Neural Congealing: Aligning Images to a Joint Semantic Atlas Supplementary Material

Dolev Ofri-Amar[1] Michal Geyer[1] Yoni Kasten[2] Tali Dekel[1]
[1]Weizmann Institute of Science [2]NVIDIA Research

A. Implementation Details

A.1. General

We use images of size 256×256 (using border padding for non-square image), which is the resolution the STN takes as input. The rigid STN resizes the images to the resolution of the atlas, which is set to be $W_{\mathcal{A}} = H_{\mathcal{A}} = 128$. For feature extraction, we use `ViT-S/8` ($D = 384$) with stride 4, as in [1]. We extract the keys from the original images and bilinearly upsample them to the atlas resolution.

A.2. Spatial Transformer Architecture Details

We use the same architectures for the STNs as in [5]. Both architectures are based on the design of the ResNet-based discriminator from StyleGAN2 [2]. Tables 3 and 4 detail the layers of the rigid and non-rigid STN respectively, and 5 and 6 detail the building blocks.

Rigid STN. The rigid mapping network consists of a ResNet backbone with a fully-connected layer at the end, which outputs four logits (o_1, o_2, o_3, o_4), to which the following activations are applied to obtain the transformation parametrization:

$$\theta = \pi \cdot \tanh(o_1) \quad , \quad R = \begin{bmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{bmatrix} \tag{1}$$

$$s = \exp(o_2) \tag{2}$$

$$\mathbf{t} = \begin{bmatrix} o_3 \\ o_4 \end{bmatrix} \tag{3}$$

Non-Rigid STN. The non rigid mapping network consists of a ResNet backbone that outputs a 16×16 feature grid which is then fed to two small convolutional networks: the first outputs a 16×16 coarse flow field, and the second outputs weights which are used to perform $\times 8$ upsampling of the coarse flow field to the size of $128 \times 128 = H_{\mathcal{A}} \times W_{\mathcal{A}}$. The final flow is bilinearly upsampled in case of applying backward warp on inputs of resolution higher than 128.

When composing both networks, the affine matrix given by the rigid STN is applied to the non-rigid flow which results in the final sampling grid used to congeal the original input image, DINO-ViT features and saliency mask.

A.3. Loss Terms

All losses except for $\mathcal{L}_{sparsity}$ and $\mathcal{L}_{scale}$ are applied in the atlas space within the boundaries of the backward warped images. Formally, for $\mathcal{L}_{keys}$, $\mathcal{L}_{saliency}$, $\mathcal{L}_{mag}$, $\mathcal{L}_{smooth}$ and $\mathcal{L}_{center}$ the sum in the atlas space is taken over the indices $\{\mathbf{x}_{\mathcal{A}} \mid \mathcal{M}(I_i, \mathbf{x}_{\mathcal{A}}) \in I_i\}$.

We detach the atlas saliency $S_{\mathcal{A}}$ for losses which should not have an impact on the joint saliency, which are $\mathcal{L}_{keys}$ and local $\mathcal{L}_{smooth}$ (see details next).

Rigidity Loss $\mathcal{L}_{smooth}$. Recall the term $\mathcal{L}_{smooth}$, defined as in [3], is formally defined by:

$$\mathcal{L}_{smooth} = \frac{1}{N \cdot N_{\mathcal{A}}} \sum_{i=1}^{N} \sum_{x_{\mathcal{A}}} \left(\left\| J^T J \right\|_F + \left\| \left(J^T J \right)^{-1} \right\|_F \right) \tag{4}$$

where N is the number of images, $N_{\mathcal{A}}$ is the number of pixels in the atlas, and J is the Jacobian matrix of $\mathcal{M}$ at $\mathbf{x}_{\mathcal{A}}$. The term is used to prevent the non-rigid mapping from distorting the shared content by encouraging as rigid as possible mapping. The Jacobian matrix is defined by:

$$J = \frac{1}{\Delta} \begin{bmatrix} \mathbf{j}_1 & \mathbf{j}_2 \end{bmatrix} \in \mathbb{R}^{2 \times 2} \tag{5}$$

where

$$\mathbf{j}_1 = \mathcal{M}\left(I_i, \mathbf{x}_{\mathcal{A}} + \Delta \cdot \begin{bmatrix} 1 \\ 0 \end{bmatrix} \right) - \mathcal{M}(I_i, \mathbf{x}_{\mathcal{A}}), \tag{6}$$

$$\mathbf{j}_2 = \mathcal{M}\left(I_i, \mathbf{x}_{\mathcal{A}} + \Delta \cdot \begin{bmatrix} 0 \\ 1 \end{bmatrix} \right) - \mathcal{M}(I_i, \mathbf{x}_{\mathcal{A}}) \tag{7}$$

and Δ corresponds to the offset in pixels. This encourages both singular values of J to be 1, which is what is required for a rigid mapping. We apply both local and global constraints, with $\Delta = 1$ and $\Delta = 20$ respectively.

In practice, similarly to the other losses, we apply the local rigidity loss only within the salient parts of the atlas (similarly to Eq. (5) in the main paper).

A.4. Training

We train the atlas and the STNs jointly. We first bootstrap the rigid STN for 1000 epochs, and then train both the

rigid and non-rigid components on their own (separate objective function), when only the non-rigid network affects the atlas training. We train for a total of 8000 epochs, and use Adam optimizer [4] with a learning rate of $1 \cdot 10^{-4}$ for the STNs and $8 \cdot 10^{-4}$ for the atlas. Training on a set of 10 images on a Tesla V100-SXM2-32GB takes 1.2 hrs and uses 3.4GB of GPU memory, or 1.8 hrs and 5.3GB of GPU memory in case of including horizontal flips.

Loss coefficients. The loss coefficients we used for all experiments are as follows:

$$\mathcal{L} = \mathcal{L}_{keys} + \lambda_s \mathcal{L}_{saliency} + \lambda_r \mathcal{L}_{reg_{\mathcal{M}}} + \lambda_a \mathcal{L}_{reg_{\mathcal{A}}} \quad (8)$$

with $\lambda_s = 1.25, \lambda_r = 0.025, \lambda_a = 0.75$.

$$\mathcal{L}_{keys} = \lambda_l L_2 + \mathcal{D}_{cos} \quad (9)$$

with $\lambda_l = 0.875$.

$$\mathcal{L}_{reg_{\mathcal{M}}} = \lambda_{s_1} \mathcal{L}_{scale} + \lambda_{s_2} \mathcal{L}_{mag} + \mathcal{L}_{smooth} \quad (10)$$

with $\lambda_{s_1} = 8, \lambda_{s_2} = 80$. As mentioned in Appendix A.3, we apply $\mathcal{L}_{smooth}$ both locally and globally, namely:

$$\mathcal{L}_{smooth} = \mathcal{L}^{\Delta=1}_{smooth} + \lambda_{s_3} \mathcal{L}^{\Delta=20}_{smooth} \quad (11)$$

with $\lambda_{s_3} = 3.5$.

$$\mathcal{L}_{reg_{\mathcal{A}}} = \mathcal{L}_{center} + \lambda_p \mathcal{L}_{sparsity} \quad (12)$$

with $\lambda_p = 0.075$. As mentioned in Sec. 3.2 in the main paper, we apply $\lambda_p \mathcal{L}_{sparsity}$ both to the atlas saliency and the atlas keys, namely:

$$\mathcal{L}_{sparsity} = \mathcal{L}^{S_{\mathcal{A}}}_{sparsity} + \lambda_k \mathcal{L}^{K_{\mathcal{A}}}_{sparsity} \quad (13)$$

with $\lambda_k = 0.044$. For $\mathcal{L}^{S_{\mathcal{A}}}_{sparsity}$ we set the relative weight between the L1- and L0-approximation terms to be $\gamma = 2$.

The entire objective function is multiplied by a scalar $c \cdot \mathcal{L}$, where $c = 4000$.

A.4.1 Congealing under extreme deformations.

Similarly to [5], we include an option of allowing horizontal flips in a given set, which is also used for the training of subsets of SPair 71K and CUB-200-2011 (Sec. 4.2 in the main paper). The flipping is done during training: we train the STNs with both the original images and the flipped images, and update the atlas only according to the orientation that currently has a lower semantic loss (keys loss).

Due to the extreme deformations present in the subsets of SPair 71K and CUB-200-2011, to increase robustness, we further reduce the local and global rigidity coefficients to be $\times 0.25$ its original value and the global rigidity to be $\lambda_{s_3} = 0.9$. In addition, the atlas representation is gradually updated during training, i.e., the images used to update the atlas are added one-by-one where every 100 epochs the image with the lowest key loss is added. We observed that this training scheme is more stable and allows faster convergence for these sets. For the *Bicycle* set of SPair-71K, since many images contain only one wheel, we fix the atlas with the image that is most semantically similar to the average keys of the set and train the set with a fixed atlas.

block	layer	output size
0	bilinear downsample (using conv2d)	$3 \times 128 \times 128$
1	convL(64, 1, 1, 0, fusedLeakyReLU)	$64 \times 128 \times 128$
2	ResBlock((64, 3, 1, 1), (128, 3, 2, 0), (128, 1, 2, 0))	$128 \times 64 \times 64$
3	ResBlock((128, 3, 1, 1), (512, 3, 2, 0), (512, 1, 2, 0))	$512 \times 32 \times 32$
4	ResBlock((512, 3, 1, 1), (512, 3, 2, 0), (512, 1, 2, 0))	$512 \times 16 \times 16$
5	ResBlock((512, 3, 1, 1), (512, 3, 2, 0), (512, 1, 2, 0))	$512 \times 8 \times 8$
6	ResBlock((512, 3, 1, 1), (512, 3, 2, 0), (512, 1, 2, 0))	$512 \times 4 \times 4$
7	convL(512, 1, 2, 0, fusedLeakyReLU)	$512 \times 4 \times 4$ (flattened)
8	linear + fusedLeakyReLU	1×512
9	linear	1×4

Table 3. *Architecture of rigid STN.*

block	layer	output size
0	convL(64, 1, 1, 0, fusedLeakyReLU)	$64 \times 128 \times 128$
1	ResBlock((64, 3, 1, 1), (128, 3, 2, 0), (128, 1, 2, 0))	$128 \times 64 \times 64$
2	ResBlock((128, 3, 1, 1), (512, 3, 2, 0), (512, 1, 2, 0))	$512 \times 32 \times 32$
3	ResBlock((512, 3, 1, 1), (512, 3, 2, 0), (512, 1, 2, 0))	$512 \times 16 \times 16$
4	ResBlock((512, 3, 1, 1), (512, 3, 1, 1), (512, 1, 1, 0))	$512 \times 16 \times 16$
5	convL(512, 3, 1, 1, fusedLeakyReLU)	$512 \times 16 \times 16$
6	conv2d(512, 3, 1, 1) + ReLU + conv2d(2, 3, 1, 1)	$2 \times 16 \times 16$ (coarse flow)
7	conv2d(512, 3, 1, 1) + ReLU + conv2d(576, 3, 1, 1)	$576 \times 16 \times 16$ (upsampling weights)

Table 4. *Architecture of non-rigid STN.*

ResBlock((channels1, kernel1, stride1, padding1), (channels2, kernel2, stride2, padding2), (channels3, kernel3, stride3, padding3))		
order	type	layer
0	conv1	convL(channels1, kernel1, stride1, padding1, fusedLeakyReLU)
1	conv2	convL(channels2, kernel2, stride2, padding2, blur, fusedLeakyReLU)
2	skip	convL(channels3, kernel3, stride3, padding3, blur)

Table 5. *Architecture of a ResBlock.*

convL(channels, kernel, stride, padding, blur, fusedLeakyReLU)	
order	layer
0	blur (upfirdn2d)
1	conv2d(channels, kernel, stride, padding)
2	fusedLeakyReLU

Table 6. *Architecture of a convL layer.*

B. Point Correspondence Between A Pair of Images

As in [5], our method can find dense correspondences between a pair of images. For each image pair $\{I_A, I_B\}$, we transfer the ground truth keypoints $\mathbf{k}_A \in I_A$ to I_B. This is done by mapping $\mathbf{k_A}$ to the atlas, obtaining $\mathbf{k}_{\mathcal{A}}$ then mapping it to I_B. Recall that our mapping $\mathcal{M} = \mathcal{M}_r \circ \mathcal{M}_f$ is defined from the atlas to each image. For mapping $\mathbf{k}_A$ to the

Figure 9. *Limitations.* Sets containing symmetric objects under large rotations may converge partially due to relative position between semantic parts. In addition, our method is not designed to align images depicting more than one instance of the shared mode.

atlas, we first compute the inverse of the rigid transformation, which has a closed-form solution (inverse of an affine matrix). Then, since there is no closed form for obtaining the inverse of $\mathcal{M}_f$, we follow [5], and approximate the inverse using nearest neighbors. Finally, we map $\mathbf{k}_\mathcal{A}$ to I_B by bilinearly sampling the mapping grid of I_B.

C. Ablation Study: No Atlas Regularization

As discussed in the main paper (Sec. 4.3), the sparsity regularization on the atlas assists our framework in capturing the dominant shared content, while ignoring noise and background clutter. Sample cases can be seen in Fig. 10: for *Guitars*, some background content is captured by the atlas w/o this regularization. In *Art Cats*, the sparsity regularization allows us to only focus on the face, while ignoring unshared regions even if they are initially considered to be salient (cat's body, third column from the right).

References

[1] Shir Amir, Yossi Gandelsman, Shai Bagon, and Tali Dekel. Deep vit features as dense visual descriptors. *ECCVW What is Motion For?*, 2022. 1

[2] Tero Karras, Samuli Laine, Miika Aittala, Janne Hellsten, Jaakko Lehtinen, and Timo Aila. Analyzing and improving the image quality of stylegan. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2020. 1

[3] Yoni Kasten, Dolev Ofri, Oliver Wang, and Tali Dekel. Layered neural atlases for consistent video editing. *ACM Transactions on Graphics (TOG)*, 40(6):1–12, 2021. 1

[4] Diederik P Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980*, 2014. 2

[5] William Peebles, Jun-Yan Zhu, Richard Zhang, Antonio Torralba, Alexei A Efros, and Eli Shechtman. Gan-supervised dense visual alignment. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 13470–13481, 2022. 1, 2, 3

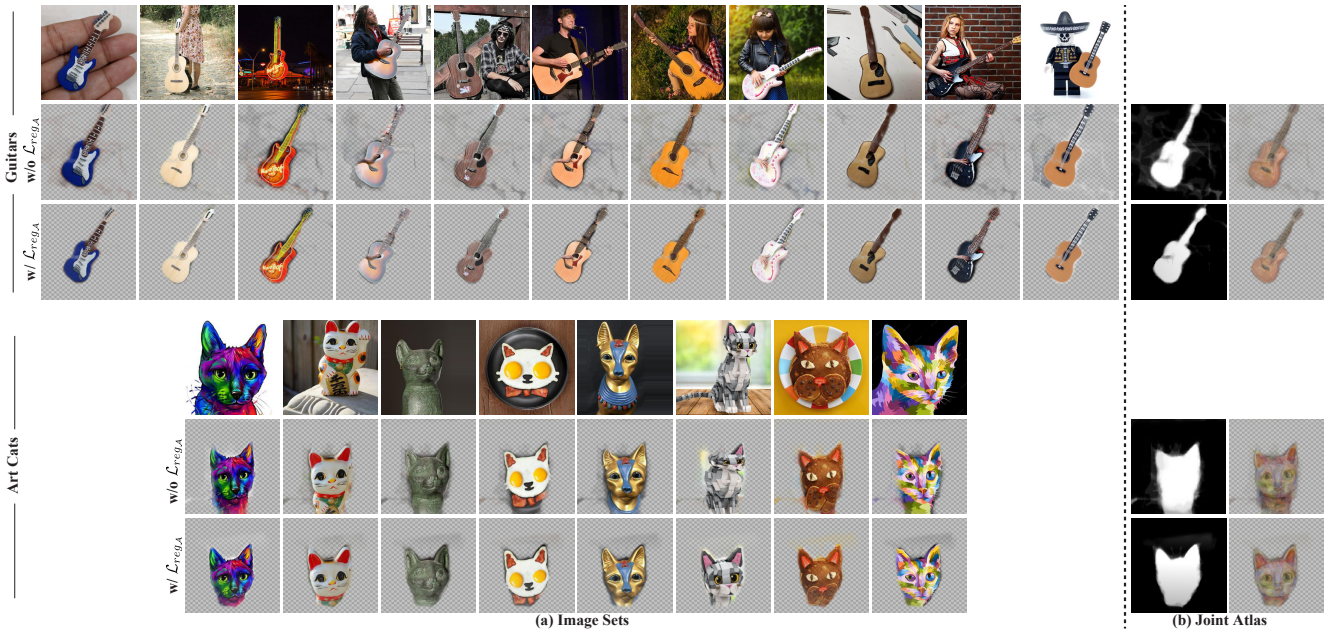

Figure 10. *Results without* $\mathcal{L}_{\mathbf{reg}_\mathcal{A}}$. For each set, we show the original images, and the congealed images with and without $\mathcal{L}_{\mathbf{reg}_\mathcal{A}}$; on the right are the average image in atlas space and the atlas saliency. This regularization encourages cleaner atlases, e.g., ignoring background clutter in *Guitars*, and allows us to better capture the dominant shared content, e.g., focusing only on the face in *Art Cats* allows us to better align the third cat from the right.