Appendix for Neural Textured Deformable Meshes for Robust Analysis-by-Synthesis

A. Implementation Details

A.1. Differentiable Transformation Function

The differentiable transformation function takes inputs of an UV fragment $\mathcal{U}$ (which is obtained via interpolate the UV values of vertices using the barycentric weight from rasterization) and a feature F on the image coordinate, and output the surface features $\hat{F}$ with a visibility mask $\mathcal{V}$. The transformation compute gradient to both $\mathcal{U}$ and F.

Forward. As Figure 1 shows, for each four adjacent pixels $\mathcal{P} = (p_1, p_2, p_3, p_4)$ pairs on the $\mathcal{P} \in \mathcal{U}$, we first check if all of them are on the object. For each on-object pixel pair, we find the corresponded quadrilateral on the surface $\mathcal{S}$. Then we compute the barycentric coordinates inside the quadrilateral, which gives four weights w_1, w_2, w_3, w_4 on each surface pixel. Then we weighted sum the four feature vector on p_1, p_2, p_3, p_4 to compute the final value on the output surface feature:

$$F_{(u,v)} = w_1 \cdot F_{p_1} + w_2 \cdot F_{p_2} + w_3 \cdot F_{p_3} + w_4 \cdot F_{p_4} \quad (1)$$

In our implementation, for those cases that a pixel on surface is covered by multiple quadrilateral, we take a average of the value via store the total weight per surface pixels $\hat{w}_{(u,v)} = \sum_{\mathcal{P} \in \mathcal{U}} \sum_k^4 w_k$. Then the visibility mask is computed as the area that $\hat{w}_{(u,v)} > 0$. Also, in order to get rid of the quadrilateral cross the left to right or top to bottom boundary, we skip the quadrilateral larger than a threshold (set to be 0.2 of the surface size).

Backward. In the backward process, assume the final loss is computed as L and the up stream loss to the layer is $\frac{\partial L}{\partial F_{(u,v)}}$ on each surface pixel. The overall loss to the input feature is computed as:

$$\frac{\partial L}{\partial F_p} = \sum_{\mathcal{P} \in \mathcal{U}} w_k \cdot \frac{\partial L}{\partial F_{(u,v)}} \quad (2)$$

where $k \in \{1, 2, 3, 4\}$ is the index of p_k in the $\mathcal{P}$. We compute the index lookup table during the forward process and save it for usage in backward. On the other hand, the gradient to $u \in \mathcal{U}$ is computed via:

$$\frac{\partial L}{\partial u_p} = \frac{\partial L}{\partial F_{(u,v)}} \cdot \sum_{\mathcal{P} \in \mathcal{U}} \frac{\partial F_{(u,v)}}{\partial w_k} \cdot \frac{\partial w_k}{\partial u_{p_k}} \quad (3)$$

where

$$\frac{\partial w_k}{\partial u_{p_k}} = \frac{\partial w_k}{\partial u_{p_k}} \cdot \frac{\sum_{j \neq k}^4 w_j}{\sum_j^4 w_j} - \sum_{i \neq k}^4 \frac{\partial w_i}{\partial u_{p_i}} \frac{w_i}{\sum_j^4 w_j} \quad (4)$$

and

$$\frac{\partial F_{(u,v)}}{\partial w_k} = \sum_{c \in C} F^c_{(u,v)}, \quad (5)$$

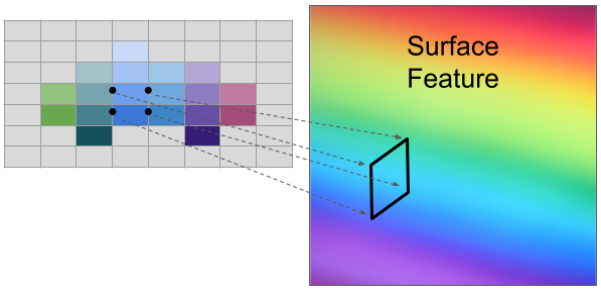

Figure 1. Each four adjacent pixel in the UV fragment is interpolated as a quadrilateral with barycentric coordinates.

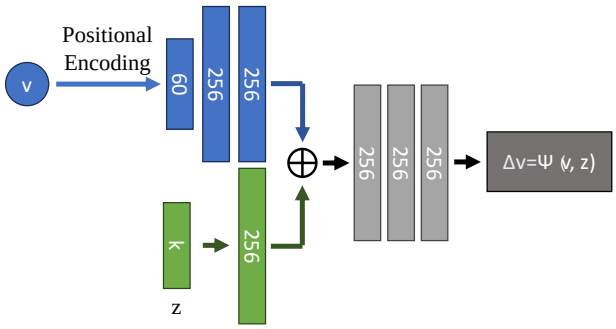

Figure 2. Network structure of mesh deformation MLP.

C is dimension of the features. Note, in practice, we don't look up each pixel pairs on $\mathcal{U}$ to compute the sum, instead, we store the lookup and weight w_k for each pixel (maximum 10 quadrilaterals per pixel) in the surface coordinate to reduce computation costs.

We implement the function using CUDA and packed as a PyTorch auto-gradient function, which are potentially useful for future projects, *e.g.* texture extraction.

A.2. Deformable Mesh

Figure 2, show the network structure of the MLP network which controls the deformation of the meshes.

A.3. Contrastive Loss

Figure 3 shows the implementation details of our training loss. Specifically, in each training step, we sample

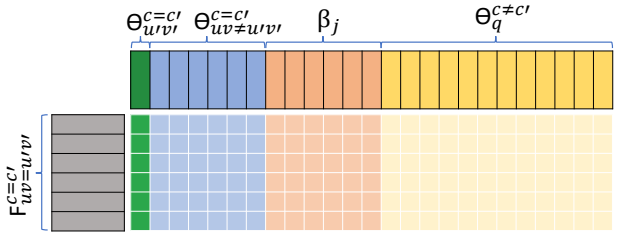

Figure 3. Training Loss.

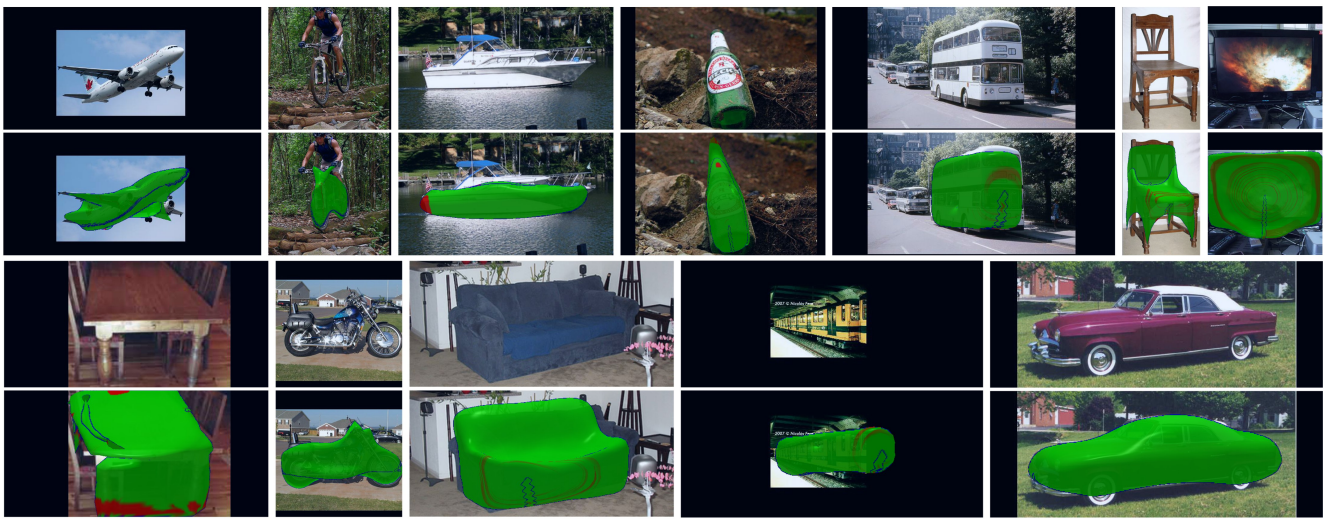

Figure 4. Qualitative results of all 12 categories on PASCAL3D+ dataset [6].

1000 uv pairs on the transformed image features F. For each sampled feature from the training data $uv = u'v'$, we sample the corresponding feature from the neural texture $\theta_{uv=u'v'}$, and concatenate it with 3200 negative samples uv from 20 pixels away from $u'v'$ (specifically 2000 features from the background β_j and 200 features from every other category $\theta_q^{c \neq c'}$). Subsequently, we compute the cosine similarity between the image features and other collected features and apply the cross-entropy loss. Such loss will make features from the same location on the object similar while making it different from other parts, backgrounds, and other classes. When applying the cross-entropy loss, we use $W_{ML} = 1, W_{Con} = 1, W_{Class} = 1$ and $W_{BG} = 0.1$.

A.4. Multi-Task Mask RCNN

To compare DMNT with traditional multi-head network architectures, we extend a Mask R-CNN model [1] with a pose estimation head. More specifically, the MT Mask R-CNN model consists of three heads, a classification head, a pose estimation head, and a mask segmentation head. We adopt the classification head and mask segmentation head from the official PyTorch [3] implementation. We follow the loss functions defined in [1]. To provide the ground-truth masks for Mask R-CNN, we compute the projection of the known CAD models given the annotated principal points and 3D poses. In terms of the pose estimation head, we follow [2,4] that formulate the pose estimation as a classification task. We follow the implementations in [7] to reproduce the results. Formally, the MT Mask R-CNN is end-to-end supervised by a multi-task loss given by

$$\mathcal{L} = \mathcal{L}_{\text{cls}} + \mathcal{L}_{\text{box}} + \mathcal{L}_{\text{mask}} + \mathcal{L}_{\text{pose}} \quad (6)$$

B. Additional Qualitative Results

In this section, we provide additional qualitative results to demonstrate the capabilities of DMNT under various settings.

PASCAL3D+. We visualize the predictions of DMNT on PASCAL3D+ dataset [6] in Figure 4. By learning a 3D deformable neural representation, DMNT solve multiple tasks from a holistic perspective. As we can see from Figure 4, DMNT predicts accurate object poses, good amodal segmentations, as well as class labels.

Occluded PASCAL3D+. We also visualize the results on Occluded PASCAL3D+ dataset [5] in Figure 5. As we can see, DMNT is very robust to partial occlusion and can accurately capture the 3D pose and object boundaries from the visible part of the object.

Failed examples. To investigate the limitations of our model, we also looked into some failed cases from PASCAL3D+ [6] and Occluded PASCAL3D+ dataset [5], which are shown in Figure 6. In Figure 6(a), DMNT failed to predict accurate amodal segmentation because the wheels of the car in the background resemble the wheels of the motorbike. In Figure 6(b), the novel parts of the aeroplane, i.e., wings, are largely dominated by the occluders, and DMNT failed to predict good object pose. DMNT couldn't predict good object boundaries in Figure 6(c) since this boat has a different shape from the known CAD models in PASCAL3D+ and the body of the boat is heavily occluded.

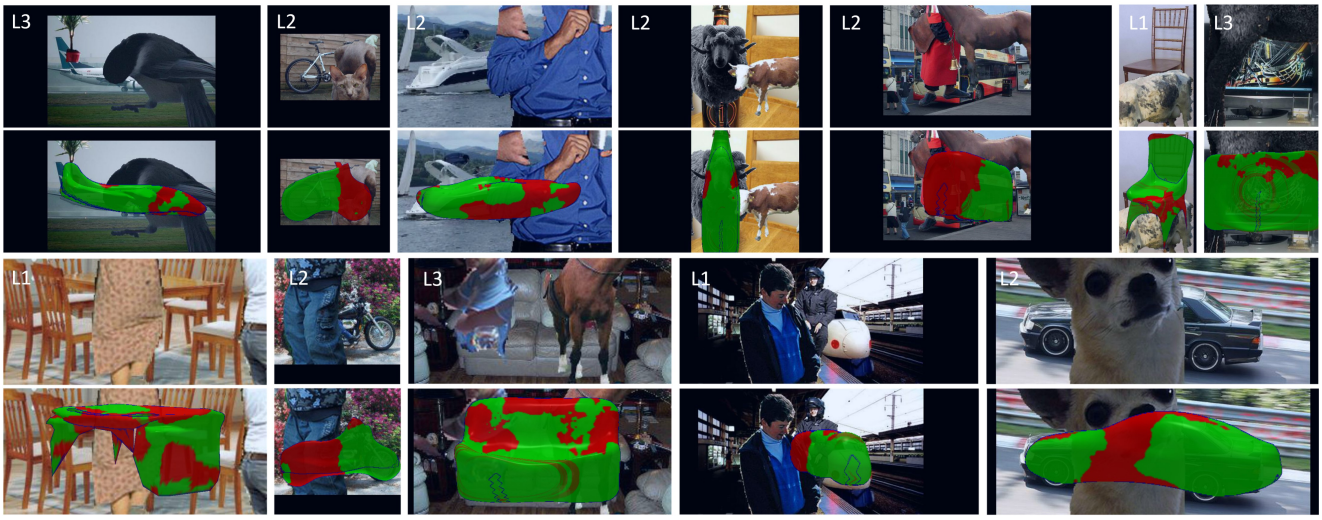

Figure 5. Qualitative results of all 12 categories on Occluded PASCAL3D+ dataset [5].

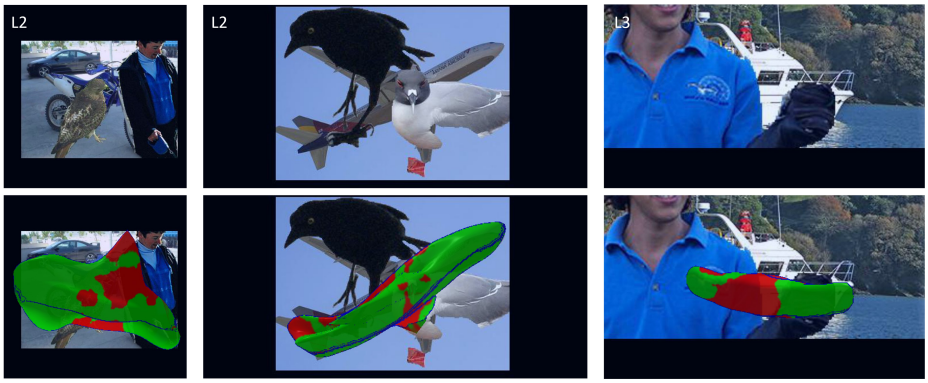

Figure 6. Some failed examples from PASCAL3D+ dataset [6] and Occluded PASCAL3D+ dataset [5].

References

[1] Kaiming He, Georgia Gkioxari, Piotr Dollar, and Ross Girshick. Mask r-cnn. In *Proceedings of the IEEE International Conference on Computer Vision (ICCV)*, Oct 2017. 2

[2] Arsalan Mousavian, Dragomir Anguelov, John Flynn, and Jana Kosecka. 3d bounding box estimation using deep learning and geometry. In *Proceedings of the IEEE conference on Computer Vision and Pattern Recognition*, pages 7074–7082, 2017. 2

[3] Adam Paszke, Sam Gross, Francisco Massa, Adam Lerer, James Bradbury, Gregory Chanan, Trevor Killeen, Zeming Lin, Natalia Gimelshein, Luca Antiga, et al. Pytorch: An imperative style, high-performance deep learning library. *Advances in neural information processing systems*, 32, 2019. 2

[4] Shubham Tulsiani and Jitendra Malik. Viewpoints and keypoints. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 1510–1519, 2015. 2

[5] Angtian Wang, Yihong Sun, Adam Kortylewski, and Alan L Yuille. Robust object detection under occlusion with context-aware compositionalnets. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 12645–12654, 2020. 2, 3

[6] Yu Xiang, Roozbeh Mottaghi, and Silvio Savarese. Beyond pascal: A benchmark for 3d object detection in the wild. In *IEEE winter conference on applications of computer vision*, pages 75–82. IEEE, 2014. 2, 3

[7] Xingyi Zhou, Arjun Karpur, Linjie Luo, and Qixing Huang. Starmap for category-agnostic keypoint and viewpoint estimation. In *Proceedings of the European Conference on Computer Vision (ECCV)*, pages 318–334, 2018. 2