SUPPLEMENTARY MATERIAL

Author(s) Name(s)

Author Affiliation(s)

1. BASELINE ARCHITECTURE

1.1. Detailed baseline architecture

Overview. The network takes as input M and N visual descriptors and keypoint positions from images A and B, indexed by $\mathcal{A} \coloneqq \{1, ..., M\}$ and $\mathcal{B} \coloneqq \{1, ..., N\}$, respectively. Input descriptors and keypoint positions are denoted as $\{{}^0\mathbf{x}_i^I\}_i$ and $\{{}^0\mathbf{p}_i^I\}_i$, respectively, where $I \in \{A, B\}$. The network updates descriptors by L attentional layers. Finally, the network outputs a partial assignment matrix $\mathbf{P} \in [0, 1]^{M+1 \times N+1}$ whose (i, j) element represents the confidence that the i-th keypoint in the image A and the j-th one in the image B are matched. The last column and rows of P represents dustbins for unmatched keypoints. We obtain point correspondences by selecting the pairs with confidences larger than the threshold τ.

Positional encoding Our network uses positional encoding that encodes absolute keypoint positions as in the existing matchers [1, 2]. In our architecture, positional encoding is performed in each layer. The positional encoding in the ℓ-th layer is calculated as follows:

$$ {}^{\ell-1}\mathbf{x}_i^I \leftarrow {}^{\ell-1}\mathbf{x}_i^I + \mathrm{MLP}({}^0\mathbf{p}_i^I). \tag{1}$$

where ${}^{\ell-1}\mathbf{x}_i^I$ denotes descriptors updated from $\ell - 1$-th layer. Some keypoint detectors, e.g., SIFT [3], output the orientation φ_i and scale η_i of each keypoint as well as the position. These geometric information are added to the positinoal encodings [4] as follows:

$$ \mathbf{x}_i^I \leftarrow \mathbf{x}_i^I + \mathrm{MLP}(\left[\mathbf{p}_i^I | \cos\varphi_i^I | \sin\varphi_i^I | \eta_i^I\right]). \tag{2}$$

Self-Cross attention. The ℓ-th attentioanl layer takes as input descriptors $\{{}^{l-1}\mathbf{x}_i^I\}_i$ and update them into $\{{}^{l}\mathbf{x}_i^I\}_i$. The layer subscripts are omitted hereafter for ease of reading. Descriptors are updated on the basis of the message passing style like SuperGlue as follows:

$$ \mathbf{x}_i^I \leftarrow \mathbf{x}_i^I + \mathrm{MLP}\left(\left[\mathbf{x}_i^I | \mathbf{m}_i^{I \leftarrow S}\right]\right), \tag{3}$$

where MLP, $[\cdot|\cdot]$, and $\mathbf{m}^{I \leftarrow S}$ denote a Multi-Layer Perceptron (MLP), concatenation, and a mesasge from a source image $S \in \{A, B\}$ to a target image I, respectively. The messages are calculated on the basis of the attention mechanism. Self attention computes a message $\mathbf{m}_i^{I \leftarrow I}$ and cross attention does $\mathbf{m}_i^{I \leftarrow S}$, where $S = \{A, B\} \backslash I$. In the attention computation, the descriptors are converted by learnable linear transformations to queries, keys and values as $\mathbf{q}_i = \mathbf{W}_q \mathbf{x}_i^I$, $\mathbf{k}_j = \mathbf{W}_k \mathbf{x}_j^S$ and $\mathbf{v}_j = \mathbf{W}_v \mathbf{x}_j^S$, respectively. Let $\mathbf{K}$ and $\mathbf{V}$ be $\mathbf{K} = [\mathbf{k}_1, ..., \mathbf{k}_{|S|}] \in \mathbb{R}^{d \times |S|}$ and $\mathbf{V} = [\mathbf{v}_1, ..., \mathbf{v}_{|S|}] \in \mathbb{R}^{d \times |S|}$, respectively. The message $\mathbf{m}_i^{I \leftarrow S}$ is computed by attention as

$$ \mathbf{m}_i^{I \leftarrow S \top} = \mathrm{Softmax}\left(\frac{\mathbf{q}_i{}^\top \mathbf{K}}{\sqrt{d}}\right)\mathbf{V}^\top. \tag{4}$$

We extend all the above attentions to multi-head ones in practical implementation.

Matching module. This module in the ℓ-th block temporarily performs matching to obtain an assignment matrix $\mathbf{P}$. The method of calculation is the same as the one proposed in LightGlue. First, we calculate a score matrix $\mathbf{S} \in \mathbb{R}^{M \times N}$ as follows:

$$ S_{i,j} = \mathrm{Linear}(\mathbf{x}_i^A)^\top \mathrm{Linear}(\mathbf{x}_j^B), \tag{5}$$

where Linear is a linear transformation with learnable parameters. Then, the score σ_i^I is computed as

$$ \sigma_i^I = \mathrm{Sigmoid}(\mathrm{Linear}(\mathbf{x}_i^I)). \tag{6}$$

Finally, an assignment matrix is obtained as

$$ P_{i,j} = \sigma_i^A \sigma_j^B \mathrm{Softmax}(S_{i,j})_i \mathrm{Softmax}(S_{i,j})_j. \tag{7}$$

This matching method is faster than the sinkhorn algorithm [5] used in some matchers [1, 2], and we can obtain an assignment matrix in each layer without large overhead.

2. TRAINING

2.1. An example of strong data augmentation

We performed strong data augmentation including large scale and rotation as described in the paper. Fig. 1 shows an example of an augmented image pair.

2.2. Other details about training

We trained local feature matchers by only MegaDepth dataset. Unlike existing works [1, 6], homography pre-training is not performed. For both SIFT and SuperPoint, we extract 2048

Fig. 1: **An example of data augmentation**

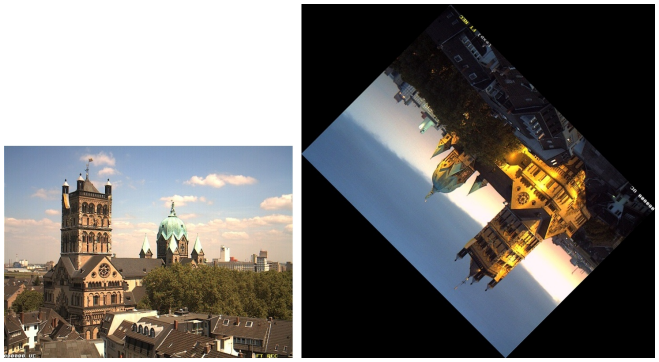

Fig. 2: **An example of a rotated image pair (Rotation angle is 225°).**

keypoints from all the training images. We trained local feature matchers for 60 epochs without data augmentation. If we perform the data augmentation, matchers are trained for 180 epochs since the training becomes difficult for matchers and the convergence gets slow. We use the Adam optimizer with the learning rate of 10^{-4}, where, in the training with data augmentation, the learning rate is exponentially decayed by 0.95 in each epoch after 120 epoch. Note that the parameters of SuperPoint are frozen during training, and only matchers are trained.

3. EXPERIMENTAL DETAILS

3.1. Experiments on Homography estimation

Experimental setup of rotation test. For rotation evaluation, we rotate an image of an image pair from 0° to 315° in 45° increment, and 8 datasets are created for the respective rotation angles. Fig. 2 shows an example of rotated image pairs. We resize images so that their smaller dimension is 480 and extract 1024 local features per images with NMS of 3 pixel. Note that rotated images have the same resolution except for the empty black areas in Fig. 2.

Experimental setup of rotation test. For scale evaluation, one of two images is scaled by the factor of 1.0, 2.0 and 3.0 to create 3 datasets. Fig. 3 shows an example of scaled image pairs. We resize reference images so that their smaller dimension is 300 and extract 1024 local features per images with NMS of 3 pixel. Target images are resized so that their smaller dimension is 300, 600, and 900 for respective scaling factors.

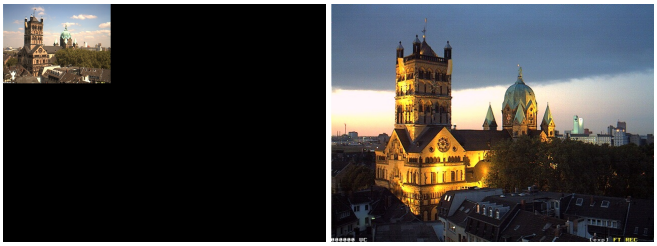

Fig. 3: **An example of a scaled image pair (Scaling factor is 3.0).**

Experimental setup of SIM2E. For SIM2E dataset, we resized all images so that their smaller dimension is 480 and extract 1024 local features per images with NMS of 3 pixel. Table 2 shows the results for overall SIM2E dataset. SN and DA are effective also for SIM2E.

3.2. Experiments on MegaDepth1500

We resized images so that their larger dimension is 1600 and extract 2048 local features per images. NMS radius of SuperPoint and SIFT is 3 and 5, respectively. We split image pairs of MegaDepth1500 according to scale changes between image pairs. A scale change between an image pairs is defined as described below. First, we detect keypoints of SuperPoint for all the images of MegaDepth1500. Then, GT point correspondences for each image pair are obtained by using a GT essential matrix and intrinsic parameters. We calculate a covariance matrix $C^I \in \mathbb{R}^{2\times 2}$ of keypoint positions for each image and the two eigen values (e_1^I, e_2^I), where I means an index of an input image, namely, $I \in A, B$. We denote a larger eigen value and a smaller eigen value of each image as e^I_{larger} and $e^I_{smaller}$, which is defined as

$$e^I_{larger} = \max\left\{e^I_1, e^I_2\right\}, e^I_{smaller} = \min\left\{e^I_1, e^I_2\right\}. \quad (8)$$

Finally, the scale gap g between the image pair is defined as

$$g = \max\left\{ \max\left\{ \frac{\sqrt{e^A_{larger}}}{\sqrt{e^B_{larger}}}, \frac{\sqrt{e^B_{larger}}}{\sqrt{e^A_{larger}}} \right\} , \max\left\{ \frac{\sqrt{e^A_{smaller}}}{\sqrt{e^B_{smaller}}}, \frac{\sqrt{e^B_{smaller}}}{\sqrt{e^A_{smaller}}} \right\} \right\}. \quad (9)$$

4. QUALITATIVE VISUALIZATION

4.1. Visualizations for MegaDepth1500 dataset

Fig. 4 and 5 show qualitative visualizations on SIM2E dataset using SIFT and SuperPoint, respectively.

	matcher	HPatches (Rotation)			HPatches (Scaling)		
		prec @3px	DLT 1px 3px / 5px	LO-RANSAC 1px 3px / 5px	prec @10px	DLT 5px / 10px / 15px	LO-RANSAC 5px / 10px / 15px
SIFT	NN (ratio test)	84.7	1.1 / 3.8 / 5.4	21.2 / 55.9 / 68.5	86.5	6.2 / 9.8 / 12.7	57.8 / 71.1 / 76.7
	SGMNet	78.1	10.2 / 31.2 / 40.8	20.2 / 50.8 / 62.2	95.5	52.0 / 67.9 / 74.9	63.7 / 76.9 / 82.4
	LightGlue	85.1	15.4 / 43.3 / 54.4	22.1 / 57.7 / 70.0	95.2	54.4 / 68.6 / 75.0	**65.3** / 78.5 / 83.8
	Baseline	45.2	8.1 / 19.3 / 23.7	12.8 / 28.2 / 34.1	93.8	51.2 / 66.0 / 72.8	65.0 / 78.1 / 83.3
	+SN	89.2	19.5 / 56.5 / 69.3	23.6 / 62.4 / 74.7	95.6	54.5 / 69.5 / 76.3	65.2 / 78.2 / 83.6
	+DA	**92.1**	**21.7** / 59.1 / 71.6	**23.7** / 62.8 / **75.1**	**97.1**	56.9 / 71.8 / 78.3	**65.3** / **78.7** / **84.0**
	+SN+DA	**92.1**	**21.7** / **59.4** / **71.9**	23.6 / **62.9** / **75.1**	97.0	**57.6** / **72.8** / **79.5**	**65.3** / **78.7** / **84.0**
SuperPoint	NN mutual	14.7	0.0 / 0.2 / 0.4	7.3 / 17.9 / 22.4	59.5	4.1 / 6.4 / 8.0	47.8 / 63.3 / 70.4
	SuperGlue	35.4	11.3 / 24.0 / 28.5	13.5 / 26.9 / 31.3	96.8	59.8 / 75.5 / 81.7	**62.5** / **78.0** / **83.8**
	SGMNet	33.9	9.3 / 21.9 / 26.6	13.3 / 31.9 / 27.9	95.1	51.3 / 68.4 / 75.8	43.4 / 63.2 / 72.4
	LightGlue	33.7	10.8 / 22.9 / 27.1	12.0 / 24.2 / 28.4	**97.3**	**60.1** / **76.1** / **82.3**	62.1 / 77.7 / 83.7
	Baseline	31.6	8.3 / 19.4 / 23.7	11.1 / 22.7 / 26.9	95.2	56.2 / 72.6 / 79.3	61.3 / 76.8 / 82.7
	+SN	32.5	9.0 / 19.0 / 24.0	11.4 / 23.2 / 27.4	95.4	54.7 / 71.7 / 78.8	60.3 / 76.1 / 82.2
	+DA	86.9	28.6 / 62.5 / 73.8	32.2 / 66.3 / 77.2	94.4	55.7 / 71.2 / 77.4	60.4 / 75.6 / 81.4
	+SN+DA	**87.9**	**30.1** / **64.0** / **75.4**	**32.6** / **66.8** / **77.8**	96.4	57.4 / 73.7 / 80.3	61.4 / 77.2 / 83.3

Table 1: **Results on HPtaches and Rotated HPatches.** Best and second-best values are in bold and underlined, respectively.

	matcher	prec 3px	AUC DLT 1px / 5px	AUC RANSAC 1px / 5px
SIFT	NN (ratio test)	81.8	1.4 / 4.5	33.2 / 73.0
	SGMNet	71.8	13.0 / 38.1	32.2 / 64.6
	LightGlue	76.9	16.1 / 43.2	34.8 / 73.8
	Baseline	34.6	8.6 / 17.6	17.5 / 29.6
	Baseline+SN	86.4	23.9 / 63.2	37.5 / 79.7
	Baseline+DA	90.5	28.8 / 69.5	**38.1** / 80.4
	Baseline+SN+DA	**90.9**	**29.2** / **70.3**	**38.1** / **80.5**
SuperPoint	NN mutual	10.8	0.0 / 0.1	8.3 / 18.2
	SuperGlue	30.3	10.8 / 23.1	17.5 / 30.5
	SGMNet	28.7	8.6 / 20.6	17.3 / 31.2
	LightGlue	25.6	10.0 / 20.4	13.6 / 24.1
	Baseline	23.9	7.7 / 17.7	12.6 / 22.7
	Baseline+SN	23.8	8.5 / 18.5	13.0 / 23.0
	Baseline+DA	88.2	34.5 / 73.2	48.4 / 86.3
	Baseline+SN+DA	**90.0**	**36.6** / **74.9**	**49.4** / **87.1**

Table 2: **Results on SIM2E.** Best and second-best values are in bold and underlined, respectively.

5. REFERENCES

[1] Paul-Edouard Sarlin, Daniel DeTone, Tomasz Malisiewicz, and Andrew Rabinovich, "Superglue: Learning feature matching with graph neural networks," in *Proceedings of the IEEE/CVF conference on computer vision and pattern recognition*, 2020, pp. 4938–4947.

[2] Jianyun Xu, Xin Tang, Yushi Zhu, Jie Sun, and Shiliang Pu, "Sgmnet: Learning rotation-invariant point cloud representations via sorted gram matrix," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2021, pp. 10468–10477.

[3] David G Lowe, "Distinctive image features from scale-invariant keypoints," *International journal of computer vision*, vol. 60, pp. 91–110, 2004.

[4] Ostap Viniavskyi, Mariia Dobko, Dmytro Mishkin, and Oles Dobosevych, "Openglue: Open source graph neural net based pipeline for image matching," *arXiv preprint arXiv:2204.08870*, 2022.

[5] Richard Sinkhorn and Paul Knopp, "Concerning nonnegative matrices and doubly stochastic matrices," *Pacific Journal of Mathematics*, vol. 21, no. 2, pp. 343–348, 1967.

[6] Philipp Lindenberger, Paul-Edouard Sarlin, and Marc Pollefeys, "Lightglue: Local feature matching at light speed," in *Proceedings of the IEEE/CVF International Conference on Computer Vision*, 2023, pp. 17627–17638.

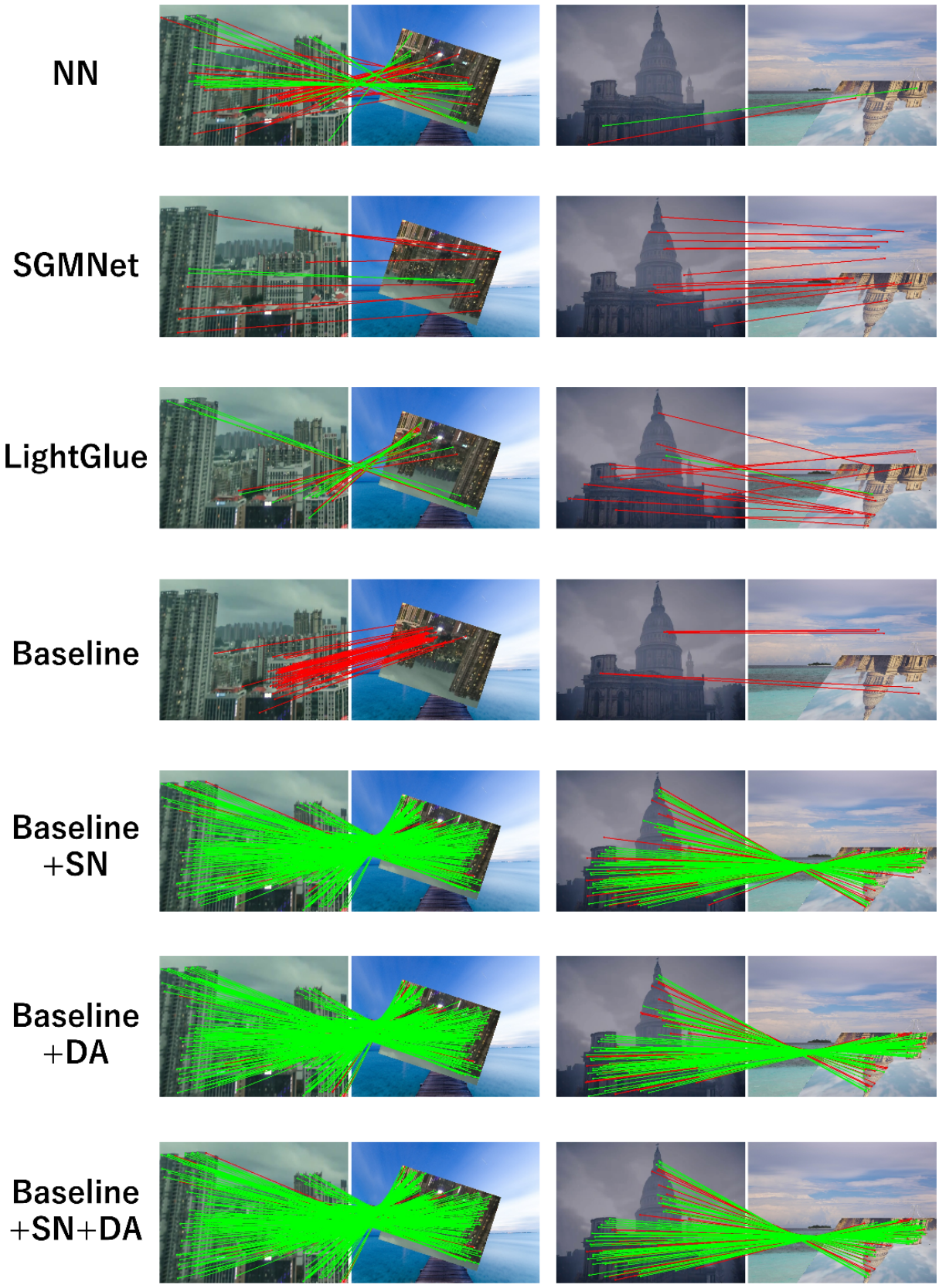

Fig. 4: **Visualizations on SIM2E (SIFT).** Green and red lines represent true and false matches, respectively.

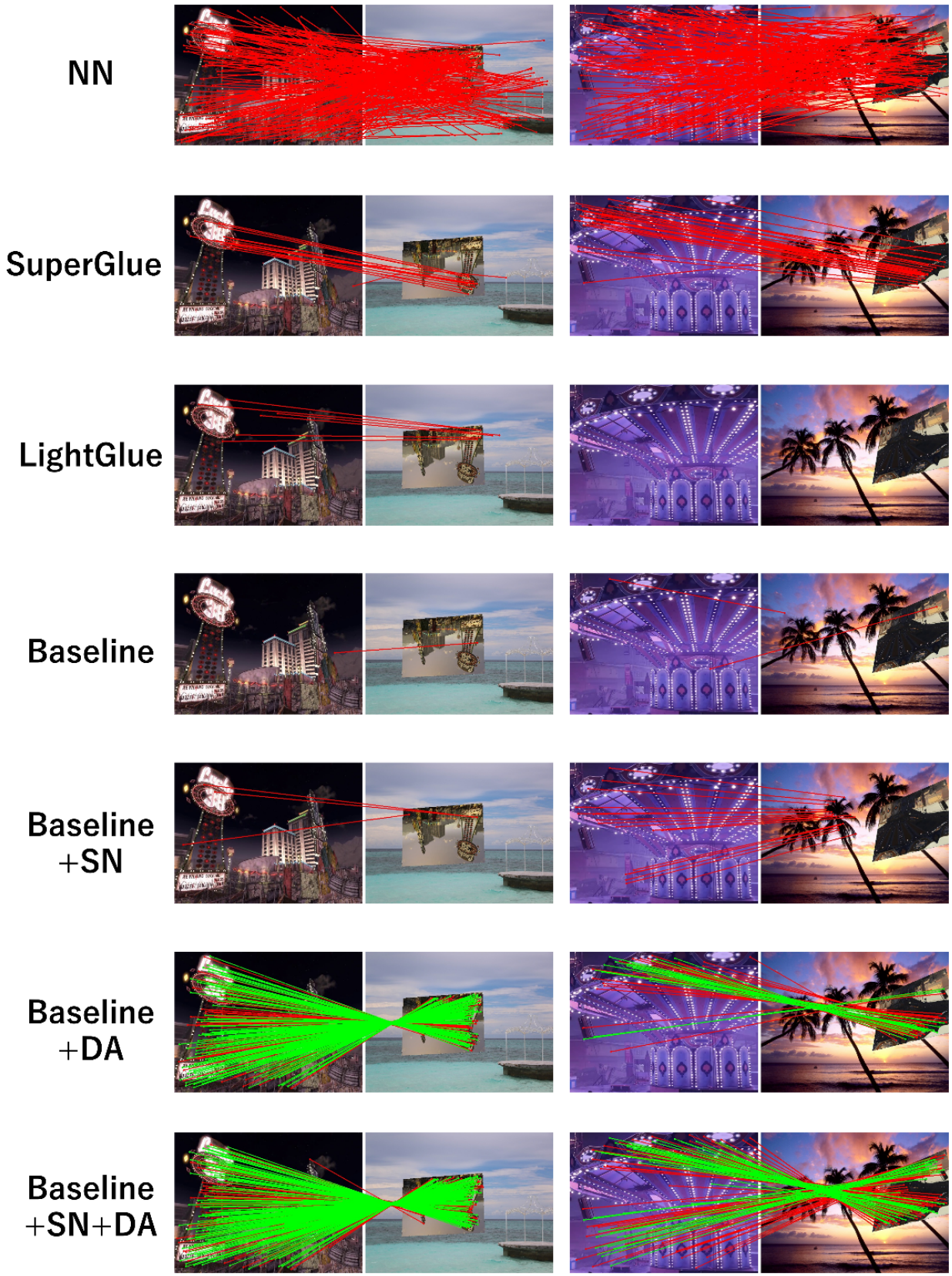

Fig. 5: **Visualizations on SIM2E (SuperPoint).** Green and red lines represent true and false matches, respectively.