Joint Stroke Tracing and Correspondence for 2D Animation
— Supplemental Material —

HAORAN MO, Sun Yat-sen University, China
CHENGYING GAO*, Sun Yat-sen University, China
RUOMEI WANG, Sun Yat-sen University, China

1 FRAMEWORK DETAILS

1.1 ASTM

The ASTM-P and the ASTM-S share a similar network architecture, as shown in Table 1.

Table 1. Architecture of ASTM-P and ASTM-S.

Layer Type	Kernel Size	Stride	Normalization Function	Activation Function	Output (ASTM-P)	Output (ASTM-S)
Input	-	-	-	-	$2 \times 256 \times 256$	$2 \times 192 \times 192$
CoordConv	-	-	-	-	$4 \times 256 \times 256$	$4 \times 192 \times 192$
Convolutional	3×3	2	Instance Norm.	ReLU	$32 \times 128 \times 128$	$32 \times 96 \times 96$
Convolutional	3×3	1	Instance Norm.	ReLU	$32 \times 128 \times 128$	$32 \times 96 \times 96$
Convolutional	3×3	2	Instance Norm.	ReLU	$64 \times 64 \times 64$	$64 \times 48 \times 48$
Convolutional	3×3	1	Instance Norm.	ReLU	$64 \times 64 \times 64$	$64 \times 48 \times 48$
Convolutional	3×3	2	Instance Norm.	ReLU	$128 \times 32 \times 32$	$128 \times 24 \times 24$
Convolutional	3×3	1	Instance Norm.	ReLU	$128 \times 32 \times 32$	$128 \times 24 \times 24$
Convolutional	3×3	1	Instance Norm.	ReLU	$128 \times 32 \times 32$	$128 \times 24 \times 24$
Convolutional	3×3	2	Instance Norm.	ReLU	$256 \times 16 \times 16$	$256 \times 12 \times 12$
Convolutional	3×3	1	Instance Norm.	ReLU	$256 \times 16 \times 16$	$256 \times 12 \times 12$
Convolutional	3×3	1	Instance Norm.	ReLU	$256 \times 16 \times 16$	$256 \times 12 \times 12$
Convolutional	3×3	2	Instance Norm.	ReLU	$512 \times 8 \times 8$	$512 \times 6 \times 6$
Convolutional	3×3	1	Instance Norm.	ReLU	$512 \times 8 \times 8$	$512 \times 6 \times 6$
Convolutional	3×3	1	Instance Norm.	ReLU	$512 \times 8 \times 8$	$512 \times 6 \times 6$
Reshape	-	-	-	-	32,768	18,432
Fully Connected	-	-	-	ReLU	256	256
Fully Connected	-	-	-	ReLU	128	128
Fully Connected	-	-	-	Tanh	5	3

1.2 Starting Point Prediction Network

The architecture of starting point prediction network in the starting point matching model is shown in Table 2.

*Corresponding author.

Authors' addresses: Haoran Mo, Sun Yat-sen University, Guangzhou, China, mohaor@mail2.sysu.edu.cn; Chengying Gao, Sun Yat-sen University, Guangzhou, China, mcsgcy@mail.sysu.edu.cn; Ruomei Wang, Sun Yat-sen University, Guangzhou, China, isswrm@mail.sysu.edu.cn.

Table 2. Architecture of starting point prediction network.

Image Type	Layer Type	Kernel Size	Stride	Normalization Function	Activation Function	Output
Reference	Input	-	-	-	-	$2 \times 256 \times 256$
Reference	CoordConv	-	-	-	-	$4 \times 256 \times 256$
Reference	Convolutional	3×3	1	Instance Norm.	ReLU	$16 \times 256 \times 256$
Reference	Convolutional	3×3	2	Instance Norm.	ReLU	$32 \times 128 \times 128$
Reference	Convolutional	3×3	1	Instance Norm.	ReLU	$32 \times 128 \times 128$
Reference	Convolutional	3×3	2	Instance Norm.	ReLU	$64 \times 64 \times 64$
Reference	Convolutional	3×3	1	Instance Norm.	ReLU	$64 \times 64 \times 64$
Reference	Convolutional	3×3	1	Instance Norm.	ReLU	$64 \times 64 \times 64$
Reference	Convolutional	3×3	2	Instance Norm.	ReLU	$128 \times 32 \times 32$
Reference	Convolutional	3×3	1	Instance Norm.	ReLU	$128 \times 32 \times 32$
Reference	Convolutional	3×3	1	Instance Norm.	ReLU	$128 \times 32 \times 32$
Reference	Convolutional	3×3	2	Instance Norm.	ReLU	$256 \times 16 \times 16$
Target	Input	-	-	-	-	$2 \times 256 \times 256$
Target	CoordConv	-	-	-	-	$4 \times 256 \times 256$
Target	Convolutional	3×3	1	Instance Norm.	ReLU	$16 \times 256 \times 256$
Target	Convolutional	3×3	2	Instance Norm.	ReLU	$32 \times 128 \times 128$
Target	Convolutional	3×3	1	Instance Norm.	ReLU	$32 \times 128 \times 128$
Target	Convolutional	3×3	2	Instance Norm.	ReLU	$64 \times 64 \times 64$
Target	Convolutional	3×3	1	Instance Norm.	ReLU	$64 \times 64 \times 64$
Target	Convolutional	3×3	1	Instance Norm.	ReLU	$64 \times 64 \times 64$
Target	Convolutional	3×3	2	Instance Norm.	ReLU	$128 \times 32 \times 32$
Target	Convolutional	3×3	1	Instance Norm.	ReLU	$128 \times 32 \times 32$
Target	Convolutional	3×3	1	Instance Norm.	ReLU	$128 \times 32 \times 32$
Target	Convolutional	3×3	2	Instance Norm.	ReLU	$256 \times 16 \times 16$
Both	Concat	-	-	-	-	$512 \times 16 \times 16$
Both	Convolutional	3×3	1	Instance Norm.	ReLU	$256 \times 16 \times 16$
Both	Convolutional	3×3	1	Instance Norm.	ReLU	$256 \times 16 \times 16$
Both	Convolutional	3×3	2	Instance Norm.	ReLU	$512 \times 8 \times 8$
Both	GAP	8×8	-	-	-	$512 \times 1 \times 1$
Both	Reshape	-	-	-	-	512
Both	Fully Connected	-	-	-	Tanh	2

1.3 Stroke Prediction Network

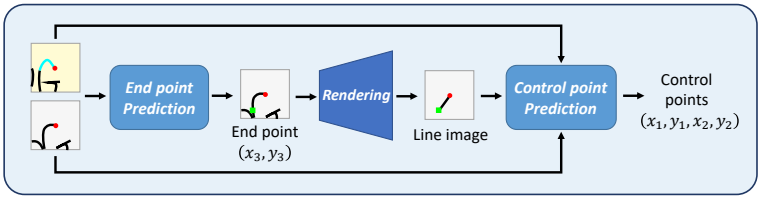

Fig. 1. Workflow of stroke prediction network.

In the stroke tracing and correspondence model, we introduce a stroke prediction network with a two-stage prediction workflow, with the first one for the end point and the second for the two control points, as illustrated in Fig. 1. This is inspired by a common user practice when tracing a stroke. We tend to find and fix the end point first, and then drag the two control points to make the stroke aligned to the drawing. Hence, we simulate this practice through the

Table 3. Architecture of stroke prediction networks. "a/b" in "Output" means differences between end point prediction network and control point prediction network.

Image Type	Layer Type	Kernel Size	Stride	Normalization Function	Activation Function	Output
Reference	Input	-	-	-	-	$3 \times 192 \times 192$
Reference	CoordConv	-	-	-	-	$5 \times 192 \times 192$
Reference	Convolutional	3×3	2	Instance Norm.	ReLU	$32 \times 96 \times 96$
Reference	Convolutional	3×3	1	Instance Norm.	ReLU	$32 \times 96 \times 96$
Reference	Convolutional	3×3	2	Instance Norm.	ReLU	$64 \times 48 \times 48$
Reference	Convolutional	3×3	1	Instance Norm.	ReLU	$64 \times 48 \times 48$
Reference	Convolutional	3×3	2	Instance Norm.	ReLU	$128 \times 24 \times 24$
Reference	Convolutional	3×3	1	Instance Norm.	ReLU	$128 \times 24 \times 24$
Reference	Convolutional	3×3	1	Instance Norm.	ReLU	$128 \times 24 \times 24$
Reference	Convolutional	3×3	2	Instance Norm.	ReLU	$256 \times 12 \times 12$
Reference	Convolutional	3×3	1	Instance Norm.	ReLU	$256 \times 12 \times 12$
Reference	Convolutional	3×3	1	Instance Norm.	ReLU	$256 \times 12 \times 12$
Target	Input	-	-	-	-	$2/3 \times 192 \times 192$
Target	CoordConv	-	-	-	-	$4/5 \times 192 \times 192$
Target	Convolutional	3×3	2	Instance Norm.	ReLU	$32 \times 96 \times 96$
Target	Convolutional	3×3	1	Instance Norm.	ReLU	$32 \times 96 \times 96$
Target	Convolutional	3×3	2	Instance Norm.	ReLU	$64 \times 48 \times 48$
Target	Convolutional	3×3	1	Instance Norm.	ReLU	$64 \times 48 \times 48$
Target	Convolutional	3×3	2	Instance Norm.	ReLU	$128 \times 24 \times 24$
Target	Convolutional	3×3	1	Instance Norm.	ReLU	$128 \times 24 \times 24$
Target	Convolutional	3×3	1	Instance Norm.	ReLU	$128 \times 24 \times 24$
Target	Convolutional	3×3	2	Instance Norm.	ReLU	$256 \times 12 \times 12$
Target	Convolutional	3×3	1	Instance Norm.	ReLU	$256 \times 12 \times 12$
Target	Convolutional	3×3	1	Instance Norm.	ReLU	$256 \times 12 \times 12$
Both	Concat	-	-	-	-	$512 \times 12 \times 12$
Both	Convolutional	3×3	2	Instance Norm.	ReLU	$512 \times 6 \times 6$
Both	Convolutional	3×3	1	Instance Norm.	ReLU	$512 \times 6 \times 6$
Both	Convolutional	3×3	1	Instance Norm.	ReLU	$512 \times 6 \times 6$
Both	Reshape	-	-	-	-	18,432
Both	Fully Connected	-	-	-	ReLU	256
Both	HyperLSTM	-	-	-	ReLU	256
Both	Fully Connected	-	-	-	Tanh	2/4

two-stage workflow. After predicting the end point in the first stage, we map it to global coordinates via the inverse transformation operation in the main workflow, and then render a full-size line image from the starting point to the predicted end point (green dot in Fig. 1). The line image is cropped into a line image patch based on the transformed window. It serves as an additional input of the control point prediction network, revealing the fixed position of the end point. This design is found to increase accuracy of the predictions for the three points.

The two prediction networks share a similar architecture, consisting of separated convolutional layers for encoding the reference and target patches respectively and a recurrent neural network (RNN) to fuse the features and output the predictions, as shown in Table 3.

2 DATASET DETAILS

Category of Shapes. There are 250 categories from TU-Berlin dataset [Eitz et al. 2012] in training set and 226 in validation set.

Distribution w.r.t Number of Strokes. In our assembled dataset, the stroke number ranges from 4 to 40. The distribution is shown in Table 4. We show some simple and complicated examples in Fig. 2.

Table 4. Distribution w.r.t number of strokes.

	Training set		Validation set	
#stroke	#example	Percentage	#example	Percentage
4-9	163	2%	21	2%
10-14	718	7%	66	7%
15-19	1269	13%	125	13%
20-24	1579	16%	149	15%
25-29	1551	16%	154	15%
30-34	1511	15%	169	17%
35-40	3209	32%	316	31%

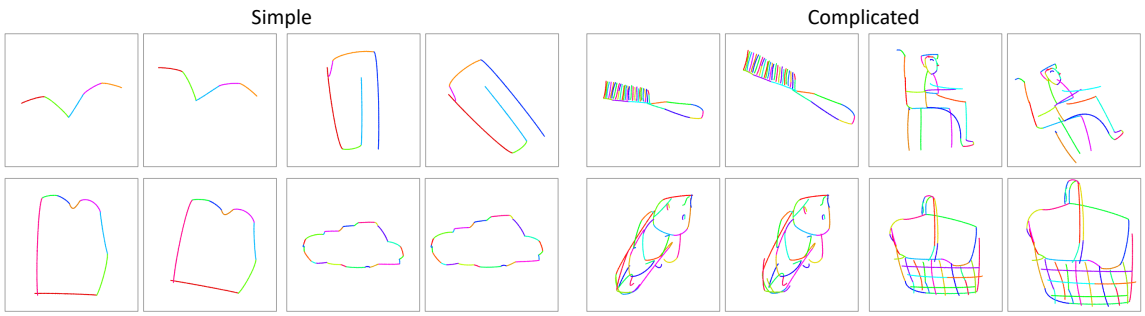

Fig. 2. Simple and complicated cases in the dataset.

Level of Deformation. We use Euclidean distance of control points on the strokes to measure the level of deformation. As shown in main paper, the deformations for creating paired data with vector stroke correspondence are sampled according to probabilities, so there are examples without any deformation (distance = 0px). The maximum averaged point distance is 90.45px. The distribution is shown in Table 5. We show examples with low and high levels of deformation in Fig. 3.

Table 5. Distribution w.r.t averaged point distance reflecting the level of deformation.

	Training set		Validation set	
Point distance (px)	#example	Percentage	#example	Percentage
0-10	2684	27%	412	41%
11-20	2634	26%	239	24%
21-30	1987	20%	163	16%
31-40	1324	13%	103	10%
41-50	757	8%	52	5%
51-60	414	4%	18	2%
61-70	147	1%	10	1%
71-80	41	0.4%	1	0.1%
81-91	12	0.1%	2	0.2%

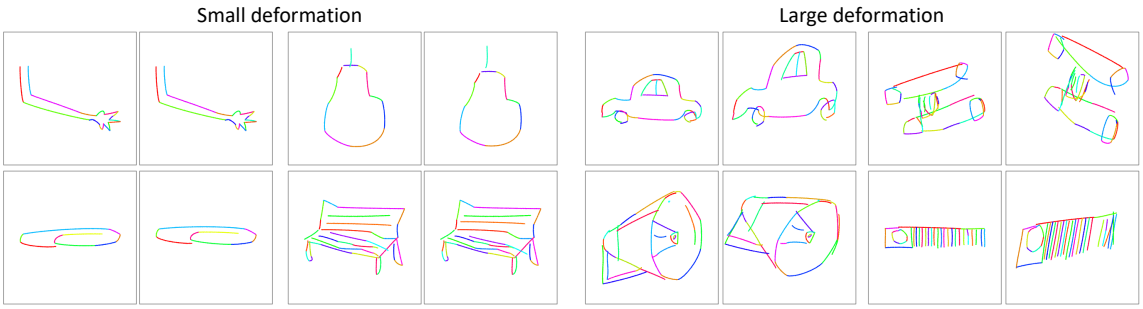

Fig. 3. Examples with small and large deformations in the dataset.

3 COMPARISONS WITH EXISTING APPROACHES

3.1 Comparisons with Raster Correspondence-based Methods

More results can be seen in Fig. 4 and Fig. 5.

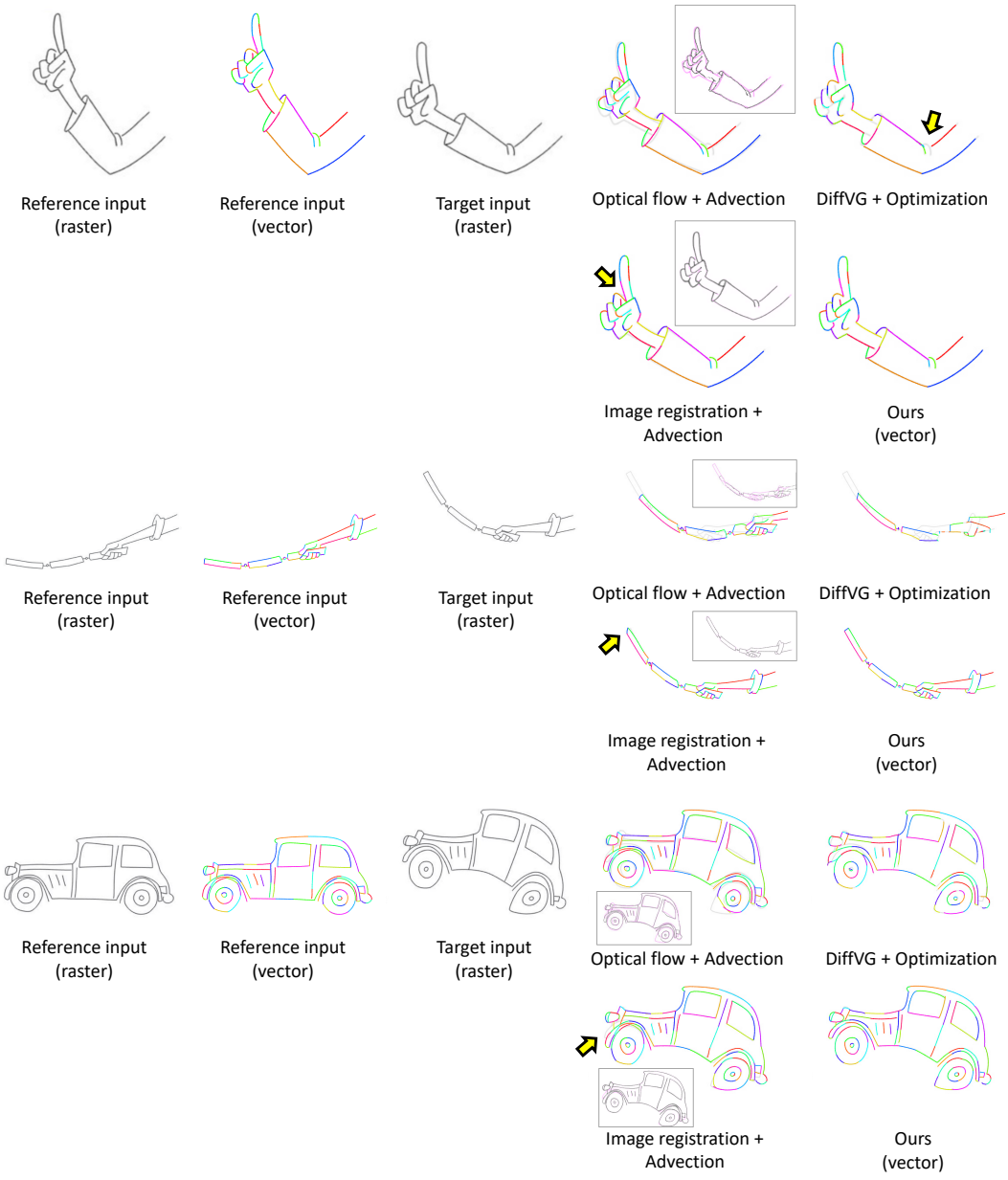

Fig. 4. Comparisons with baseline methods based on raster correspondence. Sub-figures in square box are warped reference images by the predicted optical flows or registration maps (magenta drawing denote the target).

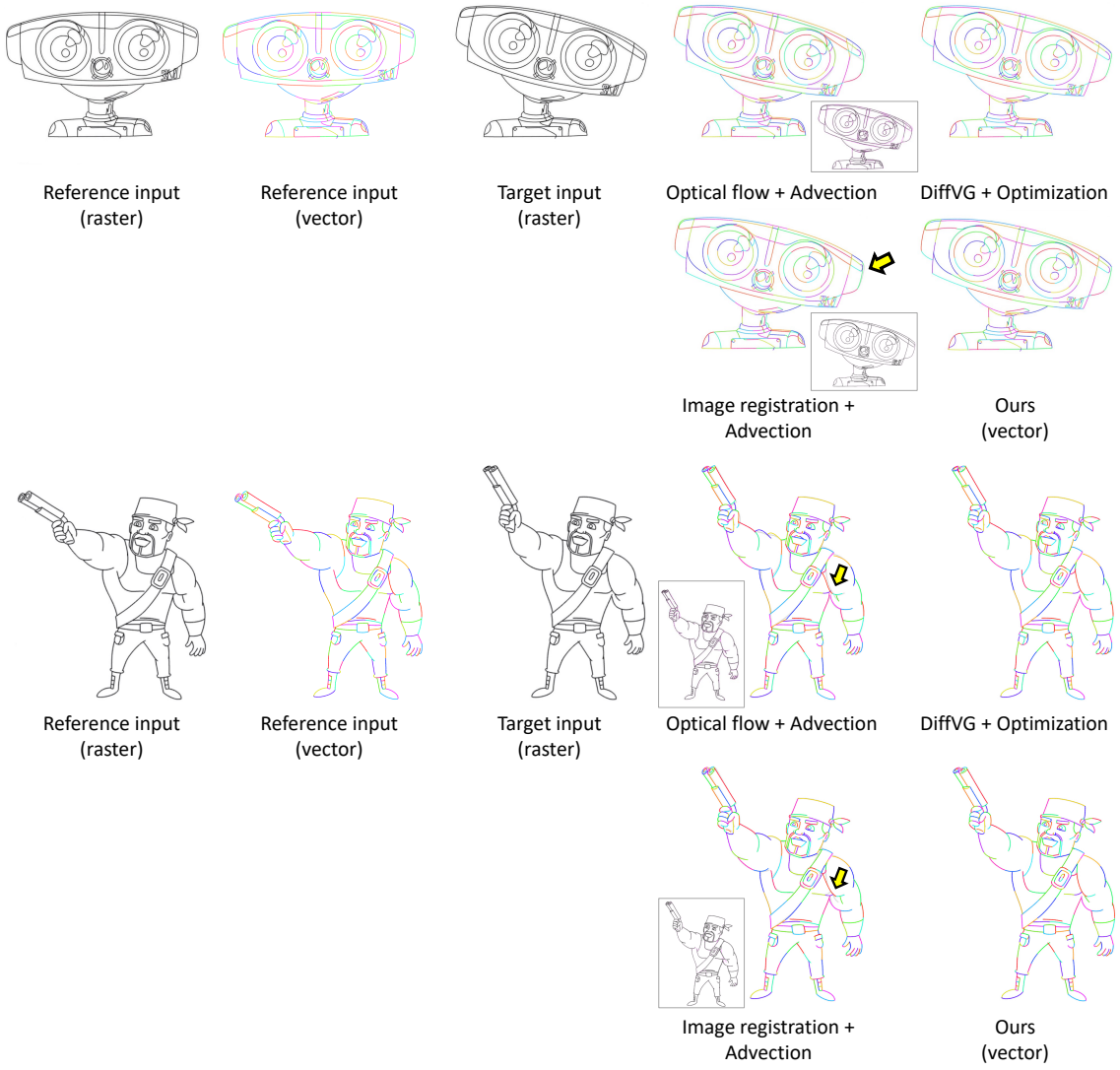

Fig. 5. Comparisons with baseline methods based on raster correspondence. Sub-figures in square box are warped reference images by the predicted optical flows or registration maps (magenta drawing denote the target).

3.2 Image Registration with APES

We try APES [Xu et al. 2022] designed for 2D cartoon frames as one of the image registration solution. However, the performance is poor, as suggested by the warped reference images by the predicted correspondence maps in Fig. 6, where the registrations exhibit scattered pixels and discontinuous lines. We notice this is probably due to the low-quality training data, *i.e.*, correspondence maps in the CreativeFlow+ dataset [Shugrina et al. 2019].

In this dataset, ground-truth pixelwise correspondence maps between animated frames are provided for a supervised training. Ideally, the correspondence map should warp the source frame into that aligned with the target frame. However, as shown in Fig. 7, we find the warpings are not fully aligned, even in consecutive frames with slight spatial variation. The warped results indicate that there are quite a few noisy correspondences that map the stroke pixels to the empty space (see arrows). The ambiguity in the training data is detrimental to a robust image registration model. The subsequent control point advection process based on the noisy correspondence tends to produce unreasonable vector strokes.

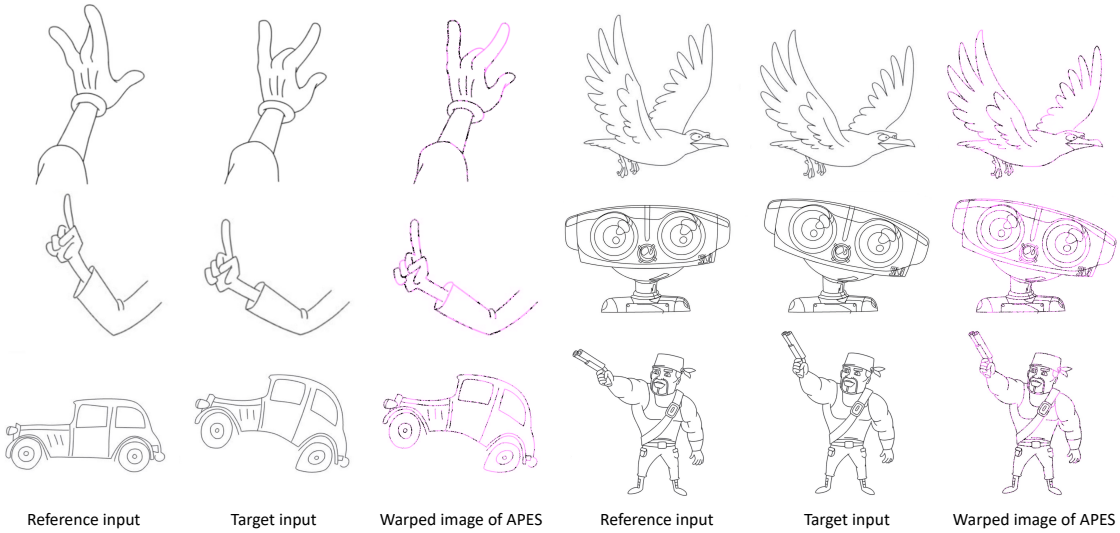

Fig. 6. Correspondence results predicted by APES [Xu et al. 2022].

Fig. 7. We warp a source frame (the frame 1) into the target frames (the rest) with the respective correspondence maps from the CreativeFlow+ dataset [Shugrina et al. 2019]. The first row shows the original frames and the second one shows the warped ones. The undesirable warpings indicate that the ground-truth correspondence maps are noisy and in low quality.

3.3 AlphaContours + Functional maps

As shown in the main paper and Fig. 8, while similar sketch shapes are computed by the AlphaContours algorithm [Myronova et al. 2023], the functional maps exhibit incorrect vertex-wise correspondence that is detrimental to the final stroke-level correspondence. We find this is due to the varying stroke numbers and lengths between the input reference vector drawing and the target one vectorized by PolyVectorization [Bessmeltsev and Solomon 2019] that tends to produce a great number of short polylines.

The AlphaContours algorithm [Myronova et al. 2023] first samples points on the vector drawings. When the target vector drawing contains much more segments than the reference, a larger number of sampling points are generated as the vertices for subsequent triangulation process. As can be seen in Fig. 9, the triangles vary significantly in even a similar shape (see arrows), and those in the target are denser. The different distributions of the triangles mislead the functional map algorithm [Ovsjanikov et al. 2012], which results in unreasonable vertex correspondence.

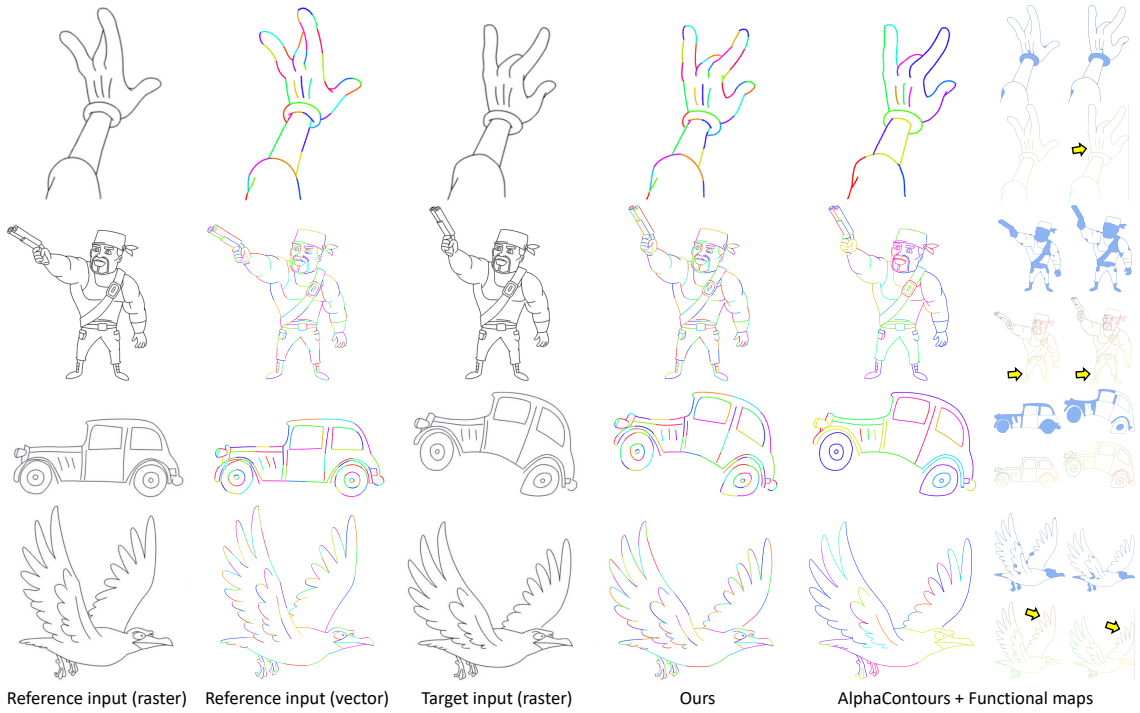

Fig. 8. Comparisons with a baseline method based on vector stroke correspondence. Sub-figures on the rightmost are sketch shapes computed by AlphaContours [Myronova et al. 2023] (top) and functional maps indicating the correspondence [Ovsjanikov et al. 2012] (bottom).

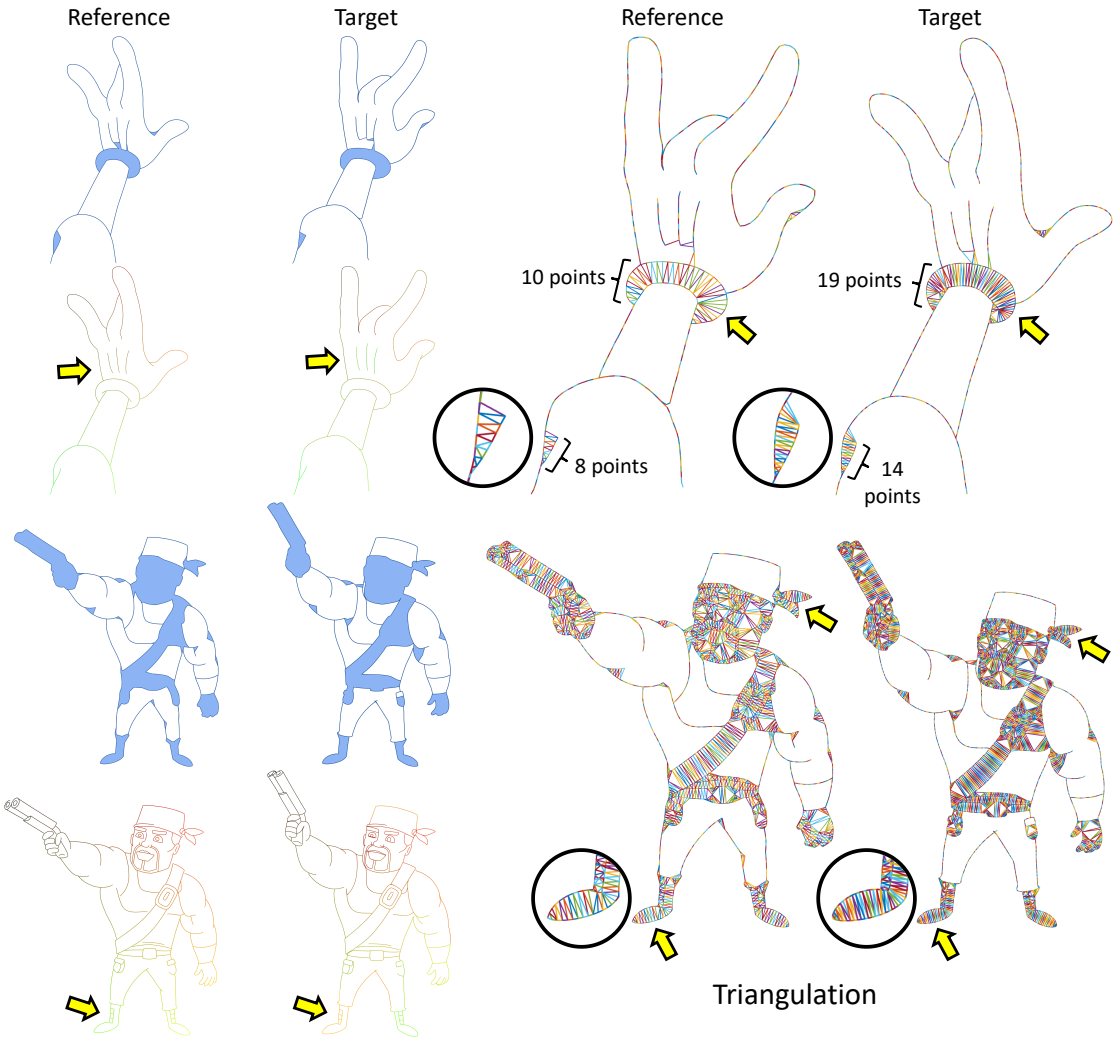

Fig. 9. The varying triangulations within a similar sketch shapes give rise to unsatisfied functional maps. Left-top: sketch shapes. Left-bottom: functional maps.

4 MORE RESULTS

We show more results of our approach in clean line drawings in the validation set of our dataset (Fig. 10) and real rough drawings (Fig. 11).

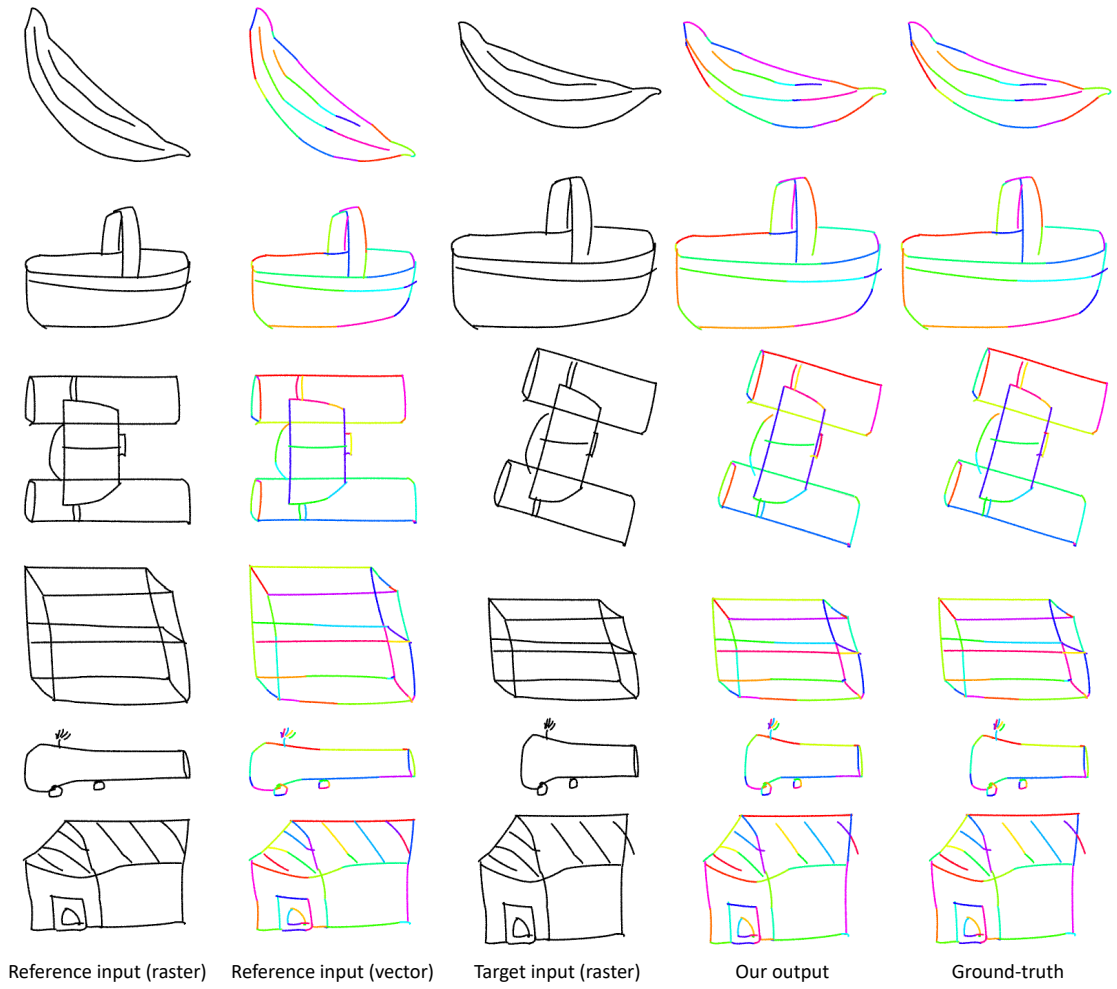

Fig. 10. Results on examples in the validation set of our dataset.

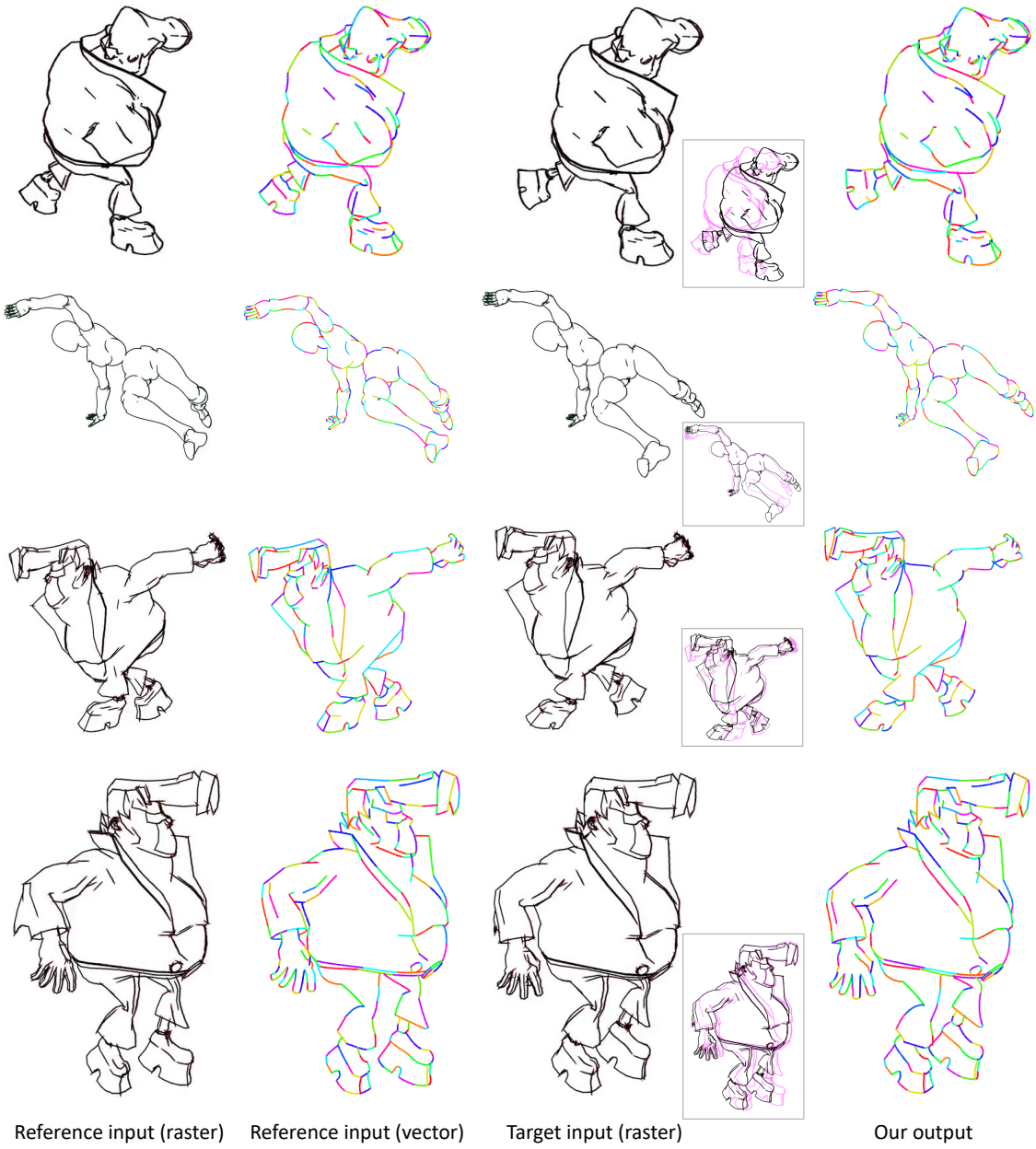

Fig. 11. Results on rough drawings. Sub-figures in boxes show differences between consecutive frames, with magenta for the previous frames and black for the current ones.

5 EFFECTIVENESS OF ASTM

Figures 12, 13 and 14 show more results of our ASTM's robustness to patches with non-linear deformations, including the predicted transformations of these examples and those in the main paper.

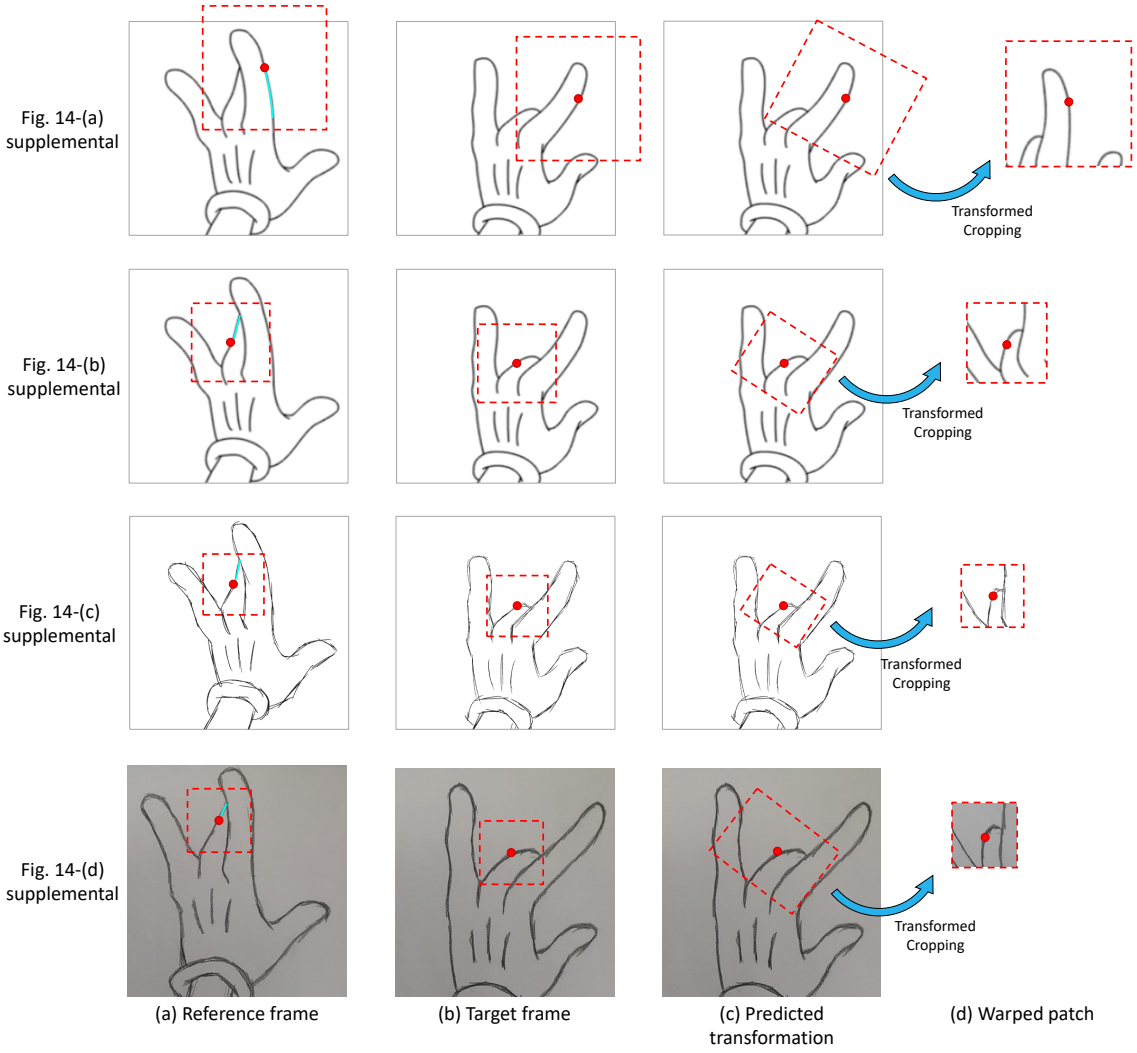

Fig. 12. Effectiveness of ASTM that predicts a transformation to roughly align the target patch with the reference. Reference strokes at current step are highlighted in blue. See images of index at the left most for more clear visualization.

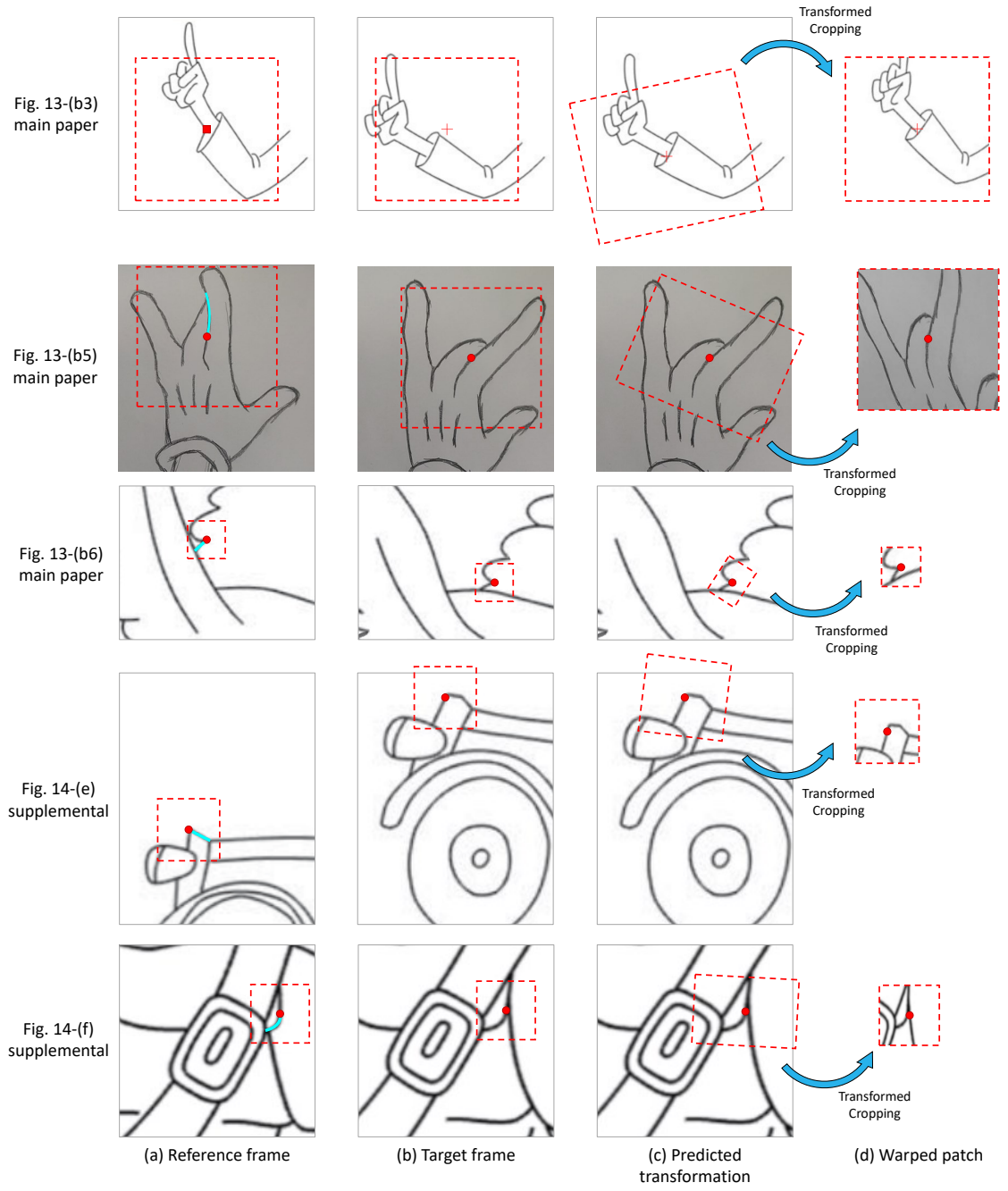

Fig. 13. Effectiveness of ASTM that predicts a transformation to roughly align the target patch with the reference. Reference strokes at current step are highlighted in blue. See images of index at the left most for more clear visualization.

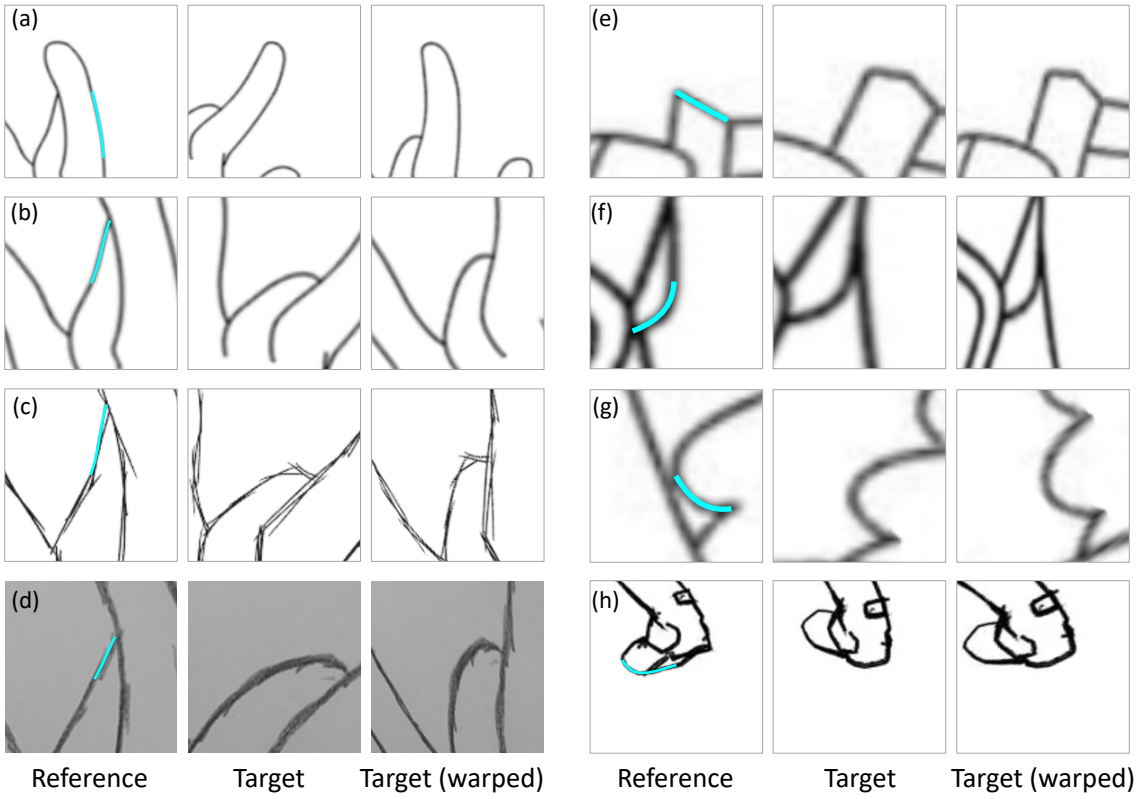

Fig. 14. More examples of warped patches from the target. Reference strokes at current step are highlighted in blue.

6 SENSITIVITY TO INITIAL VECTORIZATION

6.1 Robustness to Stroke Order

We show more results of robustness to stroke order in Fig. 15 (clean drawings) and Fig. 16 (rough drawings).

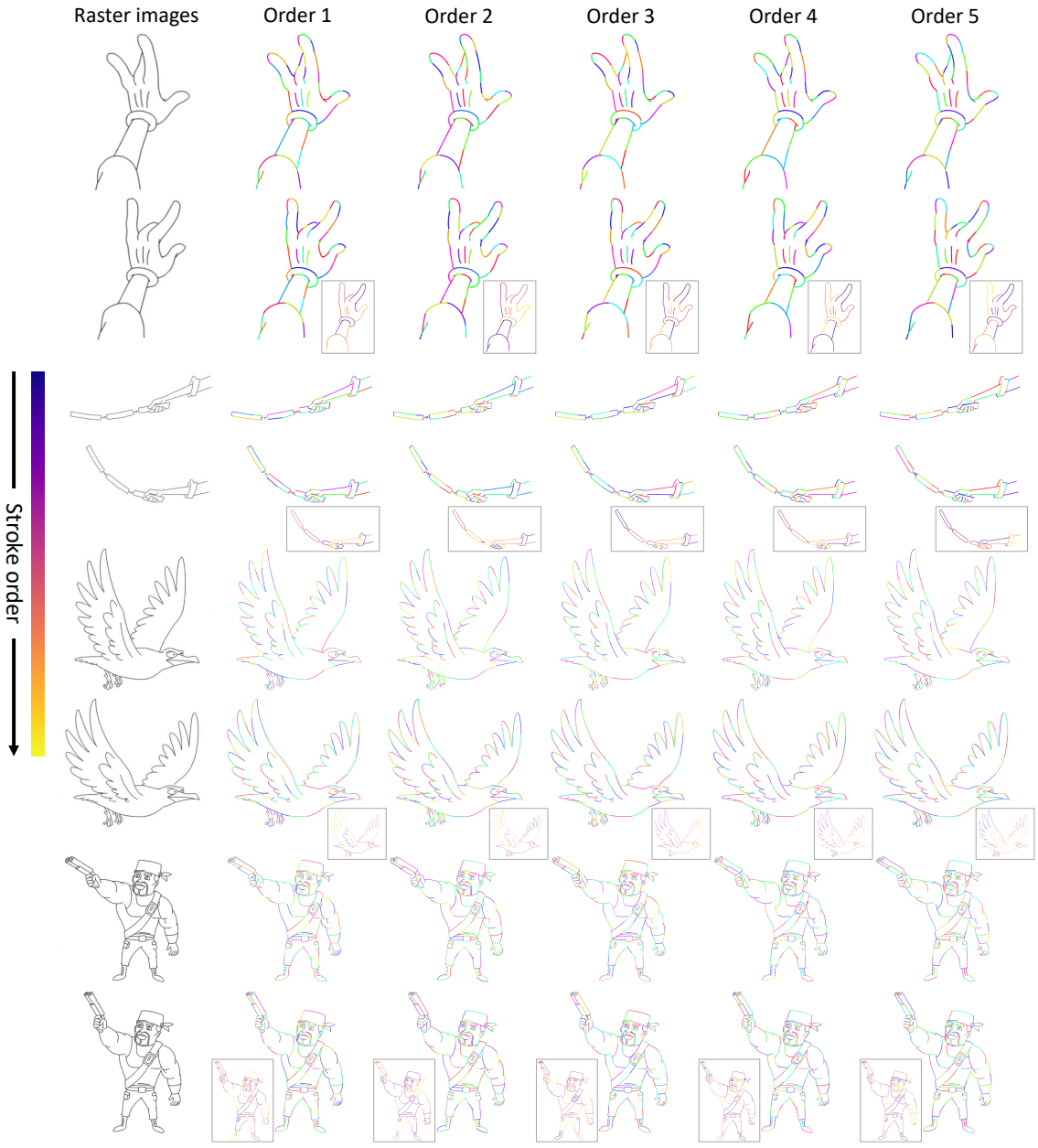

Fig. 15. Results on clean drawings with different stroke orders. Sub-figures show the stroke orders.

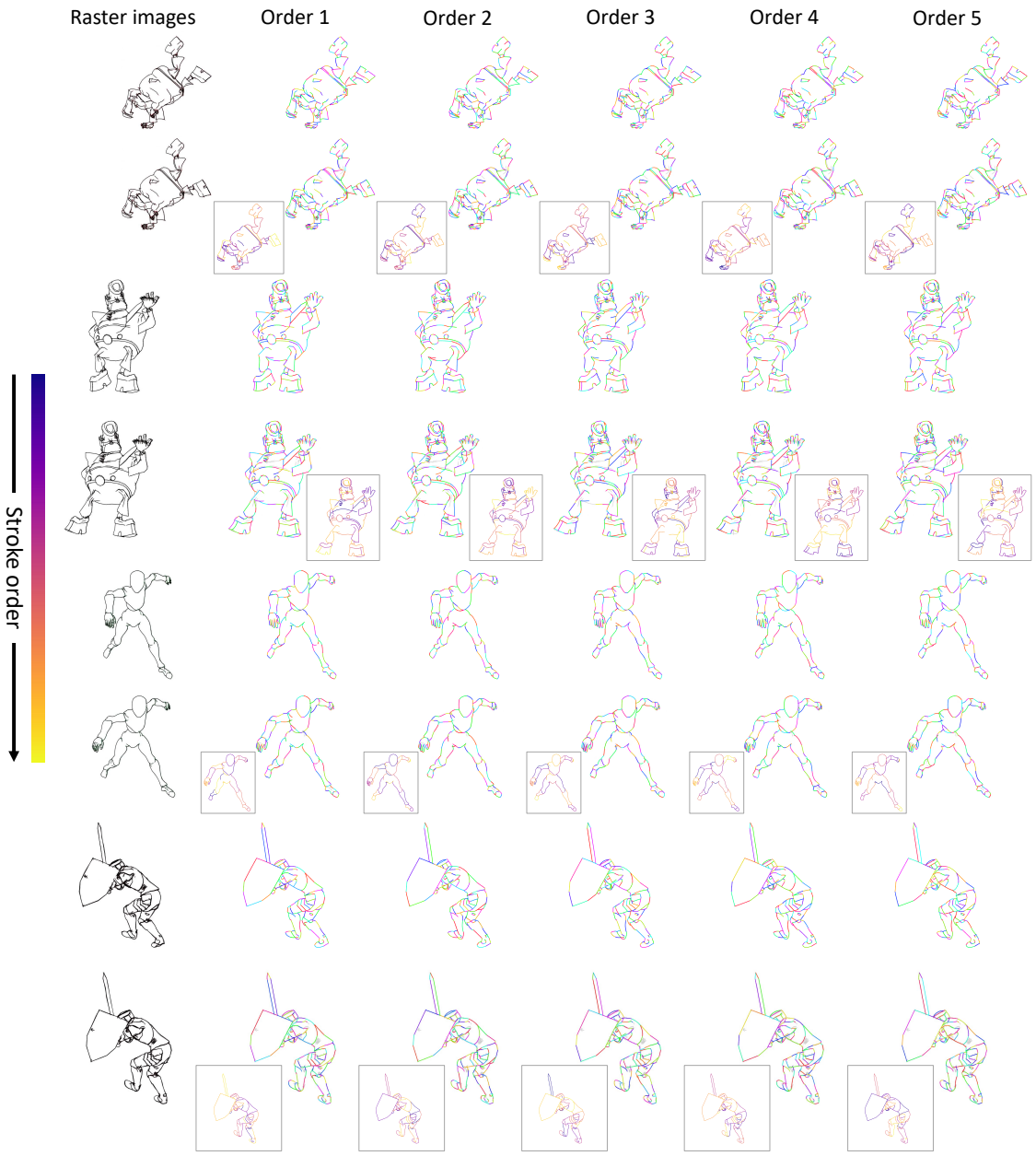

Fig. 16. Results on rough drawings with different stroke orders. Sub-figures show the stroke orders.

6.2 Sensitivity to Stroke Number or Length

We show more results of sensitivity to stroke number or length in Fig. 17 (clean drawings) and Fig. 18 (rough drawings).

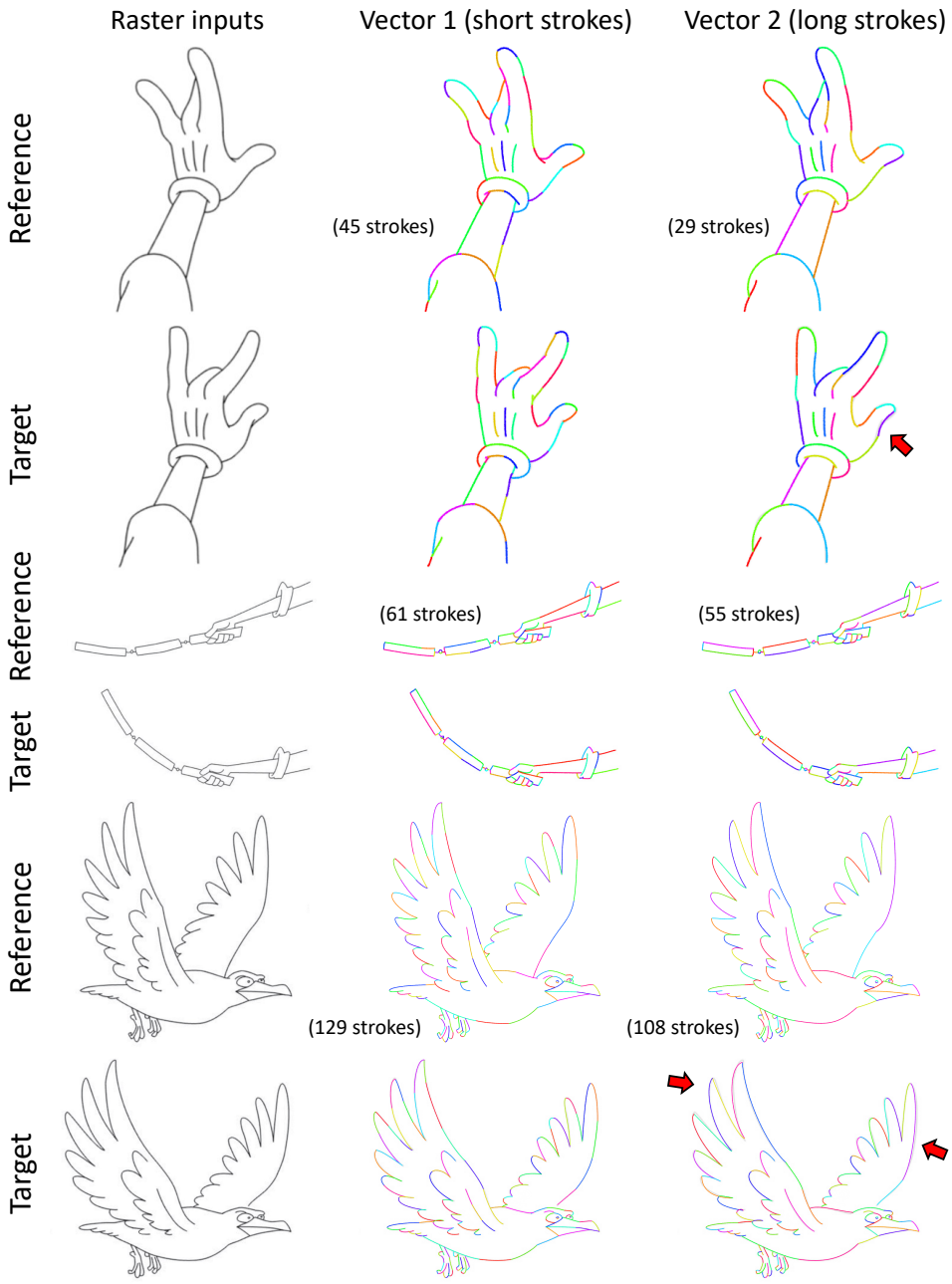

Fig. 17. Results on clean drawings with different numbers or lengths of strokes.

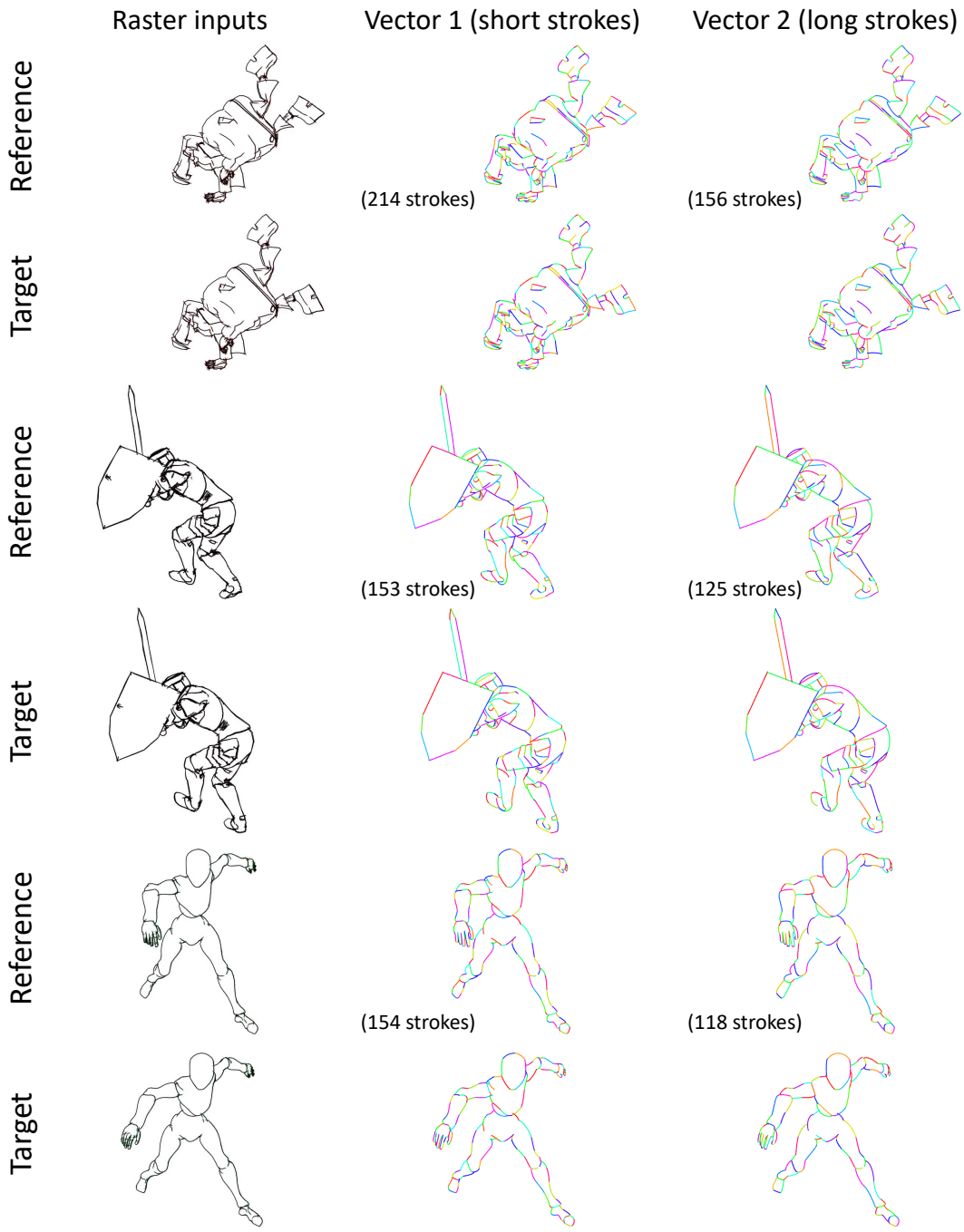

Fig. 18. Results on rough drawings with different numbers or lengths of strokes.

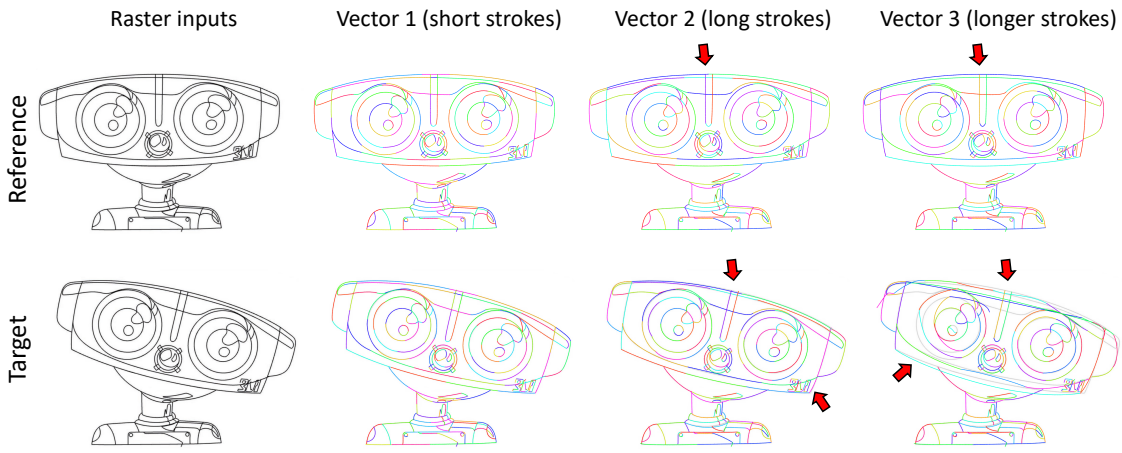

Fig. 19. Failure case with extremely long strokes.

As discussed in main paper, our approach performs less than satisfactory on some long strokes with worse alignment with the target (see red arrows in Fig. 17 and Fig. 19). We show a case with extremely long strokes in Fig. 19, where the predictions of such strokes fail. The bad strokes also have a negative impact on other strokes.

7 RUNTIME COMPARISON

The numerical runtime is shown in Table 6.

Table 6. Runtime comparison.

Image	#stroke	Resolution	Ours (auto)	Ours (fix)	Ours (total)	Registration	DiffVG	Manual operation
Arm	38	480	0:15	0:31	0:46	1:11	5:30	3:16
Hand	45	380	0:20	0:14	0:34	1:07	5:01	3:37
Stick	61	680	0:24	0:31	0:55	0:50	6:56	4:03
Car	97	640	0:37	1:35	2:12	1:07	6:41	8:55
Eagle	129	1024	0:51	2:18	3:09	1:36	11:05	11:57
Paladin	153	750	0:55	0:45	1:40	1:37	8:21	11:49
Xbot	154	750	0:54	0:37	1:31	2:31	8:03	12:30
Bigvegas1	214	750	1:17	0:21	1:38	1:05	8:29	8:22
Robot	239	1193	1:33	1:50	3:23	2:31	12:38	1:51
Gunman	274	1024	1:48	1:24	3:12	1:34	11:35	13:36
Bigvegas2	307	750	1:48	1:34	3:22	2:14	8:20	15:19

8 LIMITATIONS AND DISCUSSION

Here we have more discussions of our approach, apart from the occlusion and topology change that have been analyzed in main paper.

8.1 Can the system handle more keyframes?

While our approach is shown to work on multiple raster keyframes, error accumulation does exist due to the imperfect prediction. We also show an example with 5 keyframes in Fig. 20. More keyframes tend to induce occlusion (*e.g.,* the muzzle of the gun (yellow arrows)), topology changes (*e.g.,* the underarm and fingers (red arrows)), and large distortion of strokes (green arrows). These issues are too challenging for our automatic framework to handle, and thus the results become worse and worse. The results can be imported into an interactive system CACANi for post-hoc manual refinement and such a solution is shown to be more effective than the conventional workflow fully by hand in the main paper. To avoid incremental error accumulation across frames that may require more time to fix the incorrect strokes, we think manual fix every 1 to 2 predicted frames would be a good choice to achieve a relatively high efficiency when processing a long sequence of keyframes.

Besides the post-hoc refinement, live corrections during the inference process is also a potential solution. This method with timely fix further reduces error accumulation within a single frame. Since a specialized user interface is required, it could be a future extension of our work.

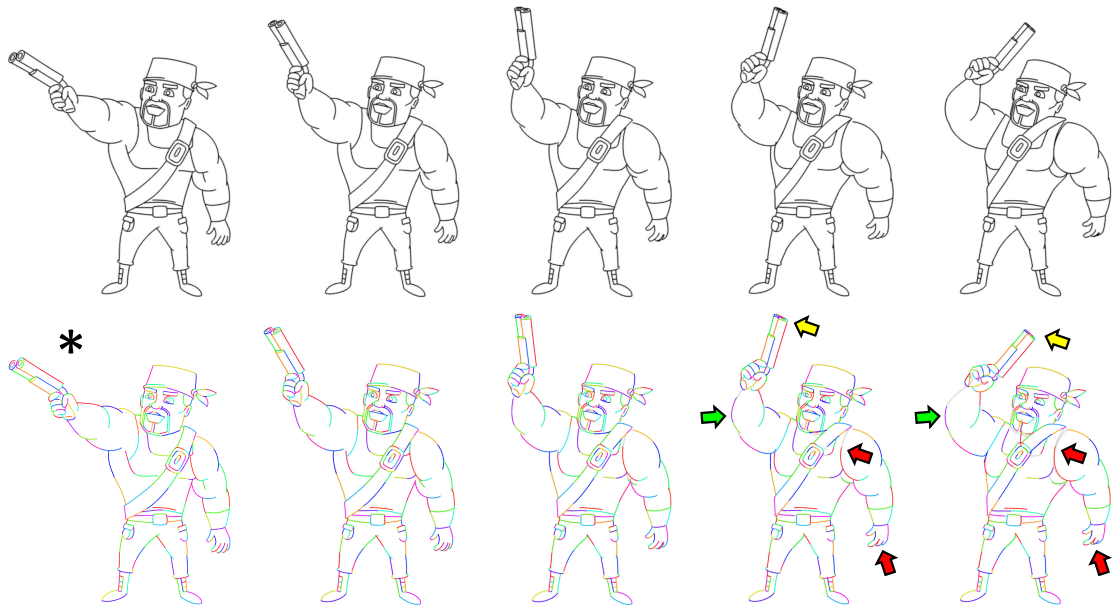

Fig. 20. Results on multiple keyframes. Only vector drawings of the starting frames (with marks '∗') are provided.

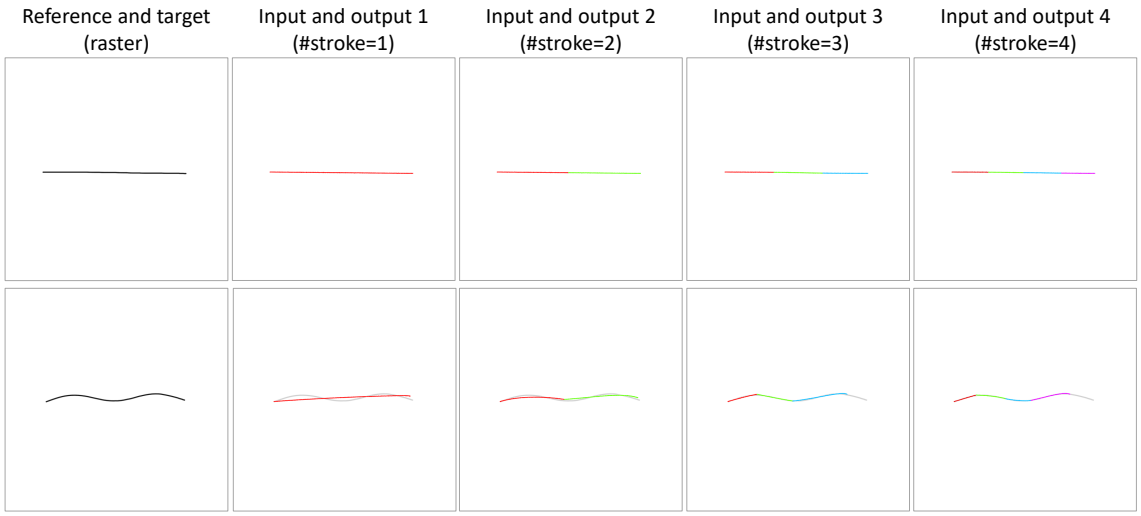

Fig. 21. Results of complex non-rigid motions, *e.g.*, a straight line evolving into a bump. We create different numbers of strokes in the vector inputs.

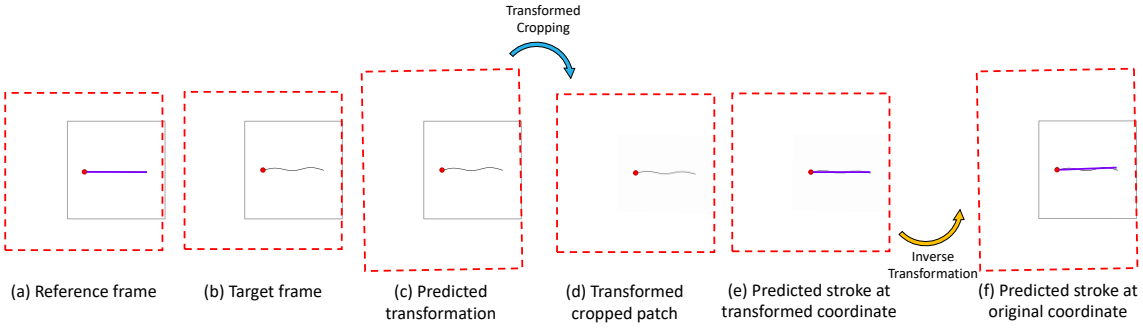

Fig. 22. Illustration of how ASTM works. There is one stroke in the input vector drawing.

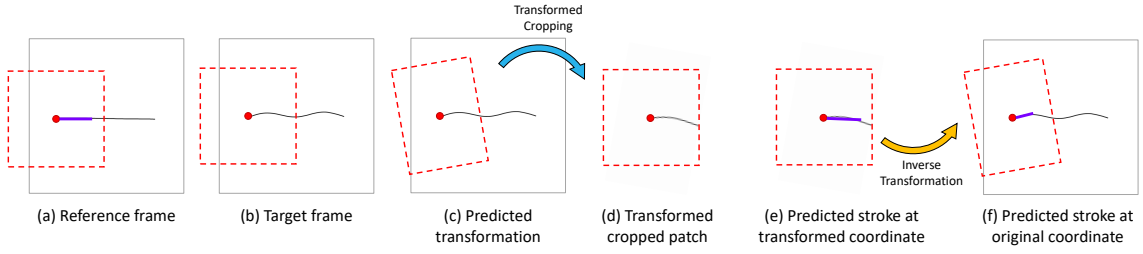

Fig. 23. Illustration of how ASTM works. There are three stroke in the input vector drawing.

8.2 How does the system handle complex non-rigid motions?

As discussed in limitation section in main paper, our proposed adaptive spatial transformation module (ASTM) leverages the visual cues in two patches to predict linear transformations that make them roughly aligned, even when non-linear deformations exist. However, for drawings with complex non-rigid motions, especially when visual cues are insufficient, such as a straight line evolving into a highly curved bump, the ASTM fails.

As shown in Fig. 21, our method fails to redraw the bump no matter how many strokes we define. We show why our method fails in this case with illustrations in Fig. 22 (#stroke=1) and Fig. 23 (#stroke=3). When there is only one stroke in the reference drawing (Fig. 22), it is difficult to fit the bump by definition. Our ASTM tends to squash the bump image by stretching the window vertically (c), such that the deformed bump is less curved (d) to approximate the straight line in the reference image. When three strokes are used as the reference vector (Fig. 23), we notice the ASTM tends to rotate the window anticlockwise (c) to make the resulting patch less curved (d). Afterwards, while the predicted stroke covers the flat part of the stroke (e), it is shorted than expected indeed (f). The observations above indicate that our ASTM works well on making two patches as similar as possible with linear transformations. However, they are incorrect in the presence of complex non-rigid motions, especially when stroke proportions should be carefully considered. Non-linear transformations could be adopted to address this issue as discussed in the limitation section of the main paper.

REFERENCES

Mikhail Bessmeltsev and Justin Solomon. 2019. Vectorization of line drawings via polyvector fields. *ACM Transactions on Graphics (TOG)* 38, 1 (2019), 1–12.

Mathias Eitz, James Hays, and Marc Alexa. 2012. How do humans sketch objects? *ACM Transactions on graphics (TOG)* 31, 4 (2012), 1–10.

Mariia Myronova, William Neveu, and Mikhail Bessmeltsev. 2023. Differential Operators on Sketches via Alpha Contours. *ACM Transactions on Graphics* 42, 4 (2023).

Maks Ovsjanikov, Mirela Ben-Chen, Justin Solomon, Adrian Butscher, and Leonidas Guibas. 2012. Functional maps: a flexible representation of maps between shapes. *ACM Transactions on Graphics (ToG)* 31, 4 (2012), 1–11.

Maria Shugrina, Ziheng Liang, Amlan Kar, Jiaman Li, Angad Singh, Karan Singh, and Sanja Fidler. 2019. Creative flow+ dataset. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*. 5384–5393.

Zhan Xu, Matthew Fisher, Yang Zhou, Deepali Aneja, Rushikesh Dudhat, Li Yi, and Evangelos Kalogerakis. 2022. Apes: Articulated part extraction from sprite sheets. In *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*.