Physical-space Multi-body Mesh Detection Achieved by Local Alignment and Global Dense Learning — Supplementary Materials

Haoye Dong[1], Tiange Xiang[2], Sravan Chittupalli[1], Jun Liu[1], Dong Huang[1]
[1]Carnegie Mellon University
[2]Stanford University
{donghaoye,liujun,donghuang}@cmu.edu, schittup@andrew.cmu.edu, xtiange@stanford.edu

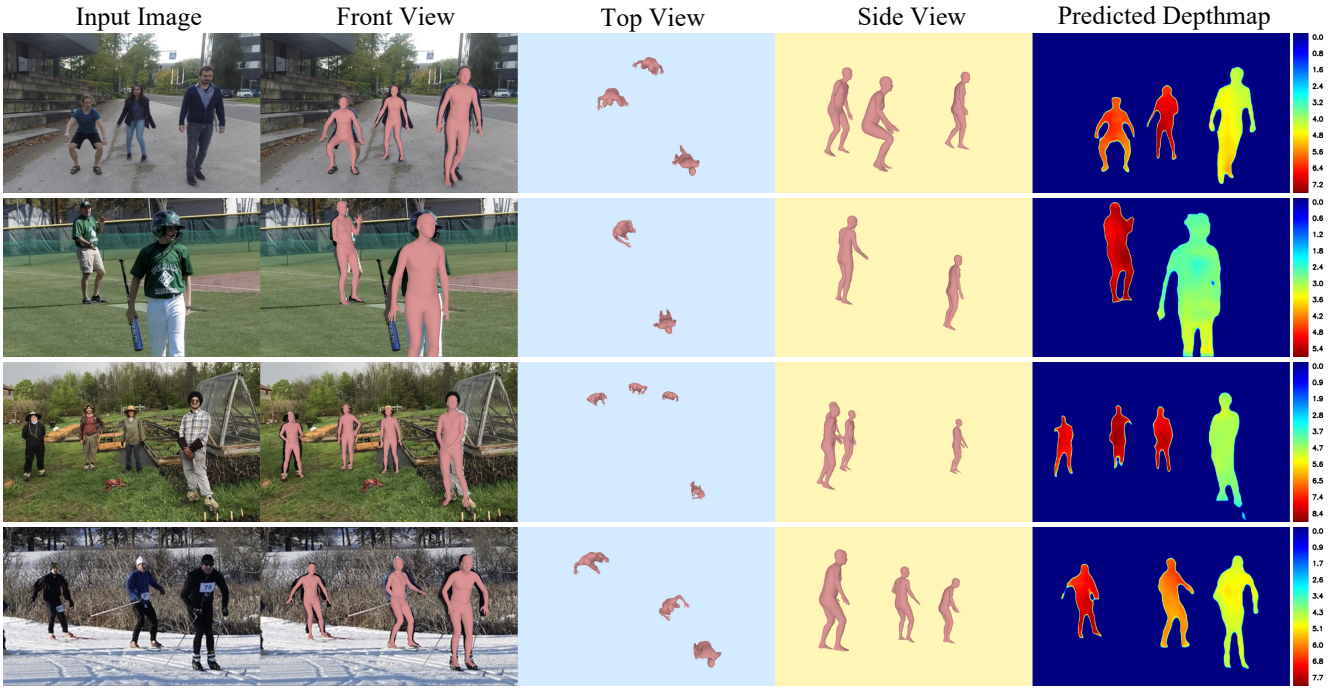

Figure 1. The proposed PMMD results on out-of-distribution (unseen scenarios) images. Since no GT mesh is available, only PMMD meshes are displayed in Front View | Top View | Side View of the camera 3D coordinate. The Predicted Dethmaps are displayed in the last column. PMMD is robust to the person in the side profile, with truncated legs, and in the wild fields.

1. Additional Visualizations

The proposed PMMD against GT in the 360-degree view. Most existing works visualize detected meshes as 2D image-space projections or in a pseudo 3D space induced by weakly prospective assumption. The 2D-image-space visualization obscures incorrect mesh scaling and translation along the depth direction. The pseudo-3D-space visualization only shows relative layouts among meshes instead of the absolute sizes and locations.

To validate PMMD meshes in physical sizes and locations, we directly render PMMD meshes (pink) against the GT meshes (gray), ROMP(blue), and BEV(yellow) in the physical global 3D coordinate. In the video file "**single_image_360views.mp4**" attached to this supplementary package, we show single-image mesh detection results in 360-degree synthesized camera views. Under many challenging scenarios, most PMMD meshes closely match the GT meshes in poses, sizes, and locations.

Out-of-distribution Images. In Fig.1-Fig.2, we show PMMD mesh detection results on images from many unseen scenarios. Since there is no GT mesh to compare with, we only show PMMD results in the camera 3D coordinates. PMMD is robust to the person in the side profile, with trun-

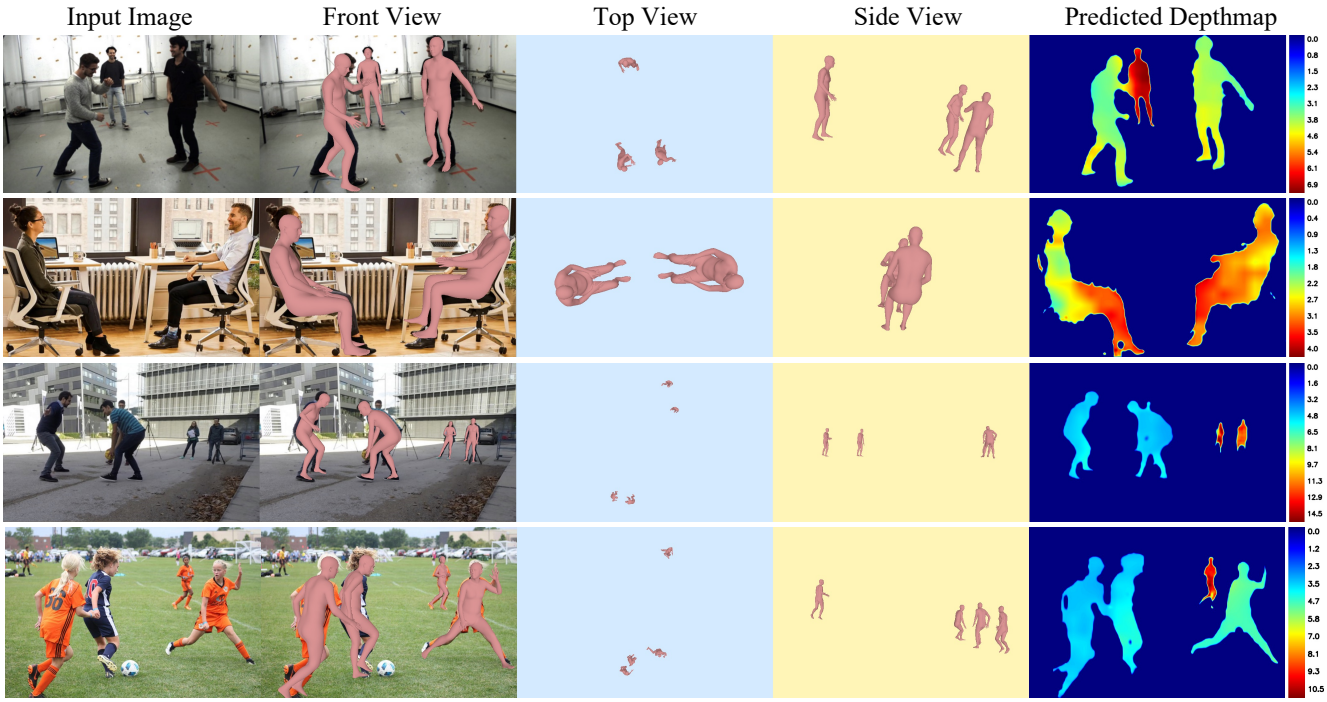

Figure 2. The proposed PMMD results on out-of-distribution (unseen scenarios) images. Since no GT mesh is available, only PMMD meshes are displayed in Front View | Top View | Side View of the camera 3D coordinate. The Predicted Dethmaps are displayed in the last column. PMMD is robust to the person in the side profile and under heavy occlusion.

cated legs, in the wild fields, and under heavy occlusion.

Failure Cases. In Fig. 3, we observed two main failure cases of PMMD under its current configuration:

- Failure on small person detection. This failure can be improved by bigger input sizes (bigger than the current 512x832) and stronger backbones such as the Swin-Transformer [1].
- Failure on depth estimation of the children meshes. This failure can be improved using the SMPL-Age model and GT age annotations used in BEV [24].

2. Training Data and Recipes

Data/Annotations. All recent works on SMPL mesh estimation used very different extra training data and annotations. Many of them were not released for license issues. For instance, VIBE, SPIN, CRMH, and BMP used the licensed codes from [18] to generate high-quality GT SMPL meshes on Human3.6M. This makes it impossible to conduct a strictly fair comparison among all algorithms. It is also very difficult for future researchers to move forward.

We consider the advances in algorithms and data/annotations to be equally critical driving forces to the research. To the best knowledge within the scope of this work, we provide all extra training data below and method-specific configurations in Table 1.

Among the compared methods in the main paper Table.1, the list of all training datasets are: **(a)** 3DPW train set [26], **(b)** Human3.6M train set [4], **(c)** MPI-Inf-3DHP [21], **(d)** UP [14], **(e)** MS-COCO [17], **(f)** MPII [2], **(g)** LSP [6], **(h)** LSP extend [7], AICH [27] (extra: **(i)** MuCo-3DHP [20], **(j)** OH [25], **(k)** PoseTrack [1], **(l)** Crowdpose [15] **(m)** Human3.6M test set [4], **(n)** PennAction [31], **(o)** InstaVariety [9], **(p)** Kinetics-400 [3], **(q)** AMASS [19]), **(r)** AGORA [23], **(s)** CLIFF-coco and CLIFF-mpii (pseudo-GT) [16], **(t)** Relative depth order annotation and age group classfication (used in BEV) [24], **(u)** SUN360 [28], **(v)** MTP [22].

Besides the extra training databases, some methods leverage extra priors which are usually associated with extra data collection and annotations. The list of extra priors in training are: **(i)** temporal smoothness [11], **(ii)** person-to-person depth order [24], **(iii)** ground plane [8,12], **(iv)** shape deformation by ages [24].

PMMD does not rely on additional annotations (t-v) or priors (i-iv), yet produced far better global metrics than existing methods (see metrics reported in Table 1 in the main submission).

Training Recipes. Due to the complexity of using so

[1]Swin Transformer Mask-RCNN. https://github.com/microsoft/Swin-Transformer

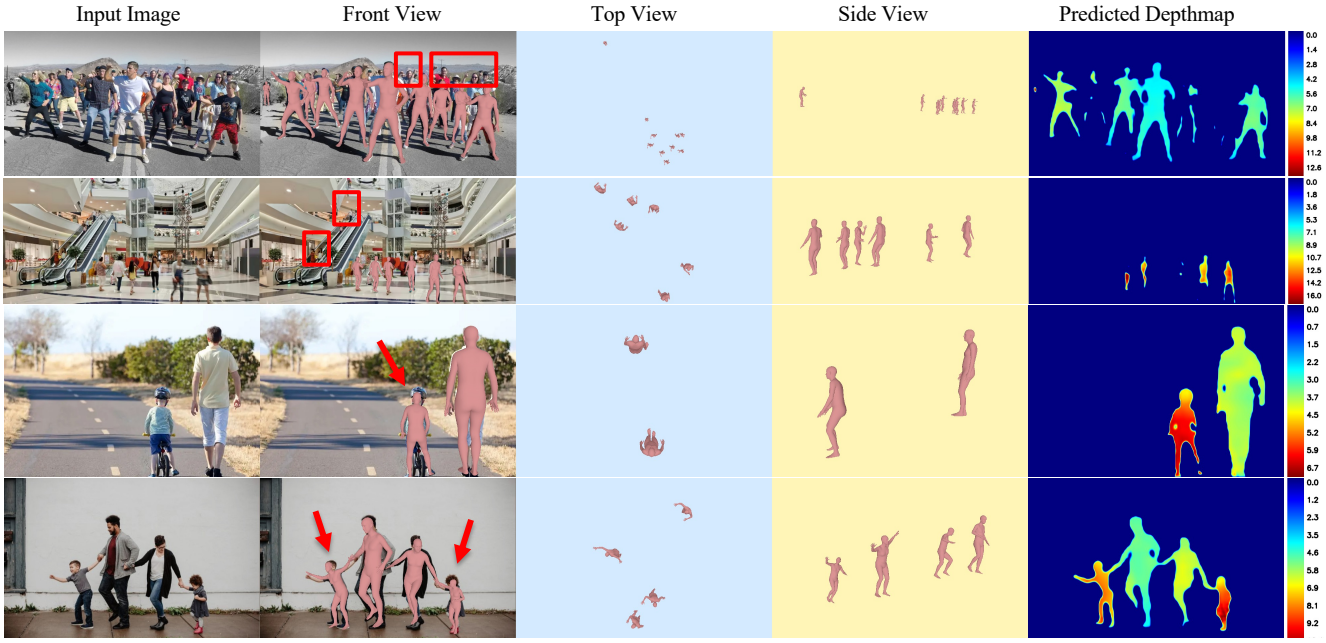

Figure 3. Failure cases. Detected meshes are displayed in Front View | Top View | Side View. The Predicted Dethmaps are displayed in the last column. In the 1st-2nd row, PMMD fails to detect some small bodies (marked by red rectangles). In the 3rd-4th row, PMMD fails to locate the child meshes (pointed by red arrows) because only the adult SMPL model was used in training PMMD.

Table 1. Comparison on method-specific training data/annotations and priors.

Method	Training datasets	Extra Priors
Existing Model(GT bbox+Est. Camera-pose)		
SPEC [12]	(a, b, c, e, u, v)	(iii)
Existing Model(2D Detector+Weak-persp. depth)		
OpenPose+SPIN [13]	(b*, c, e, f, g, h)	-
YoloV3+VIBE [11](video)	(a, b*, c, k, n, o, p, q)	(i)
BMP [30]	(b*, c, e, f, g, h, i, k)	-
CRMH [5]	(b*, c, e, f, g, h, k)	-
Faster-RCNN+OCHMR [10]	(b*, c, e, f, g, h, k)	-
ROMP [29]	(a, b, c, d, e, f, g, h, i, j, k, l, m)	-
Existing Model(2D Detector+Est. depth)		
BEV [24](SMPL-Age)	(a-v)	(ii, iv)
Ours		
PMMD	(a, b, c, e, f, g, h, r, s)	-

[1]Datasets with star ("x*") indicate their GT were generated by [18].
[2]Datasets with underline ("x") indicate that their SMPL GT is not publicly released by the time of WACV2024 deadline.

many databases above, all the existing advances depend on method-specific and evaluation-data-specific training recipes. To our knowledge of existing methods [5, 24, 29], the most common elements among previous recipes are the sampling probabilities of databases used on each loss and in each training step. Here we only present the recipe that makes PMMD work well.

(a) Data sampling by losses: For $\mathcal{L}_{Detection}$, we used all datasets that are annotated with the object class labels and 2D bounding boxes of person. For $\mathcal{L}_{j3d}$ and $\mathcal{L}_{j2d}$, we used the 2D joint annotations provided in MS-COCO [17], LSP [6], LSP extend [7], MPII [2], and 3D joint annotations in Human3.6M [4], 3DPW [26], MPI-Inf-3DHP [21] respectively. Note that for the two datasets that have ground truth SMPL coefficients (Human3.6M, 3DPW), we used the 3D joints sampled from the ground truth SMPL meshes for training instead. For $\mathcal{L}_{\theta}$ and $\mathcal{L}_{pose}$, the losses were only computed on the datasets that have ground truth SMPL coefficients [4, 26]. For the global losses $\mathcal{L}_{Global}$, $\mathcal{L}_{GRes}$ and $\mathcal{L}_{GVertex}$, we only calculated these losses on the datasets that provide frame-wise camera motion metrics (extrinsics) [4, 26]. For the mask loss $\mathcal{L}_{Mask}$, we used the ground truth person instance mask annotations in MS-COCO [17].

(b) Data Sampling by Training Steps:

- In Step-1: Sampling probability of datasets are listed: 0.6(3DPW), 0.6(Human3.6M), 0.3(COCO), 0.2(AGORA), 0.3(LSPET), 0.3(LSP), 0.3(MPII), 0.1(MPI_INF_3DHP), 0.4(CLIFF-coco peseudo-GT), and 0.4(CLIFF-mpii peseudo-GT).

- In Step-2: Sampling probability of datasets are listed: 0.8(3DPW), 0.5(Human3.6M), 0.15(COCO), 0.05(MPII), 0.1(AGORA), 0.05(MPII), 0.05(MPI_INF_3DHP), 0.3(CLIFF-coco peseudo-GT), and 0.3(CLIFF-mpii peseudo-GT).

3. Aligning Global Coordinates for Training

In our work, to utilize different datasets collected under different camera intrinsics, we align input images, GT mesh

translation and GT meshes accordingly for consistent training and fair evaluation.

We first define a common intrinsic matrix $\mathbf{K} \in \Re^{3\times3}$ for projecting meshes to the image plane:

$$\begin{bmatrix} f & 0 & w/2 \\ 0 & f & h/2 \\ 0 & 0 & 1 \end{bmatrix}, \tag{1}$$

where w, h are the size of the image plane and $f = 1000$ is the focal length manually set following [5, 29, 30].

For GT translation alignment, given the image resizing scale factor, the image shifting pixel offsets, and the real and manually-set camera focal lengths, the aligning process is implemented as in Algorithm 1. In practice, we achieve such alignment on-the-fly in the dataloader for individual input images. The alignment process is also triggered during our proposed global padding augmentation. For any resizing/shifting augmentation on an input image, we adjust the corresponding GT translations of all meshes in this image accordingly.

For GT SMPL alignment, we transform the problem of calculating the absolute GT global translation into the problem of calculating the local-to-global translations. Giving the GT camera extrinsic $\mathbf{E} \in \Re^{4\times4}$ and the GT SMPL coefficients $\theta \in \Re^{10\times1}, \beta \in \Re^{24\times3}$ in the world space, we calculate the ground truth translation vector $\mathbf{t}_{gt} \in \Re^{1\times3}$ from the local space to the global (camera) space in a preprocessing step. Specifically, a translation vector $\mathbf{t}_{gt}$ is an average distance of the same mesh between the local space and global space after aligning the mesh root rotation to the global space. The procedure is implemented in Algorithm 2 in PyTorch style. We processed all the annotated meshes to obtain GT local-global translations $\mathbf{t}_{gt}$, which are later used in baseline and PMMD training.

Algorithm 1 Aligning GT translations

```
# f: pre-defined common focal length
# f_real: dataset-aware real focal length
# scale: scale factor of image resizing
# t: aligned GT image-camera translation,
#    of the shape [3,]
# pos_x: pixel offset on x axis
# pos_y: pixel offset on y axis
# img_shape: the shape of network input,
#            of the shape [w,h,3]
def align_translation(f, f_real, scale,
                      t, pos_x, pos_y, img_shape):
    # scale depth by image resizing scale
    t[-1] /= scale

    # align x axis
    offset = img_shape[0] // 2 - pos_x
    t[0] -= (t[-1] * offset) / f_real

    # align y axis
    offset = img_shape[1] // 2 - pos_y
    t[1] -= (t[-1] * offset) / f_real

    # align z axis
    t[-1] /= f_real / f

    return t
```

Algorithm 2 Calculating GT local-global translations

```
# E: ground truth extrinsics
# pose: ground truth SMPL pose coefficients
# shape: ground truth SMPL shape coefficients

def obtain_GT_translation(E, pose, shape):
    # transform root rotation to camera space
    root_cam = pose[0] @ E[:3,:3]
    pose_cam = pose
    pose_cam[0] = root_cam

    # construct GT mesh with camera space rotations
    M_rot = SMPL(pose_cam, shape)

    # construct GT mesh in world space
    M = SMPL(pose, shape)

    # transform GT mesh from world to camera
    M_cam = torch.ones(6890, 4)
    M_cam[:,:3] = M
    M_cam = M_cam @ E
    M_cam = M_cam[:,:3] / M_cam[:,[-1]]

    # obtain GT image-camera translations
    t = torch.mean(M_cam - M_rot)

    # t in the shape [3,]
    return t
```

4. Cuda-based Center-Padding RoI-Align

We design a Cuda-based Center-Padding RoI-Align module. The process is implemented as in Algorithm 3. This function is a CUDA kernel that performs forward pass computation of Region of Interest (RoI) pooling with center padding on input feature maps. It takes the following arguments:

- $nthreads$ ($integer$): The number of threads to be launched for the kernel.
- $bottom\_data$ (pointer to $scalar\_t$): A pointer to the input feature map tensor in row-major order.
- $bottom\_RoIs$ (pointer to $scalar\_t$): A pointer to the tensor containing RoIs in the format $(batch\_index, x1, y1, x2, y2)$ where $(x1, y1)$ and $(x2, y2)$ are the top-left and bottom-right corners of the RoI, respectively.
- $spatial\_scale$ ($scalar\_t$): A scalar representing the spatial scale factor.
- $sample\_num$ ($integer$): The number of sampling points in each bin. If it is set to 0, then it is set to the ceiling value of RoI width or height divided by the corresponding pooled width or height.
- $channels$ ($integer$): The number of channels in the input feature map.

- $height$ ($integer$): The height of the input feature map.
- $width$ ($integer$): The width of the input feature map.
- $pooled\_height$ ($integer$): The height of the output pooled feature map.
- $pooled\_width$ ($integer$): The width of the output pooled feature map.
- top_data (pointer to $scalar\_t$): A pointer to the output tensor in row-major order.

The function performs RoI pooling by dividing each RoI into $pooled\_height \times pooled\_width$ non-overlapping bins and taking the maximum value of each bin. Center padding is applied to the RoI before the pooling operation, such that the RoI is made square and then padded on both sides if necessary to make its height and width equal to the maximum of the two dimensions. The function then performs bilinear interpolation to sample values from the input feature map at evenly spaced points within each bin. Finally, the output values are averaged over the number of sampling points within each bin.

References

[1] Mykhaylo Andriluka, Umar Iqbal, Eldar Insafutdinov, Leonid Pishchulin, Anton Milan, Juergen Gall, and Bernt Schiele. Posetrack: A benchmark for human pose estimation and tracking. In *CVPR*, pages 5167–5176, 2018. 2

[2] Mykhaylo Andriluka, Leonid Pishchulin, Peter Gehler, and Bernt Schiele. 2d human pose estimation: New benchmark and state of the art analysis. In *CVPR*, pages 3686–3693, 2014. 2, 3

[3] Joao Carreira and Andrew Zisserman. Quo vadis, action recognition? a new model and the kinetics dataset. In *CVPR*, pages 6299–6308, 2017. 2

[4] Catalin Ionescu, Dragos Papava, Vlad Olaru, and Cristian Sminchisescu. Human3.6m: Large scale datasets and predictive methods for 3d human sensing in natural environments. *IEEE Transactions on Pattern Analysis and Machine Intelligence*, 36(7):1325–1339, jul 2014. 2, 3

[5] Wen Jiang, Nikos Kolotouros, Georgios Pavlakos, Xiaowei Zhou, and Kostas Daniilidis. Coherent reconstruction of multiple humans from a single image. In *CVPR*, pages 5579–5588, 2020. 3, 4

[6] Sam Johnson and Mark Everingham. Clustered pose and nonlinear appearance models for human pose estimation. In *bmvc*, volume 2, page 5. Citeseer, 2010. 2, 3

[7] Sam Johnson and Mark Everingham. Learning effective human pose estimation from inaccurate annotation. In *CVPR 2011*, pages 1465–1472. IEEE, 2011. 2, 3

[8] Angjoo Kanazawa, Michael J Black, David W Jacobs, and Jitendra Malik. End-to-end recovery of human shape and pose. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7122–7131, 2018. 2

[9] Angjoo Kanazawa, Jason Y Zhang, Panna Felsen, and Jitendra Malik. Learning 3d human dynamics from video. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 5614–5623, 2019. 2

[10] Rawal Khirodkar, Shashank Tripathi, and Kris Kitani. Occluded human mesh recovery. In *CVPR*, pages 1715–1725, June 2022. 3

[11] Muhammed Kocabas, Nikos Athanasiou, and Michael J. Black. VIBE: Video inference for human body pose and shape estimation. In *CVPR*, 2020. 2, 3

[12] Muhammed Kocabas, Chun-Hao P. Huang, Joachim Tesch, Lea Müller, Otmar Hilliges, and Michael J. Black. Spec: Seeing people in the wild with an estimated camera. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, pages 11035–11045, October 2021. 2, 3

[13] Nikos Kolotouros, Georgios Pavlakos, Michael J Black, and Kostas Daniilidis. Learning to reconstruct 3d human pose and shape via model-fitting in the loop. In *ICCV*, 2019. 3

[14] Christoph Lassner, Javier Romero, Martin Kiefel, Federica Bogo, Michael J Black, and Peter V Gehler. Unite the people: Closing the loop between 3d and 2d human representations. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 6050–6059, 2017. 2

[15] Jiefeng Li, Can Wang, Hao Zhu, Yihuan Mao, Hao-Shu Fang, and Cewu Lu. Crowdpose: Efficient crowded scenes pose estimation and a new benchmark. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10863–10872, 2019. 2

[16] Zhihao Li, Jianzhuang Liu, Zhensong Zhang, Songcen Xu, and Youliang Yan. Cliff: Carrying location information in full frames into human pose and shape estimation. In *ECCV*, 2022. 2

[17] Tsung-Yi Lin, Michael Maire, Serge Belongie, James Hays, Pietro Perona, Deva Ramanan, Piotr Dollár, and C Lawrence Zitnick. Microsoft coco: Common objects in context. In *European conference on computer vision*, pages 740–755, 2014. 2, 3

[18] Matthew M. Loper, Naureen Mahmood, and Michael J. Black. MoSh: Motion and shape capture from sparse markers. *ACM Transactions on Graphics, (Proc. SIGGRAPH Asia)*, 33(6):220:1–220:13, Nov. 2014. 2, 3

[19] Naureen Mahmood, Nima Ghorbani, Nikolaus F Troje, Gerard Pons-Moll, and Michael J Black. Amass: Archive of motion capture as surface shapes. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 5442–5451, 2019. 2

[20] Dushyant Mehta, Helge Rhodin, Dan Casas, Pascal Fua, Oleksandr Sotnychenko, Weipeng Xu, and Christian Theobalt. Monocular 3d human pose estimation in the wild using improved cnn supervision. In *2017 international conference on 3D vision (3DV)*, pages 506–516. IEEE, 2017. 2

[21] Dushyant Mehta, Srinath Sridhar, Oleksandr Sotnychenko, Helge Rhodin, Mohammad Shafiei, Hans-Peter Seidel, Weipeng Xu, Dan Casas, and Christian Theobalt. Vnect: Real-time 3d human pose estimation with a single rgb camera. *ACM Transactions on Graphics (TOG)*, 36(4):1–14, 2017. 2, 3

[22] Lea Muller, Ahmed AA Osman, Siyu Tang, Chun-Hao P Huang, and Michael J Black. On self-contact and human pose. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 9990–9999, 2021. 2

[23] Priyanka Patel, Chun-Hao P. Huang, Joachim Tesch, David T. Hoffmann, Shashank Tripathi, and Michael J. Black. Agora: Avatars in geography optimized for regression analysis. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, pages 13468–13478, June 2021. 2

[24] Yu Sun, Wu Liu, Qian Bao, Yili Fu, Tao Mei, and Michael J. Black. Putting people in their place: Monocular regression of 3D people in depth. In *IEEE/CVF Conf. on Computer Vision and Pattern Recognition (CVPR)*, June 2022. 2, 3

[25] Zhang Tianshu, Huang Buzhen, and Wang Yangang. Object-occluded human shape and pose estimation from a single color image. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2020. 2

[26] Timo von Marcard, Roberto Henschel, Michael Black, Bodo Rosenhahn, and Gerard Pons-Moll. Recovering accurate 3d human pose in the wild using IMUs and a moving camera. In *ECCV*, 2018. 2, 3

[27] Jiahong Wu, He Zheng, Bo Zhao, Yixin Li, Baoming Yan, Rui Liang, Wenjia Wang, Shipei Zhou, Guosen Lin, Yanwei Fu, et al. Ai challenger: A large-scale dataset for going deeper in image understanding. *arXiv preprint arXiv:1711.06475*, 2017. 2

[28] Jianxiong Xiao, Krista A Ehinger, Aude Oliva, and Antonio Torralba. Recognizing scene viewpoint using panoramic place representation. In *2012 IEEE Conference on Computer Vision and Pattern Recognition*, pages 2695–2702. IEEE, 2012. 2

[29] Sun Yu, Bao Qian, Liu Wu, Fu Yili, Michael J. Black, and Mei Tao. Monocular, one-stage, regression of multiple 3d people. In *arxiv:2008.12272*, August 2020. 3, 4

[30] Jianfeng Zhang, Dongdong Yu, Jun Hao Liew, Xuecheng Nie, and Jiashi Feng. Body meshes as points. In *CVPR*, 2021. 3, 4

[31] Weiyu Zhang, Menglong Zhu, and Konstantinos G Derpanis. From actemes to action: A strongly-supervised representation for detailed action understanding. In *Proceedings of the IEEE International Conference on Computer Vision*, pages 2248–2255, 2013. 2

Algorithm 3 Cuda-based Center-Padding RoI-Align

```
template <typename scalar_t>
__global__ void RoIPadCenterForward(const int nthreads, const scalar_t *bottom_data,
                                const scalar_t *bottom_RoIs,
                                const scalar_t spatial_scale,
                                const int sample_num, const int channels,
                                const int height, const int width,
                                const int pooled_height, const int pooled_width,
                                scalar_t *top_data) {
  CUDA_1D_KERNEL_LOOP(index, nthreads) {
    // (n, c, ph, pw) is an element in the aligned output
    int pw = index % pooled_width;
    int ph = (index / pooled_width) % pooled_height;
    int c = (index / pooled_width / pooled_height) % channels;
    int n = index / pooled_width / pooled_height / channels;

    const scalar_t *offset_bottom_RoIs = bottom_RoIs + n * 5;
    int RoI_batch_ind = offset_bottom_RoIs[0];
    scalar_t RoI_start_w = offset_bottom_RoIs[1] * spatial_scale;
    scalar_t RoI_start_h = offset_bottom_RoIs[2] * spatial_scale;
    scalar_t RoI_end_w = (offset_bottom_RoIs[3] + 1) * spatial_scale;
    scalar_t RoI_end_h = (offset_bottom_RoIs[4] + 1) * spatial_scale;

    // Force malformed RoIs to be 1x1
    scalar_t RoI_width = fmaxf((scalar_t)RoI_end_w - RoI_start_w, 0.);
    scalar_t RoI_height = fmaxf((scalar_t)RoI_end_h - RoI_start_h, 0.);

    // make square
    scalar_t square_length = fmaxf(RoI_width, RoI_height);
    scalar_t RoI_width_offset = RoI_width;
    scalar_t RoI_height_offset = RoI_height;
    RoI_width = square_length;
    RoI_height = square_length;

    // Padding
    scalar_t pad_w = 0;
    scalar_t pad_h = 0;
    if (RoI_height > RoI_height_offset) {
      pad_h = (RoI_height - RoI_height_offset) / 2;
    }
    if (RoI_width > RoI_width_offset) {
      pad_w = (RoI_width - RoI_width_offset) / 2;
    }

    scalar_t bin_size_h = RoI_height / pooled_height;
    scalar_t bin_size_w = RoI_width / pooled_width;
    if (ph * bin_size_h > RoI_height_offset + pad_h || pw * bin_size_w > RoI_width_offset + pad_w
      || ph * bin_size_h < pad_h || pw * bin_size_w < pad_w) {
      top_data[index] = 0;
    } else {
      const scalar_t *offset_bottom_data = bottom_data + (RoI_batch_ind * channels + c) * height * width;
      int sample_num_h = (sample_num > 0) ? sample_num : ceil(RoI_height / pooled_height);
      int sample_num_w = (sample_num > 0) ? sample_num : ceil(RoI_width / pooled_width);

      scalar_t output_val = 0;
      for (int iy = 0; iy < sample_num_h; iy++) {
        const scalar_t y = RoI_start_h - pad_h + ph * bin_size_h + (scalar_t)(iy + scalar_t(.5f))
                           * bin_size_h / (scalar_t)(sample_num_h);
        for (int ix = 0; ix < sample_num_w; ix++) {
          const scalar_t x = RoI_start_w - pad_w + pw * bin_size_w + (scalar_t)(ix + scalar_t(.5f))
                           * bin_size_w / (scalar_t)(sample_num_w);
          scalar_t val;
          if (x >= width || y >= height){
            val = 0;
          }
          else{
            val = bilinear_interpolate<scalar_t>(offset_bottom_data, height, width, y, x); // feature map coords
          }
          output_val += val;
        }
      }
      output_val /= (sample_num_h * sample_num_w);
      top_data[index] = output_val;
    }
  }
}
```