Unsupervised Learning of 3D Object Categories from Videos in the Wild

Supplementary material

1. Additional implementation details

In this section, we provide more detailed information about the dense image descriptors Φ as well as the neural radiance field Ψ. Furthermore, we give more insights into the training process.

1.1. Dense image descriptors

This section describes in more detail the dense pixel-wise embeddings $\Phi(I_t)$ introduced in Section 3.3 in the main paper.

For a given source image I_t, the embedding field $\Phi(I_t)$ is composed of 3 different types of features: 1) learned $5 \cdot 32$-dimensional dense pixel-wise features output by a deep convolutional encoder network $\Phi_{\text{U-Net}}$, 2) raw image rgb colors $I_t \in \mathbb{R}^{3\times H\times W}$, and 3) the segmentation mask $m_t \in \mathbb{R}^{1\times H\times W}$.

Dense feature extractor $\Phi_{\text{U-Net}}$. The architecture of the U-Net inside $\Phi_{\text{U-Net}}$ is defined as follows (a detailed visualisation is present in Fig. 2). A source image $I^{\text{src}} \in \mathbb{R}^{3\times H\times W}$, masked by m^{src} (retrieved from Mask-RCNN), is fed into a ResNet-50 which returns spatial features from intermediate convolutional layers (*layer1, layer2, layer3, layer4, layer5*), and the final linear ResNet layer which outputs global features $\mathbf{z}_{\text{CNN}}$, i.e. non-spatial. Each feature layer including the global one is then passed through a 1x1 convolution to equalize the size of all feature channels to 32. The spatial features are further bilinearly upsampled to the spatial size of the source image and concatenated along the channel dimension to create a dense embedding field $\Phi_{\text{U-Net}}(I_t) \in \mathbb{R}^{5\cdot 32\times H\times W}$.

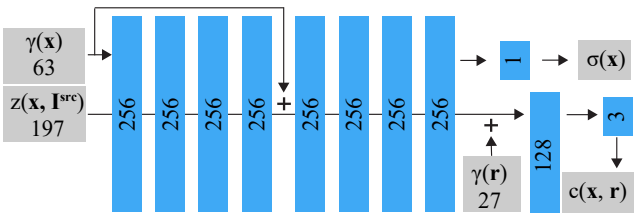

Figure 1: The neural radiance field Ψ is represented by an MLP. It takes as input the warp-conditioned embedding $\mathbf{z}(\mathbf{x})$, the harmonic positional embedding $\gamma(\mathbf{x})$ and to account for view point variations the harmonic directional embedding $\gamma(\mathbf{r})$. It returns the rgb and opacity values.

Neural radiance field Ψ. Our scene is represented by a neural radiance field Ψ similar to [1] with the only difference that we additionally condition the field with our warp-conditioned ray embedding, see Fig. 1.

1.2. Training details

We trained both the U-Net encoder $\Phi_{\text{U-Net}}$ and the neural radiance field Ψ with Adam optimizer. We set the batch size to 8 and the learning rate to 1e-4. Our method as well as all baselines were trained on an NVIDIA Tesla V100 for 7 days. For all raymarching baselines and our method, we shoot 1024 rays per iteration through random image pixels in Monte-Carlo fashion. For each ray we first uniformly sample 128 times along the ray in order to retrieve a coarse rendering (voxel or mlp based depending on the method used). In the second pass we sample each ray 128 times based on probabilistic importance sampling following [1].

For the mesh baseline we shoot rays for each pixel per iteration and use soft rasterization to predict the surface intersection. In addition to the losses used for the other baselines as well as our method, we additionally use a negative IoU loss L_{iou}, a Laplacian loss L_{lap} and smoothness loss L_{sm} according to [2] and weighted them with 1.0, 19.0, 1.0 respectively.

2. Additional qualitative results

Additional qualitative results are available presented in Fig. 4 and Fig. 3. Also, we provide more qualitative results on our project webpage: https://henzler.github.io/publication/unsupervised_videos/. The page contains comparison of our method to baselines by showing the scenes from the train-test or test subsets rendered from a viewpoint that rotates around the object of interest.

3. Test-time view ablation

Furthermore, we also provide a view ablation of our method at test time. Recall that we randomly sample between 1 and 7 source images during training. During test time we evaluated our method separately on 1, 3, 5 and 7 views as input. In the main paper we provide an average of those numbers. In Table 1 we give insight into how changing the number of source views affects performance. Not surprisingly, increasing the numbers of source views consistently improves all metrics.

		AMT								Freiburg Cars							
		Train-test				Test				Train-test				Test			
	Method	1	3	5	7	1	3	5	7	1	3	5	7	1	3	5	7
ℓ_1^{VGG}	Mesh	1.163	1.167	1.168	1.169	1.160	1.161	1.163	1.163	2.030	2.029	2.028	2.023	2.170	2.168	2.166	2.167
	Voxel	1.052	1.051	1.051	1.051	1.127	1.127	1.127	1.127	1.581	1.581	1.580	1.580	2.050	2.050	2.046	2.046
	Voxel+MLP	1.041	1.040	1.040	1.040	1.131	1.130	1.130	1.130	1.469	1.468	1.468	1.468	2.067	2.063	2.063	2.064
	MLP	**0.900**	0.899	0.899	0.899	1.130	1.130	1.130	1.131	**1.391**	1.389	1.389	1.389	2.027	2.025	2.024	2.025
	Ours	0.905	**0.846**	**0.837**	**0.832**	**1.007**	**0.921**	**0.896**	**0.883**	1.450	**1.381**	**1.372**	**1.359**	**1.945**	**1.897**	**1.874**	**1.863**
IoU	Mesh	0.599	0.599	0.599	0.598	0.598	0.598	0.598	0.598	0.601	0.604	0.605	0.606	0.556	0.556	0.556	0.556
	Voxel	0.776	0.777	0.777	0.777	0.660	0.660	0.660	0.661	0.891	0.892	0.892	0.893	0.517	0.511	0.509	0.510
	Voxel+MLP	0.775	0.776	0.777	0.776	0.652	0.654	0.654	0.654	0.878	0.878	0.878	0.878	0.540	0.541	0.542	0.541
	MLP	**0.871**	0.871	0.872	0.872	0.654	0.653	0.653	0.653	0.872	0.872	0.872	0.872	0.472	0.470	0.472	0.471
	Ours	0.866	**0.884**	**0.886**	**0.889**	**0.774**	**0.788**	**0.787**	**0.787**	**0.889**	**0.897**	**0.898**	**0.897**	**0.600**	**0.624**	**0.629**	**0.632**
ℓ_1^{Depth}	Mesh	5.138	5.119	5.128	5.130	5.100	5.101	5.090	5.086	1.202	1.185	1.178	1.177	**1.062**	**1.061**	**1.063**	**1.063**
	Voxel	2.150	2.141	2.140	2.141	3.069	3.064	3.067	3.065	0.591	0.590	0.585	0.583	2.133	2.181	2.207	2.200
	Voxel+MLP	1.958	1.942	1.942	1.941	2.881	2.868	2.861	2.864	**0.478**	0.479	0.479	0.479	1.972	1.979	1.968	1.968
	MLP	**1.389**	1.378	1.377	1.377	3.583	3.587	3.590	3.593	0.595	0.593	0.594	0.593	2.521	2.530	2.519	2.520
	Ours	1.593	**1.291**	**1.201**	**1.172**	**2.186**	**1.847**	**1.802**	**1.776**	0.535	**0.467**	**0.457**	**0.453**	1.606	1.595	1.589	1.603

Table 1: We complement the evaluation of the impact of the number of source views during test time for the metrics: ℓ_1^{VGG}, ℓ_1^{Depth}, **IoU**. We report results for 1, 3, 5 and 7 source images. The best result is **bolded** where lower is better for ℓ_1^{VGG}, ℓ_1^{Depth} and higher is better for **IoU**.

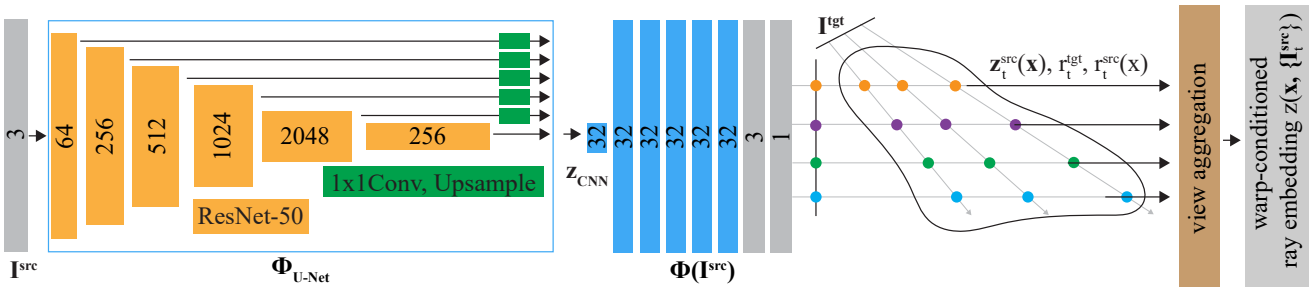

Figure 2: The input to the dense feature extractor Φ is a source image from a given view. It first makes use of a ResNet-50 ($\Phi_{\text{U-Net}}$) to retrieve the layer-wise features. Then, each layer is independently fed to a 1x1 convolution followed by bilinear upsmapling to the original input resolution. The resulting feature blocks are concatenated with the input image I^{src} and its corresponding object mask m^{src}. In case there are multiple source images available, this process is repeated for each of them. Once all per-view features are obtained the warp-conditioned ray embedding is retrieved after applying the view-aggregation.

References

[1] Ben Mildenhall, Pratul P. Srinivasan, Matthew Tancik, Jonathan T. Barron, Ravi Ramamoorthi, and Ren Ng. NeRF: Representing scenes as neural radiance fields for view synthesis. In *Proc. ECCV*, 2020. 1

[2] Nikhila Ravi, Jeremy Reizenstein, David Novotny, Taylor Gordon, Wan-Yen Lo, Justin Johnson, and Georgia Gkioxari. Accelerating 3d deep learning with pytorch3d. *arXiv:2007.08501*, 2020. 1

Figure 3: In reach row, a single source image (1st column) is processed by one of the evaluated methods (Mesh, Voxel, MLP+Voxel, MLP, **Ours** - columns 2 to 6) to generate a prescribed target view (last column). We show results on the test split.

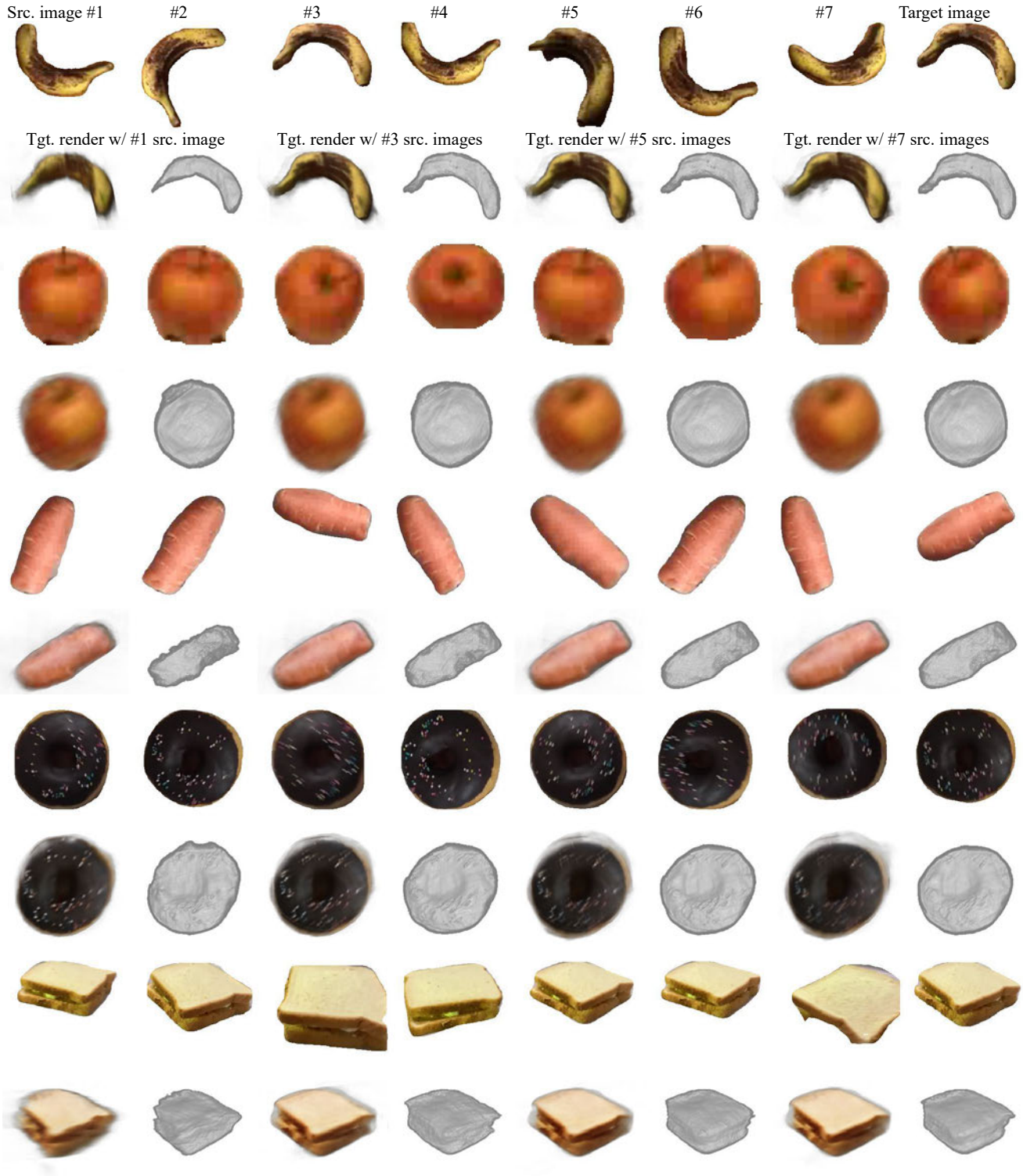

Figure 4: **Reconstruction with multiple source views.** The top row for each object shows all available source images (columns 1-7) for a given target image (top right). The bottom row contains results conditioned on 1, 3, 5 or 7 source images. In addition to the rendered new RGB views we also provide shaded surface renderings.