Online Adaptation for Consistent Mesh Reconstruction in the Wild

1 Overview

In the supplementary material, we provide additional details, discussions, and experiments to support the original submission. In the following, we first discuss our self-supervised setting in Sec. 2. We then describe evaluation of keypoint re-projection accuracy on videos in Sec. 3. Next, we show more ablation studies in Sec. 4. More qualitative results on both bird and zebra image reconstructions are present in Sec. 5. Details of the network design and implementation are discussed in Sec. 6 and Sec. 7, respectively. Finally, we describe failure cases and limitations in Sec. 8.

2 Self-supervised Mesh Reconstruction

We train the self-supervised image reconstruction model with only silhouettes and single-view images in a category. To this end, we also learn a template from scratch as in the self-supervised 3D mesh reconstruction approach [7]. Essentially, we do not apply the semantic parts from the SCOPS method [2], which means that no additional modules are required for training. After training the image model, we adapt it to each unlabeled video using the method discussed in Sec.3.2 in the submission. Since neither keypoints annotations, nor additional self-supervised blocks such as SCOPS are adopted, the results of this self-supervised single-view image reconstruction model do not outperform those of existing methods, i.e., [4] and [7], and of the proposed ACMR model (see Sec. 4.3 for the quantitative results). However, the test-time training improves the fidelity and the robustness of the reconstruction results as shown in Fig. 1. The reconstructions are more plausible after online adaptation, especially from unobserved views.

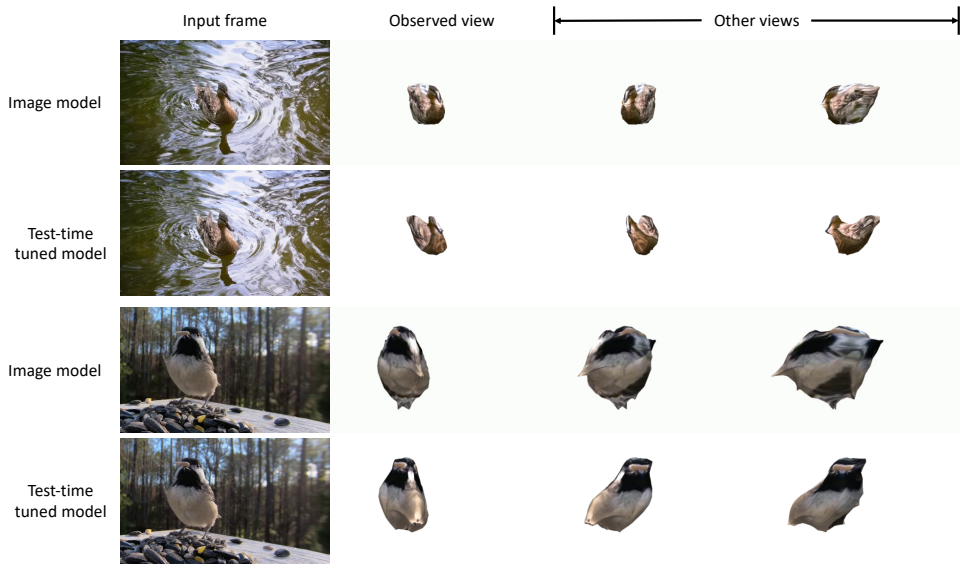


Figure 1: Comparison of the self-supervised image model and the test-time tuned self-supervised model.

34th Conference on Neural Information Processing Systems (NeurIPS 2020), Vancouver, Canada.

Table 1: Keypoint re-projection evaluation on videos. "(T)" indicates the model is test-time trained on the given video., L_c, L_t, L_s are defined in Eq.4, 5 and 6 of the main paper, respectively.

(a) Metric	(b) CMR [4]	(c) ACMR	(d) ACMR (T)	(e) ACMR-vid (T), no L_c, L_t, L_s	(f) ACMR-vid (T)
PCK@0.1 ↑	0.514	0.751	0.424	0.644	**0.794**

3 Keypoint Re-projection Accuracy on Videos

Video Keypoints Annotation. We evaluate the keypoint re-projection accuracy (see Sec 4.1 in the paper) on the 22 videos we collected. To create the ground truth keypoints, we follow the protocol of the CUB dataset [13] to annotate 15 semantic keypoints every five frames in each video, via the Labelme [12] toolkit (see Fig. 2 for the annotation interface.) Visualizations of the re-projected keypoints by different methods are visualized and compared in Fig. 3.

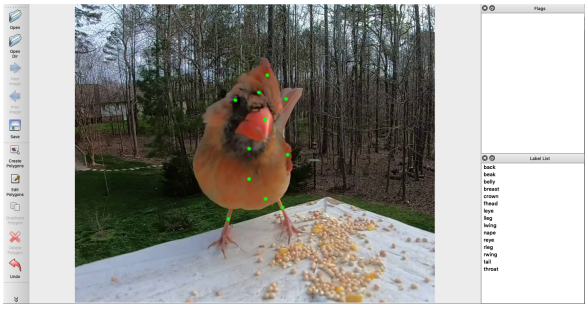


Figure 2: Keypoints annotation using the Labelme [12] toolkit.

Details of Keypoint Re-projection. Since we do not have the keypoint assignment matrix proposed in [4], we employ the canonical keypoint UV map to obtain the 3D keypoints (Sec. 3.1 in the paper). The keypoint re-projection is done by (i) warping the canonical keypoint UV map to each individual predicted mesh surface; (ii) projecting the canonical keypoint back to the 2D space via the predicted camera pose; (iii) comparing against the ground truth keypoints in 2D. This evaluation implicitly reveals the correctness of both the predicted shape and camera pose for the mesh reconstruction algorithm, especially for objects that do not have 3D ground truth annotations.

Compared to frame-wisely applying CMR [4] (Table 1 (b)) or ACMR (Table 1 (c)) discussed in Sec. 3.1 in the main paper, the test-time tuned model achieves higher PCK score, as shown in Table 1 (f). It verifies the effectiveness of the proposed test-time training procedure and the invariance constraints. Essentially, as we noted in Sec 4.1, although the original ACMR, i.e., using the ResNet-18 [1] as the image encoder with batch normalization layers [3] in Table 1 (c), achieves relatively promising results, it is hard to adapt this model to new domains like low-quality videos (e.g., when switching from the *.eval()* mode to the *.train()* mode in PyTorch [10]). The performance drops significantly after test-time tuning as shown in Table 1 (d).

4 Ablation Studies

4.1 The Role of Shape Base

To demonstrate the superiority of using a set of shape bases versus using a single template, we train a baseline model where we replace the shape combination branch with the template obtained by the CMR approach [4]. This setting is equivalent to a using a single shape base (denoted as *single base*). We show quantitative and qualitative comparisons with the proposed ACMR model in Table 2 and Fig. 4, respectively. As shown in Table 2(b) *vs.* (c), the model trained with a single base template struggles to fit the final shape when the instance is largely different from the given template (also shown in Fig. 4). In contrast, the proposed ACMR model with 8 shape bases performs favorably against the single base model.

Table 2: Quantitative comparison of the single base model with the proposed ACMR model.

(a) Metric	(b) Single base	(c) ACMR
Mask IoU ↑	0.605	0.708
PCK@0.1 ↑	0.655	0.855

4.2 ARAP Constraint in Online Adaptation

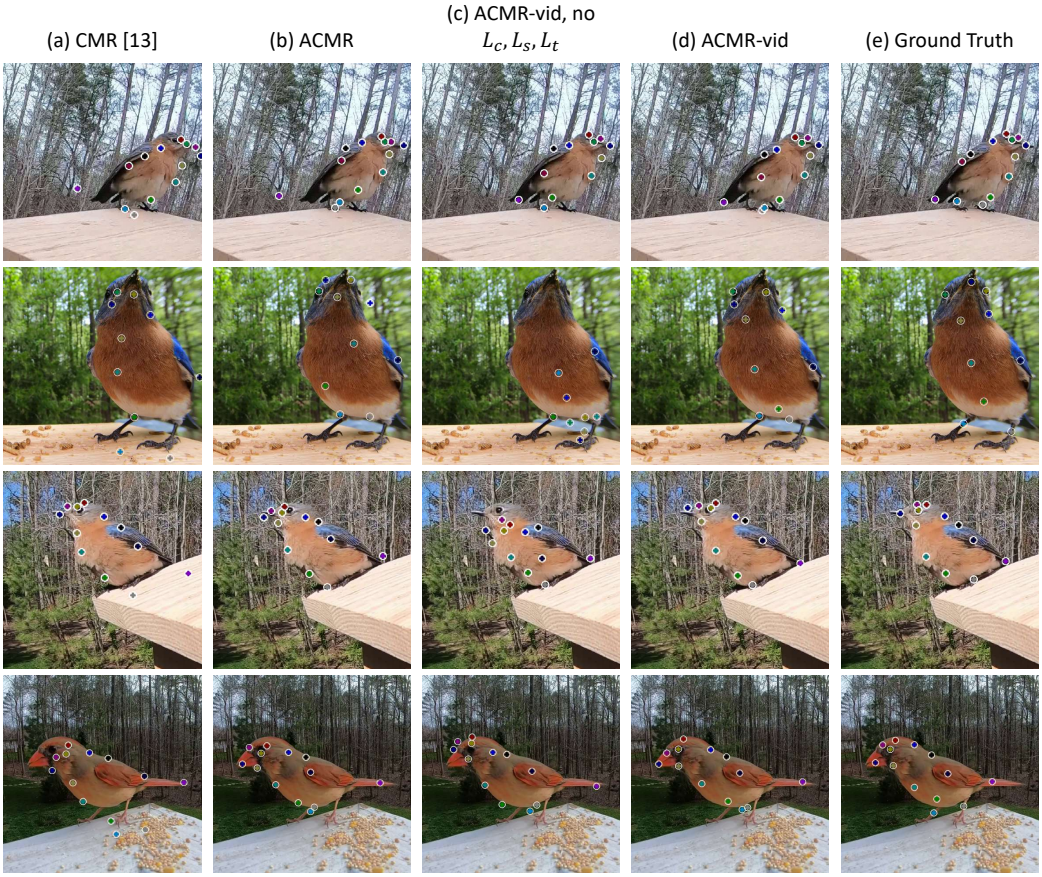


Figure 3: Visualization of re-projected keypoints on videos. We use white circles to highlight the keypoints for better visualization.

To verify the effectiveness of using the ARAP constraint in the online adaptation process, we test-time tune on the videos without this constraint. Although performing online adaptation without the ARAP constraint yields better quantitative evaluations as shown in Table 3, the reconstructed meshes are not plausible from unobserved views, as shown in Fig.6 in the submission.

Table 3: Ablation study on the ARAP constraint in online adaptation.

(a) Metric	(b) without ARAP	(c) ACMR-vid (T)
$\mathcal{J}(Mean)\uparrow$	0.875	0.868
$\mathcal{F}(Mean)\uparrow$	0.782	0.756
PCK@0.1↑	0.815	0.794

5 Qualitative Evaluations

5.1 Camera Pose Stability

To visually demonstrate the effectiveness of the test-time training procedure that stabilizes camera pose prediction, we visualize the differences in camera pose predictions between adjacent frames in Fig. 6. Compared to the model that is only trained on images, the proposed method predicts more stable camera poses that change smoothly over time.

5.2 Keypoints Re-projection for Image-based Reconstruction

We visualize re-projected keypoints on test images in Fig. 7, where the corresponding quantitative results are presented in Sec. 4.3, Table 1 of the main paper. The proposed ACMR model is able to predict more accurate keypoints compared to the CMR [4] method, especially when the bird performs an asymmetric pose, e.g. first row in Fig. 7.

Figure 4: Qualitative comparison of the single base model with the proposed ACMR model with multiple shape bases. The single base model suffers when the instance is largely different from the template, e.g. flying bird or a duck.

5.3 Single-view Image Reconstructions

In Fig. 5, we show more visualizations of reconstructions from single-view images of the test dataset of CUB birds [13] as well as comparisons with the baseline method [4]. By removing the symmetric assumption, our model is able to reconstruct objects in the input images more faithfully versus the baseline method [4] (Fig. 5(g)).

We also demonstrate the effectiveness of the ARAP constraint, as discussed in Sec.3.1 in the paper. Without this constraint, the reconstructed meshes contain unnatural spikes, as shown in Fig. 5(f).

Finally, we show reconstruction results of zebra images of the test dataset [11] in Fig. 8. Our ACMR model successfully captures motions such as head bending or walking for zebras.

6 Network Architecture

6.1 Bases Visualization

We visualize the eight shape bases obtained by applying KMeans clustering on all reconstructed meshes by the CMR [4] method for birds in Fig. 10(a). We also show the bases obtained by applying PCA to the bottleneck features of the image encoder. Note that the latter fails to discover rare shape modes (e.g., duck and flying bird) in the dataset as shown in Fig. 10(b). Thus we choose to use KMeans to obtain shape bases.

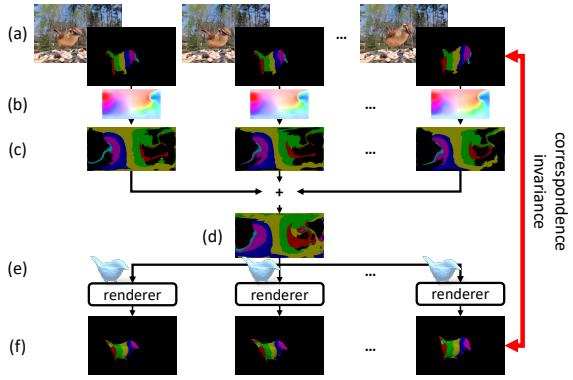


Figure 9: Part correspondence constraint. (a) Input frame and part propagations. (b) Predicted texture flows. (c) Part UV map. (d) Aggregated video-level part UV map. (e) Base shape and differentiable renderer. (f) Part rendering.

6.2 Part Correspondence Constraint

We illustrate the part correspondence constraint in details in Fig. 9. Given the propagated parts in each frame in Fig. 9(a), we map them to the UV space with the predicted texture flow in Fig. 9(b) and obtain part UV maps in Fig. 9(c). By aggregating these part UV maps, i.e., averaging, we minimize noise in each individual part UV map and obtain a video-level part UV map in Fig. 9(d). This video-level part UV map is shared by all frames in the video. Thus, for each frame, we wrap the video-level part UV map onto the base shape prediction and render it under the predicted camera pose as shown

in Fig. 9(f). Finally, we encourage consistency between part renderings and part propagations, as shown by the red arrow in Fig. 9. Through the differentiable renderer, the loss implicitly improves the predicted camera pose.

6.3 Single-view Reconstruction Network

In Fig. 11(a), we show details of our single-view reconstruction network. Given an input image, the network jointly predicts texture, shape and camera pose. By utilizing a differentiable renderer [9], we are able to utilize 2D supervision, i.e. silhouettes and input images.

6.4 Sliding Window Scheme

We show the proposed test-time tuning process in Fig. 11(b). Within each sliding window, we encourage the consistency of UV texture, UV parts as well as base shape of all frames.

7 Implementation Details

7.1 Objectives for Image Reconstruction Model

We summarize the objectives used in the single-view reconstruction model (Fig. 11(a)) discussed in Sec.3.1 in the submission as follows: (i) *foreground mask loss:* a negative intersection over union objective between rendered and ground truth silhouettes [4, 7, 5]; (ii) *foreground RGB texture loss:* a perceptual metric [4, 7, 14] between rendered and input RGB images; (iii) *mesh smoothness:* a Laplacian constraint [4, 7] to encourage smooth mesh reconstruction; (iv) *keypoint re-projection loss:* as discussed in Sec.3.1 in the paper; and (v) *the ARAP constraint:* described in Sec.3.1 in the paper. The weight for each objective is set to 3.0, 3.0, 0.0008, 5.0 and 10.0.

7.2 Objectives for Online Adaptation

We summarize the objectives used in the online adaptation process (Fig. 11(b)) in the following. Since it is feasible to predict a segmentation mask via a pretrained segmentation model, we make use of the predicted foreground mask and compute the (i), (ii), and (iii) losses (mentioned above) similarly to the image-based training. We also adopt the the ARAP constraint described in Sec.3.1 in the paper, and the three invariance constraints as discussed in Sec.3.2 for online adaptation. The weight for each objective is set to 0.1, 0.5, 0.0006, 2.0 and 2.0 (texture invariance), 1.0 (part correspondence), 1.0 (base shape invariance).

7.3 Training Details

We implement the proposed method in PyTorch [10] and use the Adam optimizer [6] with a learning rate of 0.0001 for both the image reconstruction model training and online adaptation. The weight of each objective for the image reconstruction model as well as the online adaptation process is discussed in Sec. 7.1 and Sec. 7.2, respectively.

7.4 Training on Zebra Images and Videos

We adopt a different scheme to train a single-view reconstruction model on zebras: (i) since natural zebra images labeled with keypoints are not publicly available, we adopt a synthetic dataset [11], (ii) zebras have more complex shapes with large concavities. Therefore, it is not suitable to learn the shape by deforming from a sphere primitive. Instead, we utilize a readily available zebra mesh as a template and learn motion deformation on top of it. We first train an image reconstruction model using the synthetic dataset provided by [11]. Similarly as [11], we utilize the silhouettes, keypoints, texture maps as well as partially available UV texture flow as supervision. For shape reconstruction, instead of the utilizing the SMAL parametric model [15], we use the proposed shape module, i.e. combination of base shapes and motion deformation. Due to the limited motion of zebras, we only use one base shape, which is a readily available zebra mesh with 3889 vertices and 7774 faces. For camera pose prediction, we use the perspective camera pose discussed in Sec.3 in the submission as well as in [4]. Due to the limited capacity of a single UV texture map, we also model the texture map by cutting the UV texture map into four pieces and stitch them together similarly as in [11]. We note

that this "cutting and stitching" operation does not influence the mapping and aggregation of the part UV maps discussed in Sec.3.2 in the submission.

8 Failure Cases

Our work is the first to explore the challenging task of reconstructing 3D meshes of deformable object instances from videos in the wild. Impressive as the performance is, this challenging task is far from being fully solved. We discuss failure cases and limitations of the proposed method in the following. To begin with, we focus on genus-0 objects such as birds and zebras in this work. Thus our model suffers when it is generalized to objects with large concave holes such as chairs, humans etc. Second, our work struggles to reconstruct meshes from videos with large motion and lighting changes as well as occlusion, (see Fig. 12). This is mainly due to the failure in correctly propagating parts by the self-supervised UVC model [8], which is out of scope of this work. We leave all these failure cases and limitations to future works.

Figure 5: More qualitative reconstruction results on CUB birds [13].

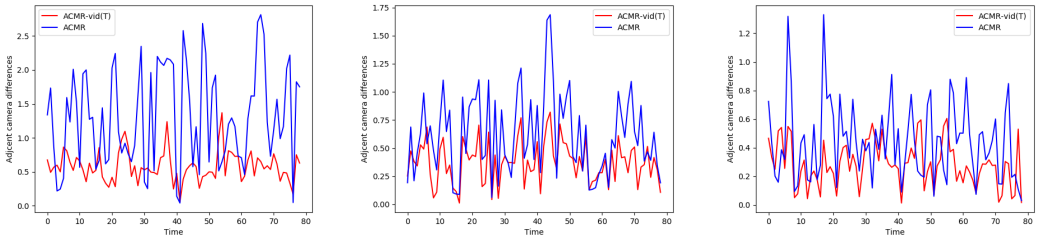


Figure 6: Camera stability visualization. We visualize differences between adjacent camera pose predictions. The blue and red lines represent the model trained only with images and the test-time tuned model, respectively.

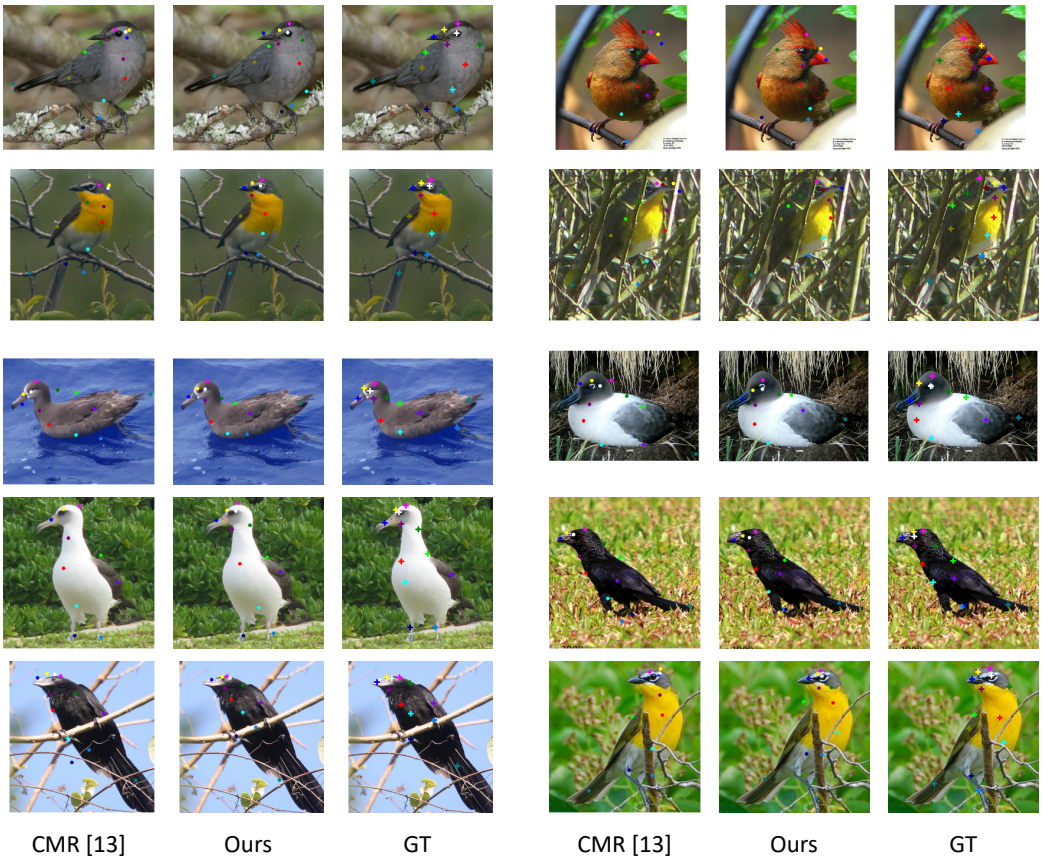


Figure 7: Visualization of re-projected keypoints of the single-view image reconstruction model.

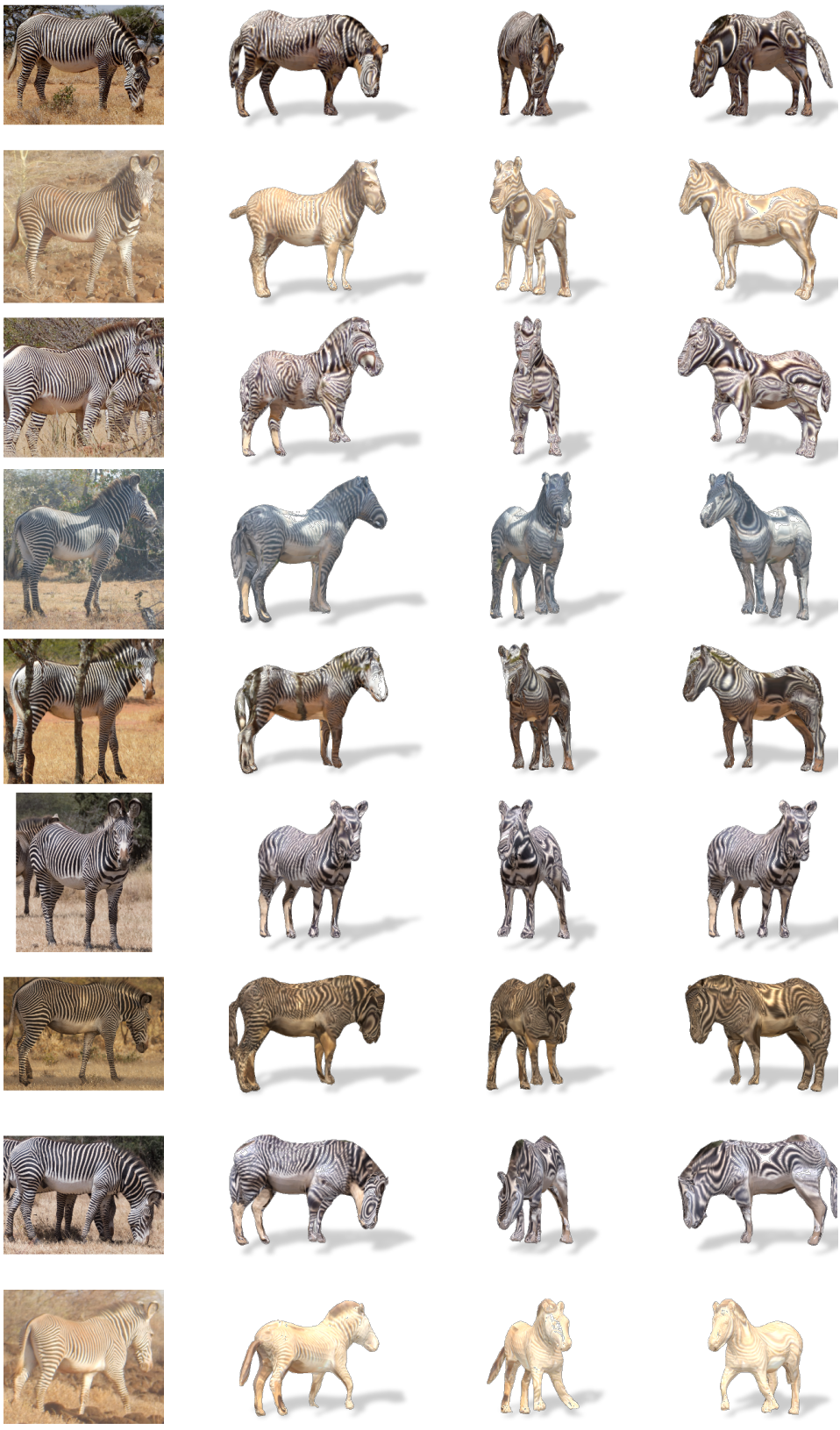
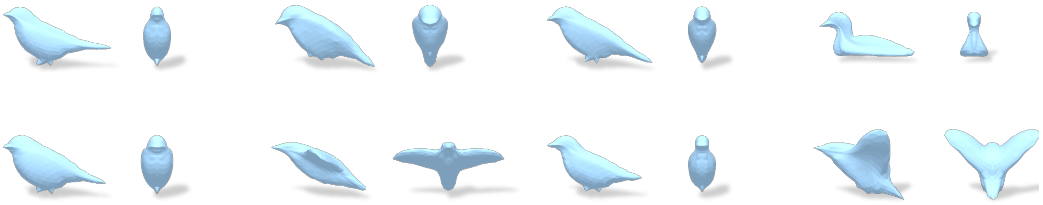
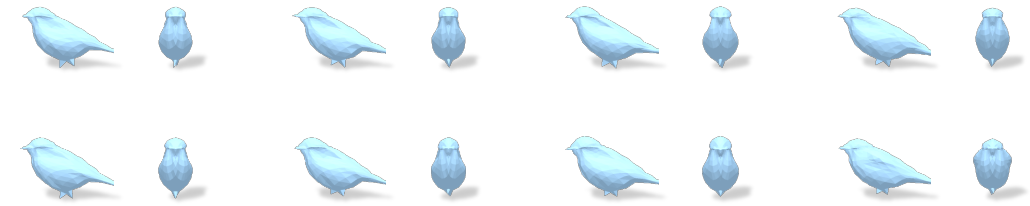


Figure 8: Visualization of reconstructed zebras.



(a) Applying KMeans to reconstructed meshes by CMR [13]



(b) Applying PCA to feature space of CMR [13]

Figure 10: Bases visualization.

(a) Image reconstruction model

(b) Illustration of the sliding window scheme

Figure 11: Frameworks. For the purposes of illustration only, we show the test-time tuning procedure in (b) with a sliding widow size of 3 and sliding stride of 2. In all our experiments, we use a sliding window size of 50 and stride of 10. The gray dashed box shows the previous sliding window while the orange box shows the current sliding window.

Figure 12: Failure cases. Our model fails when there is occlusion (e.g., $t = 50$, t stands for frame number) or large lighting changes (e.g., $t = 60$).

References

[1] K. He, X. Zhang, S. Ren, and J. Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 2

[2] W.-C. Hung, V. Jampani, S. Liu, P. Molchanov, M.-H. Yang, and J. Kautz. Scops: Self-supervised co-part segmentation. In *CVPR*, 2019. 1

[3] S. Ioffe and C. Szegedy. Batch normalization: Accelerating deep network training by reducing internal covariate shift. *arXiv preprint arXiv:1502.03167*, 2015. 2

[4] A. Kanazawa, S. Tulsiani, A. A. Efros, and J. Malik. Learning category-specific mesh reconstruction from image collections. In *ECCV*, 2018. 1, 2, 3, 4, 5

[5] H. Kato and T. Harada. Learning view priors for single-view 3d reconstruction. In *CVPR*, 2019. 5

[6] D. P. Kingma and J. Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 5

[7] X. Li, S. Liu, K. Kim, S. De Mello, V. Jampani, M.-H. Yang, and J. Kautz. Self-supervised single-view 3d reconstruction via semantic consistency. *arXiv preprint arXiv:2003.06473*, 2020. 1, 5

[8] X. Li, S. Liu, S. D. Mello, X. Wang, J. Kautz, and M.-H. Yang. Joint-task self-supervised learning for temporal correspondence. In *NeurIPS*, 2019. 6

[9] S. Liu, T. Li, W. Chen, and H. Li. Soft rasterizer: A differentiable renderer for image-based 3d reasoning. In *ICCV*, 2019. 5

[10] A. Paszke, S. Gross, F. Massa, A. Lerer, J. Bradbury, G. Chanan, T. Killeen, Z. Lin, N. Gimelshein, L. Antiga, A. Desmaison, A. Kopf, E. Yang, Z. DeVito, M. Raison, A. Tejani, S. Chilamkurthy, B. Steiner, L. Fang, J. Bai, and S. Chintala. Pytorch: An imperative style, high-performance deep learning library. In H. Wallach, H. Larochelle, A. Beygelzimer, F. d'Alché-Buc, E. Fox, and R. Garnett, editors, *NeurIPS*. 2019. 2, 5

[11] T. B.-W. M. J. B. Silvia Zuffi, Angjoo Kanazawa. Three-d safari: Learning to estimate zebra pose, shape, and texture from images "in the wild". In *ICCV*, 2019. 4, 5

[12] K. Wada. labelme: Image Polygonal Annotation with Python. https://github.com/wkentaro/labelme, 2016. 2

[13] C. Wah, S. Branson, P. Welinder, P. Perona, and S. Belongie. The caltech-ucsd birds-200-2011 dataset, 2011. 2, 4, 7

[14] R. Zhang, P. Isola, A. A. Efros, E. Shechtman, and O. Wang. The unreasonable effectiveness of deep features as a perceptual metric. In *CVPR*, 2018. 5

[15] S. Zuffi, A. Kanazawa, D. Jacobs, and M. J. Black. 3D menagerie: Modeling the 3D shape and pose of animals. In *CVPR*, 2017. 5