

# Unsupervised Monocular 3D Keypoint Discovery from Multi-View Diffusion Priors

## Supplementary Material

### A. CUB-200-2011 comparison

**Baseline reproduction.** Most unsupervised 2D keypoint methods [2–5] report their performance on the CUB-align protocol introduced in [5]. However, we note that it is challenging to ensure a fair and fully controlled comparison under this setup, as the preprocessing code, train/test splits, and the preprocessed dataset have not been publicly released by these methods. The original paper [5] specifies that images are roughly cropped using the provided landmarks, which can potentially lead to multiple plausible variants of the preprocessed dataset. Since normalized keypoint distances are highly sensitive to the crop ratio used during preprocessing, such inconsistencies make it challenging to directly compare our results with previously reported CUB-align scores.

To ensure fair comparisons under the identical evaluation protocol, we re-train baseline models that report CUB-align results in their main papers and provide official implementations [2–5]. For consistency, we use the publicly available preprocessed data from [1] and train these baselines using the same preprocessing and train/test split. For each baseline, we report both the originally published numbers and the results reproduced using their official code.

**Reproduced results and quantitative comparisons.** Table 1 (left) reports the reproduced results on the CUB-align dataset. Under the same evaluation protocol, our method outperforms other 2D baselines in terms of 2D accuracy, demonstrating that our results not only capture 3D consistent structures but also yield accurate 2D projected keypoints. The reproduced scores of StableKeypoints closely match their CUB-all (KP=4) results, likely due to their fixed-token training strategy: the model always learns with 25 top-k tokens chosen from 500 learnable keypoint embeddings, and selects the final subset via furthest point sampling, leading to consistent performance across different keypoint counts.

We additionally evaluate on the CUB-all dataset with 10 keypoints (KP=10), since 4 keypoints are often insufficient to capture overall structures. As shown in Table 1 (right), our method achieves further improvements with more keypoints and outperforms previous 2D baselines. Similar to CUB-align results, increasing the number of keypoints of StableKeypoints does not lead to the meaningful improvement due to its fixed-token sampling strategy. These results indicate that our method accurately captures structures of the objects in both 3D and 2D projections.

Table 1. Reproduced results on CUB-align and CUB-all dataset. For fair comparison, reproduced results are obtained using the identical preprocessed dataset with same train/test splits, and official source codes provided by the authors. We report reproduced results for *CUB-all*.

	<i>CUB-align</i> (KP=10)		<i>CUB-all</i>	
	Original	Reproduced	KP=4	KP=10
Lorenz <i>et al.</i> [5]	3.91	10.34	-	-
GANSeg [3]	<b>3.23</b>	15.73	-	-
Autolink [2]	4.15	8.10	13.0	11.7
StableKeypoints [4]	5.06	6.38	<b>5.6</b>	6.0
Ours	-	<b>5.16</b>	7.7	<b>5.5</b>

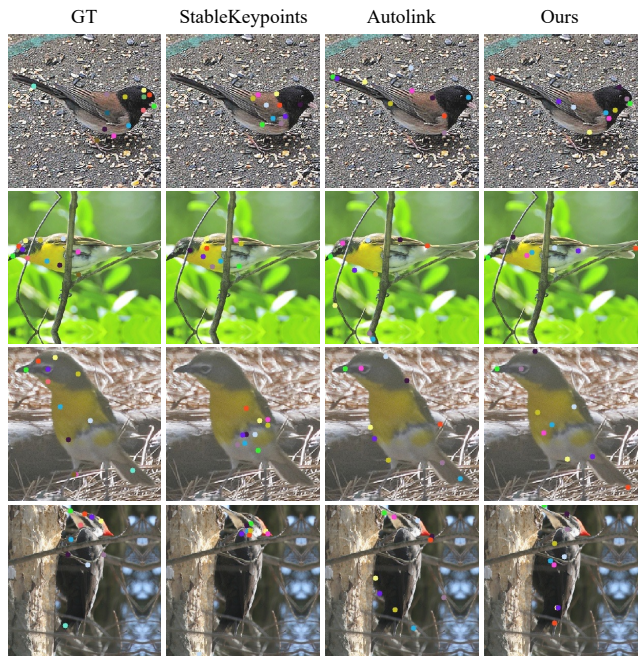
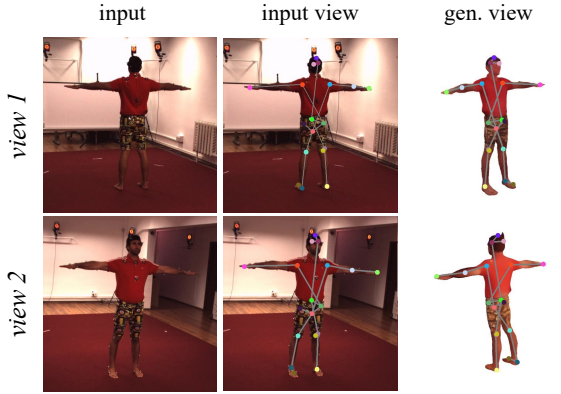
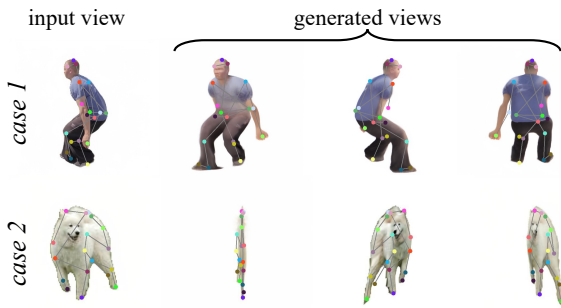


Figure 1. Qualitative comparison with unsupervised 2D keypoint estimation methods on CUB-200-2011 dataset (CUB-align).

**Qualitative comparisons.** We present qualitative comparisons of 2D keypoint predictions in Fig. 1. StableKeypoints tends to produce individually consistent keypoints but often lacks a coherent global structure across the object. AutoLink, while effective in capturing overall structure, struggles to maintain consistency under pose variations; for instance, in the first two rows, it frequently confuses the head and tail regions, assigning different keypoints to the tail depending on the viewpoint. In contrast, our method produces stable and structurally coherent keypoints



(a) Left-right ambiguity



(b) Diffusion model dependency

Figure 2. Failure cases.

across diverse poses and viewing directions. We attribute this robustness to the use of multi-view diffusion features within a 3D representation, which enables more geometry-aware keypoint reasoning.

## B. Failure cases

We illustrate our failure cases in Fig. 2. First, our method suffers from left-right ambiguity, a common issue in self-supervised keypoint learning where the model has no inherent cue to distinguish between the left and right sides [2, 7, 8]. Although our network produces consistent keypoint predictions across input and diffusion-generated views, our model has difficulty resolving the left-right ambiguity present in the input image itself. As shown in Fig. 2 (a), the model predicts the same hand (e.g., left hand) regardless of whether the input shows the front or back of the subject.

Second, since our model depends on features extracted from a pretrained diffusion model, its performance may degrade in regions where the diffusion model fails to generate plausible content. For example, as shown in *case 1* of Fig. 2 (b), the SV3D model sometimes fails to reconstruct occluded regions such as the right hand, resulting in inaccurate predictions. In another failure mode (*case 2*), SV3D

Table 2. Additional ablation results on the Human3.6M dataset. ‘default’ indicates the default configuration used in our main experiments.

	MPJPE	N-MPJPE	P-MPJPE
(a) Number of keypoints			
$N = 16$	127.99	123.93	85.57
$N = 18$ (default)	121.34	118.29	85.26
$N = 32$	119.07	116.02	85.37
$N = 48$	<b>116.45</b>	<b>113.29</b>	<b>81.30</b>
(b) Diffusion timestep			
$\tau = 300$	126.97	122.79	88.54
$\tau = 500$ (default)	<b>121.34</b>	<b>118.29</b>	<b>85.26</b>
$\tau = 700$	130.16	125.68	88.32
$\tau = 900$	131.26	127.13	88.62
(c) Number of training images			
2,200 images	134.26	130.01	90.71
4,400 images	130.85	126.74	88.96
6,600 images	127.42	123.88	85.50
8,800 images (default)	<b>121.34</b>	<b>118.29</b>	<b>85.26</b>
(d) Affine transformation			
no affine	134.54	130.69	95.09
random affine (default)	<b>121.34</b>	<b>118.29</b>	<b>85.26</b>
(e) View misalignment			
camera noise $\sigma=0.0^\circ$ (default)	121.34	118.29	<b>85.26</b>
camera noise $\sigma=0.2^\circ$	<b>121.30</b>	<b>118.18</b>	85.29
camera noise $\sigma=0.5^\circ$	124.39	121.26	86.64
camera noise $\sigma=1.0^\circ$	133.88	130.68	92.85
(f) View aggregation			
softmax (default)	<b>121.34</b>	<b>118.29</b>	<b>85.26</b>
visibility	127.12	124.29	86.30

produces inconsistent multi-view images—for instance, by flipping the subject—making it difficult for our network to predict volumetric 3D keypoints.

## C. Additional ablation study

We present additional ablation results for a more comprehensive analysis of our method. Consistent with the main paper, all ablations are conducted on the Human3.6M dataset, and we report the results with a 2-hidden-layer MLP regression.

**Number of keypoints.** We test the performance of our model with a varying number of keypoints. As reported in Table 2 (a), the performance of our model improves as the number of keypoints increases. However, as employing too many keypoints could reduce interpretability, we set the default number of keypoints  $N = 18$  in our main model. This trade-off between accuracy and interpretability can also be seen in Fig. 3.

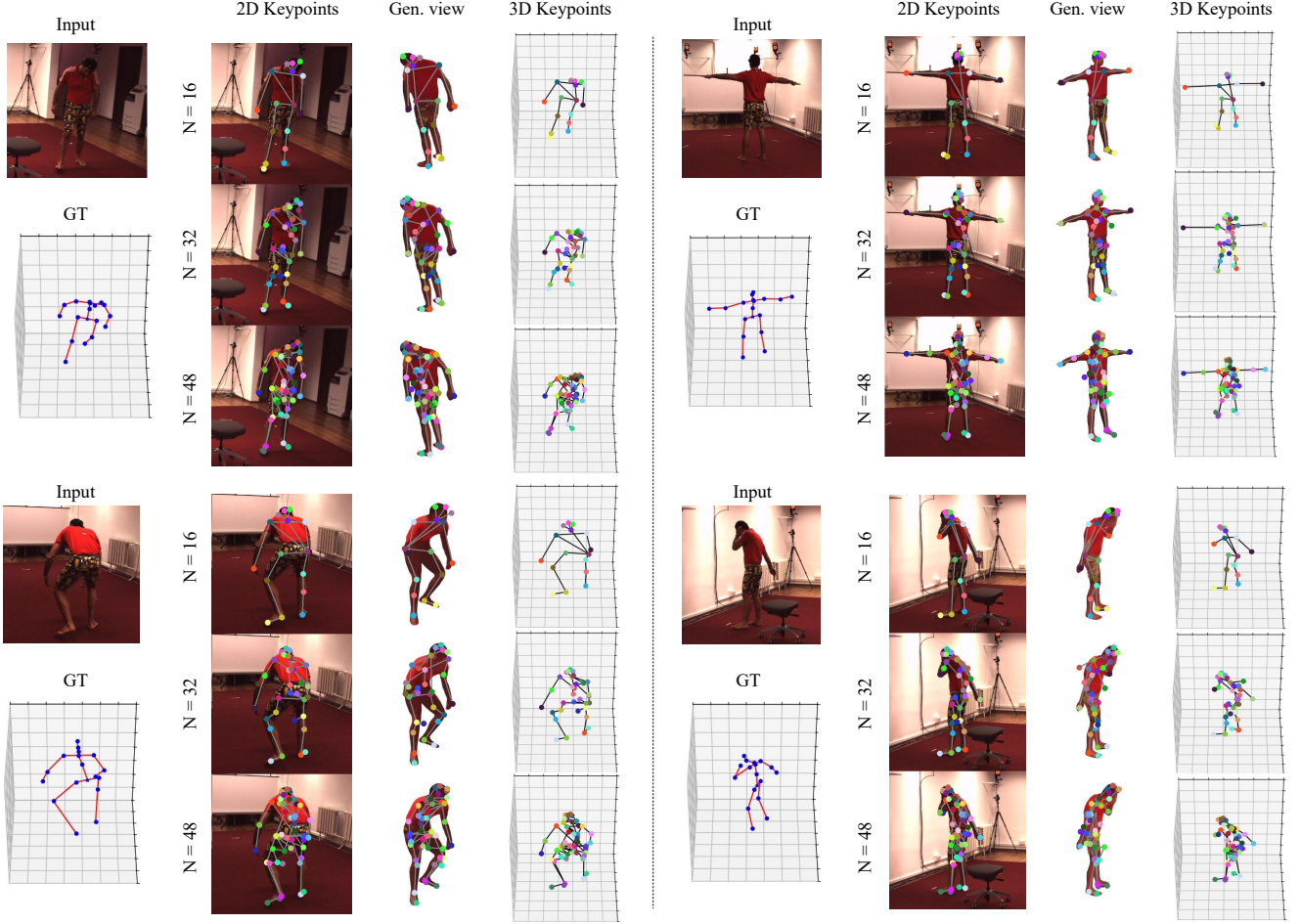


Figure 3. Keypoint prediction results according to the number of keypoints.

**Diffusion timesteps.** We analyze the performance of our model with various diffusion timesteps  $\tau$ , which are used for extracting features from the diffusion U-Net backbone. Table 2 (b) presents the ablation results on diffusion timesteps. We observe that the timestep  $\tau = 500$  yields the best results, which we set as our default configuration.

**Number of training data.** We investigate the effects of the number of training images on the performance of our model. As shown in Table 2 (c), our model delivers strong performance with fewer than 10K training images, whereas other baselines typically require at least 30K to achieve comparable results. Notably, our method only requires a set of unconstrained single-view images – without any camera parameters, temporal information, or manual annotations – which are much easier to acquire than multi-view images or 3D annotations.

**Affine transformation.** We further ablate the effect of applying an affine transformation to the input image before feeding it into the reconstruction network. As shown in Table 2 (d), removing this transformation leads to a substantial performance drop. We attribute this degradation to trivial solutions in reconstruction training: without the transformation, the reconstruction network tends to overfit to the input appearance and bypass the structural cues derived from the predicted keypoints. Applying the transformation weakens this shortcut and encourages the model to rely on structure-aware signals for reconstruction, thereby improving performance.

**View misalignment.** We evaluate the robustness of our method to view misalignment by perturbing the camera rotations used for feature unprojection with Gaussian noise. As shown in Table 2 (e), the performance degrades only slightly under mild perturbations, indicating that our method is robust to moderate inaccuracies in the poses of



Table 3. Comparison of inference-time computational cost on an NVIDIA A6000 GPU.

Model	Framework	FLOPs (GFLOPs)	VRAM (GB)	Time/Iter (ms)	Throughput (iter/s)
(2D) StableKeypoints	PyTorch	880.19	9.01	241.32	4.14
(3D) KeypointNet	TensorFlow	700.83	6.06	75.41	13.26
(3D) BKinD-3D	PyTorch	296.27	10.51	553.73	1.81
Ours - diffusion	PyTorch	7,787.39	29.48	11,930.67	0.08
Ours - keydiff3d	PyTorch	291.10	11.69	200.79	4.98

diffusion-generated views.

**View aggregation.** We further compare our default softmax-based view aggregation with a visibility-aware alternative based on depth maps predicted by VGGT. As shown in Table 2 (f), the visibility-based aggregation leads to worse performance than the default design. We attribute this degradation to errors in the predicted depth maps, which can make the visibility estimates unreliable during multi-view fusion. In contrast, the softmax-based aggregation provides more robust feature integration across views, despite not explicitly modeling visibility. This observation is also consistent with the trend reported in BKinD-3D.

### C.1. Inference Computation Cost

We provide an analysis of the inference-time computational cost in Table 3, including comparisons with existing 2D and 3D keypoint discovery baselines. Although SV3D enables our framework to estimate 3D keypoints from a single image, it also introduces substantial computational overhead. We view this as a trade-off for a capability that, to the best of our knowledge, is not supported by prior methods, namely unsupervised 3D keypoint estimation for arbitrary object categories from a single image.

Importantly, the additional cost is dominated by the diffusion stage rather than the proposed keypoint estimation module itself. As shown in Table 3, our KeyDiff3D module has computational cost comparable to existing 3D keypoint methods, while the diffusion model accounts for most of the extra FLOPs, memory usage, and latency required for monocular 3D keypoint estimation. Moreover, our framework uses diffusion features obtained from partial denoising, which is more efficient than relying on fully denoised generated images.

## D. Additional implementation details

### D.1. Additional loss

For Stanford Dogs and CUB-200-2011 datasets, we introduce a *view-invariant consistency loss* to regularize 3D keypoint prediction under the feature-space viewpoint perturbations. Given an input image, we obtain 2D projections of the predicted 3D keypoints  $\mathbf{S}$ . We then apply a random 3D rotation to the intermediate 3D volume features, predict a second set of keypoints, and project them to obtain  $\mathbf{S}_{\text{rot}}$ .

The loss encourages consistency between the two projections:

$$\mathcal{L}_{\text{vic}} = \frac{1}{N} \sum_{i=1}^N \|\mathbf{s}_i - \mathbf{s}_{\text{rot},i}\|_2^2,$$

and is added to the total loss with a weighting factor of  $\lambda_{\text{vic}} = 0.1$ .

### D.2. Model architecture details

**Feature aggregator.** The feature aggregator fuses multi-scale features extracted from different layers of the diffusion decoder. Prior to aggregation, all feature maps are bilinearly upsampled to a shared spatial resolution of  $72 \times 72$ . Each feature map is processed by a bottleneck block consisting of three convolutional layers: a  $1 \times 1$  projection layer, a  $3 \times 3$  convolution, and a final  $1 \times 1$  expansion layer. All layers are followed by GroupNorm (32 groups) and ReLU activation. In addition, the residual shortcut connection is applied through a  $1 \times 1$  convolution. The output features are projected to a unified embedding dimension (default: 384) across all layers. After processing all layers, the resulting features are aggregated via a learnable softmax-weighted sum. The weights are initialized uniformly and optimized during training. The final output is a fused feature map of shape  $[B, 384, 72, 72]$ .

**Keypoint head.** The keypoint head takes the aggregated 2D feature map as input and predicts multi-view keypoint features with  $C''$  channels for each view. This head consists of a lightweight convolutional network with three layers. The first two layers use  $3 \times 3$  convolutions followed by SiLU activation, progressively reducing the channel dimension (from 128 to 32). The final layer uses a  $1 \times 1$  convolution to project the features to  $N$  channels, followed by a Tanh activation to bound the output values. This module is shared across all views and produces a keypoint-specific feature map of a shape  $[B, C'', H, W]$ .

**Volumetric decoder.** The volumetric decoder predicts  $N$  keypoint heatmaps over a voxel grid of resolution  $72 \times 72 \times 72$  from an input 3D feature volume. To ensure computational efficiency and avoid excessive memory usage, we adopt a 3D CNN architecture inspired by V2V-PoseNet [6], where a final 3D CNN architecture outputs  $N$  volumetric heatmaps, from which 3D keypoint coordinates are estimated via soft-argmax.

**Reconstruction network.** We employ an encoder-decoder network to reconstruct RGB images conditioned on edge and keypoint heatmaps. Unlike standard U-Net or hourglass networks, we do not use skip connections between encoder and decoder layers. Instead, the decoder directly incorporates the predicted edge and keypoint maps



at each resolution level to guide structure-aware synthesis. The encoder consists of downsampling residual blocks, while the decoder performs upsampling and concatenates the corresponding edge and keypoint maps at each stage. The final output is passed through a  $1 \times 1$  convolution followed by Tanh activation.

### D.3. Data preprocessing

**Foreground mask.** To generate foreground masks for object-centric inputs, we leverage different segmentation models depending on the dataset. For the Human3.6M dataset, we use SAM with the 2D keypoint locations provided as query points. For animal datasets, such as Stanford Dogs and CUB-200-2011, we use Grounded-SAM2, which allows text prompts as input. In this case, we use the class names provided in the dataset (e.g., “dog”) as the prompt to extract object-specific masks.

**Crop and align.** To ensure that the foreground object is centered in the image, we perform cropping and alignment based on either keypoints or segmentation masks. For Human3.6M, we compute a bounding box that tightly encloses all projected 2D keypoints, and shift it to the center of the image. For other datasets, such as Stanford Dogs, we instead compute a bounding box from the foreground mask. All images are then resized to  $576 \times 576$ , which matches the input resolution of the pretrained SV3D model. During evaluation on Human3.6M, we map the predicted results back to the original image space using the inverse of the affine transformation matrix obtained from the OpenCV’s `warpAffine` function.

## E. Video results

Please refer to the accompanying file *demo.mp4* for more comprehensive descriptions of our framework, 3D keypoint results across diverse object categories, and animation results.

## References

- [1] Sandro Braun. unsupervised-disentangling. [https://github.com/theRealSuperMario/unsupervised-disentangling/tree/reproducing\\_baselines](https://github.com/theRealSuperMario/unsupervised-disentangling/tree/reproducing_baselines). 1
- [2] Xingzhe He, Bastian Wandt, and Helge Rhodin. Autolink: Self-supervised learning of human skeletons and object outlines by linking keypoints. *NeurIPS*, 35:36123–36141, 2022. 1, 2
- [3] Xingzhe He, Bastian Wandt, and Helge Rhodin. Ganseg: Learning to segment by unsupervised hierarchical image generation. In *CVPR*, pages 1225–1235, 2022. 1
- [4] Eric Hedlin, Gopal Sharma, Shweta Mahajan, Xingzhe He, Hossam Isack, Abhishek Kar, Helge Rhodin, Andrea Tagliasacchi, and Kwang Moo Yi. Unsupervised keypoints from pretrained diffusion models. In *CVPR*, pages 22820–22830, 2024. 1
- [5] Dominik Lorenz, Leonard Bereska, Timo Milbich, and Bjorn Ommer. Unsupervised part-based disentangling of object shape and appearance. In *CVPR*, pages 10955–10964, 2019. 1
- [6] Gyeongsik Moon, Ju Yong Chang, and Kyoung Mu Lee. V2v-posenet: Voxel-to-voxel prediction network for accurate 3d hand and human pose estimation from a single depth map. In *CVPR*, pages 5079–5088, 2018. 4
- [7] Supasorn Suwajanakorn, Noah Snaveley, Jonathan J Tompson, and Mohammad Norouzi. Discovery of latent 3d keypoints via end-to-end geometric reasoning. *NeurIPS*, 31, 2018. 2
- [8] Yuxuan Yang, Chen Qian, Jiefeng Li, Xiao Sun, Bin Xiao, Yichen Wei, and Limin Wang. Mask-as-supervision: Leveraging unified mask information for unsupervised 3d pose estimation. In *ECCV*, 2024. 2

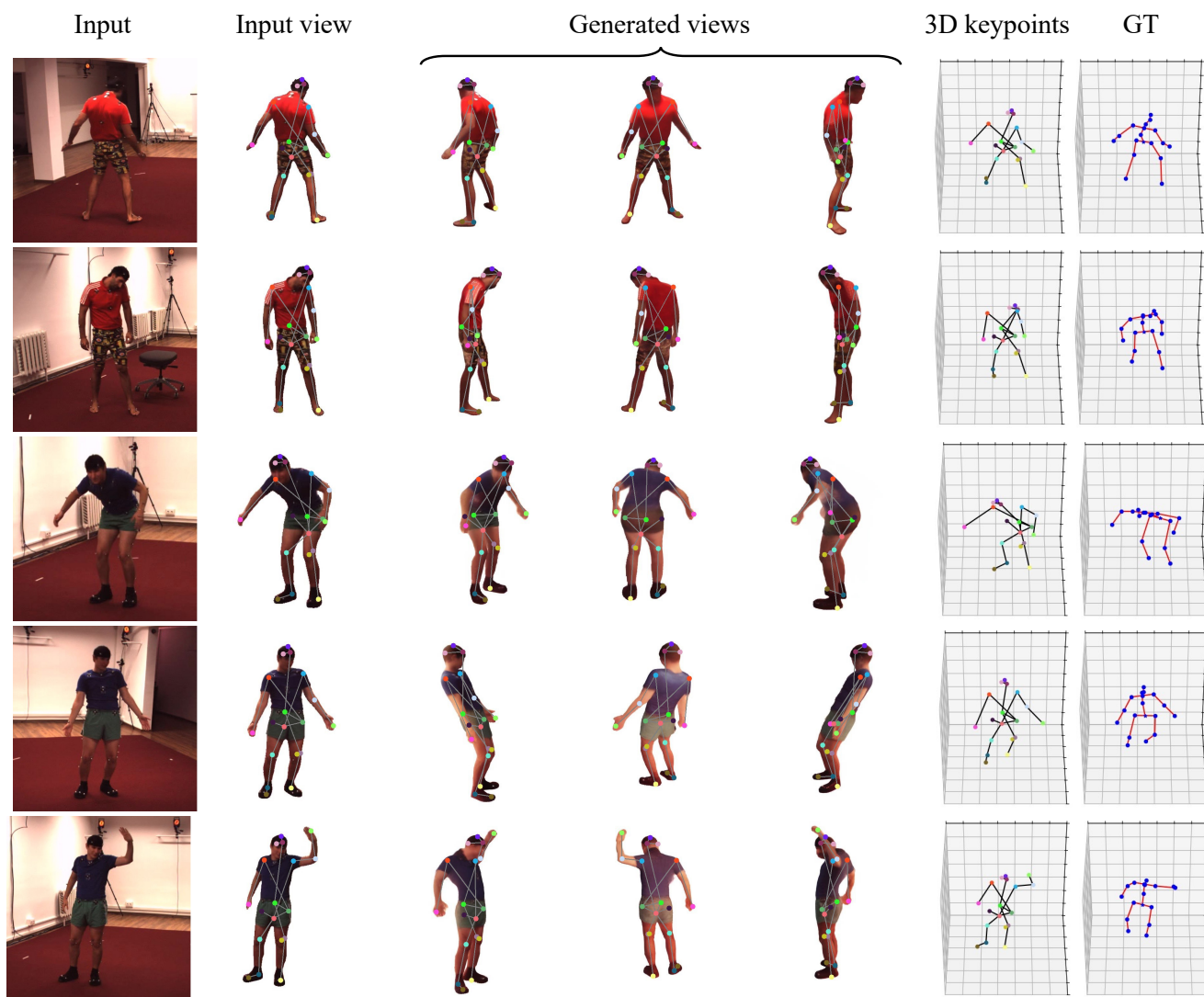


Figure 4. Additional results on Human3.6M dataset.

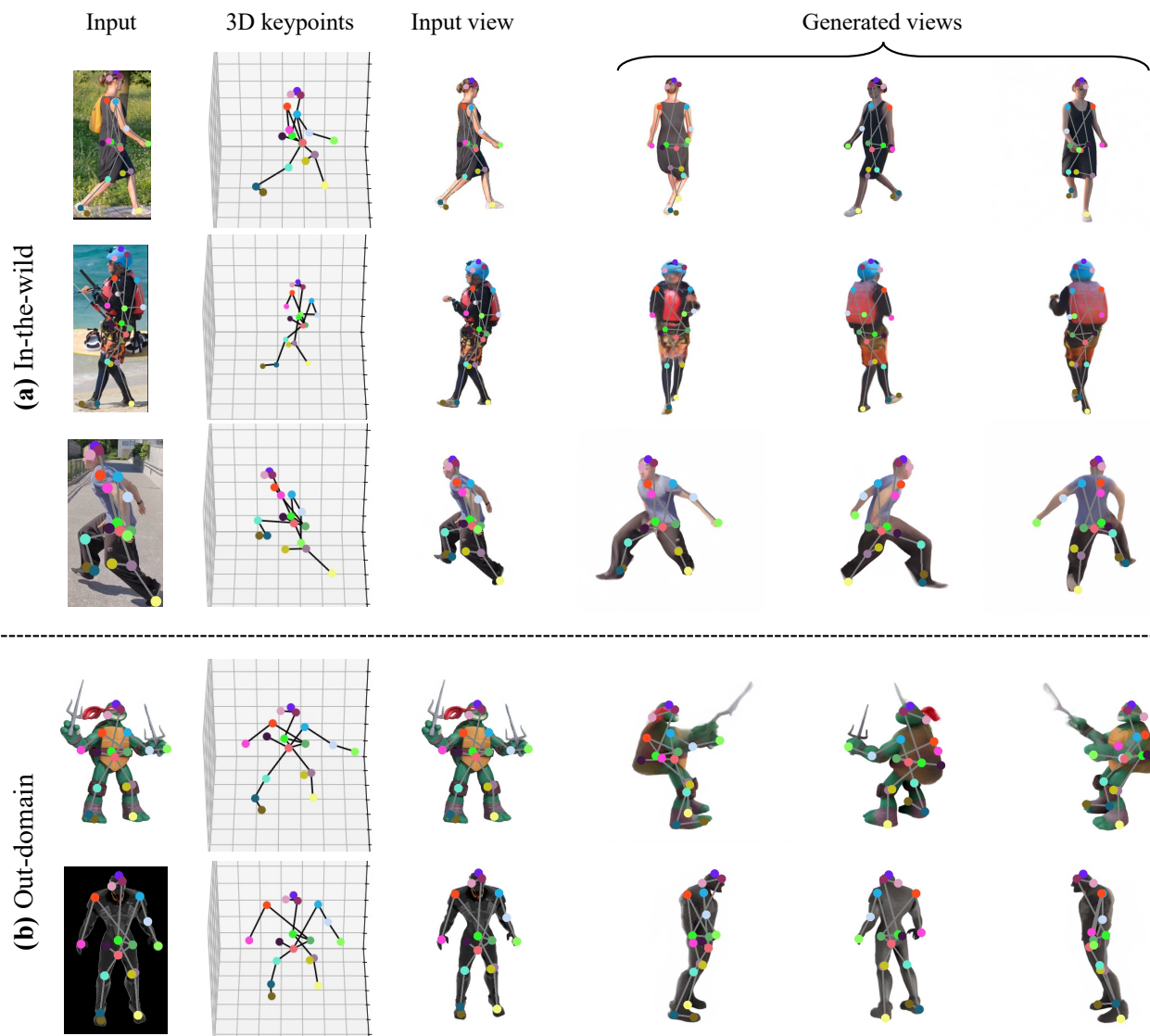


Figure 5. Results of our model trained with the Human3.6M dataset, which is tested on (a) in-the-wild and (b) out-of-domain images.



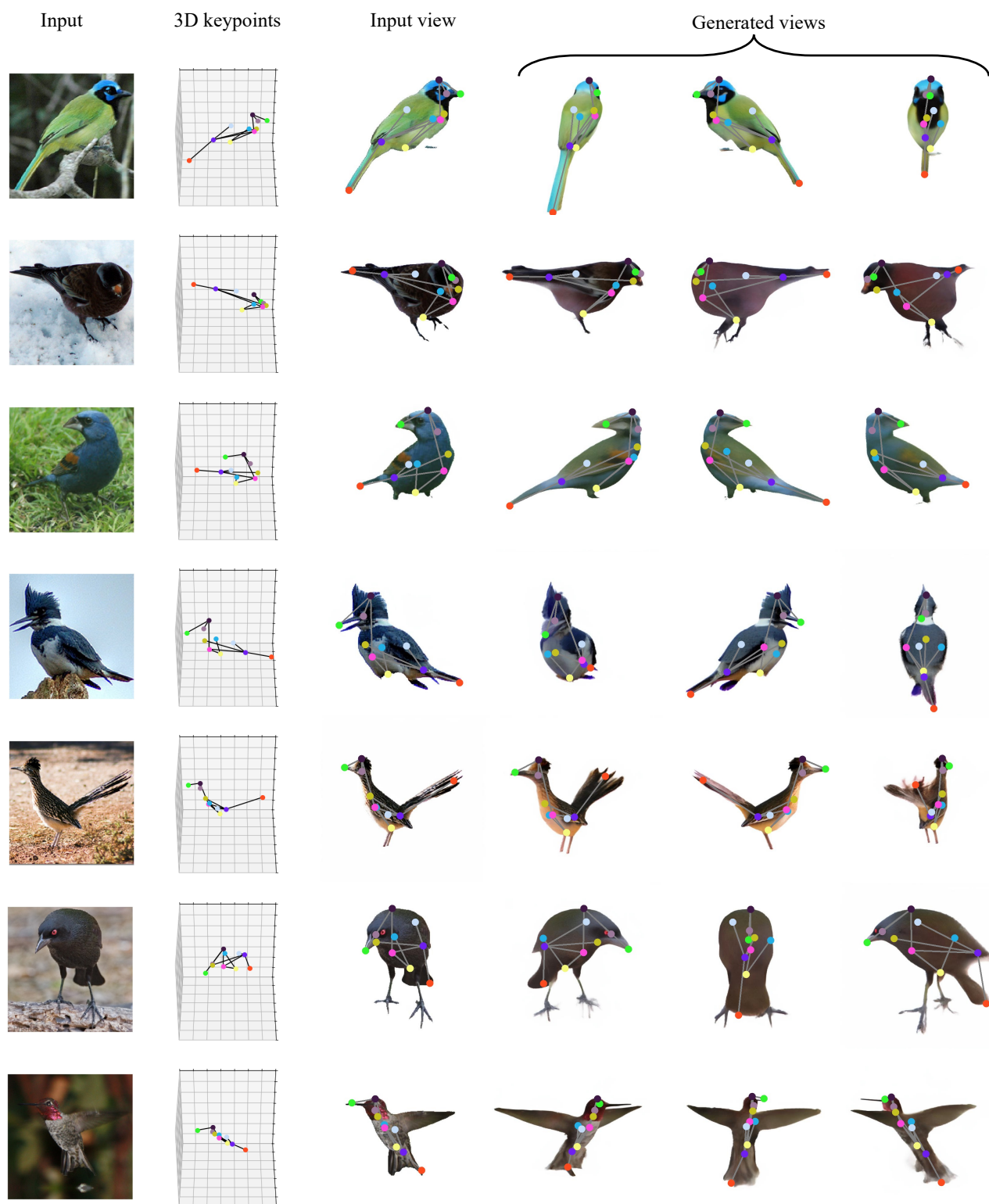


Figure 6. Additional results on CUB-200-2011 dataset (CUB-align)

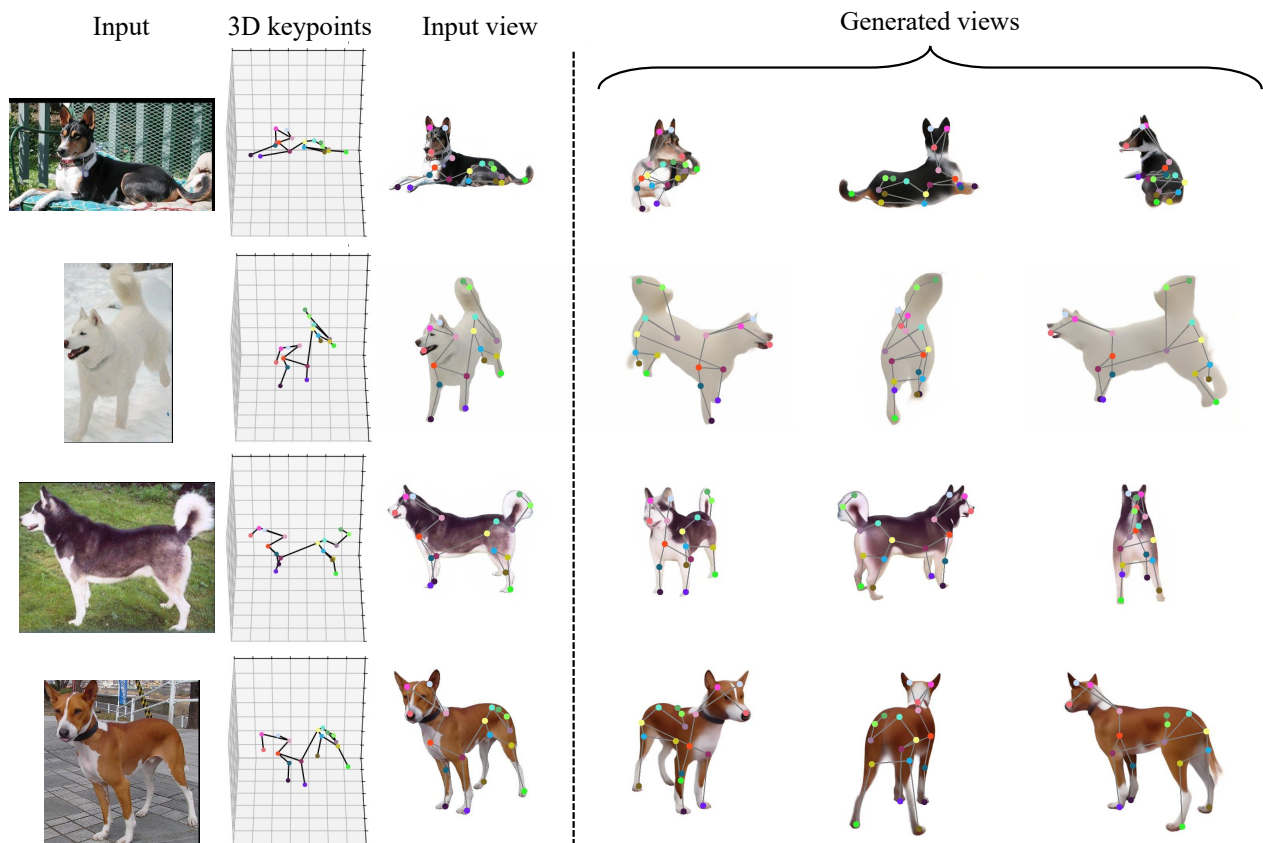


Figure 7. Additional results on Stanford Dogs dataset.

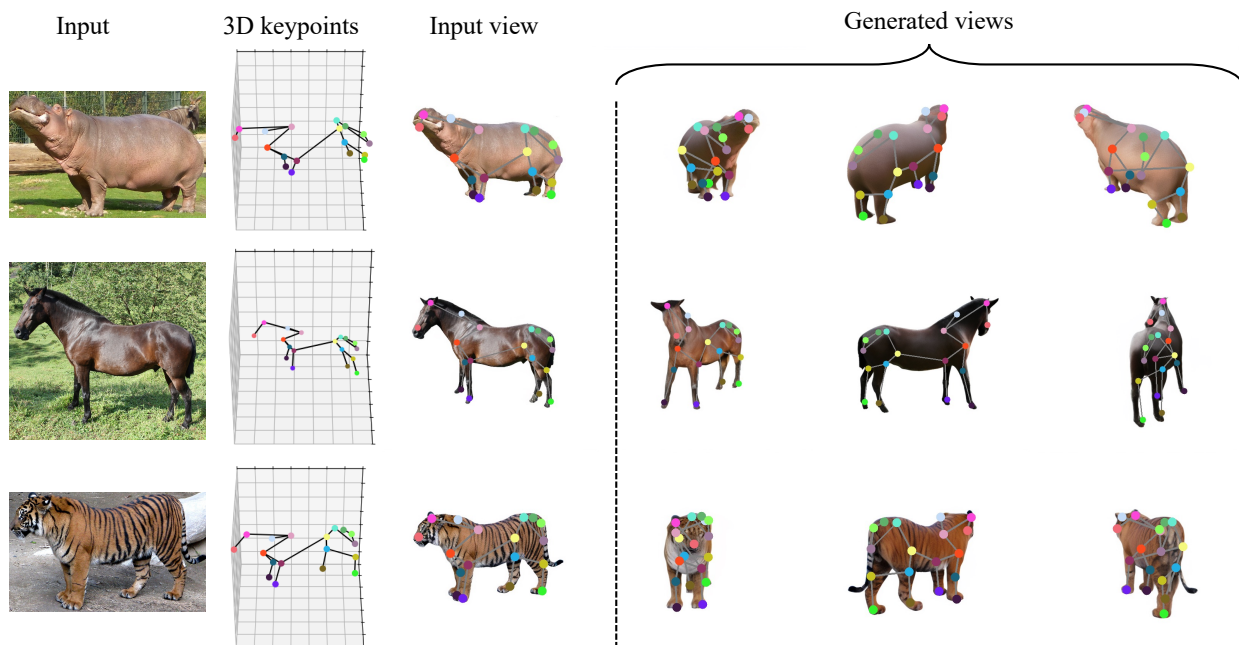


Figure 8. Additional results on the AP-10K dataset using a model trained on Stanford Dogs.