Supplementary Material: Articulation-aware Canonical Surface Mapping

Nilesh Kulkarni[1] Abhinav Gupta[2,3] David F. Fouhey[1] Shubham Tulsiani[3]
[1]University of Michigan [2]Carnegie Mellon University [3]Facebook AI Research
{nileshk, fouhey}@umich.edu abhinavg@cs.cmu.edu shubhtuls@fb.com
https://nileshkulkarni.github.io/acsm/

A. Constructing ϕ and δ

Parameterizing surface of a mesh. The surface S of a mesh is 2D manifold in 3D space hence we can construct a mapping $\phi : [0,1)^2 \to S$. We deal with triangular meshes as they are the most general form of mesh representation. Given the mapping from 2D square to a spherical mesh and another from the sphere to our template shape, our mapping from 2D manifold to the template shape is their composition. Constructing a mapping between 2D square and sphere can be understood to one analogous to latitudes and longitudes on the globe.
All our template shapes are genus-0 (isomorphic to a sphere – without holes). They have been pre-processed to have 642 vertices and 1280 faces using Meshlab's [1] quadratic vertex decimation [3]. Constructing a mapping between a sphere and a template shape corresponds to finding a mapping between faces of the spherical mesh and the faces of the template shape. To find such a mapping we need to deform the sphere to ensure that the corresponding faces have similar areas. We to do this by minimizing the squared difference of logarithm of triangle areas as the objective using Adam [7] optimizer. This optimization is an offline process and is part of preprocessing for the given category.

Parametrizing Articulation. Every template shape has a defined part hierarchy that assigns every node a parent except for the *root* node. A rigid transform is represented as translation t and rotation R. Every node, k has a rigid transform associated with. This rigid transform is applied in the frame of the part. Consider $\mathcal{T}'_k$ to be the local rigid transform at the node k represented using R and t. We define the global rigid transform at node k as $\mathcal{T}_k$.

$$\mathcal{T}_k = \mathcal{T}_l \circ \mathcal{T}'_k; \tag{1}$$
where node l is the parent of node k

Barycentric Interpolation. We use barycentric interpolation to compute the point on the surface of of the mesh for every $u \in [0,1)^2$. Given a u we map the point to the surface of the mesh of sphere, and then find the face it belongs to. We use the vertices to this face to compute it's barycentric coordinates. We then use these computed barycentric coordinates and the vertices on the corresponding face of the articulated mesh to compute the 3D location of the point.

B. Implementation Details

Our implementation uses PyTorch [10]

B.1. Network Details

We use a Resnet18 [5] encoder extracted with features after 4 blocks and 5 layer decoder. Our encoder is initialized with pretrained ImageNet [2] features. The encoder-decoder takes input an image and then outputs a 3D unit vector per pixel. We convert this unit-vector to a 2D coordinate $u \in [0,1)^2$ which is used to parameterize our 2D square.

B.2. Optimization

Parameterizing Part Transforms. We parameterize part transforms as an axis, angle representation. Every part's axis serves as a bias in the network that is learned and is same across the whole category. We predict the angle using a deep network for every part k.
Parameterizing Camera Pose. We parameterize the camera as orthographic similar to [6] where we predict R as a unit quaternion $q \in \mathcal{R}^4$, $s \in (0,\infty), t \in \mathcal{R}^2$. Also, similar to [8] we predictor 8 hypothesis for camera pose and part transforms to ease the leaning process.
Training for Articulation. We first train our network to only learn camera pose predictions (along with the CSM predictor) for 10000 iterations. We then allow the model to articulate for the rest of the training iteration. We train with Adam [7] as the optimizer using a learning rate of 10^{-4}.

B.3. Losses

Mask Consistency. We want to ensure that the rendered mask M_{rendered} of the mesh under camera π lies inside the ground truth mask of the object. We compute it by computing the euclidean distance field D_{edf} for the ground truth mask summed over all the pixels in the rendered mask.

$$L_{\text{mask-consistency}} = \sum_p M_{\text{rendered}}[p] D_{\text{edf}}[p] \tag{2}$$

Mask Coverage. Enforcing consistency is not enough as it only forces the object to lie inside the mask. We also want to ensure that all the object pixels in the foreground should be close to re-projection of some mesh vertex. This loss competes against with the consistency loss objective.

$$L_{\text{mask-coverage}} = \sum_{p \in I_f} \min_{v \in V} \|v - p\| \tag{3}$$

Mask Loss. Our mask objective is a sum of these two competing losses as follows.

$$L_{\text{mask}} = L_{\text{mask-consistency}} + L_{\text{mask-coverage}} \tag{4}$$

Visibility Loss. We render depth map of the articulated template shape under camera π as D_π and for every pixel we have the z_p as z-coordinate of the pixel in the camera frame corresponding to the point $\delta(\phi(C[p]))$ on the surface of the 3D shape.

$$L_{\text{vis}} = \sum_{p \in I_f} \max(0, z_p - D_\pi[\bar{\mathbf{p}}]); \quad \bar{\mathbf{p}} = \pi(\delta(\phi(C[p]))) \tag{5}$$

Regularization Losses. Additionally, we also add the regularization to translation prediction for part transforms, along with an entropy penalty to encourage diversity of the multi-pose predictor.

C. Evaluation

C.1. Transfer Keypoints using CSM

We use the predicted CSM map to transfer keypoints between source and target images by using their corresponding canonical surface mappings C_s and C_t. For every source keypoint at pixel p^k we map the keypoint to the point the non-articulated template shape, and then search for a pixel on the target image that maps closest to this point.

$$T^k_{s \to t} = \operatorname*{argmin}_p (\|\phi(C_t[p]) - \phi(C_s[p^k])\|) \tag{6}$$

C.2. Importance of Ground Truth Masks

We study the impact of having foreground segmentation from Mask-RCNN versus using human annotation for *training* A-CSM model. In these experiments we use pre-trained Mask-RCNN [4] on 80 COCO [9] categories. We use the CUB-2011 [11] dataset with segmentation from Mask-RCNN [4] and the ground truth annotations to compared the performance on tasks of PCK-Transfer and Key-Point (KP) Projection. We report results in Table 1 and observe that though our method has a superior performance with precise ground truth masks, the performance drop with using automatically generated inaccurate segmentation from Mask-RCNN is not significant and our methods remains competitive.

Table 1: We evaluate the performance of our model trained with ground truth (GT) and Mask-RCNN segmentation on the task of Keypoint (KP) Transfer and Keypoint (KP) Projection as described in Table 1 and 2 of the main text. We report the performance on the CUBS-2011 [11] dataset, and observe that there is not a significant performance gap when using segmentations from Mask-RCNN.

Mask Source	PCK Transfer	KP Projection
GT (Humans)	42.6	46.8
Mask-RCNN [4]	38.5	45.5

Evaluation using GT Masks. Our results in the main paper for PCK Transfer use predicted masks to transfer keypoints between two given images. We evaluated the performance of transfer by using ground truth masks on the birds dataset and observe that the performance changes on an average of 0.1 points. This implies that there is no significant disadvantage in using the predicted masks for evaluation.

D. Qualitative Sampled Results on 11 Categories

We randomly sample results for all the categories shown in the paper show their visualizations in Figure 1, Figure 2, Figure 3, Figure 4. These figure show articulations of template shape for every input image along side the CSM prediction for the foreground pixels. We observe consistent CSM predictions for various functional regions of the object. For instance, we can see the head of all the quadrupeds is greenish in color which accurately represents its mapping to green region on the template shape shown in the rightmost column. We show results over 11 categories with a wide variety of articulations.

We also show results of our method on a few videos downloaded from the internet in the video file submitted along with this supplementary. Our method is applied on a per frame basis without any temporal smoothing. We show a few screenshots from the video in Figure 5

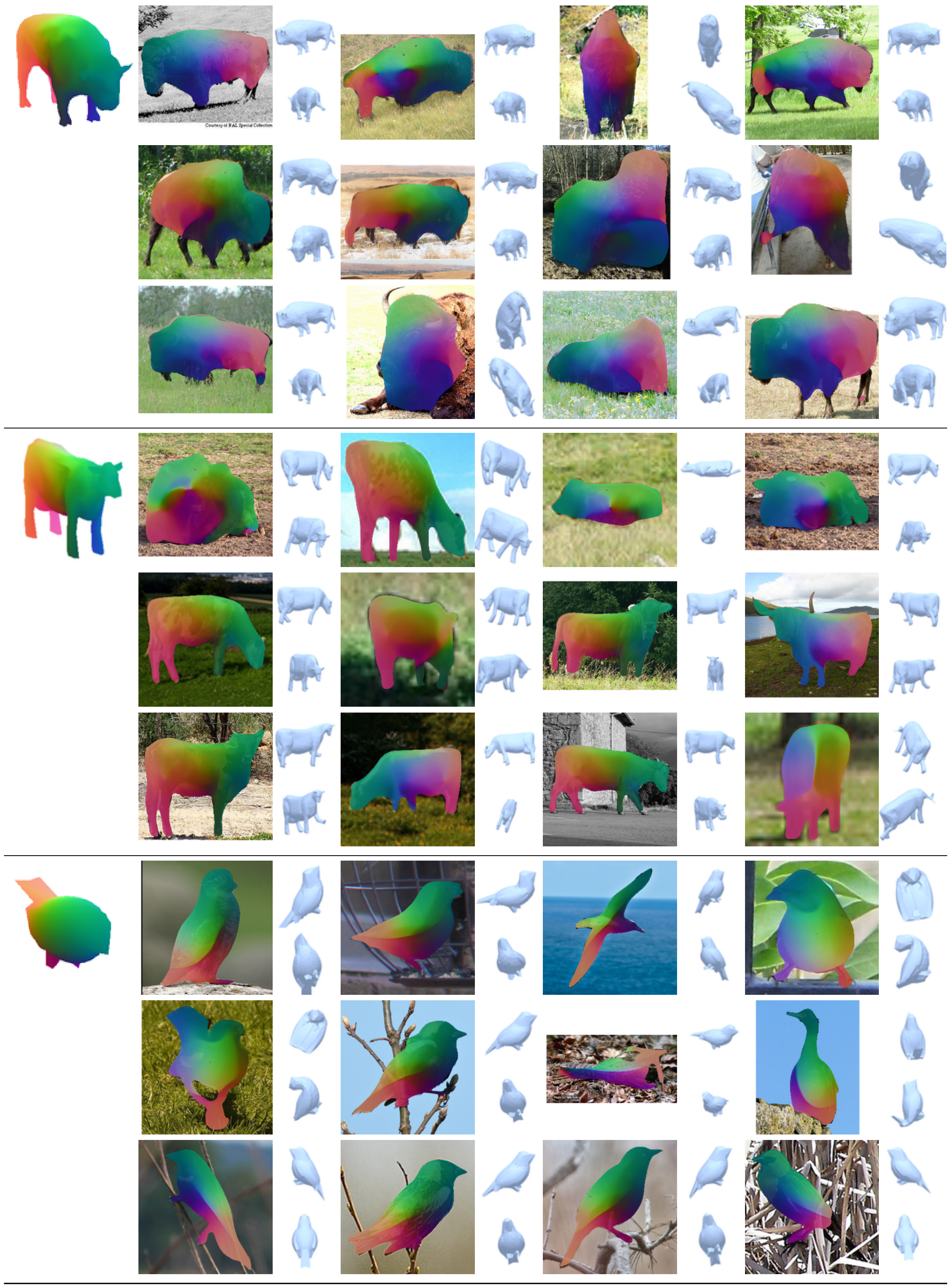

Figure 1: *Randomly sampled* results on bisons, cows, and birds

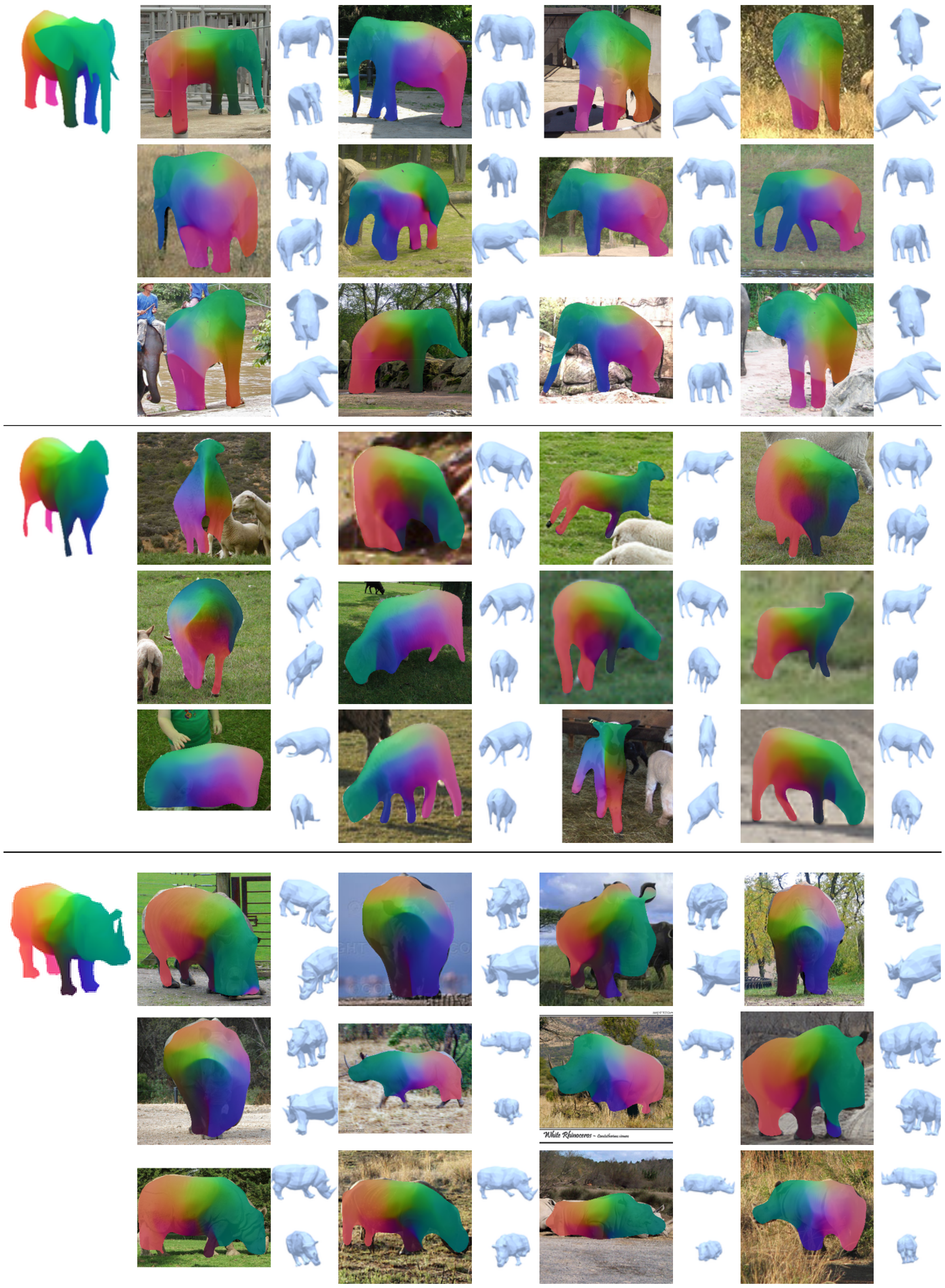

Figure 2: *Randomly sampled* results on elephants, sheeps, and rhinos

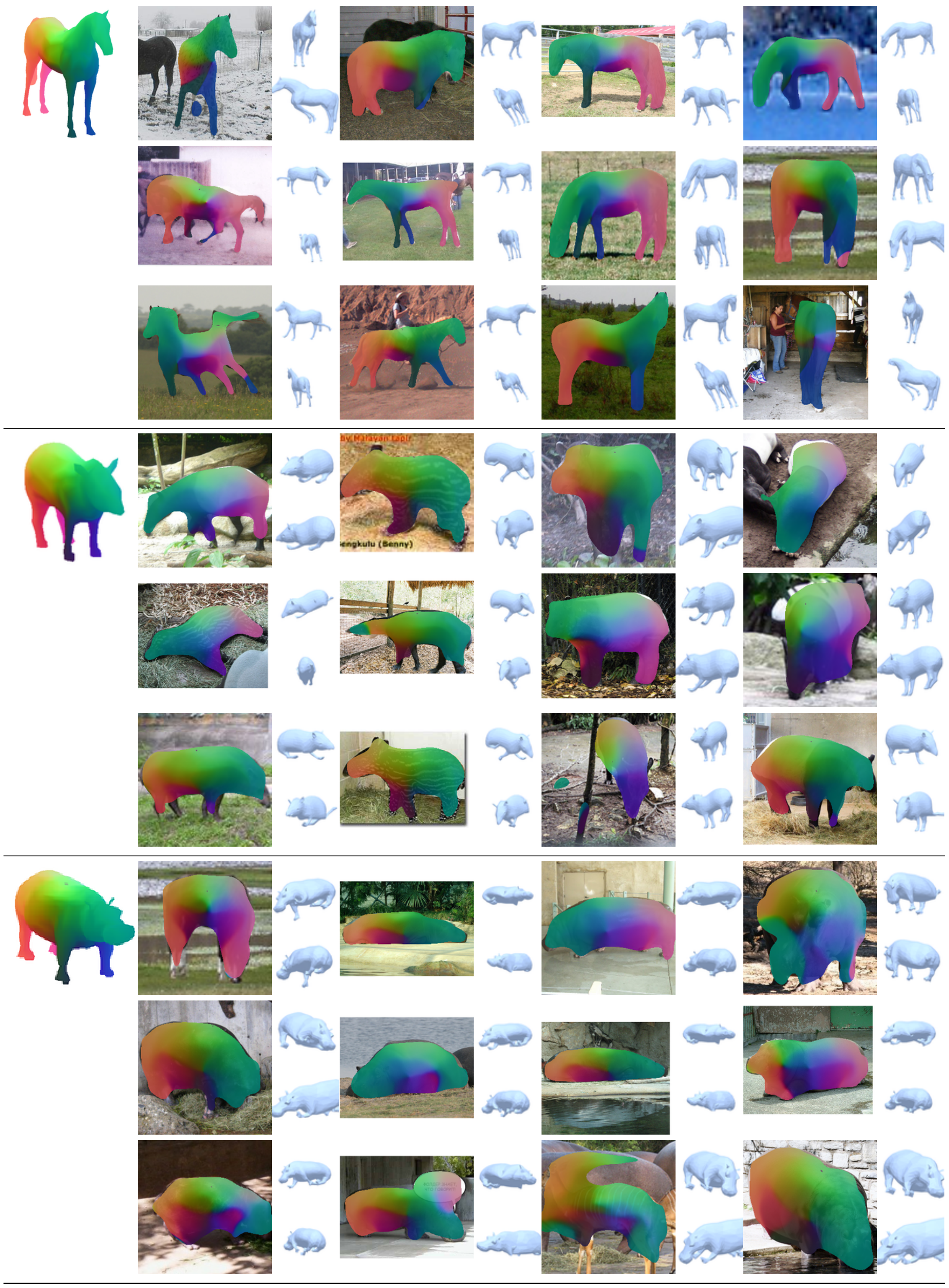

Figure 3: *Randomly sampled* results on horses, tapirs, and hippos

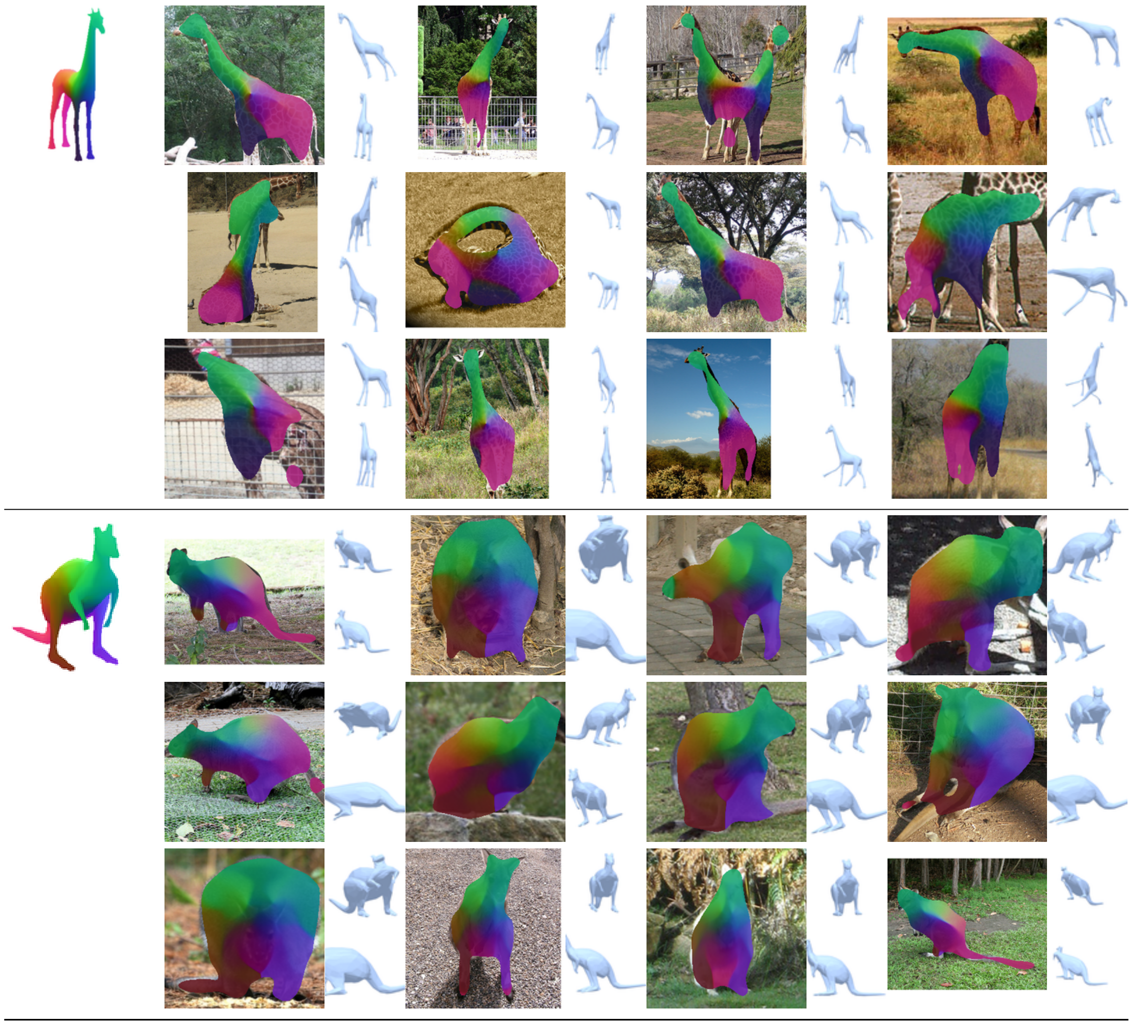

Figure 4: *Randomly sampled* results on giraffes and kangaroos

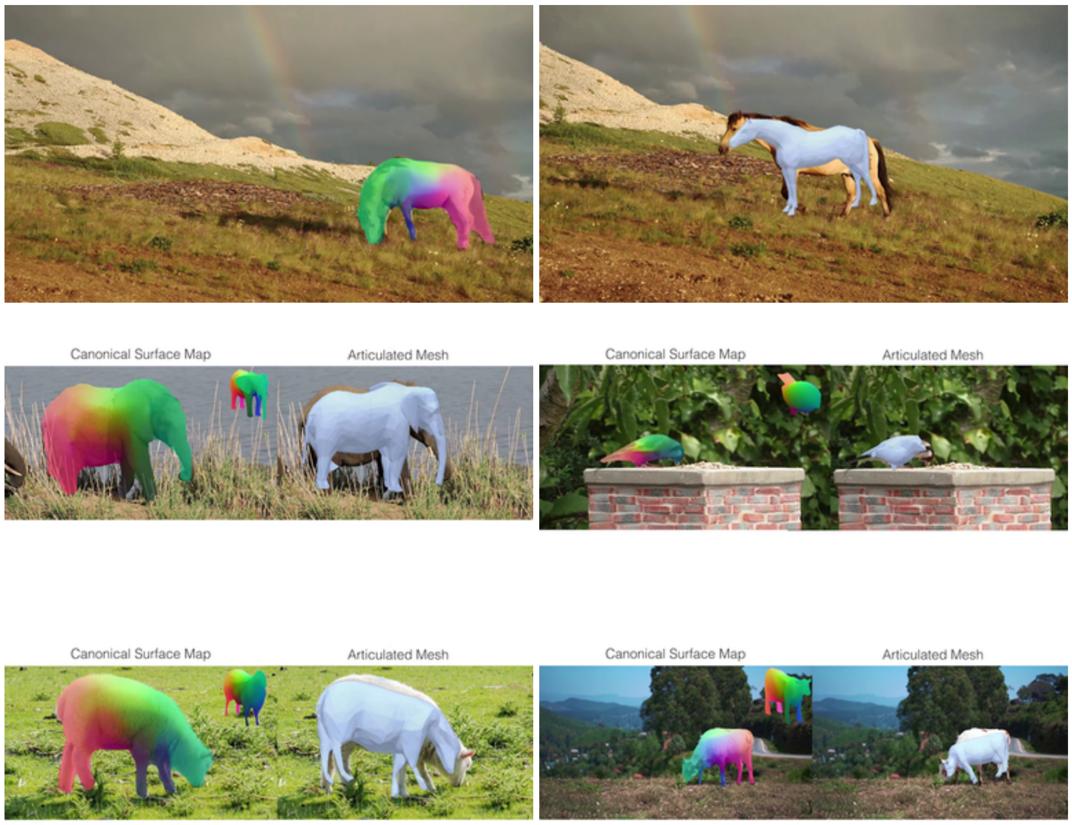

Figure 5: Screenshots of results from our method on the videos from the Internet

References

[1] Paolo Cignoni, Marco Callieri, Massimiliano Corsini, Matteo Dellepiane, Fabio Ganovelli, and Guido Ranzuglia. MeshLab: an Open-Source Mesh Processing Tool. In Vittorio Scarano, Rosario De Chiara, and Ugo Erra, editors, *Eurographics Italian Chapter Conference*. The Eurographics Association, 2008. 1

[2] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *CVPR*, 2009. 1

[3] Michael Garland and Paul S Heckbert. Surface simplification using quadric error metrics. In *SIGGRAPH*. ACM Press/Addison-Wesley Publishing Co., 1997. 1

[4] Kaiming He, Georgia Gkioxari, Piotr Dollár, and Ross Girshick. Mask r-cnn. In *ICCV*, 2017. 2

[5] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *CVPR*, 2016. 1

[6] Angjoo Kanazawa, Shubham Tulsiani, Alexei A. Efros, and Jitendra Malik. Learning category-specific mesh reconstruction from image collections. In *ECCV*, 2018. 1

[7] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 1

[8] Nilesh Kulkarni, Abhinav Gupta, and Shubham Tulsiani. Canonical surface mapping via geometric cycle consistency. In *ICCV*, 2019. 1

[9] 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 *ECCV*, 2014. 2

[10] Adam Paszke, Sam Gross, Soumith Chintala, Gregory Chanan, Edward Yang, Zachary DeVito, Zeming Lin, Alban Desmaison, Luca Antiga, and Adam Lerer. Automatic differentiation in PyTorch. In *NIPS Autodiff Workshop*, 2017. 1

[11] Catherine Wah, Steve Branson, Peter Welinder, Pietro Perona, and Serge Belongie. The caltech-ucsd birds-200-2011 dataset. 2011. 2