Matching Semantically Similar Non-Identical Objects

Supplementary Material

A. Additional matching results

In Sec. 5.1 of the main text, we evaluated non-identical object matching by inserting our method between SuperPoint [16] and LightGlue [37]. Here, "SEW" in the results refers to the case where only our weighting module is used without applying the Non-visual Object Pairing. The matching results for various image pairs show that our method enhances matchers and provides dense and consistent correspondences. The fine-grained correspondence can be applied to tasks such as object warping, as illustrated in Fig. A. To demonstrate this generality for various image pairs, we show more results in three cases: (i) same class (i.e., $y_A = y_B, D_A = D_B$), (ii) class discrepancy (i.e., $y_A \neq y_B, D_A = D_B$), (iii) domain shift (i.e., $o_A \equiv o_B$, $D_A \neq D_B$). Additionally, we present more results for complex scenes containing multiple objects in a single image.

Setup. We collected various images with objects from the COCO [36] and ImageNet [15] datasets. To prepare drawing or sketch versions of several images, we used the DreamStudio[3], which is backended by the Stable Diffusion [66]. For details of the experiment, refer to Sec. 5.1.

A.1. Success cases

Figure E shows the matching results for the same class case. As in the main text, The introduction of the proposed method turns sparse and inconsistent matching into dense and consistent one. The combination of SuperPoint and LightGlue with our method is robust to object color and shape differences and performs dense and consistent matching. Even in the case with class discrepancy, as shown in Fig. F, the proposed method successfully corresponds the same parts (e.g., eyes to eyes, nose to nose) between various animals. Figure G shows the case of domain shift between images, where our method reduces mismatching caused by image corruptions and image style changes (e.g., photos and drawings). Figure H shows cases where multiple objects are present in an image, demonstrating that our Non-visual Object Pairing selects semantically similar object pairs and provides condensed matching between the two objects. These results indicate that the proposed method can perform accurate matching in many cases under the challenging condition of non-identical object matching.

A.2. Failure cases

There are some cases of failure in non-identical object matching, as shown in Fig. I. It is difficult to match objects that are of the same class but have significantly different shapes (e.g., the presence or absence of an airplane propeller) or objects that are similar in color or size to other parts (e.g., eyes and nose that are small, round, and black) of the object. Furthermore, image corruptions sometimes hide the texture of the objects, making it difficult to pinpoint the areas to be matched.

[3]DreamStudio: https://beta.dreamstudio.ai/generate

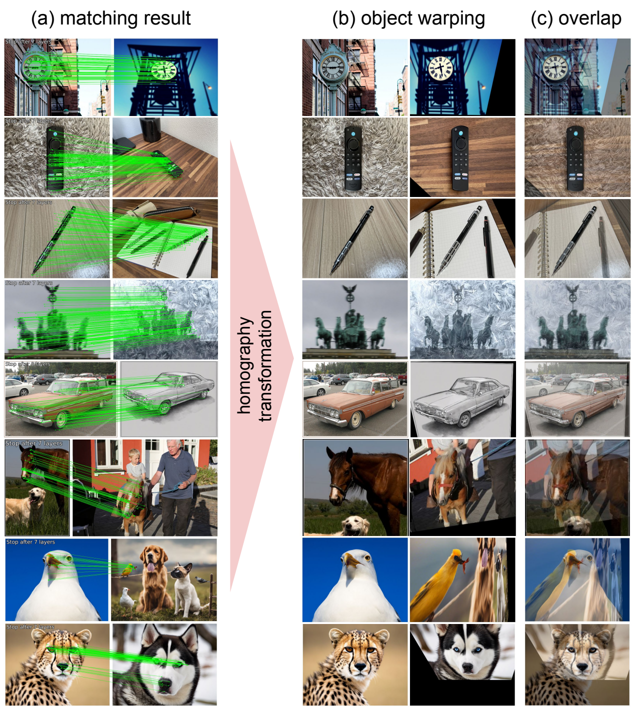

Figure A. Object warping by homography transformation for fine-grained and consistent matching using our method. (b) By warping the object in the right image to the object position in the left image and overlaying the two images, (c) it can be confirmed that the object positions are superimposed, showing consistent alignment.

B. Robust image matching details

With the same setup in Sec. 5.3 of the main text, we evaluate the effectiveness of the proposed method for LightGlue [37] and GlueStick [54] on the robustness to image corruptions and domain shift in image matching. First, we present an ablation study on the presence or absence of the Non-visual Object Pairing. Second, we consider the maximum angular error at 10° and 5° instead of 20°. The results of relative pose estimation with common corruptions [27] on one or both of the input images in the MegaDepth-1500 [79] show that the proposed method improves the robustness of LightGlue

Table A. Ablation study on the presence or absence of Non-visual Object Pairing in our method (Weighting Module: SEW, Non-visual Object Pairing: NOP). The pose accuracy (AUC) at the maximum angular error of 20° of the relative pose estimation from image pairs MegaDepth-1500 [79] under common corruptions [27]. The left side shows both images corrupted with the same type of corruption, while the right side shows only one of the input images corrupted and the other as a clean image. NOP selects similar objects between images, but the MegaDepth-1500 dataset assumes that the same object appears in both images. Therefore, there is little need to select the object to be matched, resulting in minimal accuracy differences with and without NOP.

Common Corruptions	AUC@20° (**corrupted** images)			AUC@20° (**clean and corrupted** images)		
	keypoint detector : SuperPoint			*keypoint detector : SuperPoint*		
	LG	LG+SEW	LG+SEW+NOP	LG	LG+SEW	LG+SEW+NOP
None (Clean)	**80.61**	78.42	78.91	**80.61**	78.42	78.91
Gaussian Noise	43.09	**53.27**	52.21	27.54	**41.01**	40.00
Shot Noise	43.41	**53.82**	52.81	32.10	**42.35**	41.41
Impulse Noise	44.98	**50.11**	48.94	35.95	**43.67**	42.65
Defocus Blur	32.69	**48.35**	47.37	18.25	**32.13**	31.25
Frosted Glass Blur	33.37	**47.88**	47.25	34.25	**44.80**	44.29
Motion Blur	42.12	**53.63**	52.95	50.07	**53.10**	52.46
Zoom Blur	24.40	**31.21**	30.39	34.67	**40.04**	39.36
Snow	**31.51**	30.33	30.40	56.24	**58.15**	58.00
Frost	**32.24**	31.71	31.83	62.20	**64.87**	64.74
Fog	70.99	**73.49**	72.41	76.96	**78.40**	77.37
Brightness	**75.48**	75.08	75.21	**77.14**	76.98	77.06
Contrast	**39.50**	38.47	38.77	43.09	**45.17**	45.11
Elastic Transform	54.78	**66.21**	65.69	64.93	**68.34**	67.92
Pixelate	67.94	**68.24**	68.12	66.80	**68.43**	68.37
JPEG Compression	29.54	**36.97**	36.47	47.41	**53.82**	53.45
Average	44.40	**50.58**	50.05	48.51	**54.08**	53.56

and GlueStick. In all experiments, we used the pretrained sparse matchers LightGlue [37], and GlueStick [54], and the dense matchers LoFTR [79] and EfficientLoFTR [89], downloaded from their official repositories.[4]

B.1. Abration on Non-visual Object Pairing

We conducted an ablation study to assess the impact of our Non-visual Object Pairing (NOP). Recall that the NOP assigns object pairs based on the highest cosine similarity of text embeddings (CLIP) derived from their class labels, thereby focusing the Grad-CAM-based weighting on semantically matched objects. Table B.1 presents the robustness evaluation against common corruptions applied either to both images or only one of the input images, com-Pairing results with and without the NOP. We find that the accuracy does not vary significantly between these settings. This implies that, on the MegaDepth-1500 dataset, where image pairs typically depict the same physical object—our Grad-CAM-based weighting already assigns appropriate focus without needing explicit object selection by the NOP. Hence, the NOP's influence is less pronounced on standard image matching datasets with identical objects. In contrast, its benefits become more evident for non-identical object matching tasks, where semantically similar but distinct objects require a more careful pairing mechanism.

[4]LightGlue: `https://github.com/cvg/LightGlue`
GlueStick: `https://github.com/cvg/GlueStick`
LoFTR: `https://github.com/zju3dv/LoFTR`
ELoFTR: `https://github.com/zju3dv/efficientloftr`

B.2. Robustness against common corruptions

In this experiment, common corruptions are added to both input images to demonstrate robustness against corruption. Table D shows the results of relative pose estimation for maximum angular error of 10° and 5°. As the results show, the proposed method makes the LightGlue and GlueStick robust against most types of common corruptions (roughly, 5% to 10% increase) with a slight decrease in clean accuracy. For the Noise category, LightGlue and GlueStick with our method even surpass dense matchers in both cases.

B.3. Robustness against environmental changes

In contrast to Appendix B.2, common corruptions are added to one of the image pairs to demonstrate robustness against domain shift between images. Table E shows the results of relative pose estimation for maximum angular error of 10° and 5°. As the results show, the proposed method is robust to LightGlue and GlueStick, and in particular, significantly improves the AUC of the former for all Noise, Blur, and Digital categories, and the latter for all Noise and Blur categories. For the Noise category, LightGlue and GlueStick with our method even exceed dense matchers in both cases.

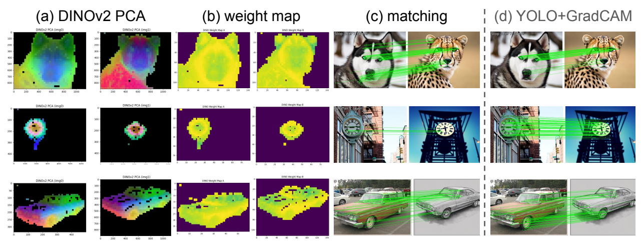

Figure B. Comparison of weighting. (a) Semantic visualization using PCA of DINOv2 features. (b) Corresponding weight maps used to reweight descriptors. (c) Matching results with DINOv2-based weights. (d) Matching results with YOLOv7 and Grad-CAM (baseline). The DINOv2-based weights do not necessarily improve the performance compared to the original configuration.

C. Other semantic models for weighting

In our main experiments, we adopted YOLOv7 [87] and Grad-CAM [73] as a fast and practical choice for generating semantic heatmaps. Here, we examine alternative semantic models and discuss their effectiveness as weighting sources for descriptors.

DINOv2 as an alternative. We tested DINOv2 [53], a vision foundation model, to generate semantic representations. First, we computed principal components of the DINOv2 features for each image to visualize the semantic regions. Figure B (a) shows the PCA-based visualization, which captures semantic regions to some extent. Then, we generated weight maps by normalizing the PCA scores, as shown in Fig. B (b). These weight maps were used to reweight keypoint descriptors, following the same procedure as our Grad-CAM-based weighting.

Effect on matching. Figure B (c) illustrates the matching results when the DINOv2-based weight maps were applied. Compared to the results using YOLOv7 and Grad-CAM (Fig. B (d)), the DINOv2-based weighting did not consistently improve the matching performance. In some cases, it even degraded the quality due to overly smooth attention over the image or misaligned semantic focus. These results indicate that strong semantic models do not always translate to better weighting for sparse feature matching. While DINOv2 performs well in recognition or segmentation tasks, its internal features may not align well with fine-grained keypoint matching needs.

D. Matching algorithm with Geo-Aware-SC

In this section, we explain how semantic correspondence is leveraged within the Geo-Aware-SC [98] framework to achieve feature matching. We introduce and compare two approaches: a Random Approach and a Heatmap-Based Approach. The first method samples pixel locations uniformly, while the second method uses a precomputed heatmap to prioritize semantically meaningful regions.

Table B. Comparison with a semantic correspondence method. We report AUC@20° of relative pose estimation using GeoAware-SC [98] and our method (SuperPoint + LightGlue + SEW) on image pairs under common corruptions. The left columns show results when both images are corrupted, while the right columns show results when only one image is corrupted (domain shift setup). Although both methods perform similarly on clean images, our method consistently outperforms GeoAware-SC under corrupted settings while being more than 10 times faster.

Common Corruptions	**corrupted** pairs		**clean / corrupted** pairs	
	GeoAware-SC	Ours	GeoAware-SC	Ours
None (clean)	55.28	**78.42**	55.28	**78.42**
gaussian_noise	29.17	**53.27**	33.57	**41.01**
shot_noise	33.82	**53.82**	36.49	**42.35**
impulse_noise	31.39	**50.11**	34.82	**43.67**
defocus_blur	42.32	**48.35**	26.51	**32.13**
glass_blur	39.15	**47.88**	39.83	**44.80**
motion_blur	39.97	**53.63**	39.36	**53.10**
zoom_blur	11.37	**21.31**	20.78	**40.04**
snow	18.44	**30.33**	35.60	**58.15**
frost	10.47	**31.71**	32.52	**64.87**
fog	48.02	**73.49**	51.31	**78.40**
brightness	49.40	**75.08**	53.48	**76.98**
contrast	34.55	**38.47**	29.58	**45.17**
elastic_transform	40.33	**66.21**	42.75	**68.34**
pixelate	48.20	**68.24**	48.92	**68.43**
jpeg_compression	33.06	**36.97**	42.61	**53.82**
Average	33.98	**50.82**	37.88	**54.08**
Time [ms/pair]	710.36	**68.43**	715.24	**68.37**

D.1. Random sampling approach

This approach uniformly selects a fixed number of points from the valid image region. In this random approach, each image is first resized to a fixed square resolution while preserving its aspect ratio. Any remaining space is padded to maintain the square shape and a binary mask is generated to indicate the valid (non-padded) region. Next, we apply bilinear interpolation to upsample the feature maps so that each pixel in the resized image can be associated with a distinct feature vector. From the valid region, we randomly select a fixed number of pixels for the source image and extract their feature descriptors, which are then normalized. The target image undergoes the same resizing and upsampling procedure, but instead of random selection, we use the entire valid region. We compute the cosine similarity between every descriptor of the source image and all descriptors of the target image. For each descriptor in the source image, the location in the target image with the highest similarity is identified as the match. Finally, these correspondences are visualized side-by-side.

D.2. Heatmap-based sampling approach

This approach leverages a precomputed semantic heatmap to guide the sampling process, ensuring that pixels in semantically important regions are more likely to be selected. The heatmap-based method follows a similar resizing and upsampling procedure, but it incorporates a precomputed heatmap to guide the selection of sample points in the source image. The heatmap is aligned to the same resolution as the resized image and normalized. Each valid pixel in the source image is associated with a heatmap value, which serves as a sampling probability. This probabilistic selection ensures that pixels in regions with higher semantic importance are more likely to be chosen. After selecting the sample points based on the heatmap, the descriptors are extracted and normalized just as in the random method. The target image also has its feature maps upsampled, and the descriptors are computed for its valid region. Each descriptor in the source image is then matched to the most similar descriptor in the target image, allowing important regions indicated by the heatmap to be prioritized during matching.

D.3. Comparison with Semantic Matching Methods

To further contextualize the effectiveness of our method, we compared it with GeoAware-SC [98], a representative semantic correspondence method. Although semantic matching techniques such as GeoAware-SC are typically designed for high-level alignment of semantically similar regions rather than fine-grained keypoint-level matching, they serve as a useful reference for evaluating robustness under challenging conditions.

The experimental setup followed that of Sec. 5.3, focusing on relative pose estimation under common corruptions. As presented in Tab. B, our method achieves significantly higher AUC@20° scores for both corrupted image pairs (50.82% vs. 33.98%) and clean-corrupted image pairs (54.08% vs. 37.88%) compared to GeoAware-SC. Moreover, the average inference time per image pair is an order of magnitude faster (68 ms vs. 710 ms), indicating practical advantages in efficiency.

E. Triangular Matching Consistency

We propose *Triangular Matching Consistency* (TMC), a new annotation-free metric for quantitatively evaluating non-identical object matching. Defining ground-truth correspondences between non-identical objects is inherently difficult, as fine-grained pixel-level matches are often ambiguous. TMC avoids this limitation by measuring the consistency between the following two matchings: Letting A, B, C be images of non-identical objects, we compute

- direct matching of (A, C)
- one-hop matching through (A, B) and (B, C),

Images A, C are expected to be very similar, and thus their direct correspondences can serve as a reliable reference. A non-identical object matching method is expected to approximate them even via the one-hop matching via semantically similar B. This design enables quantitative evaluation without annotations and naturally captures cross-object coherence. An overview of this procedure is shown in Fig. C. TMC reports three complementary aspects of performance: (i) error magnitude (RMSE), (ii) accuracy as the fraction of correct correspondences (PCK), and (iii) coverage of the baseline correspondences (Recall). We next describe the procedure step by step.

E.1. Direct and one-hop matchings

Let $\mathcal{A}$ be the matching algorithm to evaluate. Given images A, C, this returns a matching,

$$\mathcal{A}(A, C) = \left\{ (\boldsymbol{p}_i^A, \boldsymbol{p}_i^C) \right\}_{i=1}^{M_1}, \tag{13}$$

where $\boldsymbol{p}_i^A, \boldsymbol{p}_i^C \in [0,1]^2$ are the matched position vectors (i.e., normalized coordinates) in A and C, respectively, and M_1 is the number of matched pairs. Note that each position vector (say, $\boldsymbol{p}_i^A$) in the matching has an associated descriptor (say, $\boldsymbol{d}_i^A$). We denote the one-hop matching via B by

$$\mathcal{A}(A, C; B) = \left\{ (\widehat{\boldsymbol{p}}_i^A, \widehat{\boldsymbol{p}}_i^C) \right\}_{i=1}^{M_2}, \tag{14}$$

where M_2 is the number of matched pairs, and this is obtained by combining

$$\mathcal{A}(A, B) = \left\{ (\boldsymbol{q}_i^A, \boldsymbol{q}_i^B) \right\}_{i=1}^{N_1}, \tag{15}$$

$$\mathcal{A}(B, C) = \left\{ (\boldsymbol{r}_i^B, \boldsymbol{r}_i^C) \right\}_{i=1}^{N_2}. \tag{16}$$

Particularly, the Hungarian algorithm [50] is applied to pair position vectors in $\boldsymbol{q}_i^B$ and $\boldsymbol{r}_i^B$, through which the one-hop matching is found. For example, if the Hungarian algorithm pairs $(\boldsymbol{q}_i^B, \boldsymbol{r}_j^B)$, then we have $(\boldsymbol{q}_i^A, \boldsymbol{r}_j^C) =: (\widehat{\boldsymbol{p}}_i^A, \widehat{\boldsymbol{p}}_i^C)$. The implementation details are as follows.

Outlier removal. The outlier removal by RANSAC-based geometric verification is common in the literature to obtain reliable inlier matches [20, 35, 37, 79]. In TCM, the removal step is inserted before obtaining Eq. (13). Particularly, given the initial matching between A and C, geometric verification is applied via RANSAC to estimate either a homography or a fundamental matrix, depending on the scene geometry, and only inliers are retained. However, the outlier removal is not applied to $\mathcal{A}(A, B)$ and $\mathcal{A}(B, C)$ because matching between non-identical objects can be regarded as "outliers"; removing them could discard semantically meaningful pairs.

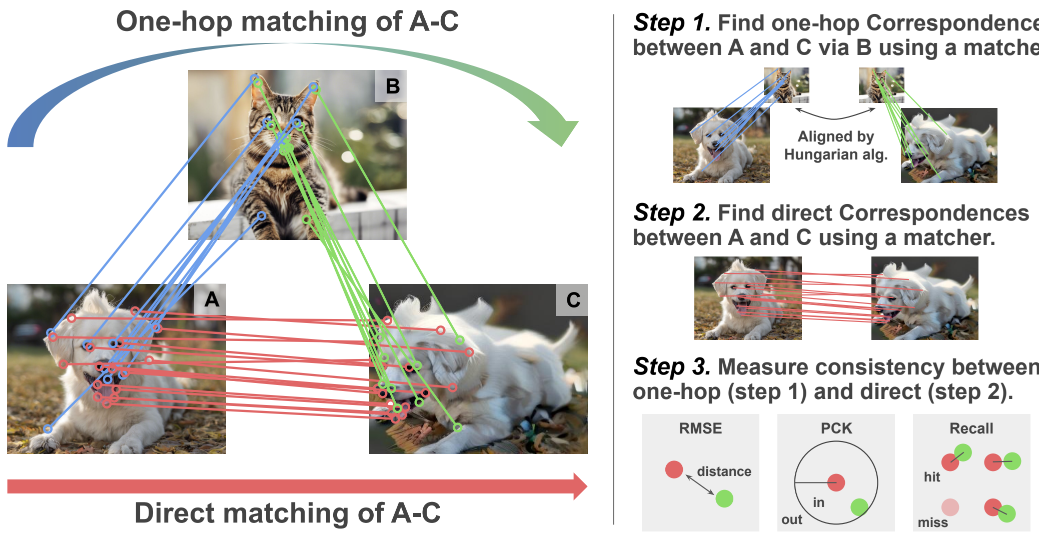
One-hop matching of A-C
B
A
C
Direct matching of A-C
Step 1. Find one-hop Correspondences between A and C via B using a matcher.
Aligned by Hungarian alg.
Step 2. Find direct Correspondences between A and C using a matcher.
Step 3. Measure consistency between one-hop (step 1) and direct (step 2).
RMSE
distance
PCK
in
out
Recall
hit
miss

Figure C. **Triangular Matching Consistency (TMC) overview.** Given two images A and C of the same object with different viewpoints (where reliable direct correspondences can be defined), and another semantically similar but non-identical object B, we compare the direct correspondences between (A, C) with the one-hop correspondences $A\leftrightarrow B\leftrightarrow C$. A method with high TMC produces one-hop matches that closely agree with the direct baseline, thereby enabling annotation-free quantitative evaluation of non-identical object matching.

Proximity used in Hungarian algorithm. The image B has two sets of position vectors, $\{\boldsymbol{q}_i^B\}_{i=1}^{N_1}$ from $\mathcal{A}(A, B)$ and $\{\boldsymbol{r}_i^B\}_{i=1}^{N_2}$ from $\mathcal{A}(B, C)$. The Hungarian algorithm determines the alignment of $M_2 := \min\{N_1, N_2\}$ vectors between them by minimizing the cost. The cost is defined as follows. First, the geometric and descriptor discrepancies are

$$\delta_{ij}^p = \|\boldsymbol{q}_i^B - \boldsymbol{r}_j^B\|, \tag{17}$$

$$\delta_{ij}^d = \|\boldsymbol{e}_i^B - \boldsymbol{f}_j^B\|, \tag{18}$$

where $\boldsymbol{e}_i^B, \boldsymbol{f}_j^B$ are the descriptors associated with $\boldsymbol{q}_i^B, \boldsymbol{r}_j^B$, respectively. The cost is then defined by

$$c_{ij} = \lambda_p\, \delta_{ij}^p + \lambda_d\, \delta_{ij}^d, \tag{19}$$

where $\lambda_p, \lambda_d \geq 0$ are hyper-parameters. In our experiments, we set $\lambda_p = 1.0$ and $\lambda_d = 0.3$.

E.2. Measuring matching consistency

Using the direct and one-hop matching, we now measure the matching consistency by three complementary metrics. The consistency is measured using the position vectors of the image C. Recall that we have $\{\boldsymbol{p}_i^C\}_{i=1}^{M_1}$ from the direct matching and $\{\widehat{\boldsymbol{p}}_i^C\}_{i=1}^{M_2}$ from the one-hop matching. Let

$$\Delta_{ij} = \|\boldsymbol{p}_i^C - \widehat{\boldsymbol{p}}_i^C\|_2, \quad i = 1, \dots, M_1,\ j = 1, \dots, M_2, \tag{20}$$

and we introduce three metrics of consistency as follows.

RMSE (Root Mean Squared Error). RMSE measures the magnitude of geometric error between the direct and one-hop matching.

$$\mathrm{RMSE} = \sqrt{\frac{1}{M_2}\sum_{j=1}^{M_2} \min_i \Delta_{ij}^2}. \tag{21}$$

Noting that the position vectors are normalized, its value ranges in $[0, \sqrt{2}]$. Smaller values indicate better consistency.

PCK (Percentage of Correct Keypoints). PCK [22] measures the proportion of accurate correspondences within a tolerance $\tau > 0$ in the normalized plane:

$$\mathrm{PCK}(\tau) = \frac{1}{M_2}\sum_{j=1}^{M_2} \mathbf{1}[\min_i \Delta_{ij} \leq \tau], \tag{22}$$

where $\mathbf{1}[\,\cdot\,]$ is the indicator function. High values at small τ indicate many highly accurate pairs. In our experiments, τ is set to 0.01, 0.05, and 0.1. To reduce sensitivity to any single threshold. PCK takes values in $[0, 1]$, and larger values indicate better consistency.

Recall. Recall measures the coverage of the direct matching $A\leftrightarrow C$ correspondences by the one-hop correspondences $A\leftrightarrow B\leftrightarrow C$.

$$\mathrm{Recall}(\tau) = \frac{1}{M_1}\sum_{i=1}^{M_1} \mathbf{1}[\min_j \Delta_{ij} \leq \tau], \tag{23}$$

Table C. Triangular Matching Consistency (TMC) results for the dog and cat categories in the Animal Faces dataset [11]. Sparse matchers (LightGlue (LG), GlueStick (GS)) with SuperPoint, and dense matchers (LoFTR, Efficient LoFTR) are shown for reference. Metrics include RMSE (lower is better), PCK, and Recall at thresholds of 0.01, 0.05, and 0.10 of the image size.

Class	Matching Method		RMSE↓	PCK↑			Recall↑		
				0.01	0.05	0.10	0.01	0.05	0.10
Dog	Sparse (+SuperPoint)	LG	0.20	48.5	60.2	66.1	51.3	62.4	67.8
		LG+SEW	**0.15**	**69.4**	**77.1**	**82.0**	**69.4**	**76.7**	**81.9**
		GS	0.22	45.0	55.8	60.3	47.9	57.2	61.8
		GS+SEW	0.17	61.8	71.3	77.4	60.4	70.1	76.2
	Dense (no detector)	LoFTR	0.26	50.4	59.3	63.7	48.7	56.1	60.9
		ELoFTR	0.20	59.6	68.8	72.5	58.4	67.5	71.2
Cat	Sparse (+SuperPoint)	LG	0.13	56.8	72.1	78.4	59.7	74.5	80.1
		LG+SEW	**0.08**	**71.2**	**84.6**	**90.4**	**71.2**	**84.4**	**90.2**
		GS	0.15	52.7	67.8	74.0	54.9	70.3	75.8
		GS+SEW	0.10	66.5	80.3	86.9	65.1	79.1	86.3
	Dense (no detector)	LoFTR	0.17	55.3	70.2	76.1	53.8	69.0	74.8
		ELoFTR	0.12	63.9	78.1	83.4	62.7	77.0	82.6

In our experiments, τ is set to 0.01, 0.05, and 0.1. Recall takes values in $[0, 1]$, and larger values indicate better coverage.

F. Additional TMC results

This section provides additional TMC results that complement the wild-category evaluation presented in Sec. 5.2. We report per-class performance for the dog and cat categories of the Animal Faces dataset [11]. The evaluation protocol follows the same triplet construction and metrics described in the main paper: error magnitude (RMSE), accuracy (PCK), and coverage (Recall) at distance thresholds of 0.01, 0.05, and 0.10 of the image size.

Table C shows, as in the wild-category results, our weighting consistently improves all matchers across all metrics. Sparse matchers combined with our module achieve substantially lower geometric error and higher accuracy than their baselines, and they also outperform dense detector-free methods in most settings. These class-wise results further confirm that our method enhances the robustness and consistency of non-identical object matching.

G. Limitations

Figure D shows failure cases caused by (a) heatmaps not covering a whole part of objects and (b) great structural differences between objects. In the former case, the importance scores take large only on particular parts of the objects. Further, the large importance scores on the irrelevant object disrupt the matching. This may be resolved using better visual explanation methods [1, 6, 44, 77, 94, 102] or vision foundation model [53]. For the latter case, objects with great structural differences cannot be matched as there is little correspondence between the objects, and eventually, regions of similar colors or textures are matched. To address these cases, one may introduce the vision and language model, such as the segment anything model [33, 61] to complement the visual information with text and part-level mask information.

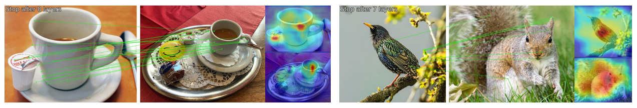
(a) heatmap not covering entire objects (b) structure discrepancy

Figure D. Failure cases: (a) high heatmap scores cover only a small region of objects, (b) the objects have clearly different structures.

H. Related Work Details

Keypoint detectors. Classically, keypoint detection was done using hand-crafted local features such as SIFT [40] and ORB [67]. Recently, learning-based methods have been developed [16, 19, 21, 25, 57, 86]. Particularly, SuperPoint [16] and DISK [84] are two popular examples. The former is based on convolutional neural networks, and the latter is based on reinforcement learning. There is also a fully differentiable and lightweight method with a minimal model configuration [21]. Unsupervised keypoint detection based on a text-to-image diffusion model [25] also exists.

Feature matchers. Feature matchers [90, 96] find a fine-grained correspondence between images. The feature matchers are categorized into two types based on the density of matching points: sparse matchers and dense matchers. The former runs faster, while the latter gives dense and high accuracy. The focus of our study lies in sparse matchers. A seminal sparse matcher, SuperGlue [69], addresses the partial assignment problem for matching by integrating the Transformer attention mechanism [85] with optimal transport [56]. The state-of-the-art sparse matcher, LightGlue [37], improves SuperGlue by introducing a hierarchical matching structure, achieving high matching accuracy and real-time processing simultaneously. GlueStick [54] utilizes line segments for structural features. In contrast to sparse matchers, dense matchers [8, 18, 30, 38, 42, 64, 79, 95, 101, 103] pursue matching accuracy rather than speed. Dense matchers input an image and infer matching using all pixels as keypoints. Therefore, it does not require a keypoint detector, so it is also called a detector-free model. LoFTR [79] proposed the first transformer-based method for dense matching, which achieves highly dense matching. Recent studies [41, 52, 100] such as Efficient LoFTR [89], enabling fast and accurate matching by token aggregation, and ASTR [95], which focuses on the role of each pixel.

Semantic correspondence models. Semantic Correspondence [3, 9, 22, 23, 32, 48, 58, 74] is a task of identifying a semantically similar point in the target image from the given point in the source image. Specifically, it enables the correspondence of parts, such as the mouth, nose, or right eye of dogs, between the source and target images. The training of a model is conducted using a specialized dataset, such as SPair-71k [48] and PF-PASCAL [22], which employ sparsely annotated part-level correspondences as supervisory data. Recently, GeoAware-SC [98] has been proposed to integrate geometric and semantic information by leveraging pre-trained features from Stable Diffusion [66] and DINOv2 [53], boosting semantic correspondence accuracy on major benchmarks [22, 48]. Additionally, various models [10, 28, 55, 75, 81, 97] have been proposed, including one that uses reverse diffusion feature maps [45] and unsupervised methods that learn semantic correspondence without annotated data [26]. It is also worth noting that semantic correlation models run significantly slower than sparse matching methods because feature extraction depends on large generative models, such as Stable Diffusion [66]. For example, it runs roughly 10 times slower in Fig. 2.

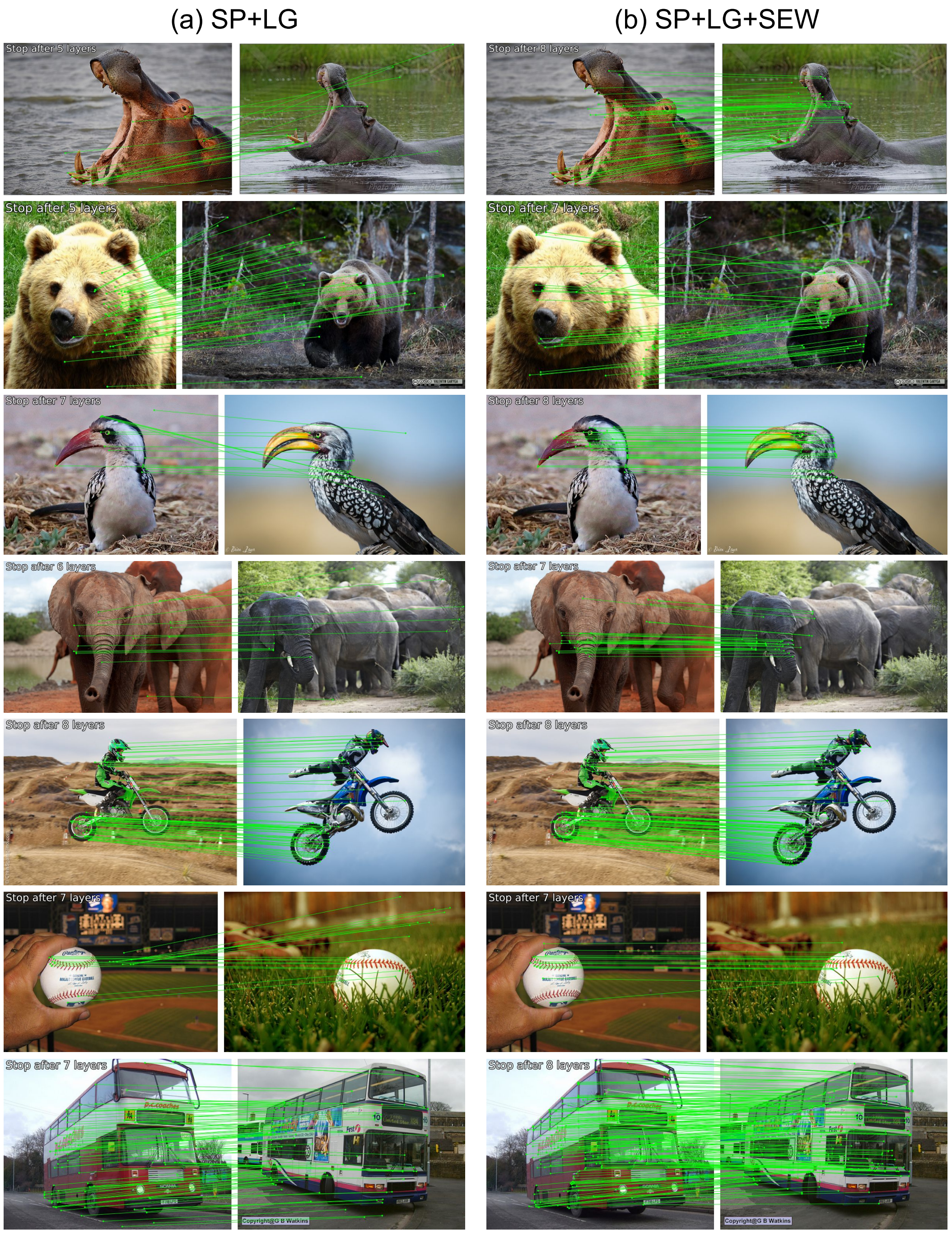

Figure E. Matching results in the same class case. Image pairs show objects of the same class in ImageNet [15] or COCO [36] datasets. (a) A combination of SuperPoint (SP) [16] and LightGlue (LG) [37] only finds a small number of correspondences, and many of them are incorrect. (b) The proposed method SEW significantly improves the matching and performs dense and consistent correspondences.

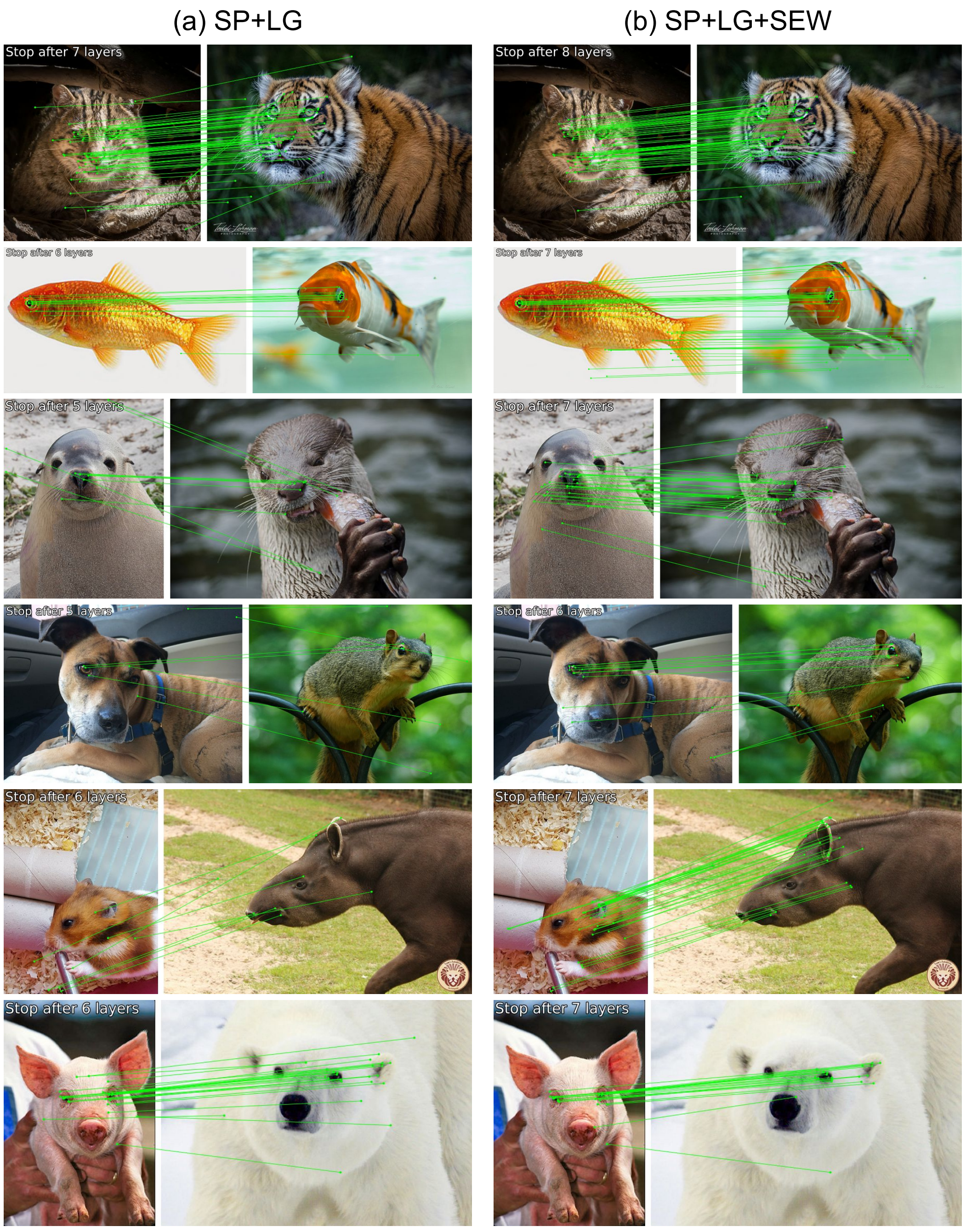

Figure F. Matching results in the class discrepancy case. Image pairs show objects with class discrepancies in ImageNet [15] or COCO [36] datasets. (a) A combination of SuperPoint (SP) [16] and LightGlue (LG) [37] finds many inconsistent and scattered matches, and it is unclear which is the correct correspondence. (b) The proposed method SEW finds almost all matches for the corresponding parts of the animal, which largely improves the matching to consistency.

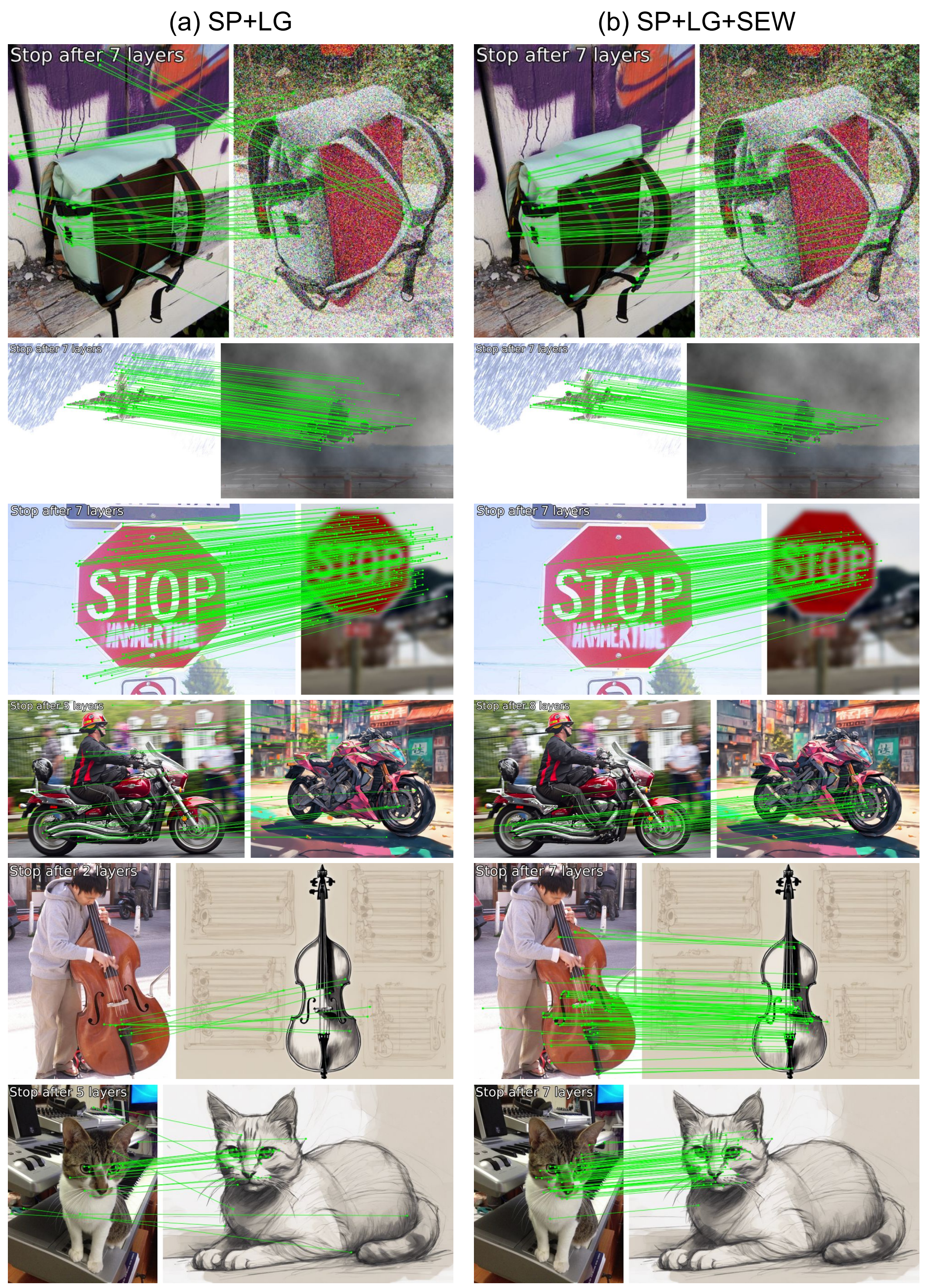

Figure G. Matching results in the domain shift case. Image pairs for matching are ImageNet [15] or COCO [36] datasets with common corruptions [27], and illustrations or sketches of a motorcycle, a violin, and a cat are generated by the Stable Diffusion [66]. (a) A combination of SuperPoint (SP) [16] and LightGlue (LG) [37] finds correct matching but also finds mismatches with many non-correspondence areas. (b) The proposed method SEW reduces these mismatches and further improves to denser matching between objects.

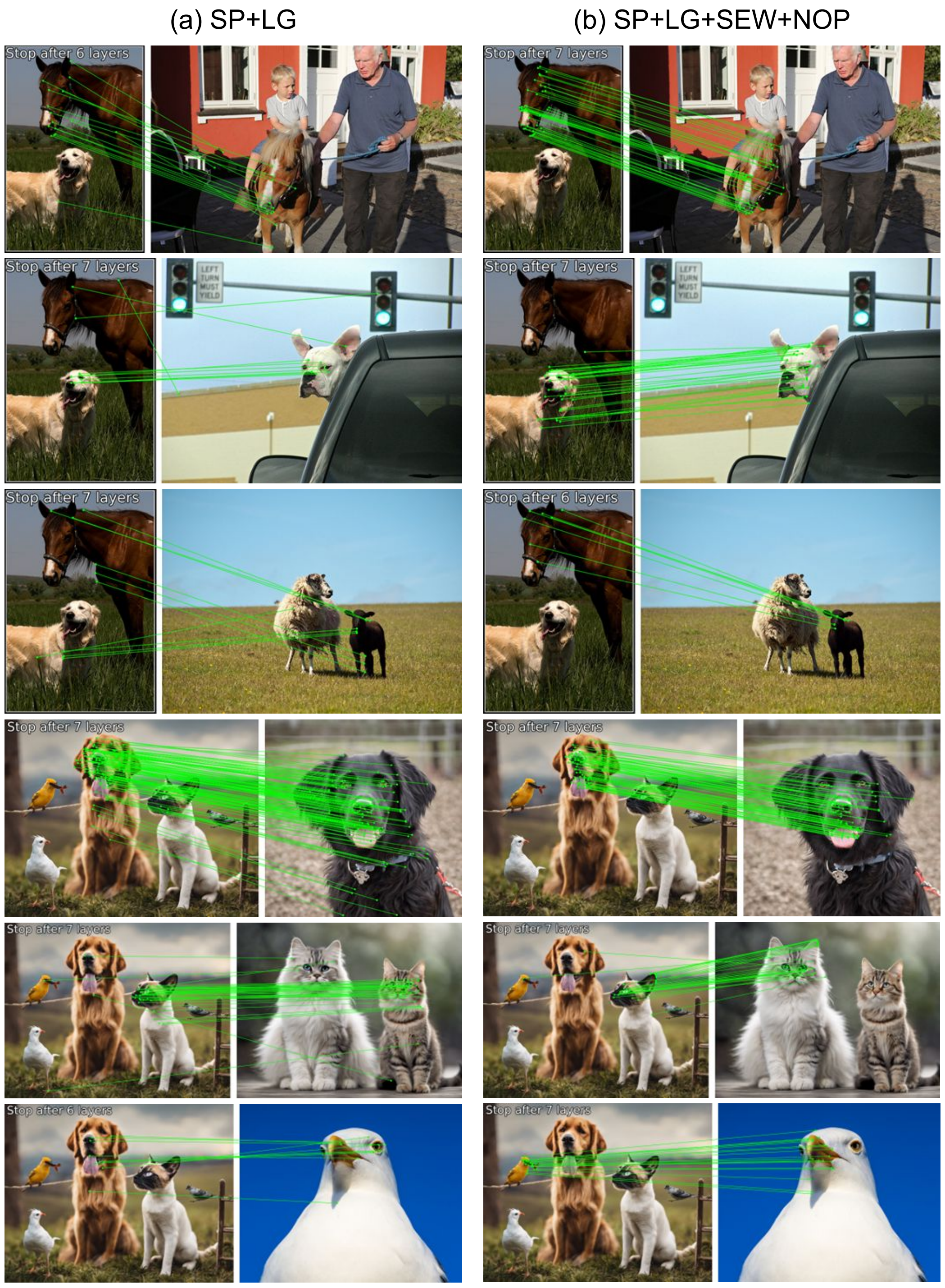

Figure H. Matching results in complex scenes containing multiple objects. (a) A combination of SuperPoint (SP) [16] and LightGlue (LG) [37] causes correspondences to scatter across multiple objects. (b) The proposed Non-visual Object Pairing (NOP) effectively selects semantically similar object pairs and provides condensed and consistent matching between the relevant parts of the objects.

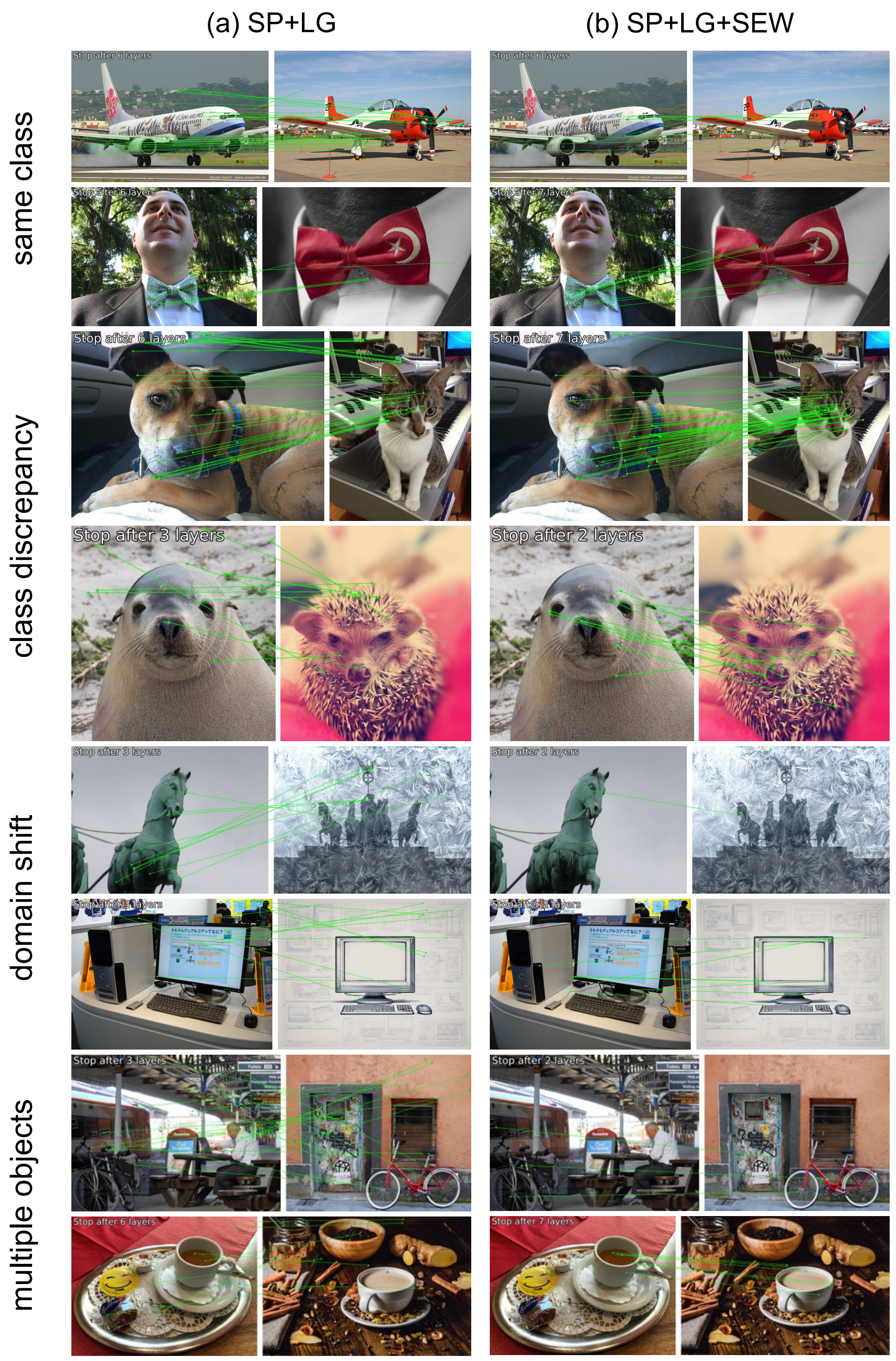

Figure I. Failure cases of non-identical object matching. (a) A combination of SuperPoint (SP) [16] and LightGlue (LG) [37] fails in matching. (b) The proposed method SEW improves several matching but still leaves many corresponding areas unmatched.

Table D. The pose accuracy (AUC) at the maximum angular error of 10° and 5° of the relative pose estimation from image pairs MegaDepth-1500 [79] under common corruptions. Both images are corrupted with the same type of corruption. Our method largely improves LightGlue (LG) [37] and GlueStick (GS) [54] for most categories and the average AUC. The results of a dense matchers, LoFTR [79] and Efficient LoFTR (ELoFTR) [89], are shown for reference. Our method improves the robustness of sparse matchers with a marginal decrease in clean accuracy.

Common Corruptions	AUC@10° with pairs of **corrupted** images					
	keypoint detector : SuperPoint				*dense matcher*	
	LG	LG+SEW	GS	GS+SEW	LoFTR	ELoFTR
None (Clean)	**67.89**	66.98	**64.11**	61.94	68.88	72.18
Gaussian Noise	27.44	**37.28**	28.64	**35.11**	20.07	20.32
Shot Noise	27.07	**39.79**	27.81	**33.42**	24.27	23.89
Impulse Noise	28.70	**37.25**	29.10	**34.44**	22.27	23.75
Defocus Blur	18.58	**24.51**	30.12	**33.78**	43.27	40.33
Frosted Glass Blur	17.24	**27.61**	29.16	**31.75**	39.21	30.43
Motion Blur	27.08	**34.85**	27.98	**35.46**	40.68	38.01
Zoom Blur	12.50	**20.23**	12.64	**20.72**	12.01	13.66
Snow	**18.35**	12.56	**12.46**	10.49	18.94	24.27
Frost	**20.86**	18.75	**17.64**	15.48	11.15	15.67
Fog	56.32	**62.62**	55.14	**57.24**	53.05	59.85
Brightness	61.10	**63.95**	**58.75**	56.51	61.88	67.22
Contrast	**24.33**	23.10	24.31	**27.45**	38.57	44.78
Elastic Transform	36.76	**47.02**	41.53	**45.52**	41.27	42.79
Pixelate	51.60	**60.66**	**47.60**	47.32	62.55	64.85
JPEG Compression	16.60	**27.73**	22.35	**27.64**	43.49	43.50
Average	29.64	**35.86**	31.02	**34.16**	35.51	36.89
	AUC@5° with pairs of **corrupted** images					
None (Clean)	**50.43**	49.71	**45.92**	43.11	52.52	56.38
Gaussian Noise	14.70	**24.44**	14.63	**21.53**	9.85	9.57
Shot Noise	13.71	**23.44**	13.44	**18.06**	12.21	11.78
Impulse Noise	15.65	**21.25**	16.90	**20.64**	10.92	11.77
Defocus Blur	8.22	**11.69**	16.59	**20.64**	27.77	25.67
Frosted Glass Blur	7.17	**18.09**	14.58	**16.12**	24.59	16.17
Motion Blur	14.72	**20.72**	15.26	**22.34**	25.37	27.82
Zoom Blur	5.38	**8.82**	6.02	**14.16**	5.17	8.42
Snow	**9.48**	8.32	**5.92**	3.87	9.91	13.79
Frost	**11.92**	11.29	**9.92**	7.72	6.09	8.75
Fog	39.01	**43.43**	35.88	**37.58**	36.94	48.00
Brightness	43.60	**47.98**	**41.86**	39.60	45.13	50.73
Contrast	11.69	**13.93**	12.75	**15.29**	23.13	29.26
Elastic Transform	20.43	**29.64**	24.29	**27.28**	24.40	25.35
Pixelate	32.97	**41.82**	**29.86**	29.66	46.40	48.41
JPEG Compression	8.00	**15.94**	9.71	**14.79**	27.77	27.45
Average	17.11	**22.72**	17.84	**20.62**	22.38	24.20

Table E. The pose accuracy (AUC) at the maximum angular error of 10° and 5° of the relative pose estimation from image pairs MegaDepth-1500 [79] under common corruptions. Only one of the input images is corrupted and the other is a clean image. Our method largely improves LightGlue (LG) [37] and GlueStick (GS) [54] for most categories and the average AUC. The results of dense matchers, LoFTR [79], and Efficient LoFTR (ELoFTR) [89] are shown for reference. Our method improves the robustness of sparse matchers with a marginal decrease in clean accuracy.

Common Corruptions	AUC@10° with pairs of **clean and corrupted** images					
	keypoint detector : SuperPoint				*dense matcher*	
	LG	LG+SEW	GS	GS+SEW	LoFTR	ELoFTR
None (Clean)	**67.89**	66.98	**64.11**	61.94	68.88	72.18
Gaussian Noise	16.63	**28.34**	28.57	**32.42**	24.17	27.01
Shot Noise	19.97	**32.50**	26.23	**29.92**	26.58	29.11
Impulse Noise	23.56	**34.74**	33.12	**36.26**	25.67	28.67
Defocus Blur	8.29	**22.43**	12.03	**18.17**	36.85	31.36
Frosted Glass Blur	19.00	**32.78**	26.39	**29.66**	40.40	34.59
Motion Blur	33.29	**39.69**	27.88	**38.30**	39.41	37.15
Zoom Blur	20.38	**26.23**	16.76	**25.67**	21.60	21.63
Snow	39.74	**41.07**	30.74	**33.00**	35.91	43.33
Frost	47.61	**51.78**	38.80	**41.62**	32.97	38.00
Fog	62.39	**64.08**	**61.11**	60.80	60.74	65.26
Brightness	**63.40**	61.75	**62.69**	60.04	65.64	69.24
Contrast	27.81	**33.12**	25.86	**28.95**	30.27	40.35
Elastic Transform	47.62	**56.66**	52.17	**55.08**	53.61	56.20
Pixelate	50.14	**56.68**	**50.33**	48.63	63.14	66.22
JPEG Compression	29.92	**40.31**	**40.93**	40.61	53.58	57.23
Average	33.98	**41.48**	35.57	**38.61**	40.70	43.02
	AUC@5° with pairs of **clean and corrupted** images					
None (Clean)	**50.43**	49.71	**45.92**	43.11	52.52	56.38
Gaussian Noise	8.42	**20.10**	16.51	**20.52**	16.68	18.54
Shot Noise	10.30	**20.52**	16.14	**19.31**	18.14	19.72
Impulse Noise	12.88	**24.11**	21.41	**25.35**	17.75	19.55
Defocus Blur	3.22	**17.92**	5.95	**15.89**	24.73	21.55
Frosted Glass Blur	8.56	**22.42**	15.54	**18.79**	26.87	22.61
Motion Blur	18.95	**26.22**	13.94	**24.36**	25.92	28.77
Zoom Blur	9.81	**19.27**	8.95	**17.68**	15.03	17.16
Snow	22.94	**26.90**	17.27	**20.56**	23.94	31.62
Frost	31.58	**37.80**	24.13	**27.95**	23.07	29.96
Fog	44.49	**45.00**	**42.65**	41.24	44.04	51.15
Brightness	**46.14**	45.24	**43.91**	41.63	48.85	52.85
Contrast	14.98	**17.15**	14.46	**17.53**	21.09	27.79
Elastic Transform	29.34	**39.29**	32.27	**35.11**	36.41	38.80
Pixelate	31.51	**38.02**	**31.82**	30.10	46.26	49.92
JPEG Compression	15.86	**27.55**	**22.24**	21.87	37.35	40.35
Average	20.60	**28.50**	21.81	**25.19**	28.41	31.36