# Supplementary Material: Stochastic Attraction-Repulsion Embedding for Large Scale Image Localization

Liu Liu [1,2], Hongdong Li [1,2] and Yuchao Dai [3]
[1] Australian National University, Canberra, Australia
[2] Australian Centre for Robotic Vision
[3] School of Electronics and Information, Northwestern Polytechnical University, Xian, China
{Liu.Liu; hongdong.li}@anu.edu.au; daiyuchao@nwpu.edu.cn

In the Supplementary Material, we describe the gradients of loss functions which jointly handle multiple negative images (Sec.1). Our implementation details are given in Sec.2 (Codes are also included). Additional experimental results are given in Sec.3.

## 1. Handling Multiple Negatives

Give a query image $q$, a positive image $p$, and multiple negative images $\{n\}, n = 1, 2, ..., N$. The Kullback-Leibler divergence loss over multiple negatives is given by:

$$L_\theta\left(q,p,n\right) = -\log\left(c^*_{p|q}\right), \tag{1}$$

For Gaussian kernel SARE, $c^*_{p|q}$ is defined as:

$$c^*_{p|q} = \frac{\exp\left(-\|f_\theta(q)-f_\theta(p)\|^2\right)}{\exp\left(-\|f_\theta(q)-f_\theta(p)\|^2\right)+\sum_{n=1}^{N}\exp\left(-\|f_\theta(q)-f_\theta(n)\|^2\right)}. \tag{2}$$

where $f_\theta(q), f_\theta(p), f_\theta(n)$ are the feature embeddings of query, positive and negative images, respectively.

Substituting Eq. (2) into Eq. (1) gives:

$$L_\theta\left(q,p,n\right) = \log\left(1+\sum_{n=1}^{N}\exp(\|f_\theta(q)-f_\theta(p)\|^2-\|f_\theta(q)-f_\theta(n)\|^2)\right) \tag{3}$$

Denote $1+\sum_{n=1}^{N}\exp(\|f_\theta(q)-f_\theta(p)\|^2-\|f_\theta(q)-f_\theta(n)\|^2)$ as $\eta$, the gradients of Eq. (3) with respect to the query, positive and negative images are given by:

$$\frac{\partial L}{\partial f_\theta(p)} = \sum_{n=1}^{N} -\frac{2}{\eta}\exp\left(\|f_\theta(q)-f_\theta(p)\|^2-\|f_\theta(q)-f_\theta(n)\|^2\right)\left[f_\theta(q)-f_\theta(p)\right], \tag{4}$$

$$\frac{\partial L}{\partial f_\theta(n)} = \frac{2}{\eta}\exp\left(\|f_\theta(q)-f_\theta(p)\|^2-\|f_\theta(q)-f_\theta(n)\|^2\right)\left[f_\theta(q)-f_\theta(n)\right], \tag{5}$$

$$\frac{\partial L}{\partial f_\theta(q)} = -\frac{\partial L}{\partial f_\theta(p)} - \sum_{n=1}^{N}\frac{\partial L}{\partial f_\theta(n)}. \tag{6}$$

Similarly, for Cauchy kernel, the loss function is given by:

$$L_\theta\left(q,p,n\right) = \log\left(1+\sum_{n=1}^{N}\frac{1+\|f_\theta(q)-f_\theta(p)\|^2}{1+\|f_\theta(q)-f_\theta(n)\|^2}\right). \tag{7}$$

Denote $1+\sum_{n=1}^{N}\frac{1+\|f_\theta(q)-f_\theta(p)\|^2}{1+\|f_\theta(q)-f_\theta(n)\|^2}$ as $\eta$, the gradients of Eq. (7) with respect to the query, positive and negative images are given by:

$$\frac{\partial L}{\partial f_\theta(p)} = \sum_{n=1}^{N}\frac{-2}{\eta\left(1+\|f_\theta(q)-f_\theta(n)\|^2\right)}\left[f_\theta(q)-f_\theta(p)\right], \tag{8}$$

$$\frac{\partial L}{\partial f_\theta(n)} = \frac{2\left(1+\|f_\theta(q)-f_\theta(p)\|^2\right)}{\eta\left(1+\|f_\theta(q)-f_\theta(n)\|^2\right)^2}\left[f_\theta(q)-f_\theta(n)\right], \tag{9}$$

$$\frac{\partial L}{\partial f_\theta(q)} = -\frac{\partial L}{\partial f_\theta(p)} - \sum_{n=1}^{N}\frac{\partial L}{\partial f_\theta(n)}. \tag{10}$$

For Exponential kernel, the loss function is given by:

$$L_\theta(q,p,n) = \log\left(1 + \sum_{n=1}^{N} \exp\left(\|f_\theta(q) - f_\theta(p)\| - \|f_\theta(q) - f_\theta(n)\|\right)\right). \tag{11}$$

Denote $1+\sum_{n=1}^{N} \exp(\|f_\theta(q) - f_\theta(p)\| - \|f_\theta(q) - f_\theta(n)\|)$ as $\eta$, the gradients of Eq. (11) with respect to the query, positive and negative images are given by:

$$\frac{\partial L}{\partial f_\theta(p)} = \sum_{n=1}^{N} -\frac{\exp\left(\|f_\theta(q) - f_\theta(p)\| - \|f_\theta(q) - f_\theta(n)\|\right)}{\eta \|f_\theta(q) - f_\theta(p)\|} \left[f_\theta(q) - f_\theta(p)\right], \tag{12}$$

$$\frac{\partial L}{\partial f_\theta(n)} = \frac{\exp\left(\|f_\theta(q) - f_\theta(p)\| - \|f_\theta(q) - f_\theta(n)\|\right)}{\eta \|f_\theta(q) - f_\theta(n)\|} \left[f_\theta(q) - f_\theta(n)\right], \tag{13}$$

$$\frac{\partial L}{\partial f_\theta(q)} = -\frac{\partial L}{\partial f_\theta(p)} - \sum_{n=1}^{N} \frac{\partial L}{\partial f_\theta(n)}. \tag{14}$$

The gradients are back propagated to train the CNN.

## 2. Implementation Details

We exactly follow the training method of [1], without fine-tuning any hyper-parameters. The VGG-16 [7] net is cropped at the last convolutional layer (conv5), before ReLU. The learning rate for the Pitts30K-train and Pitts250K-train datasets are set to 0.001 and 0.0001, respectively. They are halved every 5 epochs, momentum 0.9, weight decay 0.001, batch size of 4 tuples. Each tuple consist of one query image, one positive image, and ten negative images. The CNN is trained for at most 30 epochs but convergence usually occurs much faster (typically less than 5 epochs). The network which yields the best recall@5 on the validation set is used for testing.

**Triplet ranking loss** For the triplet ranking loss [1], we set margin $m = 0.1$, and triplet images producing a non-zero loss are used in gradient computation, which is the same as [1].

**Contrastive loss** For the contrastive loss [6], we set margin $\tau = 0.7$, and negative images producing a non-zero loss are used in gradient computation. Note that positive images are always used in training since they are not pruned out.

**Geographic classification loss** For the geographic classification method [9], we use the Pitts250k-train dataset for training. We first partition the 2D geographic space into square cells, with each cell size at $25m$. The cell size is selected the same as the evaluation metric for compatibleness, so that the correctly classified images are also the correctly localized images according to our evaluation metric. We remove the Geo-classes which do not contain images, resulting in 1637 Geo-classes. We append a fully connected layer (random initialization, with weights at $0.01 \times randn$) and Softmax-log-loss layer after the NetVLAD pooling layer to predict which class the image belongs to.

**SARE loss** For our methods (*Our-Ind.*, and *Our-Joint* ), *Our-Ind.* treats multiple negative images independently while *Our-Joint* treats multiple negative images jointly. The two methods only differ in the loss function and gradients computation. For each method, the corresponding gradients are back-propagated to train the CNN.

**Triplet angular loss** For the triplet angular loss [10], we use the N-pair loss function (Eq. (8) in their paper) with $\alpha = 45°$ as it achieves the best performance on the Stanford car dataset.

**N-pair loss** For the N-pair loss [8], we use the N-pair loss function (Eq. (3) in their paper).

**Lifted structured loss** For the lifted structured loss [5], we use the smooth loss function (Eq. (4) in their paper). Note that training images producing a zero loss ($\tilde{J}_{i,j} < 0$) are pruned out.

**Ratio loss** For the Ratio loss [2], we use the MSE loss function since it achieves the best performance in there paper.

## 3. Additional Results

**Dataset.** Table 2 gives the details of datasets used in our experiments.

**Visualization of feature embeddings.** Fig. 1 and Fig. 2 visualize the feature embeddings of the 24/7 Tokyo-query and Sf-0-query dataset computed by our method (*Our-Ind.*) in 2-D using the t-SNE [4], respectively. Images are displayed exactly at their embedded locations. Note that images taken from the same place are mostly embedded to nearby 2D positions although they differ in lighting and perspective.

**Image retrieval for varying dimensions.** Table 3 gives the comparison of image retrieval performance for different output dimensions.

Table 1: Comparison of Recalls on the Pitts250k-test, TokyoTM-val, 24/7 Tokyo and Sf-0 datasets.

| Method \ Dataset | Pitts250k-test | | | TokyoTM-val | | | 24/7 Tokyo | | | Sf-0 | | |
|---|---|---|---|---|---|---|---|---|---|---|---|---|
| | r@1 | r@5 | r@10 | r@1 | r@5 | r@10 | r@1 | r@5 | r@10 | r@1 | r@5 | r@10 |
| Our-Ind. | **88.97** | **95.50** | **96.79** | 94.49 | 96.73 | 97.30 | 79.68 | 86.67 | 90.48 | **80.60** | **86.70** | **89.01** |
| Our-Joint | 88.43 | 95.06 | 96.58 | 94.71 | 96.87 | **97.51** | 80.63 | 87.30 | 90.79 | 77.75 | 85.07 | 87.52 |
| Contrastive [6] | 86.33 | 94.09 | 95.88 | 93.39 | 96.09 | 96.98 | 75.87 | 86.35 | 88.89 | 74.63 | 82.23 | 84.53 |
| N-pair [8] | 87.56 | 94.57 | 96.21 | 94.42 | 96.73 | 97.41 | 80.00 | **89.52** | **91.11** | 76.66 | 83.85 | 87.11 |
| Angular [10] | 88.60 | 94.86 | 96.44 | **94.84** | 96.83 | 97.45 | **80.95** | 87.62 | 90.16 | 79.51 | 86.57 | 88.06 |
| Liftstruct [5] | 87.40 | 94.52 | 96.28 | 94.48 | **96.90** | 97.47 | 77.14 | 86.03 | 89.21 | 78.15 | 84.67 | 87.11 |
| Geo-Classification [9] | 83.19 | 92.67 | 94.59 | 93.54 | 96.80 | 97.50 | 71.43 | 82.22 | 85.71 | 67.84 | 78.15 | 81.41 |
| Ratio [2] | 87.28 | 94.25 | 96.07 | 94.24 | 96.84 | 97.41 | 80.32 | 87.30 | 88.89 | 76.80 | 85.62 | 87.38 |

Table 2: Datasets used in experiments. The Pitts250k-train dataset is only used to train the Geographic classification CNN [9]. For all the other CNNs, Pitts30k-train dataset is used to enable fast training.

| Dataset | #database images | #query images |
|---|---|---|
| Pitts250k-train | 91,464 | 7,824 |
| Pitts250k-val | 78,648 | 7,608 |
| Pitts250k-test | 83,952 | 8,280 |
| Pitts30k-train | 10,000 | 7,416 |
| Pitts30k-val | 10,000 | 7,608 |
| Pitts30k-test | 10,000 | 6,816 |
| TokyoTM-val | 49,056 | 7,186 |
| Tokyo 24/7 (-test) | 75,984 | 315 |
| Sf-0 | 610,773 | 803 |
| Oxford 5k | 5063 | 55 |
| Paris 6k | 6412 | 220 |
| Holidays | 991 | 500 |

**Metric learning methods** Table 1 gives the complete *Recall@N* performance for different methods. Our method outperforms the contrastive loss [6] and Geo-classification loss [9], while remains comparable with other state-of-the-art metric learning methods.

Table 3: Retrieval performance of CNNs trained on Pitts250k-test dataset on image retrieval benchmarks. No spatial re-ranking, or query expansion are performed. The accuracy is measured by the mean Average Precision (mAP).

| Method | Dim. | Oxford 5K | | Paris 6k | | Holidays |
|---|---|---|---|---|---|---|
| | | full | crop | full | crop | |
| Our-Ind. | 4096 | **71.66** | **75.51** | **82.03** | 81.07 | 80.71 |
| Our-Joint | 4096 | 70.26 | 73.33 | 81.32 | **81.39** | **84.33** |
| NetVLAD [1] | 4096 | 69.09 | 71.62 | 78.53 | 79.67 | 83.00 |
| CRN [3] | 4096 | 69.20 | - | - | - | - |
| Our-Ind. | 2048 | **71.11** | **73.93** | **80.90** | 79.91 | 79.09 |
| Our-Joint | 2048 | 69.82 | 72.37 | 80.48 | **80.49** | **83.17** |
| NetVLAD [1] | 2048 | 67.70 | 70.84 | 77.01 | 78.29 | 82.80 |
| CRN [3] | 2048 | 68.30 | - | - | - | - |
| Our-Ind. | 1024 | **70.31** | **72.20** | **79.29** | **78.54** | 78.76 |
| Our-Joint | 1024 | 68.46 | 70.72 | 78.49 | 78.47 | **83.15** |
| NetVLAD [1] | 1024 | 66.89 | 69.15 | 75.73 | 76.50 | 82.06 |
| CRN [3] | 1024 | 66.70 | - | - | - | - |
| Our-Ind. | 512 | **68.96** | **70.59** | **77.36** | 76.44 | 77.65 |
| Our-Joint | 512 | 67.17 | 69.19 | 76.80 | **77.20** | **81.83** |
| NetVLAD [1] | 512 | 65.56 | 67.56 | 73.44 | 74.91 | 81.43 |
| CRN [3] | 512 | 64.50 | - | - | - | - |
| Our-Ind. | 256 | **65.85** | 67.46 | **75.61** | 74.82 | 76.27 |
| Our-Joint | 256 | 65.30 | **67.51** | 74.50 | **75.32** | **80.57** |
| NetVLAD [1] | 256 | 62.49 | 63.53 | 72.04 | 73.47 | 80.30 |
| CRN [3] | 256 | 64.20 | - | - | - | - |
| Our-Ind. | 128 | **63.75** | **64.71** | **71.60** | **71.23** | 73.57 |
| Our-Joint | 128 | 62.92 | 63.63 | 69.53 | 70.24 | 77.81 |
| NetVLAD [1] | 128 | 60.43 | 61.40 | 68.74 | 69.49 | **78.65** |
| CRN [3] | 128 | 61.50 | - | - | - | - |

## References


[1] R. Arandjelovic, P. Gronat, A. Torii, T. Pajdla, and J. Sivic. Netvlad: Cnn architecture for weakly supervised place recognition. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 5297–5307, 2016. 2, 3

[2] E. Hoffer and N. Ailon. Deep metric learning using triplet network. In *International Workshop on Similarity-Based Pattern Recognition*, pages 84–92. Springer, 2015. 2, 3

[3] H. J. Kim, E. Dunn, and J.-M. Frahm. Learned contextual feature reweighting for image geo-localization. In *CVPR*, 2017. 3

[4] L. v. d. Maaten and G. Hinton. Visualizing data using t-sne. *Journal of Machine Learning Research*, 9(Nov):2579–2605, 2008. 2, 4, 5

[5] H. Oh Song, Y. Xiang, S. Jegelka, and S. Savarese. Deep metric learning via lifted structured feature embedding. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4004–4012, 2016. 2, 3

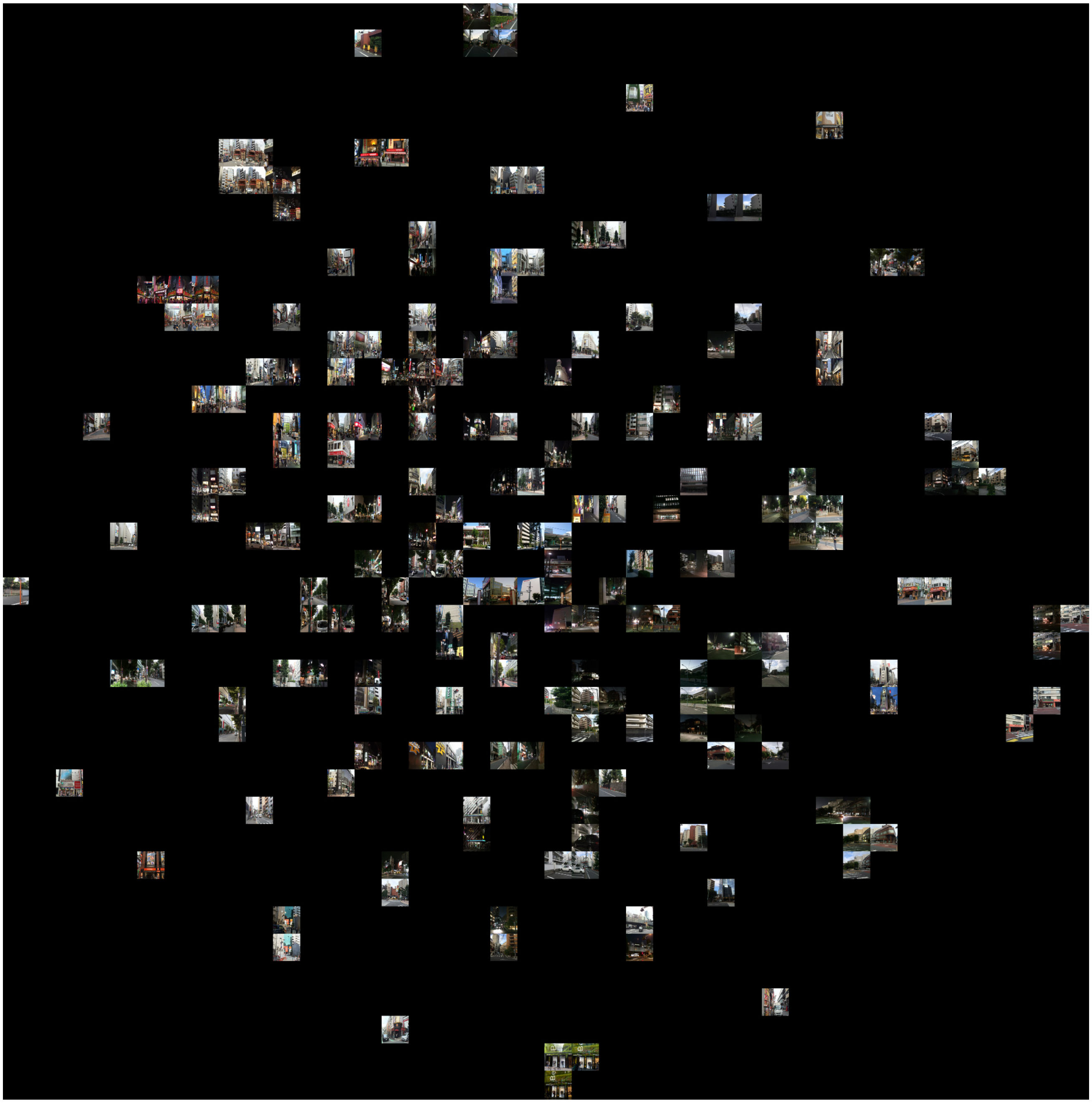

Figure 1: Visualization of feature embeddings computed by our method ( *Our-Ind.* ) using t-SNE [4] on the 24/7 Tokyo-query dataset. (Best viewed in color on screen)


[6] F. Radenović, G. Tolias, and O. Chum. Cnn image retrieval learns from bow: Unsupervised fine-tuning with hard examples. In *European Conference on Computer Vision*, pages 3–20. Springer, 2016. 2, 3

[7] K. Simonyan and A. Zisserman. Very deep convolutional networks for large-scale image recognition. *arXiv preprint arXiv:1409.1556*, 2014. 2

[8] K. Sohn. Improved deep metric learning with multi-class n-pair loss objective. In *Advances in Neural Information Processing Systems*, pages 1857–1865, 2016. 2, 3

[9] N. Vo, N. Jacobs, and J. Hays. Revisiting im2gps in the deep learning era. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017. 2, 3

[10] J. Wang, F. Zhou, S. Wen, X. Liu, and Y. Lin. Deep metric learning with angular loss. In *The IEEE International Conference on Computer Vision (ICCV)*, Oct 2017. 2, 3

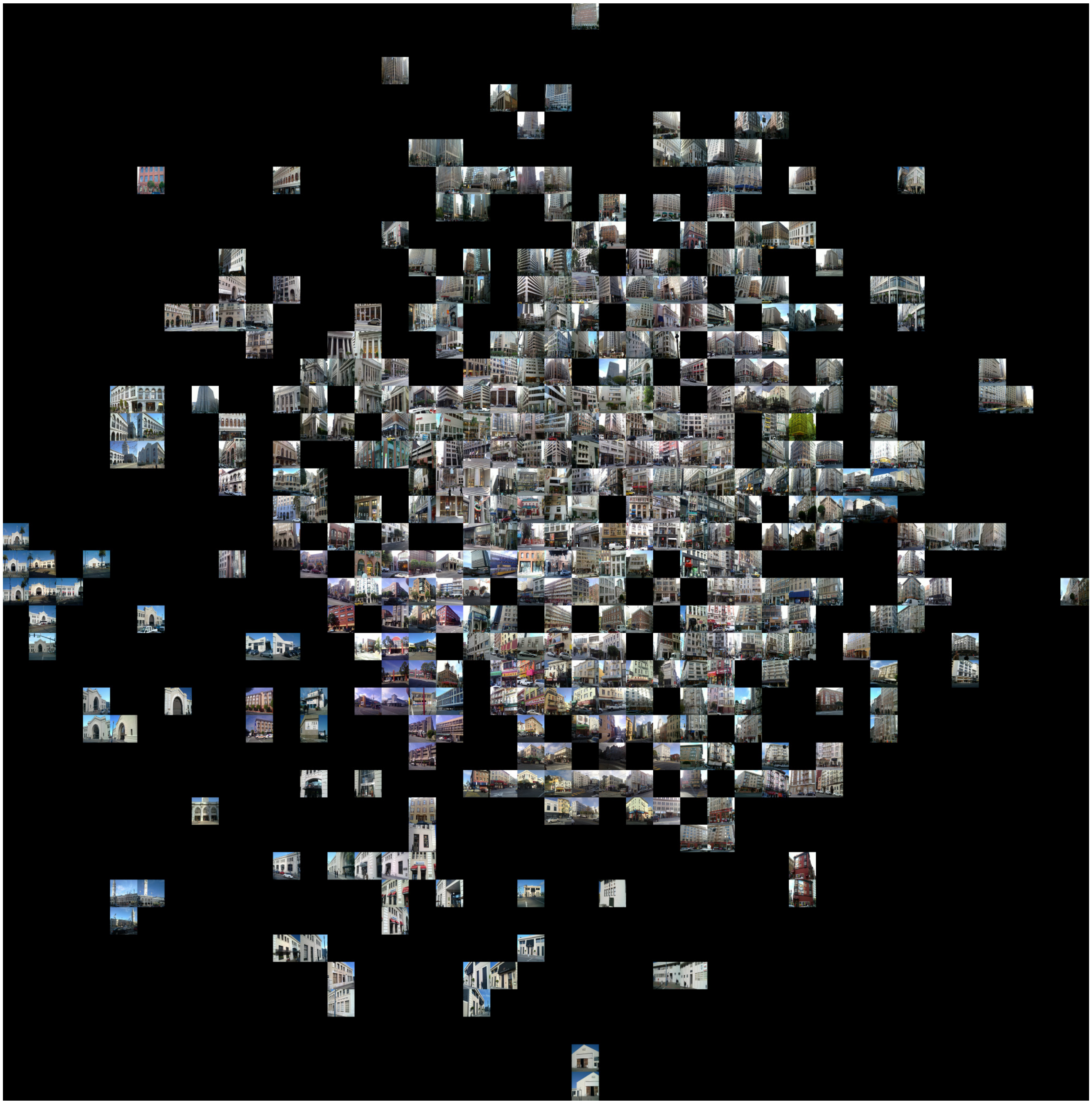

Figure 2: Visualization of feature embeddings computed by our method ( *Our-Ind.* ) using t-SNE [4] on the Sf-0-query dataset. (Best viewed in color on screen)