Supplementary Material for Correspondence Transformers with Asymmetric Feature Learning and Matching Flow Super-Resolution

Yixuan Sun[1], Dongyang Zhao[2], Zhangyue Yin[2], Yiwen Huang[2],
Tao Gui[2], Wenqiang Zhang[1,2] and Weifeng Ge[2,†]
[1]Academy of Engineering & Technology, Fudan University, Shanghai, China
[2]School of Computer Science, Fudan University, Shanghai, China
wfge@fudan.edu.cn

1. Further Analysis of ACTR

Motivation for the baseline model design: In ACTR, we use iBOT [17] pre-trained on ImageNet [4] as our feature backbone, and propose asymmetric feature learning and matching flow superresolution to achieve accurate semantic matching. Since the iBOT feature backbone [17] is a very strong backbone, we need to design baseline methods that are comparable with ACTR. We replace the asymmetric feature learning module with the commonly used symmetric one [7, 15], and replace the matching flow superresolution with a bilinear flow upsampler. To compare ACTR with the baseline models fairly, we adjust the hyperparameters in the baseline models to ensure that the amount of parameters in the baseline models is almost the same as that in ACTR. As shown in Table 4 (in the main paper), experimental results on SPair-71k [11] demonstrate that asymmetric feature learning and matching flow superresolution are vital for ACTR to achieve impressive results.

Table 1. Comparison among ACTR and other image matching methods, such as COTR [7] and GMFlow [15] on SPair-71k [12]. N/A stands for not converge. We use the publicly available codes to conduct training on SPair-71k.

Methods	Backbone	Matching Head	Param(M)	PCK@0.1
ACTR-S	iBOT-S	Biased attention	44.2	55.8
COTR	ResNet-50	Cross attention	18.4	45.0
COTR	iBOT-S	Cross attention	43.4	51.6
GMFlow	CNN	Cross attention	4.68	N/A
GMFlow	iBOT-S	Cross attention	49.3	53.7

Differences among ACTR and other image matching methods: ACTR is designed to solve the semantic matching problem, which aims to match semantics across objects or scenes with great variations in appearances and layouts. While for image matching methods, they need to get dense matching results that match objects or scenes in different views. In image matching, local feature descriptors are much more important than high-level semantics, so most image-matching methods conduct the matching process on feature maps with high resolution. ACTR exploits low-resolution features with rich semantic information to conduct asymmetric feature learning and uses matching flow superresolution to distinguish subtle local differences. While other image-matching methods conduct image matching based on low-level feature descriptors with symmetric feature learning structures.

We directly apply image matching methods, such as COTR [7] and GMFlow [15], to conduct semantic matching. We use the same training method as ACTR to train them on the SPair-71k dataset. From Table 1, we can find that the training of GMFlow does not converge and the COTR gets 45.0%. Since the amounts of learnable parameters in GMFlow and COTR are relatively small, they are promising to be adapted for semantic matching tasks. We also implemented COTR and GMFlow with iBOT-S backbone, we report the result as 51.6% and 53.7%. Especially to point out that attention blocks of COTR are used to build up our Baseline.

2. Further Quantitative Results for ACTR

2.1. Class Level Evaluation Results on SPair-71k

Table 2 shows ACTR with $256 \times 256/512 \times 512$ input resolutions surpasses all SOTA methods based on CNN feature backbones as well as CATs [3], TransforMatcher [8] and VAT [6] in iBOT-B feature backbones clearly. Note that our model performs well on difficult classes {bike, chair, pottle plant, and sheep} that previous works [3, 6, 10, 11, 16] often fail. Compared with the previous state-of-the-art method TransforMatcher [16], with the same input resolution of 256, our method outperforms it by {9.2%, 9.7%, 4.7%, 14.2%} on PCK@$\alpha_{bbox} = 0.1$ for these classes.

†: Corresponding Authors

Table 2. Per-class level quantitative evaluation results on SPair-71k [12] benchmark, ‡ stands for the method implemented with iBOT-B backbone same with ACTR, the best results are in bold.

Methods	aero.	bike	bird	boat	bott.	bus	car	cat	chai	cow	dog	hors.	mbik.	pers.	plan.	shee.	trai.	tv	all
SCOT [9]	34.9	20.7	63.8	21.1	43.5	27.3	21.3	63.1	20	42.9	42.5	31.1	29.8	35	27.7	24.4	48.4	40.8	35.6
DHPF [13]	38.4	23.8	68.3	18.9	42.6	27.9	20.1	61.6	22	46.9	46.1	33.5	27.6	40.1	27.6	28.1	49.5	46.5	37.3
CATs [3]	52	34.7	72.2	34.3	49.9	57.5	43.6	66.5	24.4	63.2	56.5	52	42.6	41.7	43	33.6	72.6	58	49.9
MMNet [16]	55.9	37	65	35.4	50	63.9	45.7	62.8	28.7	65	54.7	51.6	38.5	34.6	41.7	36.3	77.7	62.5	50.4
TransforMatcher [8]	59.2	39.3	73.0	41.2	52.5	**66.3**	55.4	67.1	26.1	67.1	56.6	53.2	45.0	39.9	42.1	35.3	75.2	68.6	53.7
CATs‡ [3]	56.7	41.3	77.8	35.0	54.8	59.8	45.2	69.9	31.4	63.7	57.6	62.5	46.7	49.1	43.2	43.5	76.4	64.1	55.2
TransforMatcher‡ [8]	57.1	47.4	**83.5**	42.3	**56.8**	57.0	55.4	**75.3**	34.5	66.1	64.2	60.2	52.8	55.2	40.5	46.0	75.1	65.8	57.9
ACTR	**65.1**	**48.5**	82.3	**50.4**	55.9	65.3	**63.1**	72.8	**35.8**	**74.1**	**70.3**	**68.9**	**58.6**	**57.1**	**46.8**	**49.5**	**84.4**	**73.3**	**62.1**
VAT [6]	56.5	37.8	73.0	38.7	50.9	58.2	40.8	70.5	20.4	72.6	61.1	57.8	45.6	48.1	52.4	39.7	77.7	**71.4**	54.2
VAT‡	58.6	47.8	83.2	45.6	52.4	67.1	61.4	73.4	30.2	76.5	67.7	66.9	48.0	53.3	46.6	44.3	84.6	60.7	59.0
ACTR$_h$	**64.9**	**54.8**	**87.6**	**49.2**	**55.7**	**74.4**	**66.5**	**80.7**	**35.3**	**82.1**	**75.2**	**71.9**	**54.0**	**62.4**	**54.9**	**53.5**	**88.7**	71.0	**65.4**

Compared with the TransforMatcher [16] with the same iBOT-B backbone as ACTR, our method also gets improvements by $\{1.1\%,\ 1.3\%,\ 6.3\%,\ 3.5\%\}$. Our ACTR$_h$ also shows an overall improvement compared with previous works and their iBOT-B feature backbone extension.

2.2. Parameters in Different Modules of ACTR

Here we provide statistics of learnable parameters for each module of ACTR. The feature backbone and asymmetric feature learning module overtake more than 98% of total parameters. While the correlation calculation (including generating matching flow) and matching flow superresolution only contain 1.8M and 0.76M learnable parameters respectively. Since the matching flow superresolution module can upscale a flow with low computation cost, ACTR can establish accurate correspondence in higher resolutions.

Table 3. Parameters of ACTR Components. Here we provide both the amount of learnable parameters and the proportion of parameters for a module in the entire model.

Structure	Param(M)	Percentage
Backbone	85.0	49.3%
Asymmetric Feature Learning	84.44	49.2%
Correlation Calculation	1.8	1.1%
Matching Flow Super-Resolution	0.76	0.4%
Total	172.8	100%

2.3. Evaluation of ACTR on MAE & DINO

We replaced iBOT ImageNet 1K pre-trained weights with that of MAE [5], DINO [2]. Results show that MAE and DINO also perform well with 61.2% and 54.0% of PCK@0.1 respectively. It indicates that self-supervised learning techniques with masked image modeling can help to learn semantic correspondences since they focus on local image patch modeling. We compared the differences among IBOT [17], DINO [2] and MAE [5]. DINO performs local-global contrastive learning that can provide scale-variant robust features. While MAE utilizes masked image modeling that can extract consistent features despite severe occlusion. iBOT exploits both contrastive learning and masked image modeling and thus gets better performance.

2.4. Further Ablations on ACTR

We provide more ablation results for micro designs in the matching flow superresolution module on Table 4. Experiments are conducted on ACTR (ImageNet 1K, resolution 256×256). First, we replaced the multi-path superresolution with a single-path superresolution by only exploiting the coarse matching flow only in the last cross-attention block. It can be found that the results drop from 62.1% to 61.0%. It indicates that the matching flows in different branches can provide complementary information for accurate semantic matching. Second, we checked whether transformer blocks with different window attention are necessary for flow superresolution. When the 4×4 window attention branch was removed, the performance dropped by 1.5%. When the 8×8 window attention branch was removed, the performance dropped by 1.3%. These results show that more diversities in transformer blocks will improve the matching accuracy. Finally, we removed the matching flow superresolution module and upscaled the coarse matching flow through a bilinear interpolation sampler, and found that the performance dropped to 59.0%. These results indicate that all the designs in the matching flow superresolution module are important to get good results.

Besides, we clarified the ablation for 'w/o dual window flow refinement' in ablation table for main paper. In main paper, we removed the 4×4 path and the performance dropped for 1.5%. Here we conduct experiment with the 8×8 path removed, the performance dropped for 1.3%. The result shows that the both path contribute to final perfor-

Table 4. Further ablations on micro designs in the matching flow superresolution module.

Methods	SPair-71K $\alpha_{bbox} = 0.1$
ACTR	62.1
w/o multi-path superresolution	61.0 (1.1↓)
w/o 4×4 branch	60.6 (1.5↓)
w/o 8×8 branch	60.8 (1.3↓)
w/o flow super-resolution	59.0 (3.1↓)
Baseline	57.7 (4.4↓)

mance. We also conduct experiment on correlation map calculation, we compared the multi-head attention with inner-production and single-head attention. Compared with our design, their performance drops by 0.6% and 3.7%. We believe this is because multi-head attention allows the model to focus on different aspects of information [14].

3. Additional Visualization

We provide more visualization results in Figure 1-11. Figure 1-6 visualize the matched key points, and Figure 7 and 8 provide an more comprehensive comparison by wrapping images through TPS [1]. We also provide more visualization results for outputs produced by different superresolution paths in Figure 9-11. We further clarify that the design of multi-path fusion can take advantage of each path and achieve better performance.

References

[1] Fred L. Bookstein. Principal warps: Thin-plate splines and the decomposition of deformations. *IEEE Transactions on pattern analysis and machine intelligence*, 11(6):567–585, 1989. 3, 10, 11

[2] Mathilde Caron, Hugo Touvron, Ishan Misra, Hervé Jégou, Julien Mairal, Piotr Bojanowski, and Armand Joulin. Emerging properties in self-supervised vision transformers. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 9650–9660, 2021. 2

[3] Seokju Cho, Sunghwan Hong, Sangryul Jeon, Yunsung Lee, Kwanghoon Sohn, and Seungryong Kim. Cats: Cost aggregation transformers for visual correspondence. *Advances in Neural Information Processing Systems*, 34:9011–9023, 2021. 1, 2, 4, 5, 6, 7, 8, 9, 10, 11

[4] Jia Deng, Wei Dong, Richard Socher, Li-Jia Li, Kai Li, and Li Fei-Fei. Imagenet: A large-scale hierarchical image database. In *2009 IEEE conference on computer vision and pattern recognition*, pages 248–255, 2009. 1

[5] Kaiming He, Xinlei Chen, Saining Xie, Yanghao Li, Piotr Dollár, and Ross Girshick. Masked autoencoders are scalable vision learners. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 16000–16009, 2022. 2

[6] Sunghwan Hong, Seokju Cho, Jisu Nam, Stephen Lin, and Seungryong Kim. Cost aggregation with 4d convolutional swin transformer for few-shot segmentation. In *European Conference on Computer Vision*, pages 108–126. Springer, 2022. 1, 2

[7] Wei Jiang, Eduard Trulls, Jan Hosang, Andrea Tagliasacchi, and Kwang Moo Yi. Cotr: Correspondence transformer for matching across images. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 6207–6217, 2021. 1

[8] Seungwook Kim, Juhong Min, and Minsu Cho. Transformatcher: Match-to-match attention for semantic correspondence. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8697–8707, 2022. 1, 2

[9] Shuda Li, Kai Han, Theo W Costain, Henry Howard-Jenkins, and Victor Prisacariu. Correspondence networks with adaptive neighbourhood consensus. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 10196–10205, 2020. 2

[10] Yanbin Liu, Linchao Zhu, Makoto Yamada, and Yi Yang. Semantic correspondence as an optimal transport problem. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 4463–4472, 2020. 1, 4, 5, 6, 7, 8, 9, 10, 11

[11] Juhong Min, Jongmin Lee, Jean Ponce, and Minsu Cho. Hyperpixel flow: Semantic correspondence with multi-layer neural features. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3395–3404, 2019. 1

[12] Juhong Min, Jongmin Lee, Jean Ponce, and Minsu Cho. Spair-71k: A large-scale benchmark for semantic correspondence, 2019. 1, 2

[13] Juhong Min, Jongmin Lee, Jean Ponce, and Minsu Cho. Learning to compose hypercolumns for visual correspondence. In *European Conference on Computer Vision*, pages 346–363. Springer, 2020. 2

[14] Jesse Vig. A multiscale visualization of attention in the transformer model. In *Proceedings of the 57th Annual Meeting of the Association for Computational Linguistics: System Demonstrations*, pages 37–42, 2019. 3

[15] Haofei Xu, Jing Zhang, Jianfei Cai, Hamid Rezatofighi, and Dacheng Tao. Gmflow: Learning optical flow via global matching. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 8121–8130, 2022. 1

[16] Dongyang Zhao, Ziyang Song, Zhenghao Ji, Gangming Zhao, Weifeng Ge, and Yizhou Yu. Multi-scale matching networks for semantic correspondence. In *Proceedings of the IEEE/CVF International Conference on Computer Vision*, pages 3354–3364, 2021. 1, 2, 4, 5, 6, 7, 8, 9, 10, 11

[17] Jinghao Zhou, Chen Wei, Huiyu Wang, Wei Shen, Cihang Xie, Alan Yuille, and Tao Kong. ibot: Image bert pre-training with online tokenizer. *ICLR*, 2022. 1, 2

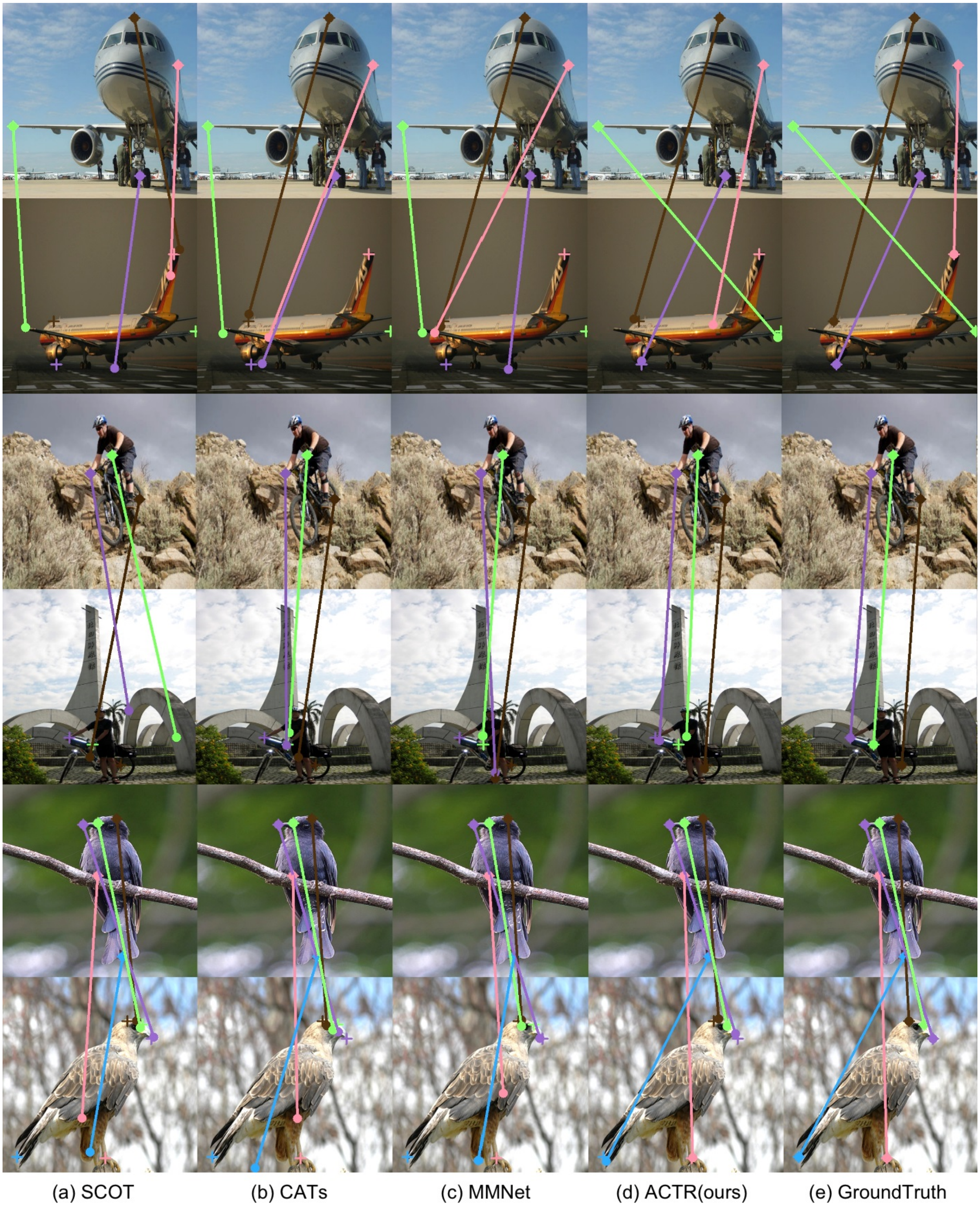

Figure 1. **Visual comparison of matched key points.** From left to right: (a) SCOT [10], (b) CATs [3], (c) MMNet [16], (d) ours ACTR and (e) the ground truth. Source and target images are in odd and even rows respectively. Crosses denote destination key points on target images.

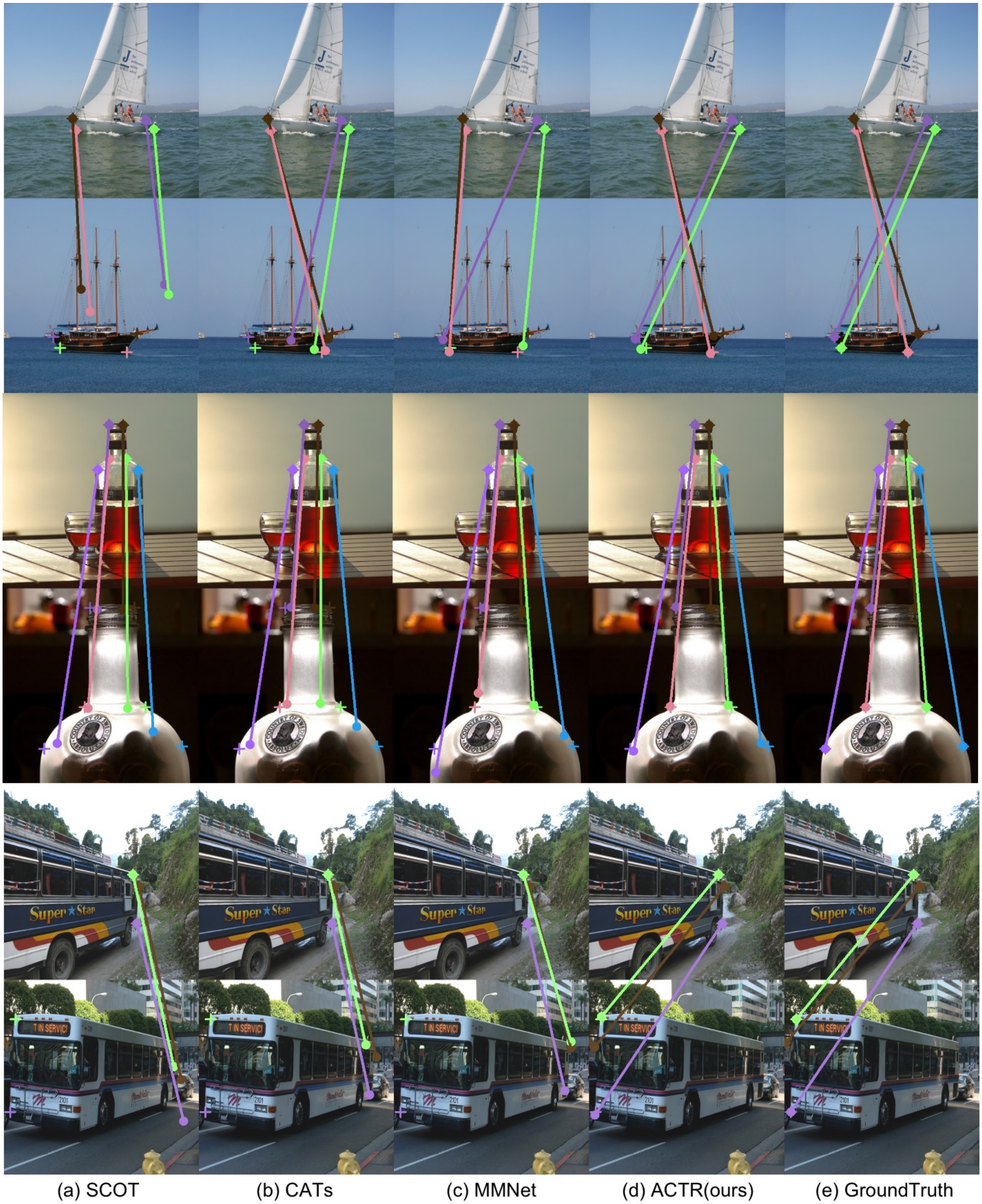

Figure 2. **Visual comparison of matched key points.** From left to right: (a) SCOT [10], (b) CATs [3], (c) MMNet [16], (d) ours ACTR and (e) the ground truth. Source and target images are in odd and even rows respectively. Crosses denote destination key points on target images.

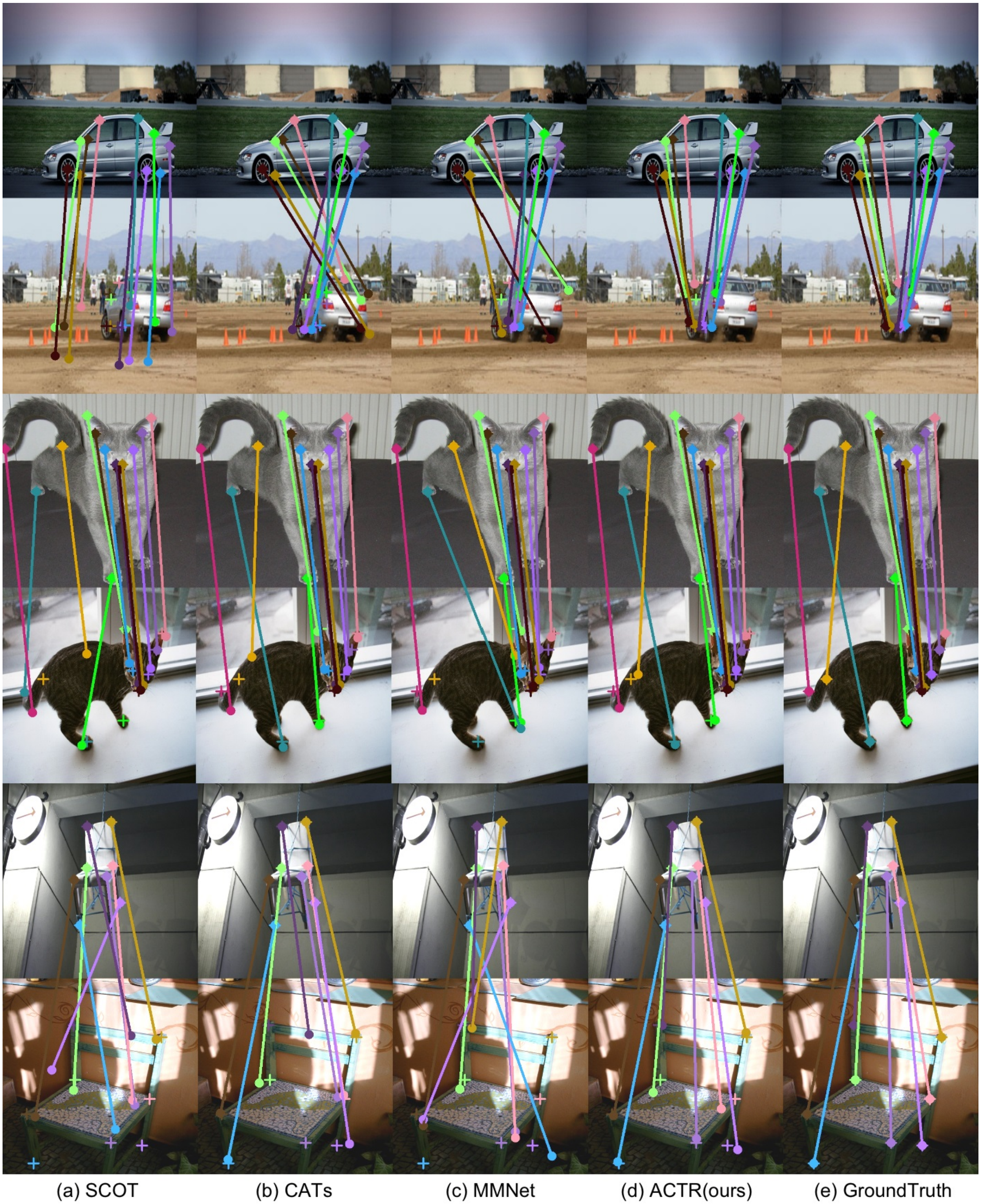

Figure 3. **Visual comparison of matched key points.** From left to right: (a) SCOT [10], (b) CATs [3], (c) MMNet [16], (d) ours ACTR and (e) the ground truth. Source and target images are in odd and even rows respectively. Crosses denote destination key points on target images.

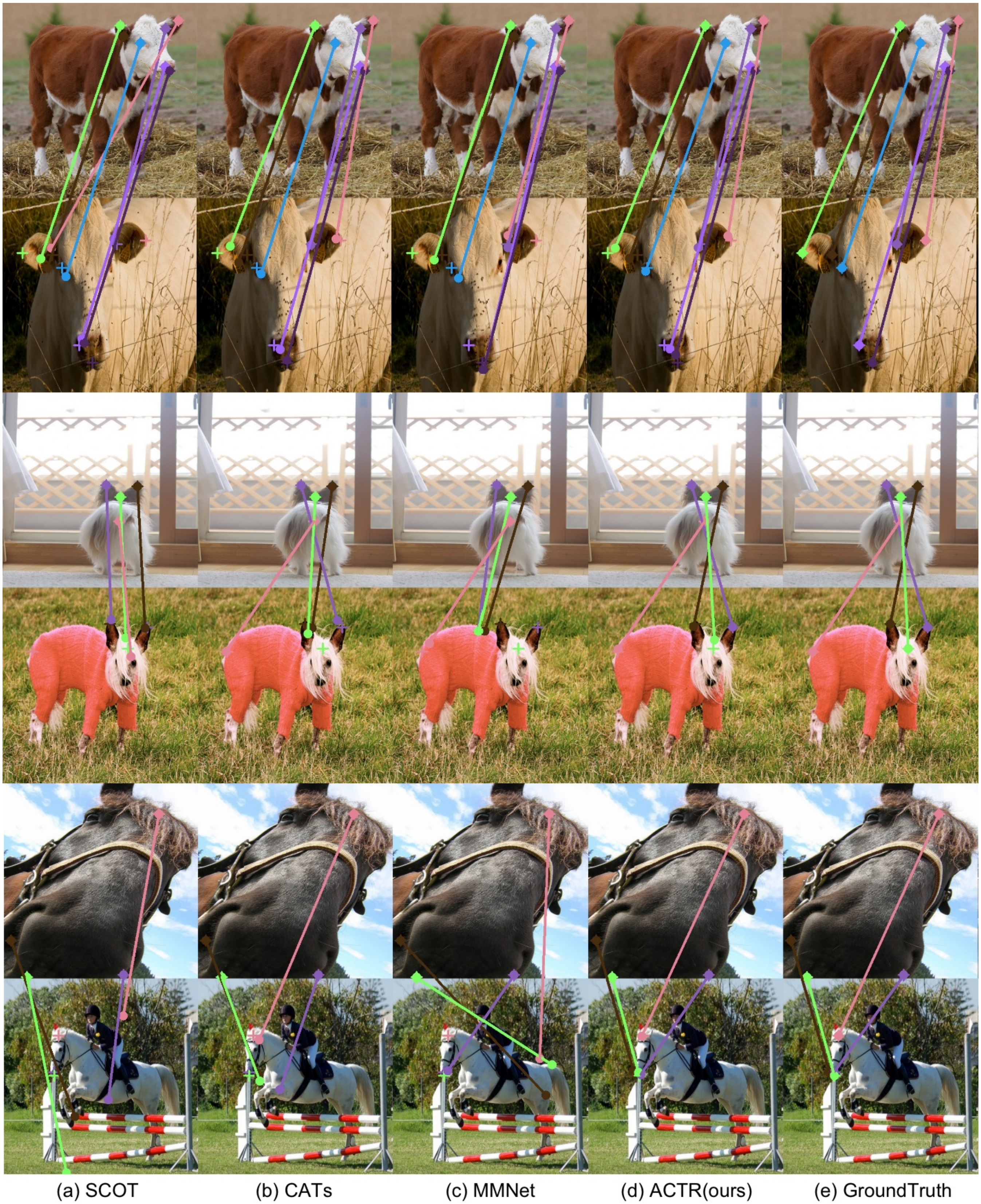

Figure 4. **Visual comparison of matched key points.** From left to right: (a) SCOT [10], (b) CATs [3], (c) MMNet [16], (d) ours ACTR and (e) the ground truth. Source and target images are in odd and even rows respectively. Crosses denote destination key points on target images.

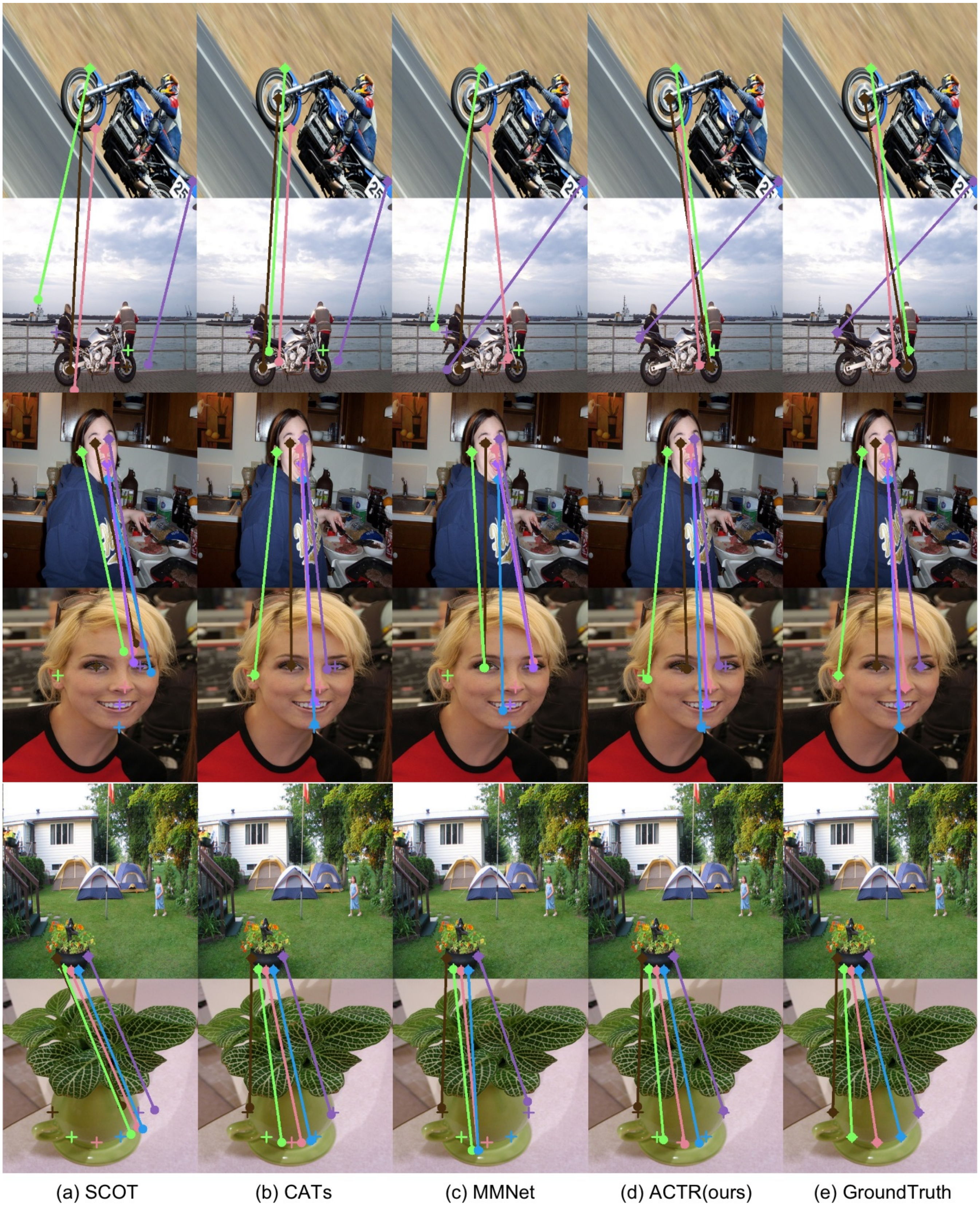

Figure 5. **Visual comparison of matched key points.** From left to right: (a) SCOT [10], (b) CATs [3], (c) MMNet [16], (d) ours ACTR and (e) the ground truth. Source and target images are in odd and even rows respectively. Crosses denote destination key points on target images.

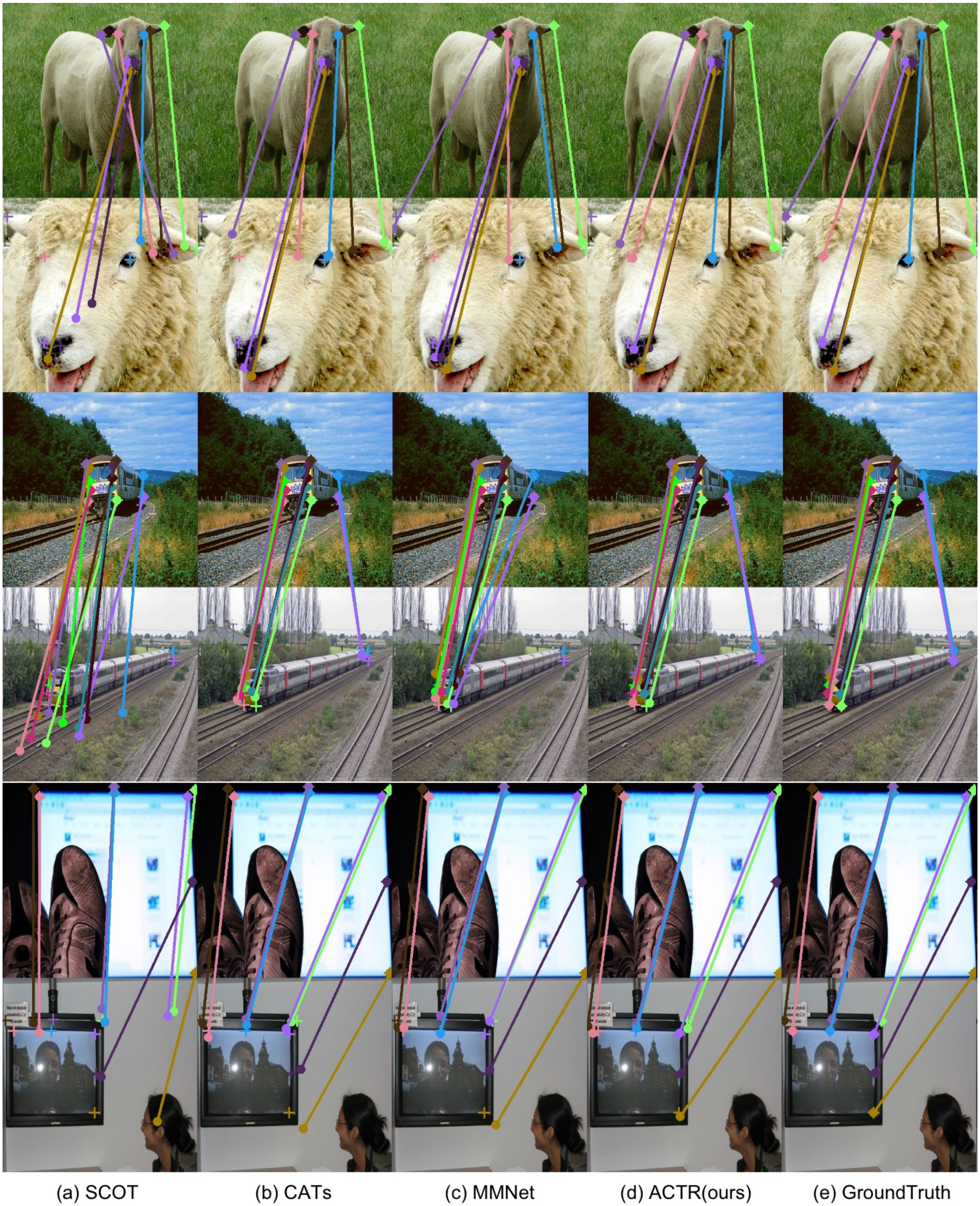

Figure 6. **Visual comparison of matched key points.** From left to right: (a) SCOT [10], (b) CATs [3], (c) MMNet [16], (d) ours ACTR and (e) the ground truth. Source and target images are in odd and even rows respectively. Crosses denote destination key points on target images.

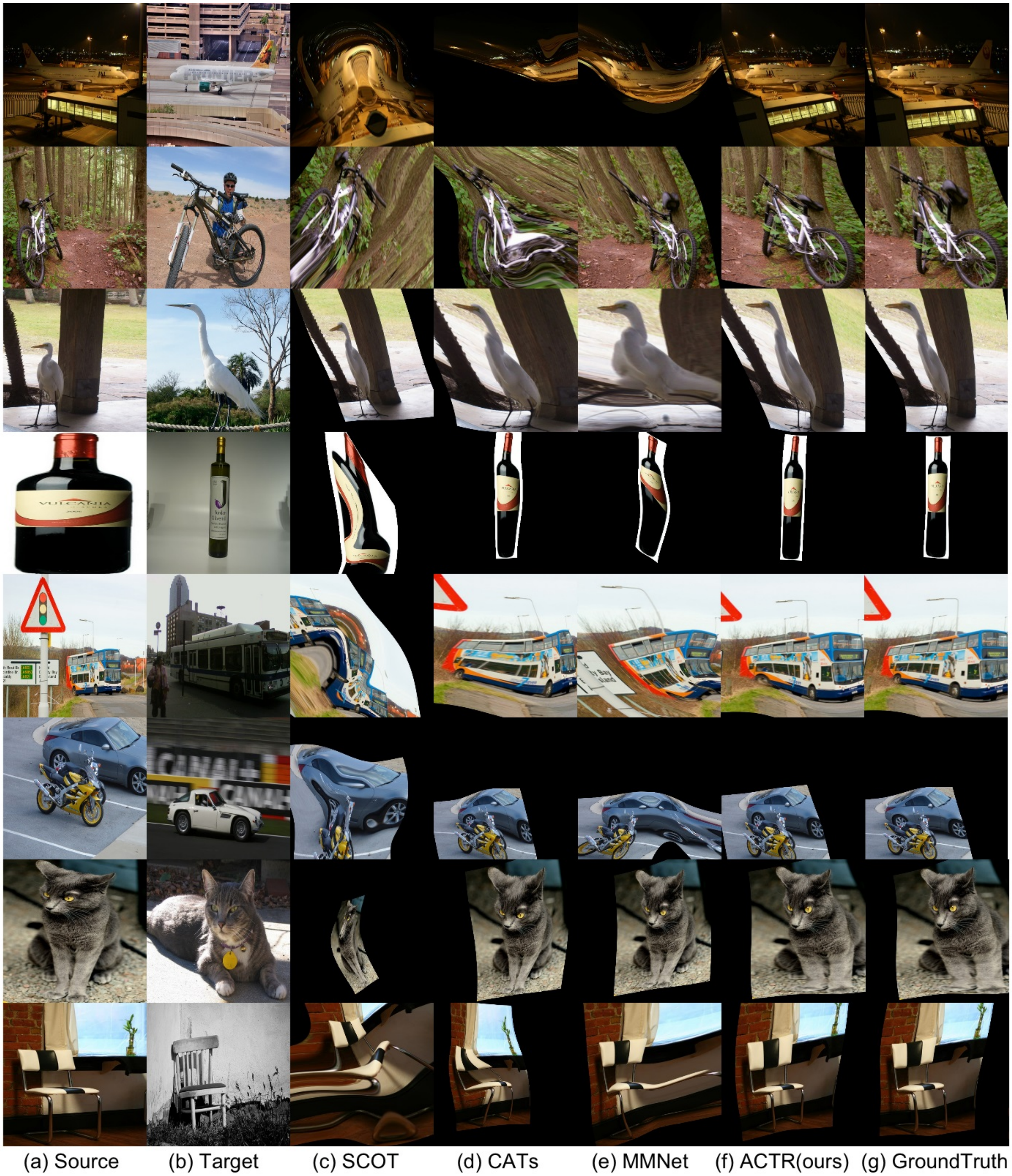

Figure 7. Dense visual correspondence generated by state-of-the-art algorithms, including SCOT [10], CATs [3], MMNet [16] and our ACTR. Images are warped with predicted key points using thin-plate splines algorithm [1].

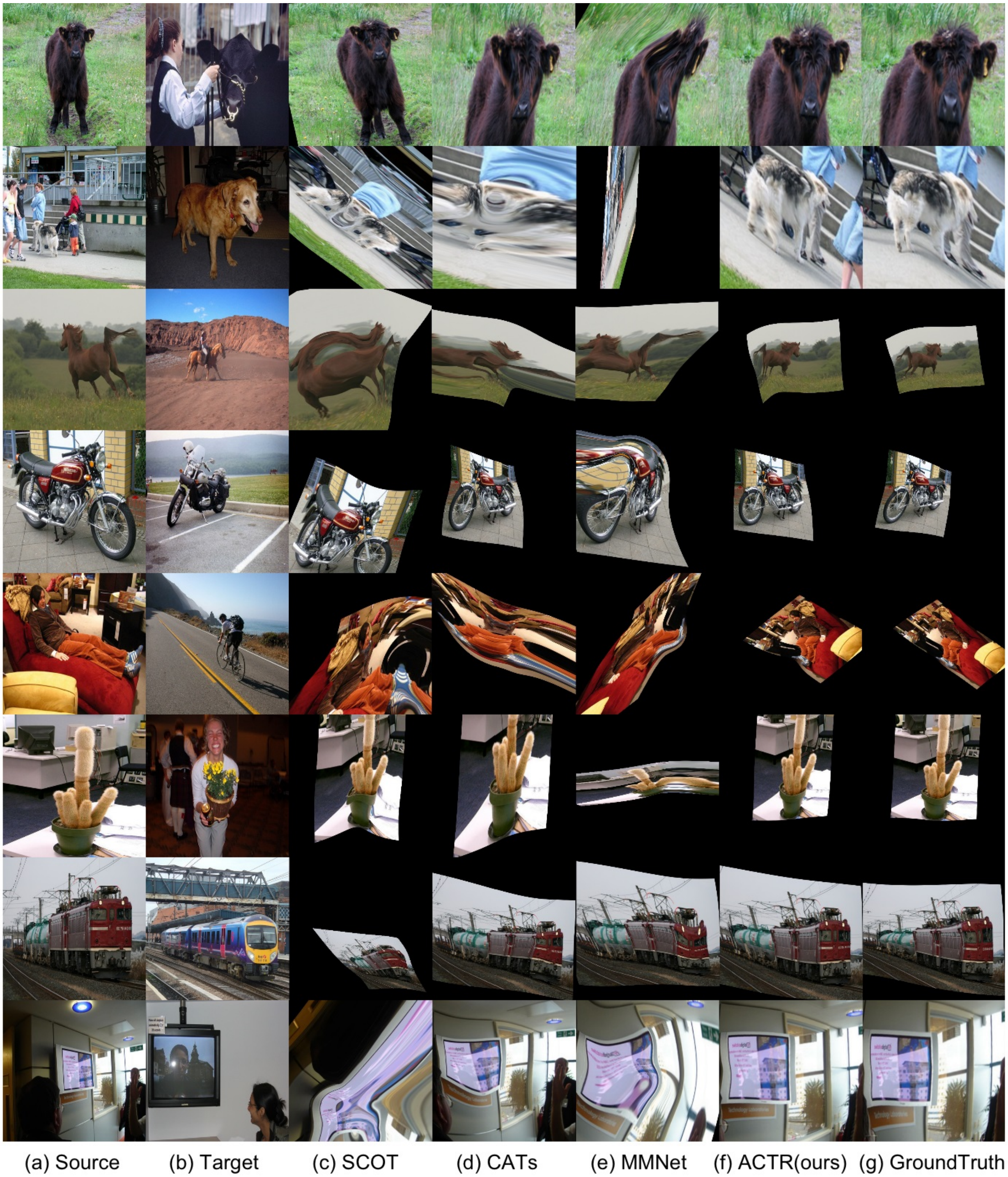

Figure 8. Dense visual correspondence generated by state-of-the-art algorithms, including SCOT [10], CATs [3], MMNet [16] and our ACTR. Images are warped with predicted key points using thin-plate splines algorithm [1].

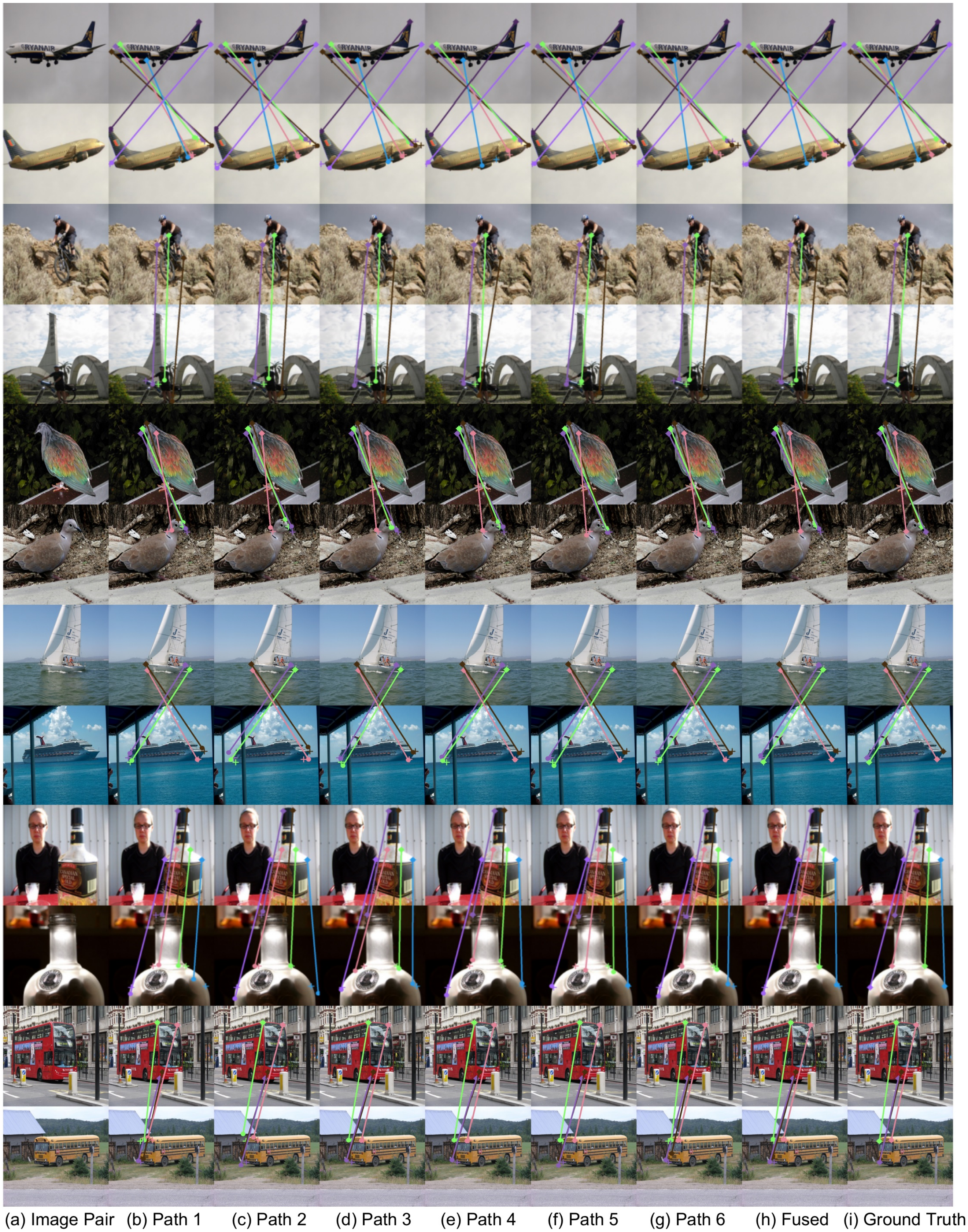

Figure 9. **Matching results of different paths.** From left to right, raw image pairs (a), matching results from path 1-6 (b-g), the output after multi-path fusion (h), and ground truth (i) are given.

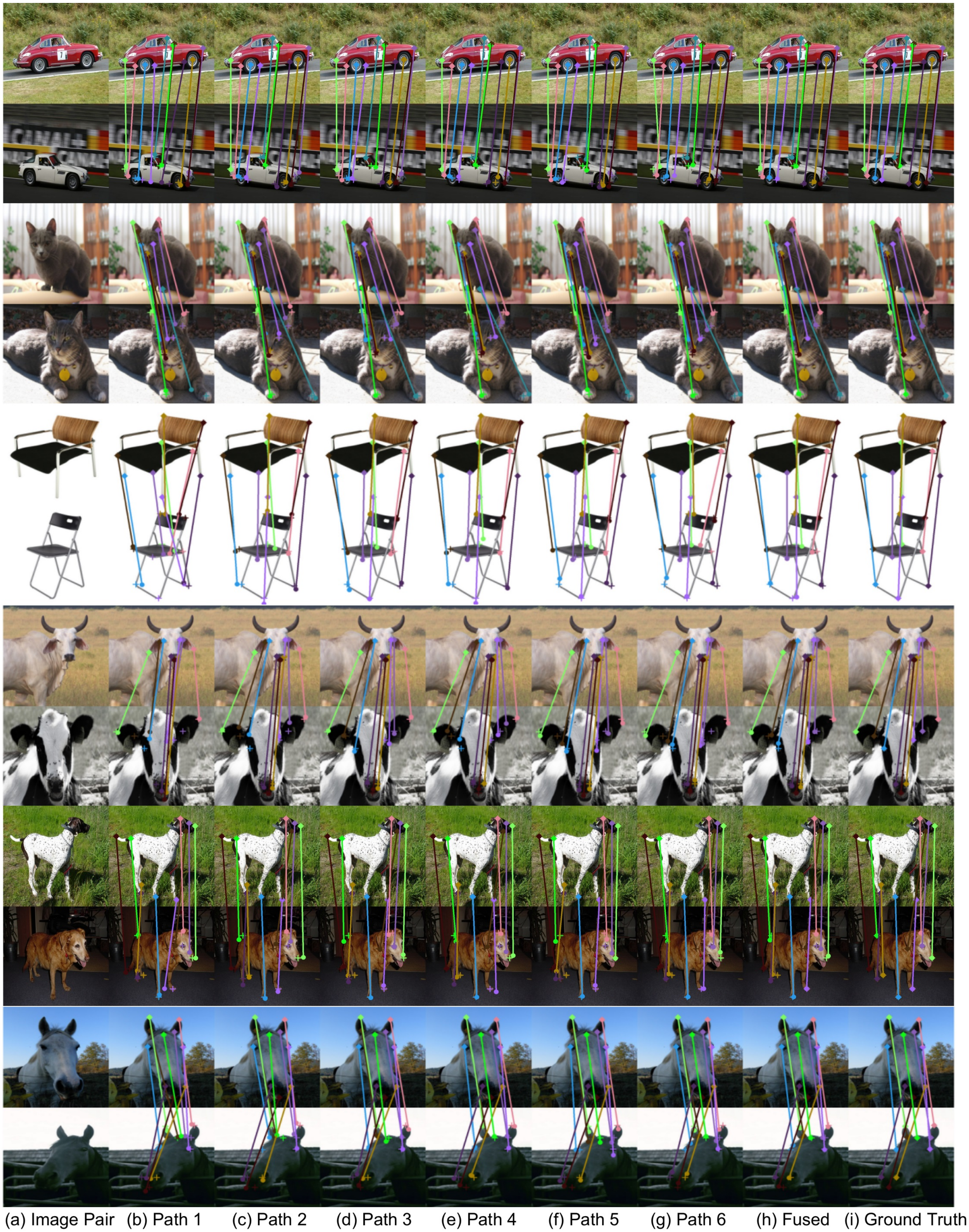

Figure 10. **Matching results of different paths.** From left to right, raw image pairs (a), matching results from path 1-6 (b-g), the output after multi-path fusion (h), and ground truth (i) are given.

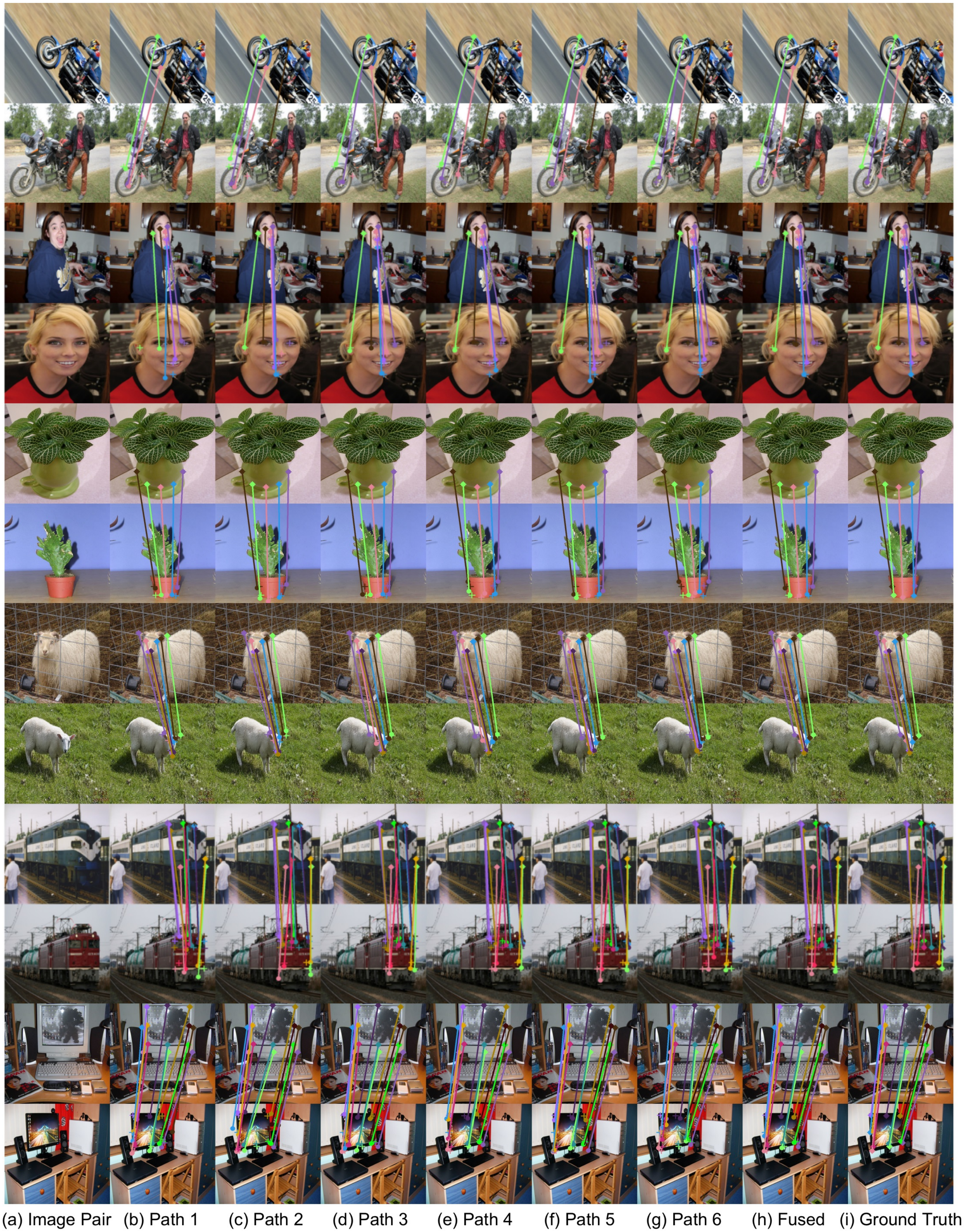

Figure 11. **Matching results of different paths.** From left to right, raw image pairs (a), matching results from path 1-6 (b-g), the output after multi-path fusion (h), and ground truth (i) are given.