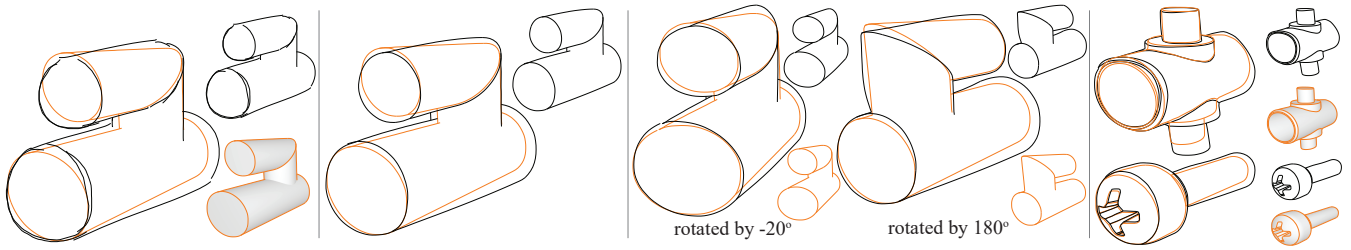
Capturing Non-Linear Human Perspective in Line Drawings

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



(a) Human sketch overlaid on input shape contours (b) Contours projected using our learned perspective (same view) (c) Contours projected using our learned perspective (other views) (d) Applying learned perspective to new shapes

Fig. 1. Human sketches (a, black) use a perspective projection that deviates from the analytical perspective model. We visualize this *perspective deviation* by overlaying the sketch over contours projected using best approximating analytic perspective (a, orange); sketch and contours shown separately as insets. We learn a model of this human deviation, producing projected contours (b-d, black) that have similar relation to their analytical counterparts. Our deviation model generalizes to other views (c) and shapes (d), maintaining similar relationship between the two sets of contours.

Artist-drawn sketches only loosely conform to analytical models of perspective projection; the deviation of human-drawn perspective from analytical perspective models is persistent and well documented, but has yet to be algorithmically replicated. We encode this deviation between human and analytic perspectives as a continuous function in 3D space and develop a method to learn it. We seek deviation functions that (i) mimic artist deviation on our training data; (ii) generalize to other shapes; (iii) are consistent across different views of the same shape; and (iv) produce outputs that appear human-drawn. The natural data for learning this deviation is pairs of artist sketches of 3D shapes and best-matching analytical camera views of the same shapes. However, a core challenge in learning perspective deviation is the heterogeneity of human drawing choices, combined with relative data paucity (the datasets we rely on have only a few dozen training pairs). We sidestep this challenge by learning perspective deviation from an individual pair of an artist sketch of a 3D shape and the contours of the same shape rendered from a best-matching analytical camera view. We first match contours of the depicted shape to artist strokes, then learn a spatially continuous local perspective *deviation* function that modifies the camera perspective projecting the contours to their corresponding strokes. This function retains key geometric properties that artists strive to preserve when depicting 3D content, thus satisfying (i) and (iv) above. We generalize our method to alternative shapes and views (ii,iii) via a self-augmentation approach that algorithmically generates training data for nearby views, and enforces spatial smoothness and consistency across all views. We compare our results to potential alternatives, demonstrating the superiority of the proposed approach. Code and models will be released upon acceptance.

CCS Concepts: • **Computing methodologies** → **Shape analysis**; **Image manipulation**.

Additional Key Words and Phrases: human perspective, sketching, non-linear perspective, line drawing

ACM Reference Format:
Jinfan Yang, Leo Foord-Kelcey, Suzuran Takikawa, Nicholas Vining, Niloy Mitra, and Alla Sheffer. 2025. Capturing Non-Linear Human Perspective in Line Drawings. *ACM Trans. Graph.* 1, 1 (September 2025), 11 pages. https://doi.org/10.1145/nnnnnnn.nnnnnnn

1 Introduction

Line drawings, or sketches, are a simple and powerful medium for conveying shapes between humans [Eissen and Steur 2008, 2011] and have long been hailed as a potential interface for bidirectional human-computer communication [Sutherland 1964]. Unfortunately, computer generated sketches lack the communication power of human ones, and computers are not yet able to fully parse human sketches [Bessmeltsev and Liu 2024].

When sketching a 3D object (e.g. Figure 1a), artists make three key choices: (i) which 3D curves to draw; (ii) what 3D-to-2D projections to employ to project the 3D curves on a 2D medium; and (iii) how

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

to stylize the 2D projections of the chosen curves. While the first and last items have been extensively researched (Section 2), the second one [Hertzmann 2024] is less studied and rarely modeled. Computer graphics applications typically use an analytical model of projection (e.g. single vanishing-point perspective or orthographic projection); in contrast, artist projection approximates but does *not* match any such analytical model [Hertzmann 2024; Pepperell and Haertel 2014]. Researchers [Gombrich 1951; Hertzmann 2024; Pepperell and Haertel 2014; Singh 2002] speculate that the deviation between artist-employed and analytical perspective models is due to a combination of artists using deliberate distortions as a key mechanism to emphasize essential features and reduce cognitive load, thereby enabling more effective communication; and inherent human imprecision. We refer to this discrepancy between analytical and artist-drawn perspectives as *human perspective deviation*.

Although this human perspective deviation has been consistently observed, we are aware of only one prior effort to model it. Specifically, Xiao et al. [2022] note the discrepancy between artist-sketched and analytically projected shape contours, and use a 2D multi-layer perceptron (MLP) to learn to deform the analytical contours towards their sketched counterparts. However, this approach collectively models all three above-mentioned drawing choices and does not offer a factorized treatment. Even when applied to contours in their training corpus, their model, trained on professional artist line-drawings, produces unnatural looking deviations containing high-frequency noise, atypical of artist choices (see Section 4).

We develop a deviation model that is designed to reflect artist perspective choices. Since there is no unified perceptual model of artist perspective deviation upon which we can draw, we propose a learning-based approach and use pairs of artist sketches, dominated by contour strokes, and renders of analytically projected contours of the artist-depicted shapes aligned to match the artist's chosen view as training data (Figure 1a). Given this data, we seek deviation models that (i) mimic artist deviation on our training data; (ii) generalize to other shapes; (iii) are consistent across different views of the same shape; and (iv) produce outputs that appear human-drawn.

A core challenge in learning perspective deviation that reflects artist choices is the heterogeneity of human drawing choices, combined with relative data paucity (the datasets we rely on have less than a couple of hundreds of training pairs). Rather than attempting to learn a single unified probabilistic model that can generate outputs that fit individual styles on demand, we learn models that reflect the perspective in individual sketches. While our model can be extended to learning from multiple sketches at once (Section 4). such "averaged" models tend to be less expressive.

We align the renderer camera parameters to best match the artist sketches and algorithmically match the rendered projected contours to artist strokes. We then use these correspondences to learn a *perspective deviation* function that maps contours to their matching strokes. Our experiments (Section 4) suggest that generalization across views and shapes requires learning a 3D, rather than a 2D, perspective deviation function. We therefore model perspective deviation using a 3D spatially varying multiplicative matrix that adjusts the analytical projection matrix; specifically, we use an MLP to define, for every point in 3D space, an associated deviation matrix. To generalize the output perspective deviation across multiple camera views, we use a self-augmentation process where we first learn artistic deviation from the sketch contour pairs above; then, we learn a deviation function across both the original input and synthetic training examples generated from the original deviation.

We train our method on 169 sketch/shape pairs sourced largely from [Gryaditskaya et al. 2019] and [Xiao et al. 2022], and show the results of applying these models to different shapes and views throughout the paper and the supplementary material. Our outputs retain the perspective of the input sketches when applied to their corresponding training shapes, both from original and novel camera views (e.g. Figure 1bc), and translate across shapes (e.g. Figure 1d). We validate our results both quantitatively and via perceptual studies, and demonstrate their superiority compared to existing and potential alternatives (Section 4).

In summary: our main contribution is to learn human perspective deviation models that capture the characteristics of individual artist choices and that generalize across views and shapes. In the process, we contribute to the understanding of human employed perspective in line drawings, and identify the relevant factors in modeling human perspective deviation. Beyond addressing the technical challenge of modeling perspective deviation for individual artists and inputs, our approach advances the understanding of how computational models can replicate human perception.

2 Related Work

Sketching with Perspective. Analytical, linear, perspective projection is ubiquitously used for precise depiction of 3D content from a given viewpoint in manually drafted technical drawings and computer generated renders. Artists are often encouraged [Eissen and Steur 2008] to aim for analytic perspective and historically have attempted to accurately reproduce it; for example, it is speculated that the Dutch masters used *camera obscura* to capture this perspective [Steadman 2002]. However, various user studies have demonstrated that artists almost never use precise linear perspective for sketching [Koenderink et al. 2016; Pepperell and Haertel 2014]. Some deviations arise due to faulty estimation [Kemp 1991; Kubovy 1986] while others are the result of artists intentionally using varying (local) perspective [Coleman et al. 2005; Hertzmann 2024; Pepperell and Haertel 2014; Schmidt et al. 2009a; Singh 2002]. Research on human perception of 2D depiction of 3D objects strongly suggests that humans make systematic errors when estimating foreshortened shapes and dimensions even for simple tasks [Koenderink and van Doorn 1991; Nicholls and Kennedy 1995; Reith and Liu 1995; Taylor and Mitchell 1997]. While studies suggest that using scaffolds for guidance [Hennessey et al. 2017; Schmidt et al. 2009a] improves the alignment of artist and analytic perspectives, artists often forego scaffolds when sketching free-hand.

Non-Photorealistic Rendering (NPR). Numerous NPR methods explore the use of line drawings for effectively conveying shape, and investigate *which* surface curves or contours to draw, e.g. [Cole et al. 2008] and *how* to stylize their 2D projections, e.g. [Hertz et al. 2023; Wang et al. 2024]. DeCarlo et al. [2003] generate collections of curves that emphasize object features; recent variants (e.g., Hähnlein et al. [2022], Liao et al. [2024]) convert CAD sequences to concept sketches, blending geometric precision with stylistic abstractions to

emulate human sketches. All above methods explicitly or implicitly rely on traditional analytical perspective.

Advances in machine learning have opened new possibilities for synthesizing line drawings from 3D shapes, including neural style transfer [Gatys et al. 2016], image-to-image translation [Isola et al. 2017], or using CLIP scoring to distill shape abstractions [Vinker et al. 2022]. Liu et al. [2020] propose a neural framework for generating contour lines directly from 3D models, showcasing the ability of neural networks to learn artistic cues. Chen et al. [2022] propose neural variants for synthesizing line drawings that simultaneously capture geometric accuracy and semantic meaning. These methods demonstrate the potential of machine learning to mimic artistic styles, but are mainly trained on synthetic renderings of 3D models using analytical projections, and learn styles rather than human perspective. These methods do not address the perspective distortion humans naturally introduce, as particularly evident when comparing NPR outputs with human-drawn sketches (see supplementary).

Closest to our work, DifferSketching [Xiao et al. 2022] use a data driven approach to learn a 2D difference function between projected contours and artist sketches (assumed to be drawn in the same view) from a large collection of paired contours and sketches. As our experiments show, despite relying on professional sketches, their method introduces notable high-frequency distortion when applied even to their training inputs. Training the method on a subset of the dataset, e.g., drawings of the same shape, or single accurate drawings increases rather than decreases this distortion. We compare theirs with our approach on their dataset in Section 4.

Sketch-Based Modeling. Sketch-based modeling systems focus on creating 3D models from 2D sketches; see [Bessmeltsev and Liu 2024; Choi et al. 2024; Olsen et al. 2009] for comprehensive surveys. Many such systems ignore the problem of perspective entirely, and use 2D contour curves drawn in the image plane as input [Dvorožňák et al. 2020; Li et al. 2018; Nealen et al. 2007; Zhang et al. 2022]; they create 3D geometry by inflating these contours and incorporate depth either by explicit annotation or relying on stroke draw order. These methods implicitly assume orthographic perspective. Other methods rely on sketched input from multiple views, where artists sketch strokes from different viewpoints onto existing 3D geometry (e.g., [De Paoli and Singh 2015; Igarashi et al. 1999; Kara and Shimada 2007]). Several methods require users to manually specify analytic perspective "scaffolds" to regularize perspective [Schmidt et al. 2009b], or use strokes to define transient surfaces to recover 3D curves [Bae et al. 2008].

Works addressing 3D reconstruction from single sketches observe that user inputs have inexact perspective, but seek to correct or sidestep this inexactness by detecting and enforcing different regularization cues [Shao et al. 2012; Xu et al. 2014], construction lines [Gryaditskaya et al. 2020], or local symmetries [Hähnlein et al. 2022]. Recent developments have shifted towards data-driven approaches by leveraging 3D datasets, synthetically rendered with a pinhole camera model with either non-photorealistic rendering or manual contour tracing, to create training and test data [Li et al. 2022; Liu et al. 2024, 2023]. When applied to human sketches, they frequently produce unexpected or inconsistent outputs, highlighting the need for frameworks that explicitly incorporate human perceptual biases.

3 Method

3.1 Overview

Modelling Human Perspective. As discussed in Section 2 while artistic perspective typically deviates from analytic one, this deviation is relatively subtle, and changes gradually across drawings, with parts of the content drawn larger or smaller relative to their analytical projection. To capture these properties, we model artists' perspective as a 3D deviation from a standard pinhole camera projection that smoothly varies across 3D space (we treat orthographic projection as a special case of perspective with the camera placed at infinity).

Setup. In computer graphics, projection is handled analytically through the camera projection matrix $\mathbf{P}$ and the modelview matrix $\mathbf{M}$. For simplicity we refer to the product of these matrices $\mathbf{C} = \mathbf{PM}$ as the camera projection matrix. When applying a perspective projection, any point $\mathbf{p} \in \mathbb{R}^3$ on a 3D shape is mapped, working in homogeneous coordinates, to $\mathbf{p} \rightarrow \mathbf{C}[\mathbf{p}; 1]$.Then, $\mathbf{C}[\mathbf{p}; 1]$ is converted to 2D points in image space by performing a perspective divide (see [Foley et al. 1996]).

We model human perspective deviation as a *local multiplicative adaptation* of the projection operator. Empirically, we found that human perspective is best modeled in a normalized world coordinate space, and not in image space. We therefore model human perspective at $\mathbf{p}$ as $\mathbf{p} \rightarrow \mathbf{D}(\mathbf{p})\mathbf{C}[\mathbf{p}; 1]$, where a $\mathbf{D}(\mathbf{p})$ is a 4×4 deviation matrix, followed by perspective division by p_w. Input shapes are normalized so their origin is at $(0, 0, 0)$ and the shape is within the $[-1, 1]^3$ unit box. We parameterize $\mathbf{D}$ using these normalized world coordinates.

We represent $\mathbf{D}$ as an MLP that takes in 3D (normalized) coordinate information and outputs 15 values. We reshape these values to a 4×4 matrix, with the last element set to 1. Given this one-to-one relation, we use $\mathbf{D}$ to represent, based on context, both the human perspective matrix as well as the MLP output.

Additional Deviation Properties. In addition to expecting our learned deviation functions $\mathbf{D}$ to change gradually across the input shapes, we aim for them to preserve core properties of the projections of the depicted curves such as slope and shape; prior research on sketch analysis [Shao et al. 2012; Xu et al. 2014] suggests that artists seek to preserve these properties in their sketches. Hence, while artists' local or global perspective deviation may be quite substantial, we do not require our deviation to be minimal across the board. Last but not least, we seek learned deviations that generalize across views and similar shapes; to this end, we explicitly seek deviations that are similar across similar views and are spatially smooth even when away from the input shape surface.

Algorithm Overview. We learn a deviation matrix $\mathbf{D}(\mathbf{p})$, (illustrated in the inset) from one or more pairs of a source sketch along with its corresponding 3D object and an estimated camera matrix that best aligns the sketch and camera views. We break the task into the following stages (Figure 2): (i) matching between sketched

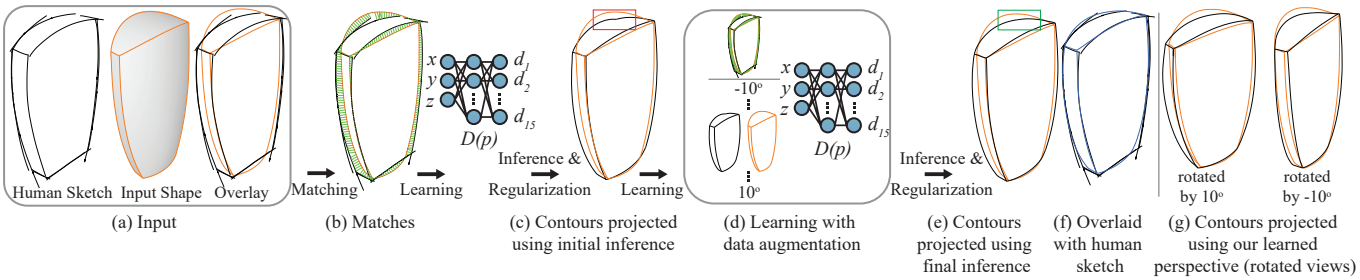
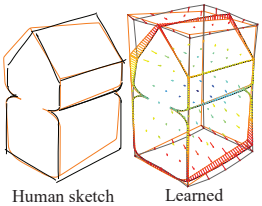


Fig. 2. Algorithm overview: Given an input sketch (black) and corresponding analytically projected 3D shape contours (orange) in a matching view (a), we match the contours to artist strokes (b). We then use a two-step process to learn the deviation between the contour and sketch projections (c-e): we learn an initial deviation function $\mathbf{D}(\mathbf{p})$ that balances satisfying the computed matches against adherence to core deviation properties and apply the learned deviation to the input contours (c); we augment our learning data with synthetic sketch/contour pairs and re-learn a deviation that best fits the augmented training set (d,e). The contours projected using our deviation align with the artist's strokes (f). Our deviation consistently generalizes to other views (g).

strokes and 3D shape contour curves computed and analytically projected using the estimated camera; (ii) modeling and learning a human perspective deviation function, as a spatially-conditioned MLP, that moves contour points toward their matched stroke positions; (iii) regularizing the curves obtained by applying the learned perspective deviation, enforcing topological and other global constraints; and (iv) self-augmentation to create additional data to retrain the MLP to better align $\mathbf{D}(\mathbf{p})$ with our global priors and better generalize to novel views. At inference, given a new shape and camera combination, we apply the learned perspective deviation function, followed by regularization (Figure 2,right). We describe the method's stages below; see supplemental material for additional details and hyper-parameter settings.



3.2 Preliminaries

We use the calibrated camera information to generate vector format occluding contours, sharp features, and surface boundaries of the input shape; referred to as *contours* throughout. We vectorized the input sketches, if needed, and resample both contour curves and sketch strokes using a fixed sampling interval rate (we denote the sampling interval length as l). In the following, unless stated otherwise, the term *curve* refers to a resampled projected contour curve and the term *stroke* refers to a resampled sketch stroke.

We use $\hat{p}$ as the 3D locations of the 2D contour vertices p. Given a vertex $\hat{p}$ and a projection matrix $\mathbf{D}$, the function $\text{proj}(\hat{p}, \mathbf{D})$ is the 2D vertex computed by taking the matrix-vector product $\mathbf{DC} \cdot \hat{p}$ in homogenous coordinates, and applying the perspective divide.

3.3 Matching Contour Curves to Sketch Strokes

First, we establish correspondences between vertices on the projected contour curves and vertices on the vectorized human sketch strokes. Intuitively, we seek to match contour vertices to nearby stroke vertices with similar tangents; we refer to this property as *compatibility*. One of the challenges in computing these correspondences is that they are not one-to-one (Fig 2ab). Sketches may contain strokes, or portions of strokes, with no matching contours, due to oversketching [Van Mossel et al. 2021]; contour curves may be depicted using multiple strokes, and some of the curve vertices may not have corresponding stroke locations. At the same time, while we do not expect exact one-to-one contour to stroke matches, we generally expect segments formed by consecutive contour vertices to match similarly consecutive stroke vertices, or in cases where a curve may correspond to multiple artist strokes or stroke sections, to match pairs of stroke vertices that form roughly parallel line segments; we refer to this property as *consistency*.

This combination of requirements differs from the classical image space matching setting where users seek to match points with similar features with no requirement for any consistency between the matches. Dropping the consistency requirement makes the problem much simpler but results in matches that do not align with artist intent (Figure 3b uses DIFT [Tang et al. 2023], Figure 3c uses our compatibility score). We account for both compatibility and consistency by formulating matching as an instance of the classical Hidden Markov Model (HMM) problem [Yoon 2009] (Figure 3de).

Given a contour curve $S := \{p_1, \dots p_n\}$, we first form, for each contour vertex p_i, a candidate set of potential matching stroke vertices $Q := \{q_1, \dots q_m\}$ on the human sketch based on the distance between these vertices in 2D image space. We then evaluate potential matches $(p_i, q_{j(i)})$ using a combined vertex-to-vertex *compatibility* score $S^v(p_i, q_{j(i)})$ and a *consistency* score $S^e(p_i, p_{i+1}, q_{j(i)}, q_{j(i+1)})$, which assesses compatibility between potential matches of consecutive curve vertices. Using the classical HMM formulation, the overall score given by matching the vertices of S to the vertices of Q is:

$$M(S, Q) := \prod_i S^v(p_i, q_i) S^e(p_i, p_{i+1}, q_i, q_{i+1}). \tag{1}$$

Using a product, rather than a sum, discourages outlier matches. We compute the matches for each curve that maximize $M(S, Q)$ using the Viterbi algorithm [1967]. To obtain a valid solution, we exclude any vertices with empty matching candidate sets, and any edges emanating from such vertices, from the per-curve score.

Compatibility Score (S^v). Given a paired curve vertex p and stroke vertex q, we define the score of using q as the match of p as a function of two terms, designed to be on the same scale, as:

$$\begin{aligned} d_a &= \|p - q\|_2 \\ d_t &= 1 - |\mathcal{T}_S(p) \cdot \mathcal{T}_Q(q)|. \end{aligned} \tag{2}$$

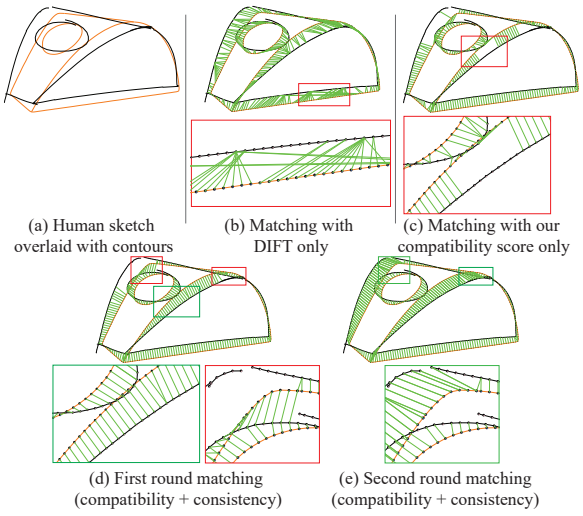


Fig. 3. We seek for artist-intended matches between contour curves and sketch strokes (a). Using only vertex-to-vertex matching scores whether feature based (DIFT [Tang et al. 2023]) (b) or our compatibility based (c) produces locally optimal but globally poor matches; accounting for consistency (d) produces better matches at contour level but can still lead to instances where multiple curves match the same stroke (see inset zoom). Our second matching step resolved these undesirable many-to-one matches (e).

The first term is the absolute distance between them, while the second term measures the similarity of the vertices' tangents and encourages matches that have similar orientations: here, $\mathcal{T}_S(p)$ is the normalized tangent vector of S at p (respectively for Q at q). The overall score for matching $q \rightarrow p$ is thus:

$$S^v(p,q) = e^{-(d_a+d_t)^2/2\sigma_1^2}. \tag{3}$$

Consistency Score (S^e). We formulate consistency purely geometrically, and prioritize matching contour edges to pairs of vertices where the line connecting these vertices has similar length and orientation to the edge ones. Given a pair of consecutive vertices p_i and p_{i+1} (potentially) respectively matching a pair of vertices $q_{j(i)}$ and $q_{j(i+1)}$, we measure consistence $S^e(p_{i,i+1}, q_i|q_{i+1})$ as:

$$\begin{aligned} d_p &= \|(p_{i+1} - p_i) - (q_{j(i+1)} - q_{j(i)})\|_2 \quad (4)\\ S^e(p_{i,i+1}, q_i|q_{i+1}) &= e^{-d_p^2/2\sigma_1^2}. \end{aligned}$$

While we expect different contours to be depicted using different strokes, enforcing global matching constraints within the optimization framework above would dramatically increase algorithm complexity, making the matching problem NP-complete. Instead, we compute matches independently for each contour curve (Fig 3d) and then identify and resolve cases where multiple curves or portions of curves are matched to the same stroke/stroke portion. We first identify vertices from different curves that match the same stroke vertex q (Figure 3d,inset). We then do another matching round, where for these conflicted curve vertices we double the distance range to search in when finding candidate sets and exclude their previously matched stroke vertices from the candidate set. For each curve vertex in conflict, we then assign either its first round stroke match, or its new second round match, depending on which solution minimizes its score (Equation (3)), see Figure 3e.

3.4 Learning Human Perspective

We use the matching results to learn a deviation function that approximately projects the original 3D locations $\hat{p}_i$ of the projected contour vertices p_i to their matching 2D stroke vertices $q_{j(i)}$, while preserving contour slope, shape, and spatial smoothness.

Formally, let $P := \{p_1, \dots p_n\}$ be all vertices on the projected contours; let $Q := \{q_1, \dots q_n\}$ be the matched vertices in the human vector sketch; for notational simplicity, we replace $q_{j(i)}$ with q_i. By abuse of notation, we identify each vertex p_i with a distortion matrix $\mathbf{D_i}$ that is the output of the MLP at $\hat{p}_i$ (i.e., $\mathbf{D_i} = \mathbf{D}(\hat{\mathbf{p}}_\mathbf{i})$), and denote the collection of all such matrices $\{\mathbf{D_1}, \dots \mathbf{D_n}\}$. Our learned MLP takes as input the coordinate $\hat{p}$, in the normalized object space $[-1, 1]^3$, and outputs a pointwise deviation matrix $\mathbf{D}_{\hat{\mathbf{p}}}$.

Overall Loss. We learn an MLP that minimizes the following overall loss function,

$$L(D) := w_1 \cdot L_{data} + w_2 \cdot L_{shape} + w_3 \cdot L_{slope} + w_4 \cdot L_S + w_5 \cdot L_{depth}. \tag{5}$$

The first term aims to project 3D contour vertices close to their matching stroke vertices; the second and third terms preserve contour shape and slopes; the fourth term ensures deviation smoothness in 3D space; and the last, regularizer, term seeks to avoid depth instabilities by preserving relative point depth under our distortion function. We set $w_1 = 0.001, w_2 = 10, w_3 = w_4 = 1, w_5 = 10^{-5}$. This setting prioritises shape preservation above all other properties, and prioritizes our general priors about deviation above the data term. A small depth regularizer is sufficient to avoid instabilities.

Data Loss (L_{data}). The term moves projected contour vertices $\text{proj}(\hat{p}_i, \mathbf{D}_i)$ toward their counterparts q_i on the human sketch, and is defined as,

$$L_{data} := \frac{1}{avg_l} \frac{1}{n} \sum_{i \in n} \alpha_i \|\text{proj}(\hat{p}_i, \mathbf{D}_i) - q_i\|_1 \tag{6}$$

where $avg_l = \frac{1}{n} \sum_{i \in n} \|p_i - q_i\|_1 + \epsilon$. We normalize each individual term by a corresponding confidence value α_i, and normalize the entire data term by the average distance between matching vertices (we add ϵ to avoid division by zero for perfect matches).

Confidence. While we seek to project 3D contour vertices $\hat{p}_i$ toward their stroke matches q_i, the matches we compute may be imperfect due to factors such as oversketching (e.g., top of shampoo bottle in Figure 2b). We therefore associate confidence values α_i with each matched pair and use these to control the degree to which they are enforced. We base these values on the difference between intrinsic contour and stroke shape at the respective 2D vertices. Since artist sketches tend to preserve curve shape, mismatches between local contour/stroke curve shapes point to potential matching errors. We use polyline angles $\measuredangle(p_i)$ and $\measuredangle(q_i)$ at p_i and q_i respectively as proxy for shape (since we use uniform sampling these serve as an approximation of curvature). We define per-vertex confidence as:

$$\alpha_i = e^{-(\measuredangle(p_i) - \measuredangle(q_i))^2/2\sigma_2^2}. \tag{7}$$

Shape Loss (L_{shape}). The term aims to ensure our deviation preserves curve shape penalizing non-uniform scale and shear (see

[Araújo et al. 2022]), while allowing for uniform scale, as:

$$L_{\text{shape}} := \sum_{i \in V} \sum_{j,k \in N(i); j \neq k} (1 - \alpha + \varepsilon) \| (\text{proj}(\hat{p}_k, \mathbf{D}_k) - \text{proj}(\hat{p}_i, \mathbf{D}_i) - (\mathbf{R_i} \frac{\|p_k - p_i\|}{\|p_j - p_i\|} (\text{proj}(\hat{p}_j, \mathbf{D}_j) - \text{proj}(\hat{p}_i, \mathbf{D}_i))) \|^2. \quad (8)$$

Here, V is the set of all vertices lying on the interior of projected contour curves, and $N(i)$ are all neighbouring vertices of the vertex i; α is the minimum confidence value α_i of the three consecutive vertices p_j, p_i, p_k forming the two edges; R_i is the rotation matrix in 2D space that rotates the vector $p_j - p_i$ to $p_k - p_i$. We choose *not* to penalize uniform scale as artists employ deviation that by design changes feature scale[Hertzmann 2024].

Slope Loss (L_{slope}). The term aims to preserve the slopes of the projected contours under our learned deviation

$$L_{\text{slope}} := \frac{1}{n-1} \sum_{i=2}^{n} \left(\hat{n}_i \cdot \frac{(\text{proj}(\hat{p}_i, \mathbf{D}_i) - \text{proj}(\hat{p}_{i-1}, \mathbf{D}_{i-1}))}{\|p_i - p_{i-1}\|} \right)^2 \quad (9)$$

where $\hat{n}_i$ is the 2D normal of the projected contour edge $(p_i - p_{i-1})$. While artists do not strictly preserve slopes, the changes they introduce are typically subtle.

Smoothness Loss (L_S). The term encourages smooth changes in the distortion matrix across view space and is computed over a set of points $\hat{S}$ containing all contour vertices $\hat{p}$ as well as the vertices of a dense grid spanning our 3D domain box $[-1, 1]^3$. Given pairs if vertices $s_i \in \hat{S}$ and $s_j \in \hat{S}$, we express smoothness as the expectation for $\mathbf{D}_i$ and $\mathbf{D}_j$ to be increasingly similar for nearby vertices:

$$L_S := \frac{1}{|\hat{S}|} \sum_{s_i \in \hat{S}} \sum_{s_j \in \hat{S}, i \neq j} e^{-(\|s_i - s_j\|)^2 / 2\sigma_1^2} \|\mathbf{D}_i - \mathbf{D}_j\|_{\text{Frob}}. \quad (10)$$

Depth Consistency Loss (L_{depth}). In post-projection space, inverting depth axis direction, either globally or locally, has no impact on the 2D projection. Thus, such inversion is not penalized by any of the terms above. While unobservable for an individual view, inversion leads to inconsistent results when the camera is rotated. Our depth consistency loss penalizes inversions by preserving the relative depth $\hat{p}_j^z$ of the transformed vertices $\hat{p}_j$ along the viewspace z-axis:

$$L_{\text{depth}} := \sum_{i,j \in n, i \neq j} \|(D_i C \hat{p}_i^z - D_i C \hat{p}_j^z) - (\hat{p}_i^z - \hat{p}_j^z)\|. \quad (11)$$

3.5 Inference and Regularization

We use our learned MLP to render any set of 3D curves from a given camera view. Specifically, we use the MLP to project 3D surface vertices $\hat{p}$ to image space, by first multiplying each vertex by the user specified camera matrix C, multiplying the result by $D(\hat{p})$ and then applying perspective divide (Figure 2c). Changing the perspective can change inter-contour occlusions and shift T-junction locations. We recover the correct contour topology via a post-inference regularization step (see supplemental material).

3.6 Self-Augmentation for Refined Learning

Our initial learning is based on potentially imperfect contour-to-stroke matches and can bake in matching imperfections (e.g., top of the shampoo in Figure 2). It also indirectly promotes consistency across views, but this is not necessarily the case. To make learning more robust to matching errors and improve generalizability, we repeat the learning step using augmented data. The primary goal of augmentation is to preserve the properties of the projected input contour curves that artists are known to preserve, thus obtaining output that appears human-like in all views.

Specifically, we use our inference to generate new pairs of contours and matching deviated contours. We render the contours of the rotated shape twice, once using an analytical camera matrix C_r and once using our inference method that multiples C_r by our learned deviation matrix $\mathbf{D}$, then regularizes the output. We then treat the deviated contours as the matched strokes of their analytical counterparts. We then use these matched contour/"sketch" pairs, plus the original contours and sketch, as training data for another learning step using the same MLP architecture and loss functions as the initial phase. Our data term, by design, has a very small weight (0.001) relative to the weights of the shape, slope, and spatial smoothness terms (10,10,1). These terms balance fitting the input sketch against strong expectations that output contours preserve input contour shape and slopes, even from novel views. During augmentation, these terms act together to counteract data artifacts rather than propagating/reinforcing them, preventing us from overfitting to the input view matches.

In our experiment, we perform this step twice, first rotating the object by $[-5, -4, \ldots, 4, 5]$ degrees around the vertical axis and fine-tuning the MLP. We repeat this process, this time augmenting the data by rotating the object by $[-10, -9, \ldots, 9, 10]$. We emperically chose the range [-10,10]: at [-5,5] we observed some artifacts under large rotations, while [-15,15] offered no additional benefit. This iterative process enhances visual consistency across different views. Without self-augmentation, small matching inaccuracies in the original view can trigger larger artifacts in close-by views. Removing augmentation produces wobbly unnatural looking curves, even for nearby views.

4 Results

Dataset and evaluation. We train our method on 169 individual pairs of sketches and corresponding shape contours: 96 from OpenSketch [Gryaditskaya et al. 2019] (6 artists, 9 shapes, 1 or 2 views), 68 from [Xiao et al. 2022] (10 artists, 8 shapes), and 5 newly collected cube sketches from 5 artists. We then evaluate the resulting 169 models on both the input shape contours rendered from different views (e.g. Figure 6) and on contours computed on other shapes (e.g. Figure 9). In addition to the representative examples shown in the paper, we include the outputs of models trained on all above pairs in the supplemental. In all figures we render contours projected via analytical perspective in *orange* and render artist sketches and contours projected using learned perspective (ours, other methods, and ablations) in *black*. We overlay artist sketches and contours projected using learned perspective over the analytical contours, to visualize the deviation between them. See supplementary for details of all evaluations below.

Alignment. Figs. 1b, 2e, 4c, 6b show our models learned on contour/sketch pairs applied to their training view contours. In each

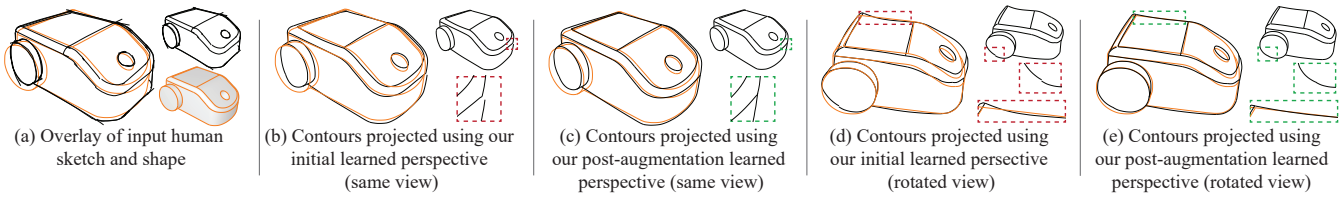


Fig. 4. Self-augmentation impact: (a) input sketch and analytically projected contours; (b) same view inference output using first learned MLP; (c) same view inference output using data augmented MLP, alternative view inference output using first learned MLP (d), alternative view inference output using data augmented MLP (e). The impact of augmentation is more notable for further away camera views.

case, the overlays show that the deviation between contours projected using our method and the analytical ones is visually *very similar* to that between the artist-sketched and analytical ones. We confirm this observation quantitatively: the L_1 chamfer distance between our contours and the artist-sketched ones is on average much smaller than that between the artist sketch and the analytical contours ($2.45e{-}3$ vs. $7.75e{-}3$, averaged over all models, all distances normalized by input image diagonal). This evaluation confirms that our inference results are indeed aligned with the training data.

Generalization. We demonstrate that our method consistently generalizes to alternative views and unseen shapes in Figures 1, 6 (views) and 1, 9, 10 (shapes). In all the figures, the overlays of our and analytical contours show visually similar deviation to that between the training sketches and their corresponding analytical contours. Figure 10 demonstrates the outcome of applying different models to the same set of analytical contours; the resulting deviations are distinctly different and visually similar to those of their respective training pairs.

Quantifying Consistency. We quantitatively evaluate consistency across views as follows. We use our learned perspective deviation D to render the contours of our input shape, with perspective deviation, from a new angle α; we then learn a new perspective deviation D' using these new contours as the input 'sketch' and apply D' to the original view contours (i.e. by rotating by $-\alpha$) and compute the Chamfer distance between these input contours rendered with D and those rendered with D'; zero value indicating perfect consistency. Visually, the results are very similar, especially for smaller α, with the average Chamfer distances between paired contours ranging from 1.3e-1 for $\alpha = \pi/10$, 3.4e-3 for $\alpha = \pi/4$, and 3.8e-3 for $\alpha = \pi/2$.

Perceptual Evaluation: Generalization. We assess how well our method generalizes across shapes by examining if viewers can discern which model was used to generate a set of projected contours. Our study shows viewers an overlay of a sketch and corresponding analytical contours on top, and two overlays of contours projected using learned perspective and their analytical counterparts below. One overlay shows our model trained on the input on top applied to the contours of an unseen shape; the second shows the result of applying a model trained on a different sketch/contour pair to the same contours. Viewers were asked "Given the relationship between the orange and black curves at the top which of the pairs of orange and black curves at the bottom exhibits a more similar relationship?". Viewers rate the outputs trained on the input pairs on top as having more similar relation to the references 63% of the time (21% other, 2% both, 14% neither), demonstrating that ours generalizes deviation present in training sketches to other inputs.

Perceptual Evaluation: Human-Like Outputs. We also assessed if our contour outputs look more human like than those generated using analytic perspective. Study participants were shown same view contours projected using both analytical and our learned perspective, and were asked to assess which output was "more likely to have been drawn by a human". Participants rated our outputs as more human-like 71% of the time, the analytical contours as more human like 12% of the time, both equally human-like 8%, neither human-like 9%. The study confirms that our outputs look more human-like than analytical contours as desired.

Comparison to Prior Work. Figure 5 compares our results to those of DifferSketching [Xiao et al. 2022] on input sketch and contour pairs that are part of their training dataset. In Figure 5c, we show the output of DifferSketching's pretrained model. DifferSketching's pretrained model (trained on their entire dataset) cannot capture individual artist choices, exhibit high-frequency noise, and deviate significantly from the corresponding artist sketches (Figure 5c). Retraining their model on a subset of the dataset (all sketches of the input shape (Figure 5d), or a single sketch/contour pair (Figure 5e)) using their default parameters (extrinsic noise = intrinsic noise = 0.3) dramatically reduces result quality. We also tried using smaller values for extrinsic and intrinsic noise (both = 0.001), and setting each of the two parameters to 0 and the other to 0.001 or 0.3. Using their model with the above settings brings the output contours spatially closer to the input when compared to the default but retains undesirable stroke "jaggies" and other stroke-level artifacts. In all cases, learning on a single input or single family of inputs (e.g. all drawings of the same shape) produces catastrophic failures like those shown in Figure 5de. Training DifferSketch by fine-tuning their pretrained model on either one input, or a small family of inputs, produces similar catastrophic failures as Figure 5de. Our method (Figure 5f) successfully trains on their inputs and generalizes across shapes. We conclude that while DifferSketching can be beneficial for other applications, it is unsuitable for ours.

For completeness, in the supplementary, we provide additional visual comparisons to methods for novel view synthesis [Liu et al. 2023], generating stylized line drawings from images [Chan et al. 2022], and to Pix2Pix [Isola et al. 2017] trained on our paired sketch and contour corpus. 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 learns perspective deviation, instead of stylization, and ours ability to do so.

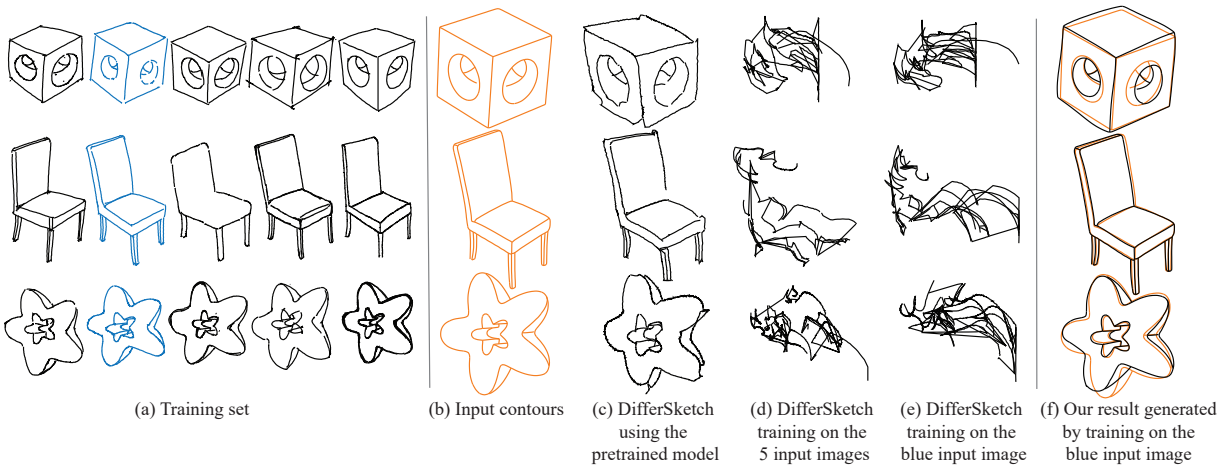


Fig. 5. Comparison versus DifferSketching [Xiao et al. 2022]. (a) same view/shape sketches in DifferSketching dataset; (b) analytically projected contours of the depicted shapes in a matching view; (c-e) Applying DifferSketching to the contours in (b): (c) DifferSketching pre-trained on full data corpus; (d) trained only on the sketches in (a) and the contours in (b); (e) trained on the blue sketch in (a) and the contour in (b); (f) Our outputs trained the blue sketch in (a) and the contour in (b). In all cases, DifferSketching outputs are distorted, and quality reduces as training set size decreases. Our method capture the artists' deviation and generalizes.

Ablations. We compared our method against a 2D version of our MLP, which attempts to learn pure 2D deviation between artist and analytical contours. While the outputs generated using these 2D only models appear reasonable in the input and nearby views, the models do not generalize to farther away views where their outputs exhibit undesirable deformation (Fig 7). We validated this observation via two studies. The first study asked participants whether our outputs, or ones generated using the 2D MLP, were more accurate rotated depictions of the original sketched shape. Participants ranked ours as more accurate 79% of the time (vs 6% 2D MLP, 7% both, 8% neither). The second study used a similar design to the generalization study above, but compared our output to that of the 2D MLP. Participants assessed the relations between our output and analytical contours as more similar to the reference than the one between 2D MLP outputs and analytical contours 72% of the time versus 17% for 2D MLP (6% both, 5% neither). These studies confirm the need for a 3D approach for encoding and learning perspective deviation.

Figure 8 evaluates our decision to use single sketch-contour pairs to learn meaningful deviation function by training our perspective MLP on larger data corpuses. As demonstrated, increasing the size of the training corpus diminishes the magnitude of the deviation; contrary to our goal of capturing training sketch deviation. See supplemental for details and additional ablations.

Applications. Learning human perspective can support a range of downstream applications. One natural direction is non-photorealistic rendering, where our approach can add a human touch to generated outputs in combination with other stylization elements. Another is content addition or editing (for instance, through generative AI), where the additions/edits would obey and respect the perspective deviations present in the input, ensuring visually coherent results. Our framework could also serve as a tool for analyzing artistic choices, such as whether artists apply similar perspective deviations across views or shapes.

5 Conclusion

We presented the first method to model and learn the perspective projection humans use when creating line drawings. As demonstrated, our method faithfully captures input sketch artist perspective and generalizes it across novel views and similar shapes.

Limitations and Future Work. The only notable failure scenarios we observed were either due to faulty vector contour extraction from our input shapes or poor input camera estimation for the input shape/sketches. Our method produces a single learned perspective deviation per input, leaving it to the user to pick the input they like best. It would be interesting to analyse the similarities and differences between the deviation functions we learn across different shapes and artists and to use the results of this analysis to select or compute the visually best deviation for a new input.

An exciting follow-up avenue for our work is to use our approach for learning deviations as a basis for a method capable of inverting artist deviation, i.e., taking free-hand sketches and producing 2D curves that are the *analytic* projection of artist intended 3D surface curves. Being able to invert deviation will improve the robustness of sketch-based 3D modeling.

Acknowledgments

We thank Congyi Zhang for their help with figures. We acknowledge the support of the Natural Sciences and Engineering Research Council of Canada (NSERC) grant RGPIN-2024-03981. Finally, this work is supported in part by the Institute for Computing, Information and Cognitive Systems (ICICS) and Advanced Research Computing (ARC) at UBC.

References

Chrystiano Araújo, Nicholas Vining, Enrique Rosales, Giorgio Gori, and Alla Sheffer. 2022. As-Locally-Uniform-As-Possible Reshaping of Vector Clip-Art. *ACM Transaction on Graphics* 41, 4 (2022). https://doi.org/0.1145/3528223.3530098

Seok-Hyung Bae, Ravin Balakrishnan, and Karan Singh. 2008. ILoveSketch: as-natural-as-possible sketching system for creating 3d curve models. In *Proceedings of the 21st annual ACM symposium on User interface software and technology*. 151–160.

Mikhail Bessmeltsev and Chenxi Liu. 2024. Fundamentals and Applications of Sketch Processing. In *Symposium on Graphics Processing (SGP 2024) Course*. https://school.geometryprocessing.org/summerschool-2024/

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

Changwoon Choi, Jaeah Lee, Jaesik Park, and Young Min Kim. 2024. 3Doodle: Compact Abstraction of Objects with 3D Strokes. *ACM TOG* 43, 4 (July 2024), 1–13.

Forrester Cole, Aleksey Golovinskiy, Alex Limpaecher, Heather Stoddart Barros, Adam Finkelstein, Thomas Funkhouser, and Szymon Rusinkiewicz. 2008. Where Do People Draw Lines? *ACM Transactions on Graphics (Proc. SIGGRAPH)* 27, 3 (Aug. 2008).

Patrick Coleman, Karan Singh, Leon Barrett, Nisha Sudarsanam, and Cindy Grimm. 2005. 3D screen-space widgets for non-linear projection. In *Proc. Computer Graphics and Interactive Techniques (GRAPHITE '05)*. 221–228.

Chris De Paoli and Karan Singh. 2015. SecondSkin: sketch-based construction of layered 3D models. *ACM Transactions on Graphics (TOG)* 34, 4 (2015), 1–10.

Doug DeCarlo, Adam Finkelstein, Szymon Rusinkiewicz, and AnthonySantella. 2003. Suggestive Contours for Conveying Shape. *Proc. SIGGRAPH* 22, 3 (jul 2003), 848–855.

Marek Dvorožňák, Daniel Sýkora, Cassidy Curtis, Brian Curless, Olga Sorkine-Hornung, and David Salesin. 2020. Monster Mash: A Single-View Approach to Casual 3D Modeling and Animation. *Proc. of SIGGRAPH ASIA* 39, 6, Article 214 (2020).

Koos Eissen and Roselien Steur. 2008. *Sketching: Drawing Techniques for Product Designers*. Bis Publishers.

Koos Eissen and Roselien Steur. 2011. *Sketching: The Basics*. Bis Publishers.

James D. Foley, Andries van Dam, Steven K. Feiner, and John F. Hughes. 1996. *Computer Graphics: Principles and Practice* (2nd ed.). Addison-Wesley, Reading, MA.

Leon A. Gatys, Alexander S. Ecker, and Matthias Bethge. 2016. Image Style Transfer Using Convolutional Neural Networks. In *2016 IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2414–2423. https://doi.org/10.1109/CVPR.2016.265

E. H. Gombrich. 1951. The Story of Art. *Journal of Aesthetics and Art Criticism* 9, 4 (1951), 339–340. https://doi.org/10.2307/426517

Yulia Gryaditskaya, Felix Hähnlein, Chenxi Liu, Alla Sheffer, and Adrien Bousseau. 2020. Lifting Freehand Concept Sketches into 3D. *ACM Transactions on Graphics (Proc. SIGGRAPH Asia)* (2020).

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).

Felix Hähnlein, Yulia Gryaditskaya, Alla Sheffer, and Adrien Bousseau. 2022. Symmetry-driven 3D reconstruction from concept sketches. In *ACM SIGGRAPH 2022 Conference Proceedings*. 1–8.

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

James W. Hennessey, Han Liu, Holger Winnemöller, Mira Dontcheva, and Niloy J. Mitra. 2017. How2Sketch: Generating Easy-To-Follow Tutorials for Sketching 3D Objects. *Symposium on Interactive 3D Graphics and Games* (2017).

Amir Hertz, Andrey Voynov, Shlomi Fruchter, and Daniel Cohen-Or. 2023. Style Aligned Image Generation via Shared Attention. (2023).

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

Takeo Igarashi, Satoshi Matsuoka, and Hidehiko Tanaka. 1999. Teddy: a sketching interface for 3D freeform design. In *Proceedings of the 26th Annual Conference on Computer Graphics and Interactive Techniques (SIGGRAPH '99)*. ACM Press/Addison-Wesley Publishing Co., USA, 409–416.

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

Levent Burak Kara and Kenji Shimada. 2007. Sketch-based 3D-shape creation for industrial styling design. *IEEE Computer Graphics and Applications* 27, 1 (2007), 60–71.

M. Kemp. 1991. The Science of Art – Optical Themes in Western Art from Brunelleschi to Seurat. 617–619 pages.

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

Jan Koenderink, Andrea van Doorn, Baingio Pinna, and Robert Pepperell. 2016. On right and wrong drawings. *Art and Perception* (9 2016).

Jan J. Koenderink and Andrea J. van Doorn. 1991. Affine structure from motion. *J. Opt. Soc. Am. A* 8, 2 (Feb 1991), 377–385.

Michael Kubovy. 1986. *The Psychology of Perspective and Renaissance Art*. Cambridge University Press, Cambridge, UK.

Changjian Li, Hao Pan, Adrien Bousseau, and Niloy J Mitra. 2022. Free2cad: Parsing freehand drawings into cad commands. *ACM (TOG)* 41, 4 (2022), 1–16.

Changjian Li, Hao Pan, Yang Liu, Alla Sheffer, and Wenping Wang. 2018. Robust Flow-Guided Neural Prediction for Sketch-Based Freeform Surface Modeling. *ACM Trans. Graph. (SIGGRAPH ASIA)* 37, 6 (2018), 238:1–238:12. https://doi.org/10.1145/3272127.3275051

Zhichao Liao, Fengyuan Piao, Di Huang, Xinghui Li, Yue Ma, Pingfa Feng, Heming Fang, and Long Zeng. 2024. Freehand Sketch Generation from Mechanical Components. In *Proc. ACM MM*. 6755–6764.

Difan Liu, Mohamed Nabail, Aaron Hertzmann, and Evangelos Kalogerakis. 2020. Neural Contours: Learning to Draw Lines from 3D Shapes. In *IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR)*.

Feng-Lin Liu, Hongbo Fu, Yu-Kun Lai, and Lin Gao. 2024. SketchDream: Sketch-based Text-To-3D Generation and Editing. *ACM Trans. Graph.* 43, 4, Article 44 (July 2024), 13 pages. https://doi.org/10.1145/3658120

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]

Andrew Nealen, Takeo Igarashi, Olga Sorkine, and Marc Alexa. 2007. FiberMesh: Designing Freeform Surfaces with 3D Curves. *ACM Transactions on Graphics (Proceedings of ACM SIGGRAPH)* 26, 3 (2007), article no. 41.

Andrea L Nicholls and John M Kennedy. 1995. Foreshortening in Cube Drawings by Children and Adults. *Perception* 24, 12 (1995), 1443–1456. https://doi.org/10.1068/p241443 PMID: 8734543.

Luke Olsen, Faramarz F. Samavati, Mario Costa Sousa, and Joaquim A. Jorge. 2009. Sketch-based modeling: A survey. *Computers & Graphics* 33, 1 (2009), 85–103. https://doi.org/10.1016/j.cag.2008.09.013

Robert Pepperell and Manuela Haertel. 2014. Do Artists Use Linear Perspective to Depict Visual Space? *Perception* 43, 5 (2014), 395–416.

Emiel Reith and Chang Hong Liu. 1995. What Hinders Accurate Depiction of Projective Shape? *Perception* 24, 9 (1995), 995–1010. https://doi.org/10.1068/p240995 PMID: 8552463.

Ryan Schmidt, Azam Khan, Gord Kurtenbach, and Karan Singh. 2009a. On expert performance in 3D curve-drawing tasks. In *SBIM*. New York, NY, USA, 133–140.

Ryan Schmidt, Azam Khan, Karan Singh, and Gord Kurtenbach. 2009b. Analytic drawing of 3D scaffolds. *ACM Trans. Graph.* 28, 5 (Dec. 2009), 1–10. https://doi.org/10.1145/1618452.1618495

Cloud Shao, Adrien Bousseau, Alla Sheffer, and Karan Singh. 2012. CrossShade: shading concept sketches using cross-section curves. *ACM Trans. Graph.* 31, 4, Article 45 (July 2012), 11 pages.

Karan Singh. 2002. A Fresh Perspective. In *Proceedings - Graphics Interface*.

Philip Steadman. 2002. *Vermeer's camera: uncovering the truth behind the masterpieces*. Oxford University Press.

Ivan E Sutherland. 1964. Sketch pad a man-machine graphical communication system. In *Proceedings of the SHARE design automation workshop*. 6–329.

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

Laura M Taylor and Peter Mitchell. 1997. Judgments of apparent shape contaminated by knowledge of reality: Viewing circles obliquely. *British Journal of Psychology* 88, 4 (1997), 653–670.

Dave Pagurek Van Mossel, Chenxi Liu, Nicholas Vining, Mikhail Bessmeltsev, and Alla Sheffer. 2021. StrokeStrip: joint parameterization and fitting of stroke clusters. *ACM Trans. Graph.* 40, 4, Article 50 (July 2021), 18 pages.

Yael Vinker, Ehsan Pajouheshgar, Jessica Y. Bo, Roman Christian Bachmann, Amit Haim Bermano, Daniel Cohen-Or, Amir Zamir, and Ariel Shamir. 2022. CLIPasso: Semantically-Aware Object Sketching. *ACM Trans. Graph.* 41, 4, Article 86 (jul 2022), 11 pages.

A. Viterbi. 1967. Error bounds for convolutional codes and an asymptotically optimum decoding algorithm. *IEEE Transactions on Information Theory* 13, 2 (1967), 260–269. https://doi.org/10.1109/TIT.1967.1054010

Haofan Wang, Peng Xing, Renyuan Huang, Hao Ai, Qixun Wang, and Xu Bai. 2024. InstantStyle-Plus: Style Transfer with Content-Preserving in Text-to-Image Generation. *arXiv preprint arXiv:2407.00788* (2024).

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.

Baoxuan Xu, William Chang, Alla Sheffer, Adrien Bousseau, James McCrae, and Karan Singh. 2014. True2Form: 3D curve networks from 2D sketches via selective regularization. *ACM Trans. Graph.* 33, 4, Article 131 (July 2014), 13 pages.

Byung-Jun Yoon. 2009. Hidden Markov Models and their Applications in Biological Sequence Analysis. *Current genomics* 10 (09 2009), 402–15. https://doi.org/10.2174/138920209789177575

Congyi Zhang, Lei Yang, Nenglun Chen, Nicholas Vining, Alla Sheffer, Francis C.M. Lau, Guoping Wang, and Wenping Wang. 2022. CreatureShop: Interactive 3D Character Modeling and Texturing from a Single Color Drawing. *IEEE Transactions on Visualization and Computer Graphics* (2022), 1–18.

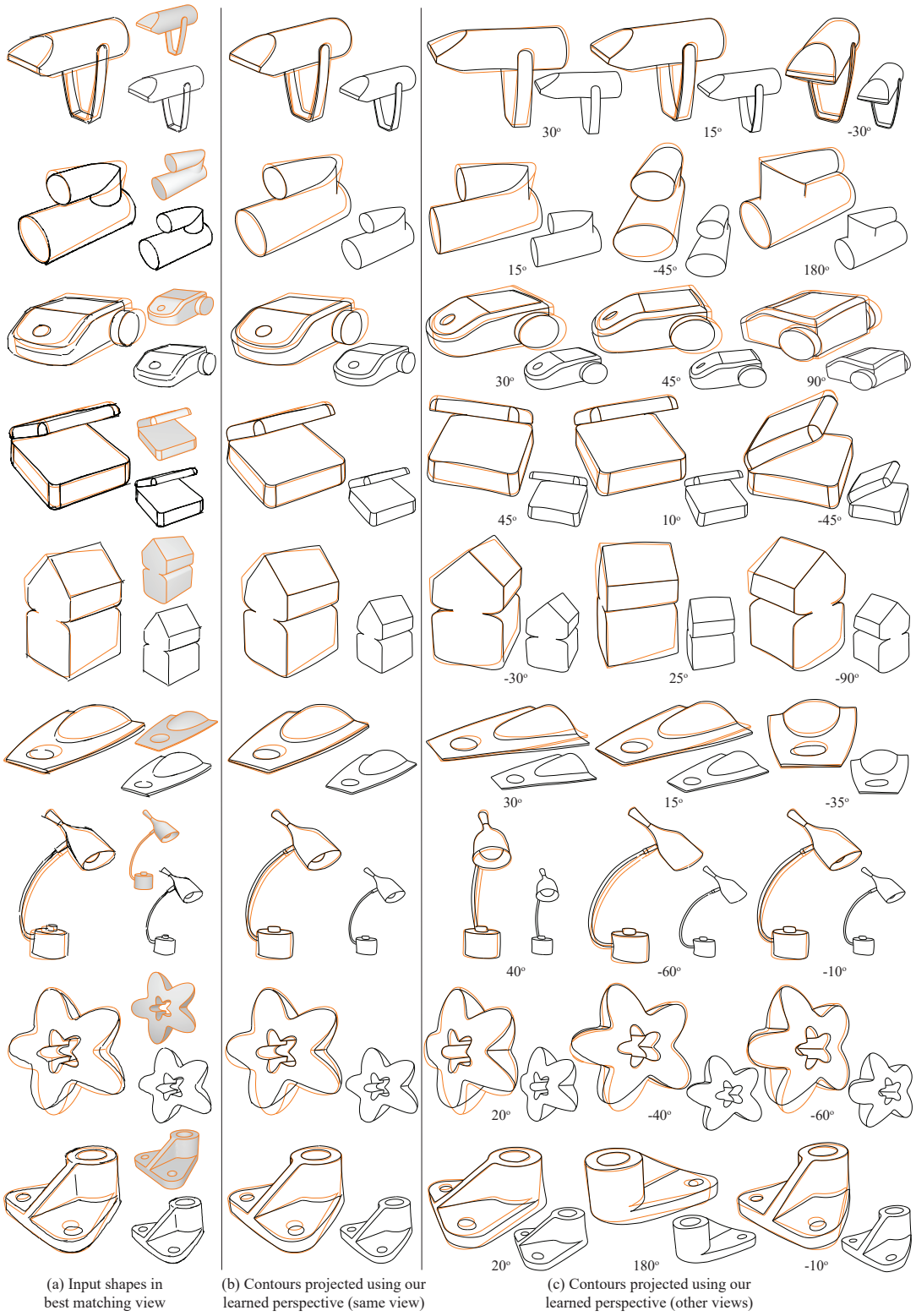


Fig. 6. A gallery of our results. We show (a) overlay of human sketches and shape contours in best matching view (insets show the sketches and contours separately); (b) our learned outputs under the same view as (a); (c) Applying our learned perspectives to contours in different other rotated views. In all examples our outputs match the sketches' perspective deviations.

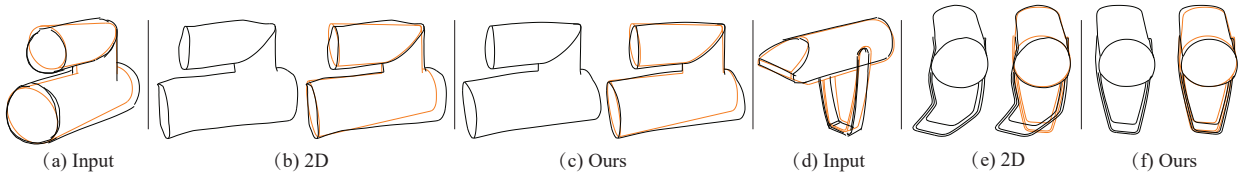


Fig. 7. Ablation against 2D MLP: (a,d) input sketches and matching view contours; (b,e) rotated view outputs generated using 2D MLP; (c,f) our results for same views. 2D MLP outputs undesirably and notably diverge from the analytical contours in such views. Our results maintain the same deviations relative to analytical contours, as the input sketches across all views.

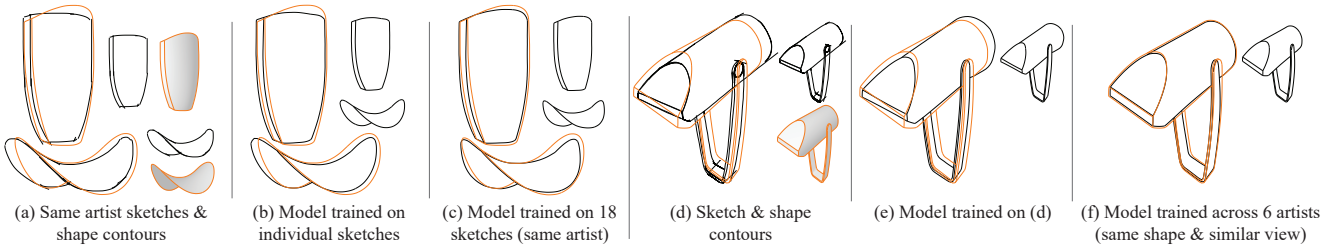


Fig. 8. Learning from multiple sketch/shape pairs: Left: learning from 18 sketches by the same artist: (a) inputs, (b) our outputs, (c) outputs of learning from 18 sketches. The result captures the common characteristics of artist's perspective, but is more subtle then one learned form a single input pair. Right: learning from same shape, approximately same view sketches from 6 artists: (d) input, (e) our output, (f) output of learning from 6 sketches. The result loses the expressiveness, with learned perspective essentially identical to analytical one.

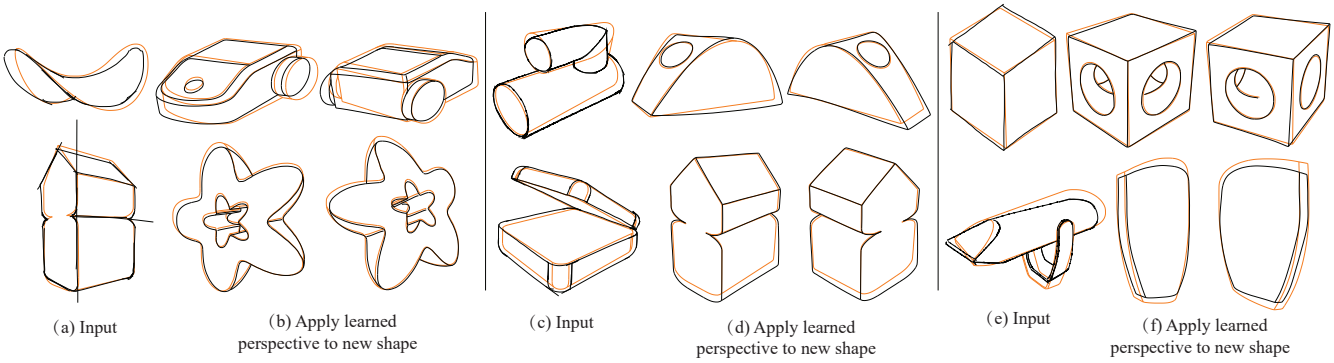


Fig. 9. Applying learned perspective to new shapes: (a,c,e) overlay of human sketches and shape contours in best matching view ; (b,d,f) Applying our learned perspectives to contours of other shapes in different views.

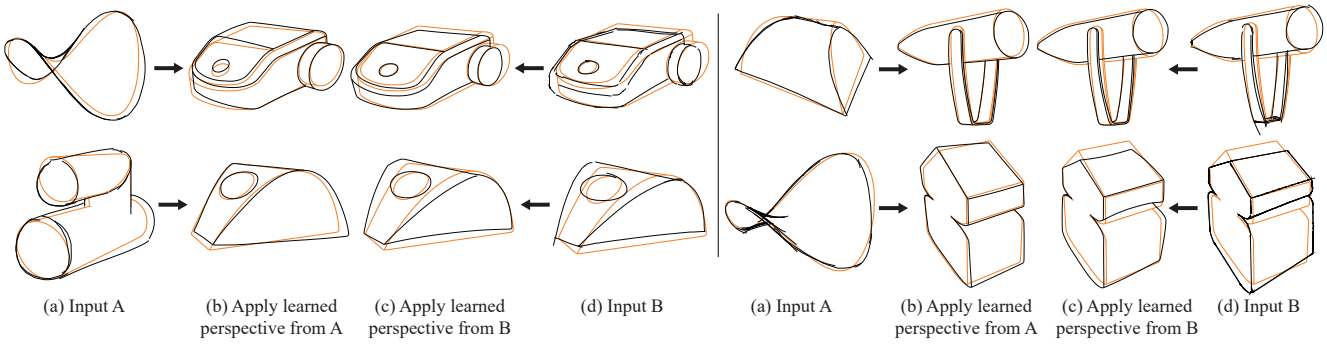


Fig. 10. Applying different learned perspectives to the same shape. As desired, applying different learned perspectives to the same analytical contours produces distinctly different projections, each consistent with its source training sketch projection.