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

Dolev Ofri-Amar¹  Michal Geyer¹  Yoni Kasten²  Tali Dekel¹
¹Weizmann Institute of Science  ²NVIDIA 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 resize the images to the resolution of the atlas, which is set to be \( W_A = H_A = 128 \). 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:

\[
\theta = \pi \cdot \tanh(o_1), \quad R = \begin{bmatrix} \cos \theta & -\sin \theta \\ \sin \theta & \cos \theta \end{bmatrix}, \quad s = \exp(o_2), \quad t = \begin{bmatrix} o_3 \\ o_4 \end{bmatrix}
\]

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 = \( H_A \) × \( W_A \). 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 \( \mathcal{L}_{\text{sparsity}} \) and \( \mathcal{L}_{\text{scale}} \) are applied in the atlas space within the boundaries of the backward warped images. Formally, for \( \mathcal{L}_{\text{keys}}, \mathcal{L}_{\text{saliency}}, \mathcal{L}_{\text{mag}}, \mathcal{L}_{\text{smooth}} \) and \( \mathcal{L}_{\text{center}} \), the sum in the atlas space is taken over the indices \( \{x_A \mid M(I_i, x_A) \in I_1\} \).

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

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

\[
\mathcal{L}_{\text{smooth}} = \frac{1}{N \cdot N_A} \sum_{i=1}^{N} \sum_{x_A} \left( \|J^T J\|_F^2 + \| (J^T J)^{-1} \|_F^2 \right)
\]

where \( N \) is the number of images, \( N_A \) is the number of pixels in the atlas, and \( J \) is the Jacobian matrix of \( M \) at \( x_A \). 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:

\[
J = \frac{1}{\Delta} \begin{bmatrix} j_1 \\ j_2 \end{bmatrix} \in \mathbb{R}^{2 \times 2}
\]

where

\[
\begin{aligned}
&j_1 = M \left(I_i, x_A + \Delta \cdot \begin{bmatrix} 1 \\ 0 \end{bmatrix} \right) - M(I_i, x_A), \\
&j_2 = M \left(I_i, x_A + \Delta \cdot \begin{bmatrix} 0 \\ 1 \end{bmatrix} \right) - M(I_i, x_A)
\end{aligned}
\]

and \( \Delta \) 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:

$$L = L_{keys} + \lambda_s L_{saliency} + \lambda_r L_{reg,M} + \lambda_a L_{reg,A} \quad (8)$$

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

$$L_{keys} = \lambda_l L_2 + D_{cos} \quad (9)$$

with $\lambda_l = 0.875$.

$$L_{reg,M} = \lambda_s L_{scale} + \lambda_s L_{mag} + L_{smooth} \quad (10)$$

with $\lambda_s = 8$, $\lambda_s = 80$. As mentioned in Appendix A.3, we apply $L_{smooth}$ both locally and globally, namely:

$$L_{smooth} = L_{smooth}^{A=1} + \lambda_s L_{smooth}^{A=20} \quad (11)$$

with $\lambda_s = 3.5$.

$$L_{reg,A} = L_{center} + \lambda_p L_{sparsity} \quad (12)$$

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

$$L_{sparsity} = L_{sparsity}^{S_A} + \lambda_k L_{sparsity}^{K_A} \quad (13)$$

with $\lambda_k = 0.044$. For $L_{sparsity}^{S_A}$ 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 $c \cdot L$, 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 $0.25$ its original value and the global rigidity to be $\lambda_s = 0.9$. 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 image of the set.

### 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 $k_A \in I_A$ to $I_B$. This is done by mapping $k_A$ to the atlas, obtaining $k_A^A$ then mapping it to $I_B$. Recall that our mapping $M = M_r \circ M_t$ is defined from the atlas to each image. For mapping $k_A$ to the

<table>
<thead>
<tr>
<th>block</th>
<th>layer</th>
<th>output size</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>bilinear downsample (using conv2d)</td>
<td>$3 \times 128 \times 128$</td>
</tr>
<tr>
<td>1</td>
<td>convL(64, 1, 1, 0, fusedLeakyReLU)</td>
<td>$64 \times 128 \times 128$</td>
</tr>
<tr>
<td>2</td>
<td>ResBlock((64, 3, 1, 1), (128, 3, 2, 0), (128, 1, 2, 0))</td>
<td>$128 \times 64 \times 64$</td>
</tr>
<tr>
<td>3</td>
<td>ResBlock((128, 3, 1, 1), (512, 3, 2, 0), (512, 1, 2, 0))</td>
<td>$512 \times 32 \times 32$</td>
</tr>
<tr>
<td>4</td>
<td>ResBlock((512, 3, 1, 1), (512, 3, 2, 0), (512, 1, 2, 0))</td>
<td>$512 \times 16 \times 16$</td>
</tr>
<tr>
<td>5</td>
<td>ResBlock((512, 3, 1, 1), (512, 3, 2, 0), (512, 1, 2, 0))</td>
<td>$512 \times 8 \times 8$</td>
</tr>
<tr>
<td>6</td>
<td>convL(512, 1, 2, 0, fusedLeakyReLU)</td>
<td>$512 \times 4 \times 4$ (flattened)</td>
</tr>
<tr>
<td>7</td>
<td>linear + fusedLeakyReLU</td>
<td>$1 \times 512$</td>
</tr>
<tr>
<td>8</td>
<td>linear</td>
<td>$1 \times 4$</td>
</tr>
</tbody>
</table>

### Table 3. Architecture of rigid STN.

<table>
<thead>
<tr>
<th>order</th>
<th>type</th>
<th>layer</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>conv1</td>
<td>convL((channels1, kernel1, stride1, padding1), (channels2, kernel2, stride2, padding2), (channels3, kernel3, stride3, padding3))</td>
</tr>
<tr>
<td>1</td>
<td>conv2</td>
<td>convL((channels1, kernel1, stride1, padding1, fusedLeakyReLU))</td>
</tr>
<tr>
<td>2</td>
<td>skip</td>
<td>convL((channels1, kernel1, stride1, padding1, fusedLeakyReLU))</td>
</tr>
</tbody>
</table>

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

<table>
<thead>
<tr>
<th>block</th>
<th>layer</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>convL(64, 1, 1, 0, fusedLeakyReLU)</td>
</tr>
<tr>
<td>1</td>
<td>ResBlock((64, 3, 1, 1), (128, 3, 2, 0), (128, 1, 2, 0))</td>
</tr>
<tr>
<td>2</td>
<td>ResBlock((128, 3, 1, 1), (512, 3, 2, 0), (512, 1, 2, 0))</td>
</tr>
<tr>
<td>3</td>
<td>ResBlock((512, 3, 1, 1), (512, 3, 2, 0), (512, 1, 2, 0))</td>
</tr>
<tr>
<td>4</td>
<td>ResBlock((512, 3, 1, 1), (512, 3, 2, 0), (512, 1, 2, 0))</td>
</tr>
<tr>
<td>5</td>
<td>convL(512, 3, 1, 1, fusedLeakyReLU)</td>
</tr>
<tr>
<td>6</td>
<td>conv2d((512, 3, 1, 1) + ReLU + conv2d((2, 3, 1, 1))</td>
</tr>
</tbody>
</table>

### Table 5. Architecture of a ResBlock.

```python
ResBlock((channels1, kernel1, stride1, padding1), (channels2, kernel2, stride2, padding2), (channels3, kernel3, stride3, padding3))
```
Table 7. PCK-Transfer, comparing different encoder models. (t) - tokens; (k) - keys. *model used in our framework.

<table>
<thead>
<tr>
<th>Model type</th>
<th>Emb. size</th>
<th>SPair-71K</th>
<th>CUB</th>
</tr>
</thead>
<tbody>
<tr>
<td>CLIP ViT-B/16 (t)</td>
<td>768</td>
<td>51.2</td>
<td>-</td>
</tr>
<tr>
<td>DINO ViT-B/8 (k)</td>
<td>768</td>
<td>59.8</td>
<td>32.4</td>
</tr>
<tr>
<td>*DINO ViT-S/8 (k)</td>
<td>384</td>
<td>53.3</td>
<td>35.2</td>
</tr>
</tbody>
</table>

Figure 9. Sample results of our method using CLIP image encoder vs DINO ViT-S/8 encoder.

Figure 10. 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.

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. 11: 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).

D. Alternative Encoder Models

Our choice of ViT-S/8 was motivated by [1], which demonstrated the power of these features for several dense inference tasks. Table 7 reports the performance of additional two models: DINO ViT-B/8 and CLIP [6] image encoder of ViT-B/16 backbone – both models have embedding size of 768, compared to 384 used in our framework. For obtaining the same features’ spatial resolution as in our setting, we set the patch embeddings’ stride to 4, as suggested in [1].

As can be seen, CLIP’s features are less effective than both DINO models for this task of dense alignment. We speculate that this may be due to the discriminate training objective of CLIP, which may lead to noisier internal features [1]. The two DINO models perform on par, where ViT-S/8 is more efficient due to the smaller feature dimension.

References

Figure 11. *Results without $\mathcal{L}_{\text{regA}}$.* For each set, we show the original images, and the congealed images with and without $\mathcal{L}_{\text{regA}}$; 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.