it.

To enable the network to operate on the frustum space, we discretize it into a 3D meshgrid. Each cell in the meshgrid store a 3D coordinate of a point. With the camera extrinsic, we can transform all the points from total N camera frustums into the shared world space $\mathcal{W}$ (details in Sec. 3.3.1). Therefore, the hand is surrounded by a point cloud $\mathbf{P}$ aggregated from N frustums. Now let us revisit the form of $\mathbf{V}$. If we consider the 778 vertices of $\mathbf{V}$ as another point cloud in $\mathcal{W}$, the goal of the second stage becomes encouraging the interaction between two point clouds: One from N camera frustums, the other from the surface of hand.

Representing the camera frustum and hand vertices in point cloud has two advantages. **(1)** Point cloud is invariant to the permutation of points. It only subject to the relative distance between points. Hence, points from different camera frustums can be easily fused to the points in $\mathbf{V}$ if they are spatially close; **(2)** Point cloud is a set of 3D points with its coordinates as a natural positional encoding, which is quite suitable for the mechanism of self and cross-attention; Given these advantages, we design a cross-set point Transformer to fuse features in point cloud (Sec. 3.3.3).

One last issue is how to embed features into the point clouds $\mathbf{P}$ and $\mathbf{V}$, since currently they are just a set of 3D coordinates. We investigate two ways in the Sec. 3.3.2. We start by introducing a *Position Embedded Aggregation* and followed by a more powerful *Projective Aggregation*.

3.3.1 Embedding Points to Camera Frustum Space

To associate each camera frustum into a shared world space $\mathcal{W}$, we propose to first embed discrete points along the camera ray direction in the frustum and then transform the embedded points to $\mathcal{W}$. A camera ray can be represented by a 3D vector $\boldsymbol{r} = (u, v, d)$, where u, v are the pixel coordinates in the image plane, and d is a depth value. Accordingly, all the camera ray vectors constitute a meshgrid of size $W \times H \times D$, where W, H are the width and height of the image, and D is the number of discrete depth values. Finally, given the camera intrinsic matrix $\mathbf{K}$ and extrinsic matrix $\mathbf{T}$ (from camera to $\mathcal{W}$), we can transfer the points in each camera's meshgrid to $\mathcal{W}$ through firstly back projecting the ray vector $\boldsymbol{r}$ into a 3D point in camera space using

Authorized licensed use limited to: ShanghaiTech University. Downloaded on October 19,2026 at 14:41:09 UTC from IEEE Xplore. Restrictions apply.

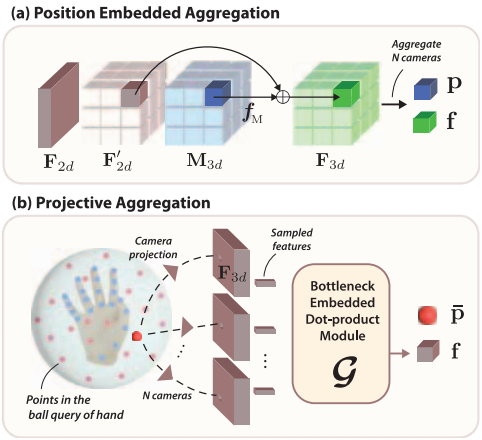


Figure 3. Illustration of the (a) position embedded aggregation and (b) projective aggregation for embedding image features into point cloud.

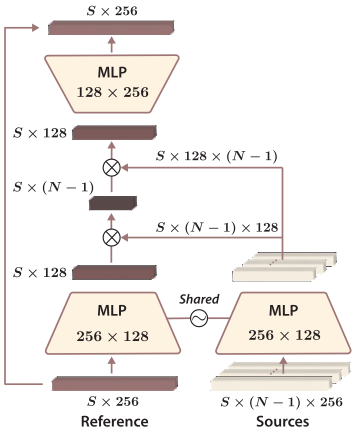


Figure 4. Architecture of $\mathcal{G}$: bottleneck embedded dot-product module for fusing features from different views.

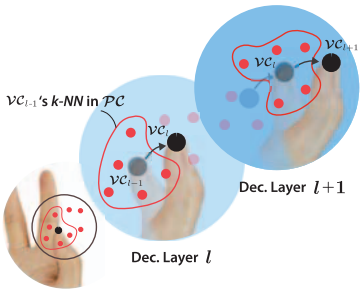


Figure 5. Vector attention: in each decoder layer, the $\mathcal{VC}$'s position will be updated by the attention between $\mathcal{VC}$ and $\mathcal{VC}$'s k nearest neighbors in $\mathcal{PC}$. Accordingly, $\mathcal{VC}$'s k nearest neighbors will be updated based on refined position of $\mathcal{VC}$.

$\mathbf{K}$, and then transform the point to $\mathcal{W}$ using $\mathbf{T}$. After aggregating points from all N cameras, we obtain the final point cloud $\mathbf{P}$ of size $M = N \times W \times H \times D$, which represents a wide range of points surrounding the hand $\mathbf{V}$.

However, only points that are spatially close to $\mathbf{V}$ are relevant. Ideally, we aim for the number of relevant points (S) to be greater than the number of vertices of hand ($S > 778$), while still being considerably smaller than M ($S \ll M$). To achieve this, we need to determine the initial position of the surface vertex, $\mathbf{V}$, based on the predicted skeleton joints, $\mathbf{X}$. While MANO's formulation provides a pretrained mapping from $\mathbf{V}$ to $\mathbf{X}$, mapping $\mathbf{X}$ to $\mathbf{V}$ is not straightforward. A basic approach for this inverse mapping is to learn a neural network $f_m : \mathbf{X} \mapsto \mathbf{V}$. However, we found that the initial $\mathbf{V} = f_m(\mathbf{X})$ may deviate significantly from $\mathbf{X}$, resulting in unstable training. Therefore, instead of learning the absolute position of $\mathbf{V}$, we choose to learn $\mathbf{V}$ as an offset of the wrist (root) joint $\mathbf{X}_w$, as follows: $\mathbf{V} = \mathbf{X}_w + f_m(\mathbf{X})$, where $\mathbf{X}_w$ is the wrist joint in the world system. Once $\mathbf{V}$ is obtained, we can retrieve its relevant points from $\mathbf{P}$ using the *ball query* [36] operation. These relevant points lie within a certain radius of $\mathbf{V}$ and are a subset of $\mathbf{P}$. We refer to them as $\bar{\mathbf{P}}$.

3.3.2 Embeding Features to Points

In this section, we discuss the approaches for embedding image features to camera frustum points ($\mathbf{P}$/ $\bar{\mathbf{P}}$, Sec. 3.3.2-A) and hand surface points ($\mathbf{V}$, Sec. 3.3.2-B), respectively.

A. Embedding Features to Camera Frustum Points ($\mathbf{P}$)

Position Embedded Aggregation. Recall in Sec. 3.3.1, we use a meshgrid of 3D points to represent the camera ray vectors inside the camera frustum. This meshgrid $\mathbf{M}_{3d}$ is a tensor of shape: $(W, H, D, 3)$. Meantime, the image feature $\mathbf{F}_{2d}$ extracted from the backbone model is a tensor with different shape: (C, H, W), where C is the numbers of channel. A basic design to assign the features in $\mathbf{F}_{2d}$ to the points in $\mathbf{M}_{3d}$ is through embedding the $\mathbf{M}_{3d}$'s positions into $\mathbf{F}_{2d}$, as shown in Fig. 3(a). Specifically, we first reshape the $\mathbf{F}_{2d}$ to $\mathbf{F}'_{2d}$: $(W, H, D, \frac{C}{D})$, where the last dimension (of size $\frac{C}{D}$) stores $\mathbf{F}_{2d}$'s 3D position-aware components. Then, we add the $\mathbf{M}_{3d}$'s positional encoding and $\mathbf{F}'_{2d}$ up for the *position embedded* feature: $\mathbf{F}_{3d}$:

$$\mathbf{F}_{3d} = f_{\mathrm{M}}(\mathbf{M}_{3d}) + \mathbf{F}'_{2d}, \tag{4}$$

where f_{M} is a series of sinusoids and MLP functions to obtain $\mathbf{M}_{3d}$' s positional encoding. The $\mathbf{F}_{3d}$ has an identical spatial arrangement to $\mathbf{M}_{3d}$. Each cell in $\mathbf{F}_{3d}$ stores a feature corresponding to the point at the same position in $\mathbf{M}_{3d}$. Therefore, the underlying idea for embedding image features into frustum points involves associating each point in $\mathbf{M}_{3d}$ with its corresponding feature in $\mathbf{F}_{3d}$. Subsequently, following the application of ball query, $\bar{\mathbf{P}}$ gathers points with different image features from different viewpoints, leading to effective feature aggregation. This approach is akin to previous methods such as PETR [29] (referred to as "3D Positional Encoder") and MVP [45] (referred to as "Ray Convolution"), which also rely on similar feature embedding strategy.

Although the above design seems reasonable, it suffers from two drawbacks. Firstly, we select $\bar{\mathbf{P}}$ instead of $\mathbf{P}$, which results in only a small portion of feature cells in $\mathbf{F}_{3d}$ being assigned to the point cloud $\bar{\mathbf{P}}$. Secondly, since each point in $\bar{\mathbf{P}}$ only carries features from one camera frustum, the features in $\bar{\mathbf{P}}$ lack consistency across different views

Projective Aggregation. To overcome the aforementioned drawbacks and facilitate more effective feature aggregation, we propose fusing features of points in $\bar{\mathbf{P}}$ from different views. As shown in Fig. 3(b), the key concept is to collect the features sampled at the 2D projection of a specific point in $\bar{\mathbf{P}}$ across N views, and combine the N in-

Authorized licensed use limited to: ShanghaiTech University. Downloaded on October 19,2026 at 14:41:09 UTC from IEEE Xplore. Restrictions apply.

dependently sampled features into a single geometry-aware feature for that point. We refer to this operation as *projective aggregation*. To sample feature at projected location in $\mathbf{F}_{3d}$, we use bilinear interpolation. To fuse the features sampled from different camera views, we modify an original non-local network [46] with bottleneck embedded dot-product design. As shown in Fig. 4, we choose the feature of one camera as reference, and the remain $N-1$ as sources. In the beginning, all the N sampled features are downsampled by an embedding projection (a MLP function). Then the features of sources and reference are completely mixed through dot-product, and up-sampled back by another embedding projection. At last, the mixed features are added to the reference for the final fused feature. Formally, the projective aggregation can be expressed as:

$$\mathbf{F} = \mathcal{G}(\mathbf{f}_1, ..., \mathbf{f}_N), \text{ where } \mathbf{f}_i = \mathbf{F}_{3d}\big(\pi(\bar{\mathbf{P}})\big)_i \tag{5}$$

where $\mathbf{F}_{3d}(\cdot)_i$ is bilinear interpolation on the i-th $\mathbf{F}_{3d}$, π is the camera projection, and $\mathcal{G}$ is the bottleneck embedded dot-product module.

Compared to features of position embedded aggregation, the feature produced by projective aggregation is geometrical-aware. For example, two points from different camera frustums will have similar fused features if they are spatially close. On the other hand, two points that are close in one image plane will still have the fused features of significant difference, if they are spatially far away.

B. Embedding Features to Hand Surface Points ($\mathbf{V}$)

Similar to the features of $\bar{\mathbf{P}}$, we gather the $\mathbf{V}$'s multi-view image feature by projective aggregation. In addition, since we are using $\mathbf{V}$ as the token in the Transformer, providing positional encoding for $\mathbf{V}$ is crucial. Following the practices of several vision-based Transformers, such as Keypoint Transformer [13], Mesh Transformer [27], and MVP [45], we utilize a set of learnable parameters: $\mathbf{S} \in \mathbb{R}^{N_V \times Z}$, as the position encoding for $\mathbf{V}$, where number of vertex: $N_V = 778$, and dimension of the encoding: $Z = 256$. We explore three different strategies for constructing $\mathbf{S}$.

(1) J-emb ($\mathbf{S}^{\mathbf{J}}$). Similar to Keypoint Transformer [13], the $\mathbf{S}^{\mathbf{J}}$ is initialized as the joint-level embedding vectors: $\mathbf{J} \in \mathbb{R}^{N_V \times Z}$. Hence, $\mathbf{S}^{\mathbf{J}} = \mathbf{J}$.

(2) G-emb ($\mathbf{S}^{\mathbf{G}}$). Similar to the Mesh Transformer [27], the $\mathbf{S}^{\mathbf{G}}$ is initialized as the concatenation ($\oplus$) of an input-dependent global image feature: $\mathbf{G} \in \mathbb{R}^Z$ and vertex-specified position: $\mathbf{V}_i^{\dagger}$ (i for i-th vertex) extracted from a zero-posed and mean-shape MANO hand template ($\dagger$). Therefore, $\mathbf{S}_i = \mathbf{G} \oplus \mathbf{V}_i^{\dagger}$. Here, $\mathbf{G}$ is the average of the N final-layer features from the N-views' image backbones ($\mathcal{F}\phi$ in Eq. (3)).

(3) G&J-emb ($\mathbf{S}^{+}$). Similar to the MVP [45], the $\mathbf{S}^{+}$ is initialized as the sum of the global image feature, $\mathbf{G}$, and joint-level embedding vectors, $\mathbf{J}$, as ${\mathbf{S}^{+}}_i = \mathbf{G} + \mathbf{J}_i$.

We empirically find that the **J-emb** achieve the best performance (see Sec. 4.2-G). Therefore, The feature of $\mathbf{V}$ consists of two terms, namely (1) feature from projective aggregation, $\mathbf{F}$ (obtained via substituting $\mathbf{V}$ for $\bar{\mathbf{P}}$ in Eq. (5)) and (2) $\mathbf{S}^{\mathbf{J}}$ as the positional encoding of $\mathbf{V}$. Following the convention used by [13, 27, 45], we refer to this positional encoded feature as **query of V** (denoted as $\mathbf{Q}$). Formally, we have $\mathbf{Q} = \mathbf{F} + \mathbf{S}^{\mathbf{J}}$.

3.3.3 Cross-set Point Transformer

Given two pointclouds:

- $\mathcal{PC} = (\bar{\mathbf{P}}, \mathbf{F})$, representing the frustum points' position $\bar{\mathbf{P}}$ and feature $\mathbf{F}$ aggregated from multi-view images;
- $\mathcal{VC} = (\mathbf{V}, \mathbf{Q})$, representing the hand vertices' position $\mathbf{V}$ and feature $\mathbf{Q}$,

we want a set operator to progressively extract the inter-set relationship and update $\mathcal{VC}$'s prediction. The attention mechanism [44] is quite a natural and powerful choice for this task. Inspired by the recent success of the self-attentive Point Transformer [53], we construct a cross-set Point Transformer to effectively capture relevant point features across different point clouds.

Our Transformer only contains one decoder with multiple decoder layers in it. As shown in Fig. 2-(b), each decoder layer consists of three sequential modules: a self-attention to perform point-wise interaction in $\mathcal{VC}$, a vector cross-attention to perform point-to-set interaction from $\mathcal{VC}$ to $\mathcal{PC}$, and a feed-forward network to regress a offset on $\mathcal{VC}$'s position ($\mathbf{V}$). This offset will be used to refine the input $\mathbf{V}$ from the previous layer. When the Transformer reaches its final output stage, the $\mathcal{VC}$'s position will also reaches its convergence based on the learnable weights in the second stage. The recursive form of each decoder layer is: $\mathcal{VC}_l = \mathcal{D}_l(\mathcal{PC}, \mathcal{VC}_{l-1})$, where $\mathcal{D}_l$ is the l-th decoder layer. The point attention between $\mathcal{VC}$ and $\mathcal{PC}$ is vector attention [52, 53], in which the attention will be applied in a local neighbor (*i.e.* the k nearest neighbor) of each query point (see Fig. 5). We formulate the three modules in each $\mathcal{D}_l$ as:

$$\begin{aligned}
&\mathbf{Q}^{\star} = \textit{self-attention}(\mathbf{Q}),\\
&\widetilde{\mathbf{Q}}_i = \sum_{\mathbf{F}_j \in \boldsymbol{\mathcal{X}}_i} \mathrm{SM}\big(\gamma(\alpha(\mathbf{Q}_i^{\star}) - \beta(\mathbf{F}_j)) + \boldsymbol{\delta}\big) \odot (\psi(\mathbf{F}_j) + \boldsymbol{\delta}),\\
&\widetilde{\mathbf{V}}_i = \mathbf{V}_i + \mathrm{FFN}(\widetilde{\mathbf{Q}}_i),
\end{aligned} \tag{6}$$

where $\boldsymbol{\mathcal{X}}_i$ is a subset of $\mathcal{PC}$, which collect points in the k nearest neighbor of the i-th hand vertex: $\mathbf{V}_i$. The $\boldsymbol{\delta}$ is the position encoding for point cloud, $\boldsymbol{\delta} = \theta(\mathbf{V}_i - \bar{\mathbf{P}}_j)$. The $\alpha, \beta, \gamma, \psi, \theta$ are learnable functions (*e.g.* MLP), $\odot$ is the Hadamard product, FFN is the feed-forward network, and SM is the *softmax* operation. Through collecting all the $\widetilde{\mathbf{V}}_i$ and $\widetilde{\mathbf{Q}}_i$, we can get the output $\mathcal{VC}_l$ of the current decoder layer $\mathcal{D}_l$.

Authorized licensed use limited to: ShanghaiTech University. Downloaded on October 19,2026 at 14:41:09 UTC from IEEE Xplore. Restrictions apply.

			Methods	Hand vertices				Hand keypoints			
				MPVPE↓	RR-V↓	PA-V↓	AUC-V↑	MPJPE↓	RR-J↓	PA-J↓	AUC-J↑
DexYCB-MV	1	ours	**POEM**	**6.13**	**7.21**	**4.00**	**0.70**	**6.06**	**7.30**	**3.93**	**0.68**
	2	A	MVP [45]	9.77	12.18	8.14	0.53	6.23	9.47	4.26	0.69
	3		PE-Mesh-TR	7.41	8.67	4.70	0.64	7.49	8.87	4.76	0.64
	4		FTL-Mesh-TR	8.75	9.80	5.75	0.59	8.66	9.81	5.51	0.59
	8	C	POEM w/o pt.	7.63	8.94	5.48	0.63	7.20	8.58	4.89	0.65
	9	D	POEM w/o Proj.	6.57	7.69	4.42	0.68	6.54	7.82	4.37	0.67
	10	F	Multi-view Fit.	7.33	8.71	5.29	0.65	7.22	8.77	5.19	0.65
HO3D-MV	11	ours	**POEM**	**17.2**	**21.45**	**9.97**	**0.66**	**17.28**	**21.94**	**9.60**	**0.63**
	12	A	MVP [45]	20.95	27.08	10.04	0.59	18.72	24.90	10.44	0.60
	13		PE-Mesh-TR	23.49	29.19	11.31	0.55	23.94	30.23	11.67	0.54
	14		FTL-Mesh-TR	24.15	33.53	10.56	0.53	24.66	34.74	10.76	0.52
	15	C	POEM w/o pt.	19.26	24.32	12.45	0.62	18.20	23.80	10.56	0.63
	16	D	POEM w/o proj.	18.83	22.26	10.83	0.63	18.48	22.73	10.39	0.63
	17	E	POEM w/o $\Delta\mathbf{v}$	19.34	24.27	11.18	0.62	19.42	25.00	10.71	0.60
OakInk-MV	18	ours	**POEM** ($\mathbf{S}^{\mathbf{J}}$)	**6.20**	**7.63**	**4.21**	**0.70**	**6.01**	**7.46**	**4.00**	**0.69**
	19	A	MVP [45]	9.69	11.75	7.74	0.53	7.32	9.99	4.97	0.64
	20		PE-Mesh-TR	8.34	9.67	5.75	0.60	8.18	9.59	5.42	0.61
	21		FTL-Mesh-TR	9.28	10.88	6.61	0.56	8.89	10.66	6.01	0.58
	22	D	POEM w/o proj	6.42	7.82	4.50	0.69	6.25	7.84	4.28	0.68
	23	E	POEM w/o $\Delta\mathbf{v}$	6.56	8.04	4.63	0.69	6.32	7.99	4.32	0.67
	24	G	$\mathbf{S}^{+}$ as pos-enc.	6.23	7.63	4.30	0.70	6.05	7.65	4.09	0.69
	25		$\mathbf{S}^{\mathbf{G}}$ as pos-enc.	6.25	7.65	4.33	0.70	6.05	7.66	4.10	0.69

Table 1. Quantitative results (mm) of evaluations **A** to **F**. The AUC are computed on the evaluation metrics of MPVPE and MPJPE. The thresholds of AUC vary from datasets. *i.e.* 0-20 mm for DexYCB-MV and OakInk-MV, and 0-50 mm for HO3D-MV.

4. Experiments and Results

4.1. Datasets

DexYCB. It contains 582K images of hand grasping objects [4]. These images consist of the observations from 8 cameras. To construct a split mode for multi-view task, we follow its official 'S0' split on train/val/test sets and filter out the frames on left hand. For each frame ID, we collect 8 images from total 8 cameras. These 8 images, along with their annotations, consist of one multi-view (*m.v.*) frame. We name the DexYCB with our multi-view split as DexYCB-MV. In all, DexYCB-MV contains 25,387 *m.v.* frames (that is, 203,096 monocular frames) in training set, 1,412 *m.v.* frames in validation and 4,951 in testing set.

HO3D. HO3D (version 3) [12] contains 103,462 images capturing hand-object interaction from up-to 5 cameras. Frames of same sequence but different cameras may be scattered in different split sets. To enable HO3D supporting multi-view task, we only select sequences with full 5 camera observations, and construct a HO3D-MV upon them. There are total 7 sequences in HO3D satisfy multi-view requirements. We select sequences with serial: 'ABF1','BB1', 'GSF1', 'MDF1' and 'SiBF1' as training set, and the remain 'GPMF1' and 'SB1' as testing set. In total, there are 9,087 *m.v.* frames (that is 45,435 monocular frames) in training set and 2,706 in testing set.

OakInk. OakInk-Image [49] is a new dataset of hand manipulating objects. It contains 230K images from the observation of 4 cameras. We follow the official 'SP2' – objects split mode and construct the OakInk-MV. Each *m.v.* frame in OakInk-MV contains the images from 4 cameras. Nearly a quarter of sequences in OakInk contain two person handing over an object (with two hands captured in image). For these sequences, we train and test each hand separately. OakInk-MV has 58,692 *m.v.* frames (that is, 234,768 monocular frames) in training set and 19,909 *m.v.* frames in testing set.

4.2. Evaluation

We report the MPJPE and MPVPE (mm), standing for the mean per keypoint (joint) and per vertex position error, respectively. Notably, since the MANO provides a pre-trained mapping from $\mathbf{V}$ to $\mathbf{X}$, The reported keypoints $\mathbf{X}$ is a by-product of the final vertex $\mathbf{V}$ from the second stage. Beside, we also inspect the MPVPE and MPVPE in a root-relative (RR) system and under the Procrustes analysis (PA). Additionally, we evaluate the percentage of correct keypoints under a range of threshold by measuring the area under the percentage curve (AUC), which tells us the model's discriminative ability on localizing keypoints.

A. POEM -vs- SOTA. POEM targets on single hand reconstruction under the multi-view RGB observations. We find that the MVP [45], which directly regress mesh parameters catered to a skinning body model (SMPL [30] and MANO [40]) using vision Transformer, is highly relevant to our model. MVP also employs another module to as-

Authorized licensed use limited to: ShanghaiTech University. Downloaded on October 19,2026 at 14:41:09 UTC from IEEE Xplore. Restrictions apply.

sociate keypoints among different persons. In our experiments, we only compare POEM with MVP w.r.t. the single-body reconstruction. Apart from MVP, we also simulate several SOTA methods for multi-view hand mesh reconstruction. The purpose of the simulation is to combine the SOTA architecture in monocular hand reconstruction with the most advanced fusion algorithms adopted by multi-view settings. With this in mind, we first explore the advanced model in the field of 3D detection in autonomous driving, as operating neural network on multi-view setting is well-explored in this field. The recent method: PETR [29] encoded the 3D position embedding (PE) of camera frustum into 2D image features, fused those position-embedded features with learnable objects' queries (corresponding to the $\mathbf{Q}$ in our model), and finally transferred those objects' queries to the objects' 3D position via a DETR [3] decoder. In terms of monocular hand mesh reconstruction, the Mesh Transformer [27] is a representative method that adopts an image-based self-attentive Transformer to directly regress all vertices on the hand mesh. Therefore, for simulating multi-view hand mesh reconstruction, we utilize the feature-position encoder of PETR to process multi-view image features and a DETR-like vision Transformer to regress the 3D vertices on hand mesh (based on the design in [27]). We denote this method "PE-Mesh-TR". Similarly, we simulate the "FTL-Mesh-TR", which utilizes the feature transform layer (FTL) [39] to fuse multi-view features and a mesh-adapted DETR decoder for mesh regression. Our POEM outperforms these methods in all metrics.

B. POEM's Two-stage Design. Recalling the problem formulation in Sec. 3.1: Eq. (1), the method MVP, PE-Mesh-TR and FTL-Mesh-TR can be represented as $P(\mathbf{X}, \mathbf{V}|\mathcal{I})$, where the hand vertices and keypoints are directly regressed from their models at the output stage. In contrast, our method is explicit formulated as two-stage model. Their comparisons are shown in row `1` *vs.* `2-4`, `11` *vs.* `12-14` and `18` *vs.* `19-21`.

C. POEM without Embedded Point. We also simulate a two-stage version of POEM but **without** embedded points design, named "POEM w/o pt". We sequentially concatenate the first stage in POEM (for predicting $\mathbf{X}$), the feature-position encoder, and Transformer decoder as in the PE-Mesh-TR. This lead inferior results, shown in row `1` *vs.* `8` and `11` *vs.* `15`.

D. POEM without Projective Aggregation. We investigate the performance of two feature-to-points embedding methods in Sec. 3.3.2. The results of the position embedded aggregation are denoted as "POEM w/o proj". Comparison between row `1` *vs.* `9`, `11` *vs.* `16` and `18` *vs.* `22` show that the later projective aggregation is effective.

E. POEM without Progressive $\mathcal{VC}$ Update. We remove all the FFN in the decoder layer only except the last one. The final $\mathbf{V}$ is thus a single-step prediction ("POEM w/o $\Delta\mathbf{V}$"). The comparison can be found in row `11` *vs.* `17` and `18` *vs.* `23`, showing that removing the progressive $\mathcal{VC}$'s update leads to inferior results.

F. POEM -vs- Multi-view Mesh Fitting. Multi-view pose estimation is a common practice for datasets to obtain hand's ground-truth [11, 37]. For instance, HOnnotate [11] aggregated visual cues from several upstream vision tasks to acquire the hand model automatically. In a similar way, we also fit a 3D hand mesh (MANO) to the multi-view 2D predictions. Our fitting objectives are three-fold: 1) 2D hand keypoints; 2) silhouette; 3) hand's anatomical constraints [50]; The pseudo ground-truth of 2D keypoints and silhouette are obtained from a pretrained model [7]. We report the final results in row `1` *vs.* `10`. POEM outperforms the time-consuming fitting-based methods for auto-labeling.

G. POEM: V's Positional Encoding. We explored three different strategies for constructing the position encoding (pos-enc) of $\mathbf{V}$, namely $\mathbf{S^J}$, $\mathbf{S^G}$, and $\mathbf{S}^+$, as described in Sec. 3.3.2-B. The results in row `18` *vs.* `24,25` show that the joint-level embedding ($\mathbf{S^J}$) best fit for our model.

H. Decoders, Cameras, and *k* Neighbors. We further examine the performance of POEM on varying the number of decoder, numbers of cameras, and numbers of k nearest neighbors, and qualitatively assess the POEM's reconstruction results. Please refer to the **Appx** for more details.

5. Discussion

Limitation. First, POEM is not able to distinguish keypoint from different hands. Hence, it only supports single hand reconstruction. Second, to fully exploit its power, the projective aggregation expects those cameras to have an apparent spatial difference. Otherwise, it will collapse to the aggregation of 2D features with adjacent coordinates.

Conclusion. In this paper, we propose POEM, which addresses the multi-view hand reconstruction using point representation. POEM directly operates on points through two novel designs, that are point-based feature fusion and cross-set point attention. Experiments on three datasets show the effectiveness of such designs. Though targeting on hand's reconstruction, the virtue brought from POEM also opens a path toward the general multi-view-based object reconstruction, which we leave as a future work.

Acknowledgments. This work was supported by the National Key R&D Program of China (No. 2021ZD0110704), Shanghai Municipal Science and Technology Major Project (2021SHZDZX0102), Shanghai Qi Zhi Institute, and Shanghai Science and Technology Commission (21511101200). We thank Anran Xu, Jiefeng Li and Kailin Li for their help, discussion and feedback.

Authorized licensed use limited to: ShanghaiTech University. Downloaded on October 19,2026 at 14:41:09 UTC from IEEE Xplore. Restrictions apply.

References

[1] Adnane Boukhayma, Rodrigo de Bem, and Philip HS Torr. 3d hand shape and pose from images in the wild. In *Computer Vision and Pattern Recognition (CVPR)*, 2019. 1, 3

[2] Samarth Brahmbhatt, Chengcheng Tang, Christopher D Twigg, Charles C Kemp, and James Hays. ContactPose: A dataset of grasps with object contact and hand pose. In *European Conference on Computer Vision (ECCV)*, 2020. 1

[3] Nicolas Carion, Francisco Massa, Gabriel Synnaeve, Nicolas Usunier, Alexander Kirillov, and Sergey Zagoruyko. End-to-end object detection with transformers. In *European Conference on Computer Vision (ECCV)*, 2020. 8

[4] Yu-Wei Chao, Wei Yang, Yu Xiang, Pavlo Molchanov, Ankur Handa, Jonathan Tremblay, Yashraj S. Narang, Karl Van Wyk, Umar Iqbal, Stan Birchfield, Jan Kautz, and Dieter Fox. Dexycb: A benchmark for capturing hand grasping of objects. In *Computer Vision and Pattern Recognition (CVPR)*, 2021. 1, 2, 7

[5] Ping Chen, Yujin Chen, Dong Yang, Fangyin Wu, Qin Li, Qingpei Xia, and Yong Tan. I2uv-handnet: Image-to-uv prediction network for accurate and high-fidelity 3d hand mesh modeling. In *International Conference on Computer Vision (ICCV)*, 2021. 1, 3

[6] Rui Chen, Songfang Han, Jing Xu, and Hao Su. Point-based multi-view stereo network. In *International Conference on Computer Vision (ICCV)*, 2019. 3

[7] Xingyu Chen, Yufeng Liu, Chongyang Ma, Jianlong Chang, Huayan Wang, Tian Chen, Xiaoyan Guo, Pengfei Wan, and Wen Zheng. Camera-space hand mesh recovery via semantic aggregation and adaptive 2d-1d registration. In *Computer Vision and Pattern Recognition (CVPR)*, 2021. 1, 3, 8

[8] Zerui Chen, Yana Hasson, Cordelia Schmid, and Ivan Laptev. Alignsdf: Pose-aligned signed distance fields for hand-object reconstruction. In *European Conference on Computer Vision (ECCV)*, 2022. 3

[9] Liuhao Ge, Zhou Ren, Yuncheng Li, Zehao Xue, Yingying Wang, Jianfei Cai, and Junsong Yuan. 3d hand shape and pose estimation from a single rgb image. In *Computer Vision and Pattern Recognition (CVPR)*, 2019. 1, 3

[10] Meng-Hao Guo, Jun-Xiong Cai, Zheng-Ning Liu, Tai-Jiang Mu, Ralph R Martin, and Shi-Min Hu. Pct: Point cloud transformer. *Computational Visual Media*, 7(2):187–199, 2021. 3

[11] Shreyas Hampali, Mahdi Rad, Markus Oberweger, and Vincent Lepetit. Honnotate: A method for 3d annotation of hand and object poses. In *Computer Vision and Pattern Recognition (CVPR)*, 2020. 8

[12] Shreyas Hampali, Sayan Deb Sarkar, and Vincent Lepetit. HO-3D_v3: Improving the accuracy of hand-object annotations of the ho-3d dataset. *ArXiv*, abs/2107.00887, 2021. 2, 7

[13] Shreyas Hampali, Sayan Deb Sarkar, Mahdi Rad, and Vincent Lepetit. Keypoint Transformer: Solving joint identification in challenging hands and object interactions for accurate 3d pose estimation. In *Computer Vision and Pattern Recognition (CVPR)*, 2022. 6

[14] Shangchen Han, Po-Chen Wu, Yubo Zhang, Beibei Liu, Linguang Zhang, Zheng Wang, Weiguang Si, Peizhao Zhang, Yujun Cai, Tomás Hodan, Randi Cabezas, Luan Tran, Muzaffer Akbay, Tsz-Ho Yu, Cem Keskin, and Robert Y. Wang. UmeTrack: Unified multi-view end-to-end hand tracking for vr. In *SIGGRAPH Asia 2022 Conference Papers*, 2022. 2

[15] Richard Hartley and Andrew Zisserman. *Multiple View Geometry in computer vision*. Cambridge university press, 2003. 3

[16] Yana Hasson, Gul Varol, Dimitrios Tzionas, Igor Kalevatykh, Michael J Black, Ivan Laptev, and Cordelia Schmid. Learning joint reconstruction of hands and manipulated objects. In *Computer Vision and Pattern Recognition (CVPR)*, 2019. 3

[17] Yihui He, Rui Yan, Katerina Fragkiadaki, and Shoou-I Yu. Epipolar transformers. In *Computer Vision and Pattern Recognition (CVPR)*, 2020. 1, 2

[18] Karim Iskakov, Egor Burkov, Victor S. Lempitsky, and Yury Malkov. Learnable triangulation of human pose. In *International Conference on Computer Vision (ICCV)*, 2019. 1, 2

[19] Mengqi Ji, Juergen Gall, Haitian Zheng, Yebin Liu, and Lu Fang. Surfacenet: An end-to-end 3d neural network for multiview stereopsis. In *International Conference on Computer Vision (ICCV)*, 2017. 3

[20] Abhishek Kar, Christian Häne, and Jitendra Malik. Learning a multi-view stereo machine. *Conference on Neural Information Processing Systems (NeurIPS)*, 2017. 3

[21] Korrawe Karunratanakul, Jinlong Yang, Yan Zhang, Michael J Black, Krikamol Muandet, and Siyu Tang. Grasping field: Learning implicit representations for human grasps. In *International Conference on 3D Vision (3DV)*, 2020. 3

[22] Nikos Kolotouros, Georgios Pavlakos, and Kostas Daniilidis. Convolutional mesh regression for single-image human shape reconstruction. In *Computer Vision and Pattern Recognition (CVPR)*, 2019. 3

[23] Deying Kong, Linguang Zhang, Liangjian Chen, Haoyu Ma, Xiangyi Yan, Shanlin Sun, Xingwei Liu, Kun Han, and Xiaohui Xie. Identity-aware hand mesh estimation and personalization from rgb images. In *European Conference on Computer Vision (ECCV)*, 2022. 3

[24] Dominik Kulon, Riza Alp Guler, Iasonas Kokkinos, Michael M Bronstein, and Stefanos Zafeiriou. Weakly-supervised mesh-convolutional hand reconstruction in the wild. In *Computer Vision and Pattern Recognition (CVPR)*, 2020. 3

[25] Yangyan Li, Rui Bu, Mingchao Sun, Wei Wu, Xinhan Di, and Baoquan Chen. Pointcnn: Convolution on x-transformed points. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2018. 3

[26] Jiahao Lin and Gim Hee Lee. Multi-view multi-person 3d pose estimation with plane sweep stereo. In *Computer Vision and Pattern Recognition (CVPR)*, 2021. 2

[27] Kevin Lin, Lijuan Wang, and Zicheng Liu. End-to-end human pose and mesh reconstruction with transformers. In *Computer Vision and Pattern Recognition (CVPR)*, 2021. 1, 3, 6, 8

Authorized licensed use limited to: ShanghaiTech University. Downloaded on October 19,2026 at 14:41:09 UTC from IEEE Xplore. Restrictions apply.

[28] Kevin Lin, Lijuan Wang, and Zicheng Liu. Mesh graphormer. In *International Conference on Computer Vision (ICCV)*, 2021. 3

[29] Ying-Hao Liu, Tiancai Wang, X. Zhang, and Jian Sun. Petr: Position embedding transformation for multi-view 3d object detection. In *European Conference on Computer Vision (ECCV)*, 2022. 2, 5, 8

[30] Matthew Loper, Naureen Mahmood, Javier Romero, Gerard Pons-Moll, and Michael J. Black. SMPL: a skinned multi-person linear model. *ACM Transactions on Graphics*, 2015. 7

[31] Gyeongsik Moon and Kyoung Mu Lee. I2l-meshnet: Image-to-lixel prediction network for accurate 3d human pose and mesh estimation from a single rgb image. In *European Conference on Computer Vision (ECCV)*, 2020. 1, 3

[32] Joo Hyun Park, Yeong Min Oh, Gyeongsik Moon, Hongsuk Choi, and Kyoung Mu Lee. HandOccNet: Occlusion-robust 3d hand mesh estimation network. In *Computer Vision and Pattern Recognition (CVPR)*, 2022. 1

[33] Georgios Pavlakos, Xiaowei Zhou, Konstantinos G. Derpanis, and Kostas Daniilidis. Harvesting multiple views for marker-less 3d human pose annotations. In *Computer Vision and Pattern Recognition (CVPR)*, 2017. 1, 2

[34] Sergey Prokudin, Christoph Lassner, and Javier Romero. Efficient learning on point clouds with basis point sets. In *International Conference on Computer Vision (ICCV)*, 2019. 2

[35] Charles Ruizhongtai Qi, Hao Su, Kaichun Mo, and Leonidas J Guibas. Pointnet: Deep learning on point sets for 3d classification and segmentation. In *Computer Vision and Pattern Recognition (CVPR)*, 2017. 3

[36] Charles Ruizhongtai Qi, Li Yi, Hao Su, and Leonidas J Guibas. Pointnet++: Deep hierarchical feature learning on point sets in a metric space. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2017. 3, 5

[37] Yuzhe Qin, Yueh-Hua Wu, Shaowei Liu, Hanwen Jiang, Ruihan Yang, Yang Fu, and Xiaolong Wang. DexMV: Imitation learning for dexterous manipulation from human videos. In *European Conference on Computer Vision (ECCV)*, 2022. 8

[38] Haibo Qiu, Chunyu Wang, Jingdong Wang, Naiyan Wang, and Wenjun Zeng. Cross view fusion for 3d human pose estimation. In *International Conference on Computer Vision (ICCV)*, 2019. 1, 2

[39] Edoardo Remelli, Shangchen Han, Sina Honari, P. Fua, and Robert Y. Wang. Lightweight multi-view 3d pose estimation through camera-disentangled representation. In *Computer Vision and Pattern Recognition (CVPR)*, 2020. 1, 2, 8

[40] Javier Romero, Dimitris Tzionas, and Michael J Black. Embodied hands: Modeling and capturing hands and bodies together. *ACM Transactions on Graphics*, 36(6), 2017. 3, 7

[41] Xiao Sun, Bin Xiao, Fangyin Wei, Shuang Liang, and Yichen Wei. Integral human pose regression. In *European Conference on Computer Vision (ECCV)*, 2018. 3

[42] Hugues Thomas, Charles R Qi, Jean-Emmanuel Deschaud, Beatriz Marcotegui, François Goulette, and Leonidas J Guibas. Kpconv: Flexible and deformable convolution for point clouds. In *International Conference on Computer Vision (ICCV)*, 2019. 3

[43] Hanyue Tu, Chunyu Wang, and Wenjun Zeng. Voxelpose: Towards multi-camera 3d human pose estimation in wild environment. In *European Conference on Computer Vision (ECCV)*, 2020. 1, 2

[44] Ashish Vaswani, Noam Shazeer, Niki Parmar, Jakob Uszkoreit, Llion Jones, Aidan N Gomez, Łukasz Kaiser, and Illia Polosukhin. Attention is all you need. *Advances in neural information processing systems*, 30, 2017. 6

[45] Tao Wang, Jianfeng Zhang, Yujun Cai, Shuicheng Yan, and Jiashi Feng. Direct multi-view multi-person 3d pose estimation. In *Conference on Neural Information Processing Systems (NeurIPS)*, 2021. 2, 5, 6, 7

[46] Xiaolong Wang, Ross Girshick, Abhinav Kumar Gupta, and Kaiming He. Non-local neural networks. In *Computer Vision and Pattern Recognition (CVPR)*, 2018. 6

[47] Shih-En Wei, Varun Ramakrishna, Takeo Kanade, and Yaser Sheikh. Convolutional pose machines. In *Computer Vision and Pattern Recognition (CVPR)*, 2016. 3

[48] Wenxuan Wu, Zhongang Qi, and Li Fuxin. Pointconv: Deep convolutional networks on 3d point clouds. In *Computer Vision and Pattern Recognition (CVPR)*, 2019. 3

[49] Lixin Yang, Kailin Li, Xinyu Zhan, Fei Wu, Anran Xu, Liu Liu, and Cewu Lu. OakInk: A large-scale knowledge repository for understanding hand-object interaction. In *Computer Vision and Pattern Recognition (CVPR)*, 2022. 1, 2, 7

[50] Lixin Yang, Xinyu Zhan, Kailin Li, Wenqiang Xu, Jiefeng Li, and Cewu Lu. CPF: Learning a contact potential field to model the hand-object interaction. In *International Conference on Computer Vision (ICCV)*, 2021. 8

[51] Yao Yao, Zixin Luo, Shiwei Li, Tian Fang, and Long Quan. Mvsnet: Depth inference for unstructured multi-view stereo. In *ECCV*, 2018. 3

[52] Hengshuang Zhao, Jiaya Jia, and Vladlen Koltun. Exploring self-attention for image recognition. In *Computer Vision and Pattern Recognition (CVPR)*, 2020. 6

[53] Hengshuang Zhao, Li Jiang, Jiaya Jia, Philip H. S. Torr, and Vladlen Koltun. Point Transformer. In *International Conference on Computer Vision (ICCV)*, 2021. 3, 6

[54] Yuxiao Zhou, Marc Habermann, Weipeng Xu, Ikhsanul Habibie, Christian Theobalt, and Feng Xu. Monocular real-time hand shape and motion capture using multi-modal data. In *Computer Vision and Pattern Recognition (CVPR)*, 2020. 1, 3

[55] Christian Zimmermann, Duygu Ceylan, Jimei Yang, Bryan Russell, Max Argus, and Thomas Brox. FreiHand: A dataset for markerless capture of hand pose and shape from single rgb images. In *International Conference on Computer Vision (ICCV)*, 2019. 1

Authorized licensed use limited to: ShanghaiTech University. Downloaded on October 19,2026 at 14:41:09 UTC from IEEE Xplore. Restrictions apply.