Supplementary: Capturing Non-Linear Human Perspective in Line Drawings

JINFAN YANG, University of British Columbia, Canada
LEO FOORD-KELCEY, University of British Columbia, Canada
SUZURAN TAKIKAWA, University of British Columbia, Canada
NICHOLAS VINING, University of British Columbia, Canada and NVIDIA, Canada
NILOY MITRA, University College London, United Kingdom and Adobe Research, United Kingdom
ALLA SHEFFER, University of British Columbia, Canada

ACM Reference Format:
Jinfan Yang, Leo Foord-Kelcey, Suzuran Takikawa, Nicholas Vining, Niloy Mitra, and Alla Sheffer. 2025. Supplementary: Capturing Non-Linear Human Perspective in Line Drawings. *ACM Trans. Graph.* 1, 1 (September 2025), 7 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

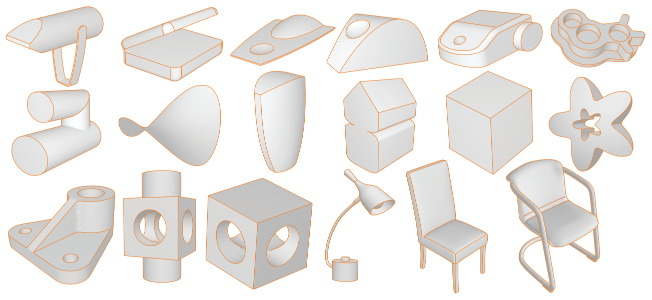


Fig. 1. All 3D model shapes used for our experiments.

1 Additional Implementation Details

Here we describe details of our implementation of learned human perspective, including sketch and NPR contour preprocessing; MLP architecture; parameter choices; our post-inference regularization step; and other technical details.

1.1 Preprocessing

Our method takes as input a paired human sketch and a 3D model with an estimated camera position matching the human sketch. For our experiments, we evaluate our method on inputs from the OpenSketch data set [Gryaditskaya et al. 2019]; the DifferSketching data set [Xiao et al. 2022], and a cube.

Preprocessing consists of four basic tasks that are orthogonal to our method: alignment of the 3D model to the input drawing; extraction of a set of NPR contours from the 3D model; vectorization of the extracted contours; and preprocessing of strokes in input human sketches. In our experiments, we used off-the-shelf methods for these tasks, discussed below. We require that the input 3D meshes are properly oriented with respect to the drawings, and consist of clean geometry with no extraneous material and no non-manifold elements. This is required to generate a clean set of contours [Bénard et al. 2019]. Inputs taken from the OpenSketch dataset include calibrated camera information, and provide code to estimate camera information including the model view matrix and intrinsic camera parameters; we use this information directly when rendering the NPR contours. For DifferSketching, we found that we had to repair the input meshes and align them by hand. This time-consuming manual process restricted the amount of data we were able to evaluate from their corpus to 9 inputs total.

For each input, we generate a rasterized set of NPR contours using the standard silhouette rendering technique [Bénard et al. 2019]. These raster NPR renders are then vectorized using the method proposed in Gutan et al. [2023], producing a set of vector strokes. In practice, these raw vectorization outputs often contain excessive points and exhibit arbitrary stroke segmentation. To address this, we preprocess the vector data in several stages. First, we evenly sample points along each vector stroke; then, using the method from [Baran et al. 2010], we fit each stroke to geometric primitives. Instead of using the fitted primitives directly, however, we leverage their endpoints to segment our strokes into subcurves with smoothly changing curvature (e.g. lines, arcs, and clothoids). As Cornucopia may oversegment contours for our purpose (for instance, splitting lines into multiple segments), we merge connected neighbouring strokes if they have the same stroke type and similar tangent and curvature. Finally, the segmented vector data is normalized to the range $[-1, 1]^2$ to standardize scale across objects.

We also preprocess the input human sketches to ensure comparability with the vectorized NPR data. First, we remove hooks at the end of strokes; we then uniformly resample them at the same rate as the NPR data. This ensure compatibility and enhances performance for subsequent matching.

Finally, we subdivide vector curves from both human sketches and NPR contours into regular polylines whose vertices are l units apart. For our experiments we set $l = 0.02$.

Authors' Contact Information: Jinfan Yang, University of British Columbia, Canada, yangjf@cs.ubc.ca; Leo Foord-Kelcey, University of British Columbia, Canada, leofk@cs.ubc.ca; Suzuran Takikawa, University of British Columbia, Canada, stakikaw@cs.ubc.ca; Nicholas Vining, University of British Columbia, Canada and NVIDIA, Canada, nvining@cs.ubc.ca; Niloy Mitra, University College London, United Kingdom and Adobe Research, United Kingdom, niloym@gmail.com; Alla Sheffer, University of British Columbia, Canada, sheffa@cs.ubc.ca.

ACM 1557-7368/2025/9-ART
https://doi.org/10.1145/nnnnnnn.nnnnnnn

Bounding Box Normalization for Human Sketches and NPR Vectors. All 3D objects are normalized to a $[-1, 1]^3$ bounding box by scaling the longest axis. We use the width and height of the human sketch to normalize the NPR vectors to fit in a $[-1, 1]^2$ bounding square.

1.2 Parameters.

We use the same parameters for all experiments in our paper. To support sketches with large perspective deviation, we set σ_1 to be a fairly large value $\sigma_1 = 10l$ (where l is the curve and stroke sampling density), ensuring that the score does not drop too fast. We set $\sigma_2 = \pi/9$ using the three sigma rule, so that once the angular difference approaches $3\sigma_2 = \pi/3$ the confidence drops to near zero. Finally, we set $\varepsilon = 0.1$ during the computation of the shape loss term L_{shape}.

1.3 Post-Inference Regularization.

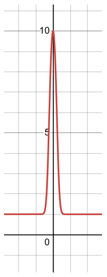
We use our learned MLP to project 3D surface vertices $\hat{p}$ to image space by first applying the camera matrix C, multiplying the result by $D(\hat{p})$, and applying perspective division. Simply rendering the resulting curves can, however, lead to several notable artifacts. First, while the deviation is subtle, it can sometimes subtly change the depth order between the projections of nearby surface curves - such that previously visible projected curves become occluded or vice versa (Figure 2bc). In this case rendering all curves results in an incoherent output. Second, changes in projection may lead to shifts in locations of T-junctions formed by partial occlusion; rendering the previous curves in this setup would lead to either gaps between the leg of the T and the top or in crossings where the leg intersects the top (Figure 2d). Similarly coincident end vertices of projected contours may no longer remain coincident. Last but not least, perspective deviation can also exaggerate small imperfections in the input contours. We eliminate these by applying a geometric smoothing step (Figure 2bc, framed corner).

We address all challenges above via a simple regularization step. We first perform a global smoothing step to better approximate local shape in the original sketch, while preserving and restoring corners ((Figure 2bc). Let p_i' be the 2D position after applying the learned distortion matrix D. We compute new 2D positions r_i for all vertices by minimizing

$$E = w_d \|r_i - p_i'\|_2^2 + w_{shape} E_{shape} + w_{coi} E_{coi} \tag{1}$$

Here the first term seeks to keep the final vertices' locations r_i at their predicted location p_i'. E_{curv} is constructed similarly to the curvature loss:

$$E_{shape} = \|r_{i+1} - r_i - R_i \frac{\|p_{i+1} - p_i\|}{\|p_i - p_{i-1}\|}(r_i - r_{i-1})\|_2^2 \tag{2}$$



$$w_{shape} = 9 * tanh(500 * (cos\theta - 1)) + 10 \tag{3}$$

Here R_i is the rotation matrix in 2D space that rotates the vector $p_i - p_{i-1}$ to $p_{i+1} - p_i$, and θ is the angle between $p_i - p_{i-1}$ and $p_{i+1} - pi$. In order to explicitly emphasize that straight lines should remain straight, we weight E_{curv} by a tanh function (inset). Finally, E_j seeks to keep curve end vertices r_i, r_j that are coincident in the original vectorized contours, coincident:

$$E_{coi} = w_j \sum_{i,j \in C, i \neq j} \|r_i - r_j\|_2^2 \tag{4}$$

Here C is the set of all contour end vertices coincident in the original vectorized contours. We then detect newly occluded curves or portions of curves and remove them from the output ((Figure 2cd; we detect occlusions in 2D by checking when curves change sides relative to one another and use depth to decide which curve portion to delete). As a final step, any T-junctions that we identified in the original vector contours are preserved by snapping any deformed legs of a previously identified T-junction that are now disconnected back into place (Figure 2de).

1.4 MLP Architecture and Training Details

We use an MLP with four hidden layers, and each hidden layer has a ReLU activation layer following it; the number of perceptrons for each layer are 16, 128, 64, and 15 for the final output layer that produces our perspective deviation matrix. We use the Adam optimizer [Kingma 2014] to train 10,000 epochs, with a learning rate of 1e-4.

1.5 Additional Evaluation Information.

We now provide additional details on our comparison setup for alternative methods; additional details on our ablations; and additional details of our three user studies.

Visual Comparison to Additional Prior Work. In addition to the comparisons to DifferSketching [Xiao et al. 2022] discussed in the main paper, we compare our method to representative approaches that address related problems that hypothetically could be used for our needs.

First, we compare our method to Zero1to3 [Liu et al. 2023], a representative approach for novel view synthesis. In theory, given an input sketch, such methods may be able to reproduce the sketched content appropriately rotated and retain the artist's perspective. Zero1to3 requires raster input, so we rasterize both the human sketch (for our main comparison; Figure 5) and our output vectors (Figure 6) to a bitmap with a white background. To generate our results, we query the model with front view rasters and horizontal camera rotations (azimuth angle) of the same degree as our output rotation. As Figure 5 shows, the pre-trained Zero1to3 model fails to generate meaningful results on many inputs. In particular, we found that Zero1to3 failed to rotate both the sketches and contour; we expect the model was not trained on similar line drawings. To confirm our experiments were done correctly, we queried their model with shaded rasters (aligned with the front view contour), and observed that this input does correctly synthesize rotated views.

In a second experiment (Figure 6), we used our initial view output as the input to Zero1to3 and performed the same rotation task (Figure 6). This experiment similarly failed to produce the expected outputs and hallucinated details that were not present in the input. After larger rotations, the original input lines become completely unstable.

Second, we compare our learned perspective results, both under original and novel viewpoints, to those produced by [Chan et al.

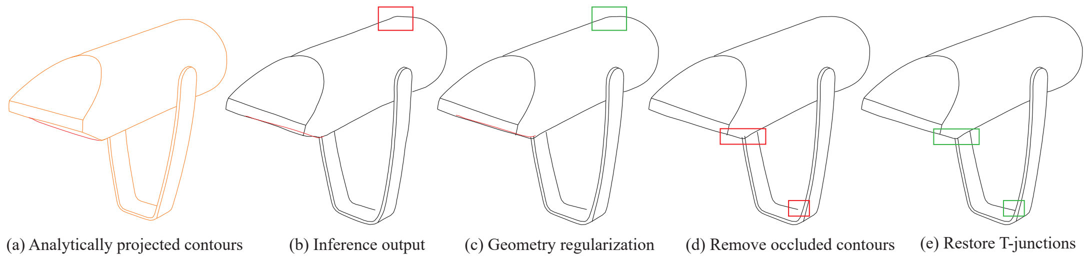

(a) Analytically projected contours
(b) Inference output
(c) Geometry regularization
(d) Remove occluded contours
(e) Restore T-junctions

Fig. 2. Topology and geometry regularization: (a) analytically projected contours; (b) Inference output. After inference, geometry regularization is applied (b,c); newly occluded contours (bcd, red) are detected and removed; (e) T-junctions that are no longer properly connected (d) are restored.

2022], a state-of-the art method for generating stylized line drawings from images. Chan et al. [2022] only requires raster input. We performed two experiments by feeding them first, the shaded images, second, the raster image of the contour lines. Their model was trained on OpenSketch-style data, aligning with our human sketch dataset. Their method focuses on stylization, and as our experiments (Figure 4) confirm, does *not* change the input perspective. As shown in Figure 4, when overlaying their outputs with the input projected vector contours, they are perfectly matched. This result holds true whether the input to their method is a shaded (Figure 4,b) or contour (Figure 4,c) render. In contrast, contours rendered by our method reproduce the perspective deviation present in the artist sketches.

Finally, we compare our outputs to those obtained by retraining *pix2pix* [Isola et al. 2017] (the PyTorch implementation) on rasterized pairs of contours and human sketches. Rather than learning perspective deviation, the resulting model produces somewhat messy, no longer stroke based, stylized outputs which retained the original analytic perspective when presented with same view or rotated contours.

In all cases, our experiments show that these alternative methods fail to reproduce human perspective deviation, showcasing both the need for a method that explicitly aims to learn perspective deviation and our method's ability to do so.

Additional Comparison vs DifferSketch Details. The authors of DifferSketching provided us with both their pre-trained models and code. Their pre-trained models are not artist-specific, and are trained on 3620 sketches. Fig. 5c in the paper shows the output of this model on input shapes in their training corpus, in a view which is identical or very similar to the ones they trained on. As the figure clearly shows, this pretrained model introduces stylization and distortion beyond the desired perspective deviation. Their model, by design, does not attempt to transfer individual artist choices. In order to try to evaluate their model's ability to mimic individual artist choices, we trained DifferSketching on smaller subsets of their corpus (all professional sketches of a given shape from a given view); and on single sketch/contour pairs (Fig. 5de, main paper). When trained on these smaller sets, output quality catastrophically deterioriated, as shown in columns d and e.

1.6 Additional Ablations and Experiments.

Perspective Comparison Across Artists. Figure 8 shows the results of applying our learned perspective from a given input sketch and object pair to another sketch of the same object from a different view by a different artist. Each of the two input sketches clearly has a very different perspective, and learning and applying this perspective produces clearly different outputs for the same camera position. However, we observe that for both views and both sets of perspectives, the outputs have cross-sections that are more circular and less elliptical compared to ground truth. Additionally, both output perspectives exhibit less foreshortening and a greater tendency towards an orthographic perspective than projected contours. This agrees with perceptual literature [Hertzmann 2024], which states that humans consistently underestimate foreshortening and prefer to draw more circular cross-sections, even under perspective.

Ablation versus 2D MLP.. We validate our design choice to construct a 3D perspective deviation function by comparing against an alternative that operates entirely in two dimensions. In this ablation, we modify our training by performing all computations exclusively in 2D. We compute our smoothess loss in two dimensions instead of three, omit the depth regularizer, and omit the 3D grid stabilization term. Fig. 7 in the main paper shows the output of this approach.

Ablation of discrete set of D matrices versus our learned function. We compare our learned distortion field versus (i) a single deviation matrix $\mathbf{D}$ to model perspective deviation throughout; or (ii) a small finite set of (9) perspective deviation matrices positioned at evenly distributed fixed points and linearly interpolated everywhere else. Results of this experiment are shown in Figure 3; neither approach satisfactorily captures human perspective deviation.

Ablation of learning on single sketch versus many sketches. We also validate our decision to use single sketch-contour pairs to learn meaningful deviation function by training our perspective MLP on larger data corpuses. We train it on one corpus consisting of all inputs drawn by the same artist (Fig. 8, main paper, left) and another containing all artists' sketches of the same object (Fig. 8, main paper, right). In both cases, the magnitude of the deviation visibly and quantitatively diminishes: in the first case the distance between the output and input contours drops to $5.69 \cdot 10^{-3}$ compared to average distance of $7.34 \cdot 10^{-3}$ for the deviations learned from single pairs (same artist different shapes). In the second case, distance drops to $3.14 \cdot 10^{-3}$ compared to average distance of $6.57 \cdot 10^{-3}$ for the deviations learned from single pairs (same shape, different artists).

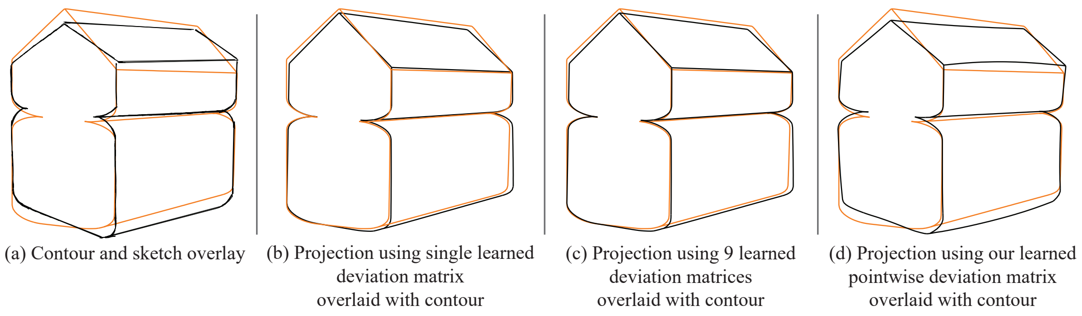

(a) Contour and sketch overlay
(b) Projection using single learned deviation matrix overlaid with contour
(c) Projection using 9 learned deviation matrices overlaid with contour
(d) Projection using our learned pointwise deviation matrix overlaid with contour

Fig. 3. Ablation. We confirm that capturing human perspective requires a smooth continuous function across object space, by comparing our outputs (d) with those generated using a single deviation matrix per input (b) or using 9 evenly-spaced and interpolated ones (c). Our output (d) reproduces human sketch (a,black) perspective much more faithfully. In each of (b,c,d) overlay of output and conotours (blue) on the left, overlay of output and sketch (green) on the right.

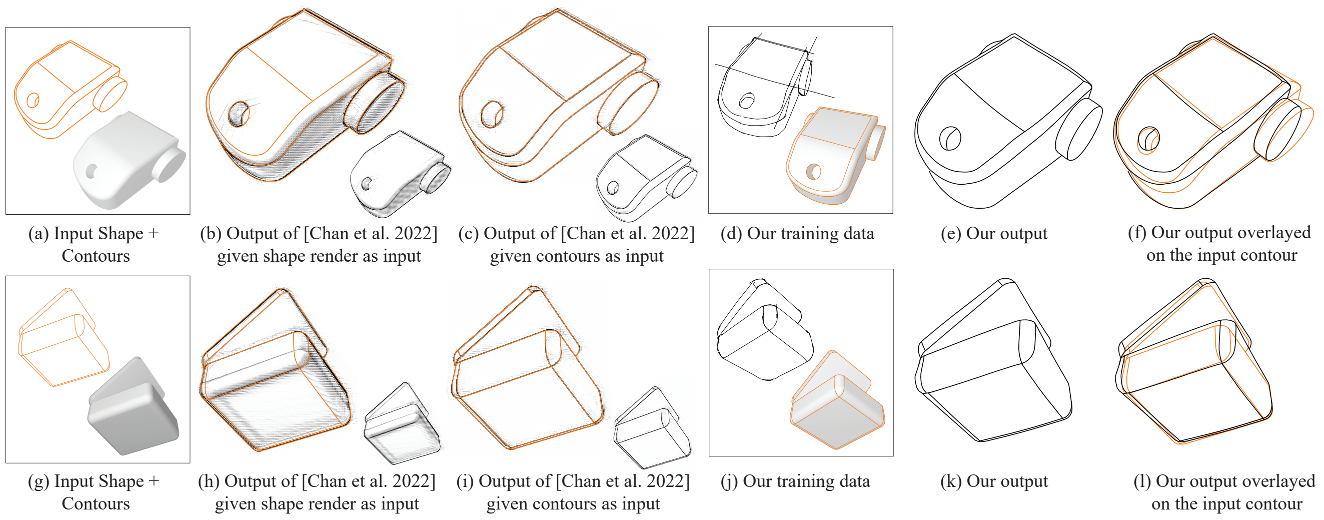

(a) Input Shape + Contours
(b) Output of [Chan et al. 2022] given shape render as input
(c) Output of [Chan et al. 2022] given contours as input
(d) Our training data
(e) Our output
(f) Our output overlayed on the input contour
(g) Input Shape + Contours
(h) Output of [Chan et al. 2022] given shape render as input
(i) Output of [Chan et al. 2022] given contours as input
(j) Our training data
(k) Our output
(l) Our output overlayed on the input contour

Fig. 4. Comparison. (a) Input shape for [Chan et al. 2022] and ours; (b) output from [Chan et al. 2022] overlaying with the shaded render of the input shape; (c) output from [Chan et al. 2022] overlaying with the input contour; (d) our paired training data; (e) our output overlaying with the shaded render of the input shape; (f) our output overlaying with the input contour. As the examples show, [Chan et al. 2022] faithfully reproduces the analytic perspective in the inputs but does not model human deviations; in contrast, we learn and reproduce the human perspective.

1.7 Applications

Our learned perspective can be part of a larger stylization or NPR pipeline. Figure 9 shows our learned perspective applied to contours that are subsequently restyled with the *CAD2Sketch* style transfer pipeline [Hähnlein et al. 2022].

1.8 Runtimes

The first training stage for our MLP takes approximately between five and ten minutes on average on an NVIDIA Tesla V100 with 16GB of memory; the second stage with augmented points from novel views takes between 1-2 hours. Our inference and regularization stage are fast and take approximately ten seconds overall.

1.9 Perceptual Study Details

We validate our method by performing four perceptual user studies.

Study 1: Generalization. Participants were shown an example drawing, consisting of an artist sketch (in black) overlaid with analytical contours (orange); and two pairs of drawings beneath it, consisting of two pairs of algorithmically generated contours (in black) with overlaid analytical contours (orange). Participants were asked the following question: "Given the relationship (distances and relative locations) between the orange and black curves in the drawing on top (A), which drawing on the bottom (B,C) exhibits a more similar relationship (distances and relative locations) between its orange and black curves to the one in the drawing above (A)? Please zoom in to see the differences. If both exhibit similar relations, select "Both". If neither has similar relations, select "Neither."" The answer options were "B", "C", "Both", and "Neither." One pair of algorithmically generated contours represents our model, trained on the input on top, applied to the contours of a different shape; the other pair of algorithmically generated contours was trained on

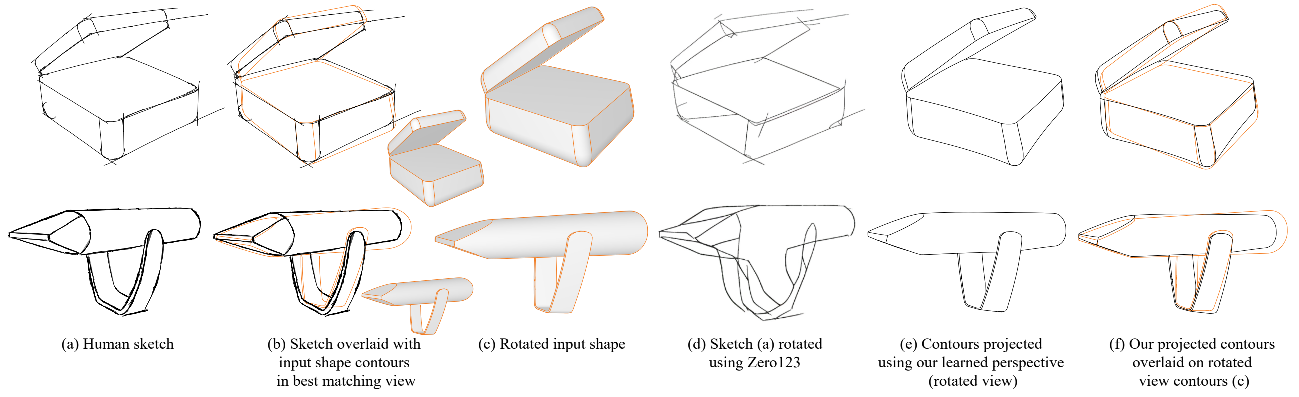

(a) Human sketch
(b) Sketch overlaid with input shape contours in best matching view
(c) Rotated input shape
(d) Sketch (a) rotated using Zero123
(e) Contours projected using our learned perspective (rotated view)
(f) Our projected contours overlaid on rotated view contours (c)

Fig. 5. Comparison versus Zero123 [Liu et al. 2023], applied to our input human sketch. Left-to-right: (a) input human sketch; (b) 3D projected contours (in blue; inset shows contours overlaid on human sketch); (c) contours in rotated view; (d) Zero123 output for this new view given sketch (a) as input; (e) contours projected using our perspective for the same novel angle. Zero123 introduces degenerate and hallucinated results and fails to preserve object shape. Our rotated view prediction correctly produces a new set of contours that align with viewer expectations and respects the perspective present in the original sketch.

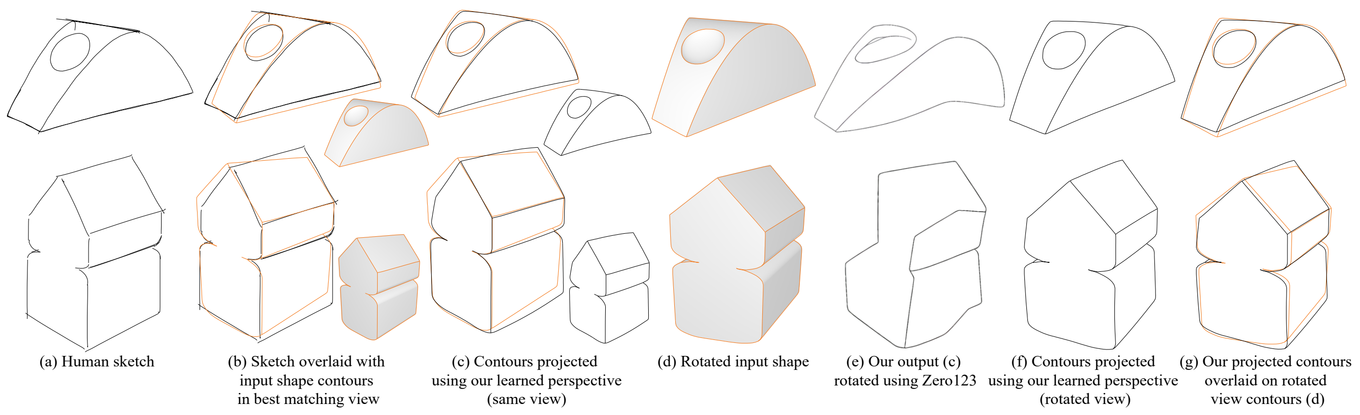

(a) Human sketch
(b) Sketch overlaid with input shape contours in best matching view
(c) Contours projected using our learned perspective (same view)
(d) Rotated input shape
(e) Our output (c) rotated using Zero123
(f) Contours projected using our learned perspective (rotated view)
(g) Our projected contours overlaid on rotated view contours (d)

Fig. 6. Comparison vs. Zero123 [Liu et al. 2023], applied this time to the same-view output of our method. Left to right: (a) input human sketch; (b) 3D projected contours (in blue); (c) our learned same-view output; (d) rotated contours under analytic projection; (e) our learned same-view output, rotated to a novel view angle using Zero123; (f) our rotated view prediction.

a different sketch/contour pair. We recruited 21 participants; each participant answered 10 questions, assigned from one of the three strata. All study data is provided in the additional supplemental material. Viewers rated the outputs trained on the input pair on top as having a more similar relation to the references 63% of the time (versus 21% other, 2% both, 14% neither); this shows that our method generalizes deviation present in training sketches to other outputs.

Study 2: Human-Like Outputs. Our second study assesses whether our output contours look more human-like than those generated with analytical perspective. Participants were shown drawings of 3D shapes and were asked to assess if they were drawn algorithmically, or by a human. Study participants were shown two images, presented side-by-side and in random order, labeled “A” and “B”. *Both* images were generated using perspective contours from our data set; one image had our learned perspective from the input human sketch applied, and the other did not. Participants were then asked: “The following drawings “A” or “B” depict the same underlying shape (from the same view). Carefully examine these drawings and identify the differences between them. Imagine the shape they depict. Which of the two drawings, “A” or “B”, would you consider more likely to have been drawn by a human?” The answer options were “A”, “B”, “Both”, and “Neither”.

We evaluated our method versus undeformed perceptual contours with a total of 54 questions, divided into three strata of 18 questions each. We recruited 45 participants; each participant answered 18 questions, assigned from one of the three strata. In total, we collected 15 answers for each question. All study data is provided in the additional supplemental material. Participants felt that our deformed contours were more likely to have been drawn by a human 71% of the time; felt that the projected contours were more likely to have been drawn by a human 12% of the time; judged both results as equally likely to be drawn by a human 8% of the time, and neither 9% of the time.

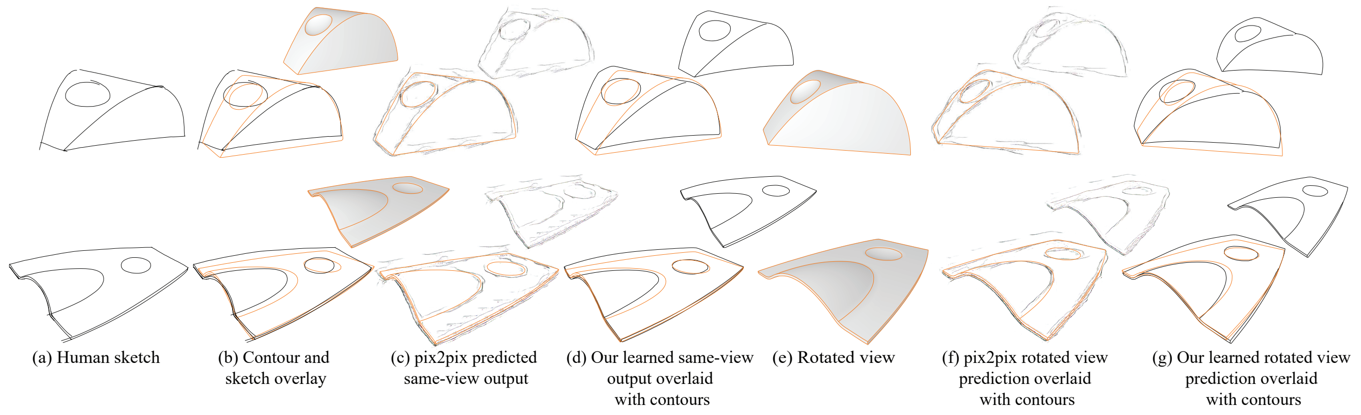

(a) Human sketch
(b) Contour and sketch overlay
(c) pix2pix predicted same-view output
(d) Our learned same-view output overlaid with contours
(e) Rotated view
(f) pix2pix rotated view prediction overlaid with contours
(g) Our learned rotated view prediction overlaid with contours

Fig. 7. Comparison to *pix2pix* [Isola et al. 2017], a conditional GAN image-to-image translator, highlights the fact that translation methods are not suited for learning perspective from sparse data. *Pix2pix* trained on our training corpus fails to apply the input human sketch perspective from (a) to the 3D projected contours (b) and introduces spurious lines (c, f); our method correctly learns and applies input sketch perspective to both same- and novel-view contours.

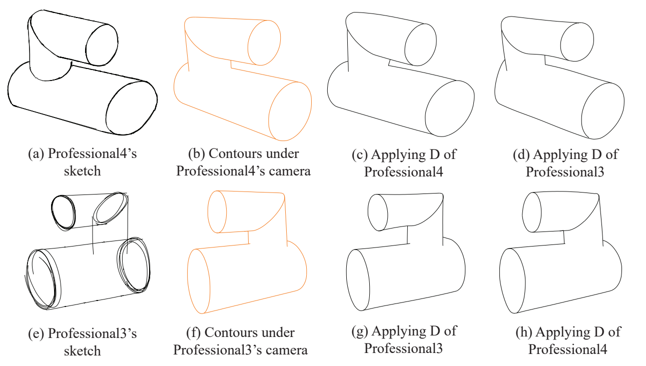

(a) Professional4's sketch
(b) Contours under Professional4's camera
(c) Applying D of Professional4
(d) Applying D of Professional3
(e) Professional3's sketch
(f) Contours under Professional3's camera
(g) Applying D of Professional3
(h) Applying D of Professional4

Fig. 8. Perspective Comparison: Given two artist sketches (a,e) with different camera views (b,f) we show the results of applying the perspectives learned from each sketch to the original (c,g) and other sketches (d,h) views. Notice how both sketches generate more circles and a closer to orthographic perspective.

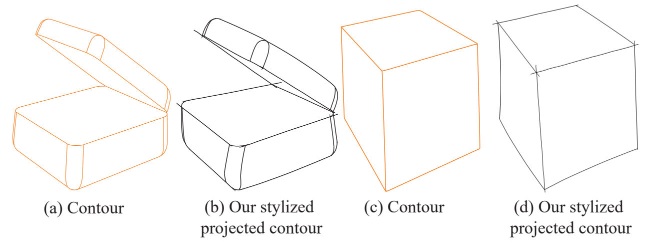

(a) Contour
(b) Our stylized projected contour
(c) Contour
(d) Our stylized projected contour

Fig. 9. Our perspective can be combined with any stylization method; here we style the contours using [Hähnlein et al. 2022].

Study 3: Accuracy Comparison versus 2D MLP. We compared our method's output directly against the 2D design alternative. Participants were shown a reference line drawing (top) and two candidate drawings (A, B; bottom), one generated by our full method and one by the 2D MLP alternative as described above. Participants were then asked the following question: "In the figure below, you are shown three line drawings, one at the top (Reference) and two below (A and B). The Reference drawings show a front-view drawing of an object, while drawings A and B depict the same shape after being rotated by a certain degree. Carefully examine these line drawings and identify the differences between them. Which of the two line drawings, "A" and "B", would you consider *more accurately* depicts the shape of the reference image after rotation? If both are equally accurate, select "Both"; if neither is accurate, select "Neither"." Options were "A", "B", "Both", and "Neither".

We recruited 15 participants in one tranche, and showed them 12 questions each, for a total of 180 responses. Participants ranked ours as more precise 79% of the time (vs 6% 2D MLP, 7% both, 8% neither). This confirms our assertion that while the 2D MLP model can replicate the artist deviation in the input view, outputs are inconsistent across views as the learned model significantly deforms the contours.

Study 4: Generalization Comparison vs 2D MLP. Our final study used the same question and experiment setup as our first generalization study, but compared our outputs to that of the 2D MLP described above. For this study, we recruited 21 participants; each participant answered 5 questions, assigned from one of the three strata. Viewers rated the outputs trained on our model as having a more similar relation to the input pair on top 72% of the time (versus 17% 2D MLP, 6% both, 5% neither).

References

Ilya Baran, Jaakko Lehtinen, and Jovan Popovic. 2010. Sketching Clothoid Splines Using Shortest Paths. *Computer Graphics Forum* (2010). https://doi.org/10.1111/j.1467-8659.2009.01635.x

Pierre Bénard, Aaron Hertzmann, et al. 2019. Line drawings from 3D models: A tutorial. *Foundations and Trends® in Computer Graphics and Vision* 11, 1-2 (2019), 1–159.

Caroline Chan, Frédo Durand, and Phillip Isola. 2022. Learning to generate line drawings that convey geometry and semantics. In *CVPR*.

Yulia Gryaditskaya, Mark Sypesteyn, Jan Willem Hoftijzer, Sylvia Pont, Frédo Durand, and Adrien Bousseau. 2019. OpenSketch: A Richly-Annotated Dataset of Product Design Sketches. *ACM Transactions on Graphics (Proc. SIGGRAPH Asia)* 38 (11 2019).

Olga Guțan, Shreya Hegde, Erick Jimenez Berumen, Mikhail Bessmeltsev, and Edward Chien. 2023. Singularity-Free Frame Fields for Line Drawing Vectorization. *Computer*

Graphics Forum (2023). https://doi.org/10.1111/cgf.14901

Felix Hähnlein, Changjian Li, Niloy J. Mitra, and Adrien Bousseau. 2022. CAD2Sketch: Generating Concept Sketches from CAD Sequences. *ACM Trans. Graph.* 41, 6, Article 279 (Nov. 2022), 18 pages. https://doi.org/10.1145/3550454.3555488

Aaron Hertzmann. 2024. Toward a theory of perspective perception in pictures. *Journal of Vision* 24, 4 (04 2024), 23–23.

Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. 2017. Image-to-Image Translation with Conditional Adversarial Networks. *CVPR* (2017).

Diederik P Kingma. 2014. Adam: A method for stochastic optimization. *arXiv preprint arXiv:1412.6980* (2014).

Ruoshi Liu, Rundi Wu, Basile Van Hoorick, Pavel Tokmakov, Sergey Zakharov, and Carl Vondrick. 2023. Zero-1-to-3: Zero-shot One Image to 3D Object. arXiv:2303.11328 [cs.CV]

Stefano Nuvoli, Alex Hernandez, Claudio Esperança, Riccardo Scateni, Paolo Cignoni, and Nico Pietroni. 2019. QuadMixer: Layout Preserving Blending of Quadrilateral Meshes. *ACM Trans. Graph.* 38, 6, Article 180 (nov 2019), 13 pages. https://doi.org/10.1145/3355089.3356542

Luming Tang, Menglin Jia, Qianqian Wang, Cheng Perng Phoo, and Bharath Hariharan. 2023. Emergent Correspondence from Image Diffusion. In *Thirty-seventh Conference on Neural Information Processing Systems*. https://openreview.net/forum?id=ypOiXjdfnU

Chufeng Xiao, Wanchao Su, Jing Liao, Zhouhui Lian, Yi-Zhe Song, and Hongbo Fu. 2022. DifferSketching: How Differently Do People Sketch 3D Objects? *ACM SIGGRAPH Asia* 41, 4 (2022), 1–16.