Appendix

A. Details on optimization-based guidance stroke generation process

In this section, we describe the details of the optimization method for generating $\boldsymbol{p}_{gt}$ described in Section 4.3, by referring to the pipeline of CLIPDraw [18] and CLIPasso [68] with modifications.

Saliency map. We use the saliency map of CLIPasso to sample the initial strokes. We first employed U2-Net [57] to remove the background from the STL-10 dataset. Then, we obtained the relevancy map by averaging the attention heads of each self-attention layer of the ViT-32/B CLIP model. Finally, we generated the saliency map by multiplying the edge map of the masked image using edges from the XDoG edge detection algorithm [74].

Initialization process As described in Section 4.3, CLIPasso stochastically samples the initialized strokes, which results in the guidance $\boldsymbol{p}_{gt}$ being highly unstable. Instead, we propose to deterministically sample the initialized stroke with low computational cost using the following process. First, when initializing the first stroke, we sample the coordinate with the highest weight of the saliency map as the first control point of the first stroke. The remaining control points are sampled by adding a small random perturbation. Next, we lower the weight of the saliency map by a value related to the distance between the sampled point. Specifically, we subtract the weight of the saliency map proportionately by the exponential of the negative squared distances scaled by σ, i.e., $\text{saliency}(u) \leftarrow \text{saliency}(u) - I(u) \cdot \exp(-(u - t_1^{(1)})/\sigma^2)$ where $\sigma = 5$, $u \in \Omega$, and $t_1^{(1)}$ denotes the first control point of the first stroke. Finally, we sample the next stroke from the modified saliency map, and the process is repeated for the number of strokes used by our model.

We observed that optimizing the stroke color with $\mathcal{L}_{percept}$ led to the generated results approaching the style of painting with colors appearing similar to the object's texture, which is not the desired as described in Sec. 5.4. To tackle this problem, we sample the color of each stroke during the initialization process and use it as the final color, instead of optimizing its value. To determine the color of each stroke, we use the color of the image corresponding to the initial position of the first control point of each stroke. Then, we refined the color with a small adjustment, $\tilde{c} = \frac{\sigma((2c-1)\beta)-\sigma(-\beta)}{\sigma(\beta)-\sigma(-\beta)}$ where $\tilde{c}$ is the adjusted color ranged from 0 to 1. $\sigma(\cdot)$ represents the logistic Sigmoid function, and β is a hyperparameter that is set to 5 to promote color contrast.

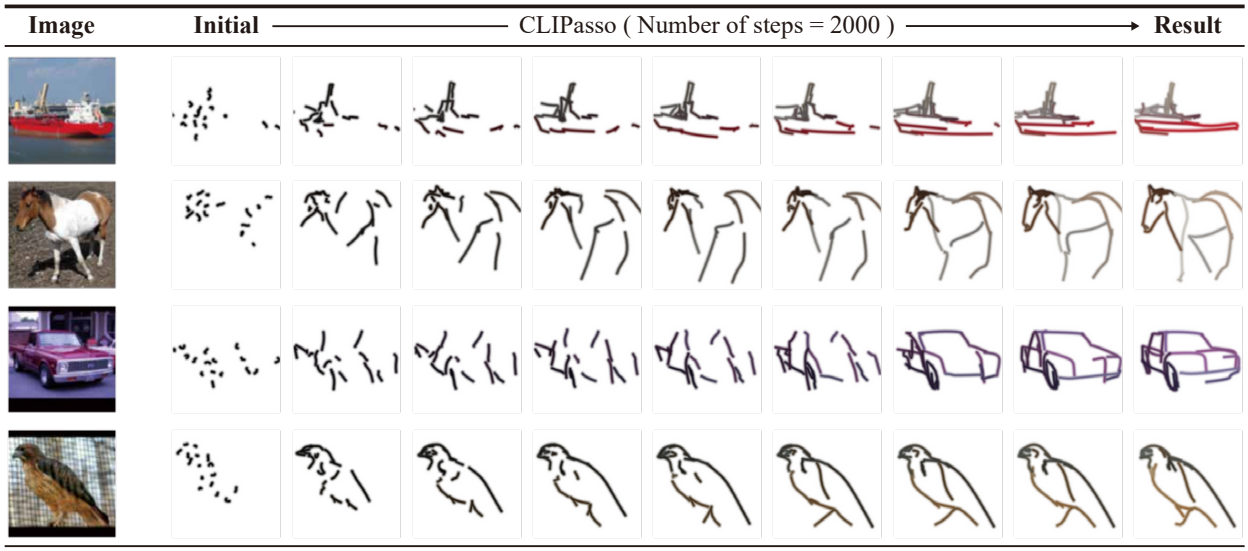


Figure 6. Visualization of optimization-based generation process. From left to right: examples of a given image, initialized stroke, and optimization results at 50, 100, 200, 400, 700, 1000, 1500, and 2000 iterations, respectively.

Hyperparameters on stroke optimization We use the same optimization scheme as CLIPasso with our initialization process. We repeat the optimization process for 2000 iterations and evaluate $\mathcal{L}_{percept}$ with augmented images with random

affine transformations. The strokes are trained with an Adam optimizer using a learning rate of 1.0. We save 8 intermediate results, namely at steps 50, 100, 200, 400, 700, 1000, 1500, and 2000, for $\boldsymbol{p}_{gt,l}$ where $l \in \{1, 2, ..., 8\}$, as shown in Figure 6.

B. Additional experimental setup

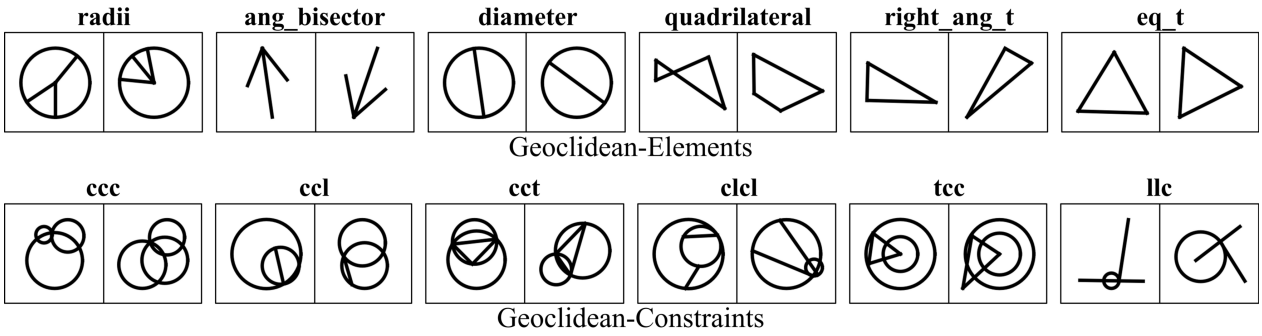


Figure 7. Example images of the Geoclidean dataset.

Geoclidean dataset The Geoclidean dataset is divided into two categories: Geoclidean-elements and Geoclidean-concepts. The concepts of Geoclidean-Elements and Geoclidean-Constraints are derived from the first book of Euclid's Elements (*e.g.*, equilateral triangle, rhomboid, or parallel lines) and from the relationship between primitive elements (*e.g.*, 3 circles with the same point of intersection), respectively. 17 concepts consist Geoclidean-Elements dataset, while 20 concepts consist Geoclidean-Constraints dataset. Since the task on Geoclidean is designed to generalize each geometric concept with a small amount of data, we use only 10 examples per class for training. The training is performed for 5000 epochs, and we evaluate the model with the weights obtained in the final epoch. To measure classification accuracy, we use 500 test images per class. The images are rescaled to have a resolution of 64×64, and we apply random augmentations such as rotation in the range of (-90°, 90°), translations with a factor of (-0.1, 0.1), and scaling with a factor of (0.8, 1.2).

Rotated MNIST dataset The training is performed for 200 epochs with 2000 training samples and 500 validation samples per rotation. To predict the transformation between unaltered and transformed images, we specifically classify the out-of-distribution transformation, namely 0, 90, 180, and 270 degrees rotation with or without horizontal reflection, resulting in 8 possible transformations. For each model trained on rotMNIST, we concatenate the two latent vectors obtained with the input images for linear probing. We rescale the image to have a 32×32 resolution and invert the pixel values to make the image with black foreground on a white background. For random augmentation, we use random cropping with a scale of (0.7, 1.0).

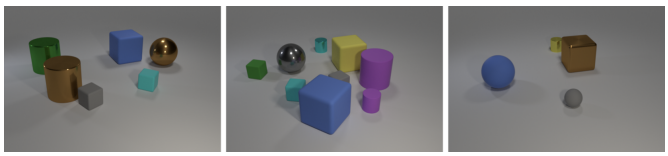


Figure 8. Example images of CLEVR dataset.

CLEVR dataset The CLEVR dataset contains rendered images of 3 to 10 objects. Each object has a set of attributes that include 8 colors, 3 shapes, 2 sizes, 2 materials, and 2D pixel positions, with question-answer pairs that evaluate a high-level understanding of the scene. On the CLEVR dataset, we trained the model for 200 epochs using 8000 training samples and 2000 validation samples, with the images resized to 128×128 resolution. Our task on CLEVR dataset explicitly requires the local geometric information of each object. To preserve such local geometric information, we chose small adjustments for random augmentation, such as random cropping with a scale of (0.9, 1.0) with weak color jittering, as large adjustments can harm these attributes and negatively affect our model (e.g., random cropping can remove the rightmost object from the scene where the label is given as the rightmost object in the original scene).

STL-10 dataset On STL-10, we train the model for 50 epochs using 90k unlabelled images for training and 10k for validation. Then, we select the model with the lowest validation loss and evaluate the performance of tasks on the CLEVR dataset. Since cropping small regions can make the scene unclear to sketch, we use random cropping with a scale (0.8, 1.0), horizontal flipping, and color jittering for random augmentation.

QMUL-ShoeV2 dataset The QMUL-ShoeV2 dataset consists of 2,000 shoe photos and 6,648 corresponding human-drawn shoe sketches. We use triplet loss as $\mathcal{L}_{embed}$, with z_{LBS+} generated from a shoe image and a corresponding sketch as input a positive pair, and z_{LBS+} for another shoe image as a negative pair. We trained the model for 1000 epochs using 200 shoes as the test split, with the images resized to 128×128 resolution. Since the sketches provided by the dataset are sketches with only black strokes, LBS also generates sketches with the stroke color fixed to black. We use random augmentation, which is the same setting as STL-10.

C. Implementation details & hyperparameter settings.

For the implementation of LBS, we selected the cubic Bézier curve for parameterizing each stroke. As the samples from rotMNIST and Geoclidean can be accurately reconstructed with a few strokes, we replace $\mathcal{L}_{percept}$ and $\mathcal{L}_{guide}$ with $\mathcal{L}_1$ loss, which does not rely on a pre-trained CLIP model. To ensure strokes do not deviate from the grid space and are standardized to be from left to the right, we penalize $\boldsymbol{p}$ as follows:

$$\mathcal{L}_{boundary} = \sum_{p \in \boldsymbol{p}} \left[\sum_{t \in p} \max(0, -1 + ||t||_\infty) \right], \tag{7}$$

$$\mathcal{L}_{align} = \sum_{p \in \boldsymbol{p}} \left[\max(0, t_{1,x} - t_{4,x}) \right], \tag{8}$$

$$\mathcal{L}_{penalty} = \mathcal{L}_{boundary} + \mathcal{L}_{align}, \tag{9}$$

where $t_{1,x}, t_{4,x}$ is the x coordinate of the first and last control point of p, respectively. The final loss is $\mathcal{L}_1 + \lambda_p \cdot \mathcal{L}_{penalty} + \lambda_e \cdot \mathcal{L}_{embed}$, where $\lambda_p = 0.1, \lambda_e = 0.01$ on Geoclidean and $\lambda_p = 1, \lambda_e = 0.1$ on rotMNIST.

We selected a 4-layer CNNs and a 2-layer Transformer decoder with a hidden size of 256 for the CNN encoder and the stroke generator. The embedding network is based on the Invariant model from DeepSets [80], and we used 2-layer MLPs for embedding each stroke and a 1-layer MLP for the aggregated embedding. We use 4 strokes for the Geoclidean dataset and 8 for the rotMMNIST dataset. The thickness is fixed for the Geoclidean dataset, and the color is fixed to be black for both experiments. We train the models using AdamW optimizer [50] with a learning rate of 1e-3.

On CLEVR, STL-10, and QMUL-ShoeV2 datasets, the CNN encoder is based on ResNet18 architecture, and the stroke generator is based on an 8-layer Transformer decoder with 512 hidden units. We use an embedding network with the same architecture as the one used in the Geoclidean experiment. To ensure consistency in stroke order, we arrange p_{gt} to follow a left-to-right order. We use 24 strokes for the CLEVR dataset, 20 for the STL-10 dataset, and 10 for the QMUL-ShoeV2 dataset, with thickness fixed for all datasets. The hyperparameters for the loss function are set to $\lambda_g = 10, \lambda_e = 0.1$ for CLEVR dataset, $\lambda_g = 1, \lambda_e = 0.1$ for STL-10 dataset, and $\lambda_g = 10, \lambda_e = 1$ for QMUL-ShoeV2 dataset. In STL-10, we assign 4 strokes as background strokes to express information about the background and trained by minimizing the $\mathcal{L}_1$ loss with the masked foreground. We use an AdamW optimizer with a learning rate of 2e-4. We use the ResNet101 architecture of the publicly available pre-trained CLIP model to measure $\mathcal{L}_{percept}$. The effects of changing the CLIP model architecture are described in Appendix G.

Training LBS with $\mathcal{L}_{LBS}$ requires a large amount of GPU memory. Therefore, to ensure memory efficiency during training of LBS (InfoNCE) in Appendix H and F, we employed a momentum encoder and queue from MoCo [26]. Specifically, we used a batch size of 32.

Evaluation protocol. All evaluations are conducted with linear probing, where we train a single linear head for evaluation. We select the model with the best validation results during training for evaluation. We train the linear head 2 times, each time for 100 epochs using the SGD optimizer with a learning rate of 1 and 0.1, respectively. We evaluate the accuracy with test samples for all epochs and learning rates and report the best value as the evaluation accuracy.

Baselines. For E(2)-CNN, we use the C_8-equivariant CNN model for rotMNIST and Geoclidean dataset, the $C_8C_4C_1$-equivariant WideResNet 16-8 model [79] for CLEVR dataset, and the $D_8D_4D_1$ model for STL-10 dataset [13]. E(2)-CNN is trained with Cross-Entropy loss in the labeled setting and InfoNCE loss in the unlabeled setting. For LtD [54], we select the Object Oriented game-different setup for rotMNIST, Geoclidean, and CLEVR datasets (LtD-diff), while selecting the original setup for STL-10 (LtD-original). For β-TCVAE [8], we adopt the hyperparameters from the public repository[2], with a latent size of 16 for rotMNIST and Geoclidean datasets, and 32 for CLEVR and STL-10. For DefGrid [21], we use a 15×15 grid for the CLEVR dataset and a 20×20 grid for the STL-10 dataset. Since our experiments use a lower resolution than the image resolution used in DefGrid, we increase λ_{area} to 0.02 and 0.1 for the CLEVR and STL-10 datasets, respectively. We conducted experiments using our implementation of the work by Novotny *et al*. [55] (referred to as GeoSSL in our paper), as no publicly available code was available.

D. Differentiable rasterizer

The rendering process r maps the set of strokes $\boldsymbol{p} \in \mathcal{P}$ generated by the stroke generator of LBS to a signal S on the physical domain Ω. We calculate the CLIP-based perceptual loss between the realized sketch $S(\Omega)$ and the original image $I(\Omega)$ by computing every pixel value in Ω from the signal $S = r(I)$. To enable the gradient of the loss to propagate back to the stroke generator, we use a differentiable rasterizer [53] that allows the mapping of $\boldsymbol{p}$ to $S(\Omega)$ to be differentiable. [53] formulated the differentiable pixel rasterization process by setting the value of each pixel to be an exponential of the negative squared distance between the pixel and the vector primitive $p = (t_1, t_2, t_3, t_4, c, w)$ (i.e., stroke) as follows:

$$r'(u; p) = \exp(-d^2_{cur}(u, p)/w^2) \cdot c, \tag{10}$$

where d_{cur} is the Euclidean distance between the parametric curve $C(s, p)$ $(0 \leq s \leq 1)$ represented by p and each pixel coordinate $u \in \Omega$.

$$\begin{aligned} d^2_{cur}(u; p) = &\min_s ||C(s, p) - u||^2_2 \\ &\text{s.t.} \quad 0 \leq s \leq 1 \end{aligned} \tag{11}$$

To better represent the curvature information of the image, we chose a cubic Bézier Curve consisting of four control points among various parametric curves. The first and last control points are set as the starting and ending points of the curve, respectively.

$$C(s, p) = (1 - s)^3 t_1 + s(1 - s)^2 t_2 + s^2(1 - s)t_3 + s^3 t_4 \tag{12}$$

To enable more pixels to be affected by strokes during the early stages of training without changing the stroke thickness while generating a sharp sketch at the later stages, we anneal the exponential value applied to Euclidean distance and w in Eq. (10). Specifically, we increase the exponential value to increase linearly from 1 to 2 during the stages of training and decrease the value of w linearly from $2w$ to w.

We use the *over* composition [53] to extend this process to multiple strokes. The over composition of two images A and B is defined as $c_{over}(A, B) = A + B(1 - A)$, and can be recursively defined for any sequence of images. For numerical stability, the cumulative product expressed as an exponentiated sum of log differences is used as follows:

$$r(\Omega; \boldsymbol{p}) = \sum_{i=1}^{k} r'(\Omega; p^{(i)}) \odot \exp\left(\sum_{j=1}^{i-1} \log\left(1 - r'(\Omega; p^{(j)})\right)\right) \tag{13}$$

However, the over composition has a problem of color blending for colored strokes (e.g., yellow color is rasterized at the intersection of red and green strokes). To ensure that the color of the stroke painted later replaces the color of the previous stroke, we modify the cumulative product as follows.

[2] https://github.com/YannDubs/disentangling-vae

$$r(\Omega; \boldsymbol{p}) = \sum_{i=1}^{k} r'(\Omega; p^{(i)}) \odot \exp\left(\sum_{j=1}^{i-1} \log\left(1 - r'_{intensity}(\Omega; p^{(j)})\right)\right), \tag{14}$$

where $r'_{intensity}$ refers to the value without multiplying the color parameter, i.e., $r'_{intensity}(u; p) = \exp(-d^2_{cur}(u, p)/w^2)$.

E. Omitted proofs

We first start with the definition of groups, group actions, and group representations.

Definition 3. *A **group** $\mathcal{G}$ is a set with a composition $\circ : \mathcal{G} \times \mathcal{G} \to \mathcal{G}$ (where we denote as $g \circ h = gh$) satisfying:*

(i) ***Associativity**: $(gh)i = g(hi), \quad \forall g, h, i \in \mathcal{G}$.*

(ii) ***Identity**: $\forall g \in \mathcal{G}$, there exists a unique $e \in \mathcal{G}$ s.t. $eg = ge = g$.*

(iii) ***Inverse**: $\forall g \in \mathcal{G}$, there exists a unique $g^{-1} \in \mathcal{G}$ s.t. $gg^{-1} = g^{-1}g = e$.*

(iv) ***Closure**: $\forall g, h \in \mathcal{G}$, we have $gh \in \mathcal{G}$.*

Definition 4. *For $\mathcal{G}$ with identity element e and a set $\mathcal{X}$, a (left) **group action** of $\mathcal{G}$ on $\mathcal{X}$ is a function $(g, x) \mapsto g.x$ that satisfies:*

(i) ***Identity**: $e.x = x, \quad \forall x \in \mathcal{X}$.*

(ii) ***Compatibility**: $g.(h.x) = (gh).x, \quad \forall g, h \in \mathcal{G}, \forall x \in \mathcal{X}$.*

where we defined the group of geometric transformations $\mathcal{G}$ acting on $\mathcal{I}$ as $(g.I)(u) = I(g^{-1}u)$ in Section 3.

Definition 5. *A **group representation** $\rho : \mathcal{G} \to GL(V)$ is a map that assigns each g to an invertible matrix $\rho(g)$, such that $\rho(gh) = \rho(g)\rho(h)$ for $\forall g, h \in \mathcal{G}$.*

Since actions on signals are linear, i.e., $g.(\alpha I + \beta I')(u) = (\alpha I + \beta I')(g^{-1}u) = \alpha(I)(g^{-1}u) + \beta(I')(g^{-1}u) = \alpha(g.I)(u) + \beta(g.I')(u)$ for any scalars α, β and signals $I, I' \in \mathcal{I}$, we can describe group action of g acting on $\mathcal{I}$ as a linear matrix $\rho(g) \in GL(|\Omega|, \mathbb{R})$ which acts on Ω. Since $\mathcal{S} \subset \mathcal{I}$, sketch shares the same physical domain Ω as $\mathcal{I}$. Thus, finding the linear operator $\rho'(g)$ for equivariance is trivial; we can let $\rho'(g)$ as the same operator which acts in the image space, $\rho' = \rho$. For example, equivariance with respect to rotation could be achieved if the sketch representation of a rotated image is equally rotated and for other arbitrary geometric transformations.

We first show that sketch representation which minimizes a metric function with some constraints, is a geometry-aware representation w.r.t arbitrary geometric transformations. Then, we show that stroke representation is also a geometry-aware representation w.r.t affine transformations.

Lemma 1. *If $\mathcal{L}$ holds the following conditions for $\forall I \in \mathcal{I}, \forall S \in \mathcal{S}, \forall g \in \mathcal{G}$:*

(i) *$\mathcal{L}(\rho(g)I, \rho(g)S) = \lambda(g)\mathcal{L}(I, S)$ where $\lambda : \mathcal{G} \to \mathbb{R}^+$.*

(ii) *$\exists! S_I$ s.t. $S_I = \arg\min_{S \in \mathcal{S}} \mathcal{L}(I, S)$*

(iii) *$\rho(g)S \in \mathcal{S}$.*

Then optimal sketch representation S_I with given $\mathcal{L}$ is a geometry-aware representation with respect to any $\mathcal{G}$.

Proof. With $\rho(g)S \in \mathcal{S}$, we can also let ρ as a group representation of $\mathcal{S}$.
Let $S'_I = \arg\min_{S \in \mathcal{S}} \mathcal{L}(I, \rho(g)^{-1}S)$.

$$\mathcal{L}(I, S_I) = \min_{S \in \mathcal{S}} \mathcal{L}(I, S) = \min_{S \in \mathcal{S}} \mathcal{L}(I, \rho(g)^{-1}S) = \mathcal{L}(I, \rho(g)^{-1}S'_I), \tag{15}$$

$$\therefore\ \rho(g)^{-1}S'_I = S_I,\ \arg\min_{S\in\mathcal{S}} \mathcal{L}(I,\rho(g)^{-1}S) = \rho(g)\arg\min_{S\in\mathcal{S}} \mathcal{L}(I,S). \tag{16}$$

$$\begin{aligned}
S_{\rho(g)I} &= \arg\min_{S\in\mathcal{S}} \mathcal{L}(\rho(g)I, S) \\
&= \arg\min_{S\in\mathcal{S}} \mathcal{L}(\rho(g)I, \rho(g)\rho(g)^{-1}S) \\
&= \arg\min_{S\in\mathcal{S}} \lambda(g)\mathcal{L}(I, \rho(g)^{-1}S) \quad \leftarrow \textit{condition.}(i) \\
&= \arg\min_{S\in\mathcal{S}} \mathcal{L}(I, \rho(g)^{-1}S) \\
&= \rho(g)\arg\min_{S\in\mathcal{S}} \mathcal{L}(I, S) \quad \leftarrow \textit{equation.}(16) \\
&= \rho(g)S_I
\end{aligned} \tag{17}$$

Therefore, S_I is a geometry-aware representation w.r.t any geometric transformations, where $\rho' = \rho$. □

We now define optimal sketch function $[r\circ f]^*(I) = S_I$ which minimizes $\mathcal{L}$ in Corollary 1, and show that $\mathcal{L}_{percept}$ in Sec. 4.1 can satisfy the first condition of Lemma 1 w.r.t affine transformations in Corollary 2.

Corollary 1. *If Lemma 1 holds, a function $[r\circ f]$ that minimizes the metric between I and $[r\circ f](I)$, i.e., $[r\circ f]^* = \arg\min_f \mathcal{L}(I, [r\circ f](I))$, generates a optimal sketch representation $[r\circ f]^*(I) = S_I$.*

Proof. $\mathcal{L}(I, [r\circ f]^*(I)) = \min_f \mathcal{L}(I, [r\circ f](I)) = \min_{S\in\mathcal{S}} \mathcal{L}(I,S) = \mathcal{L}(I, S_I)$.
Since optimal sketch representation is unique, $[r\circ f]^*(I) = S_I$. □

Corollary 2. *For any $\mathcal{L}' : \mathcal{I}\times\mathcal{S}\to\mathbb{R}$, $\mathcal{L}(I,S) = \mathbb{E}_g[\mathcal{L}'(\rho(g)I, \rho(g)S)]$ satisfies the first condition of Lemma 1.*

Proof. Since $\mathcal{G}$ is closed for its composition,

$$\begin{aligned}
\mathcal{L}(\rho(g)I, \rho(g)S) &= \mathbb{E}_{g'}[\mathcal{L}'(\rho(g')\rho(g)I, \rho(g')\rho(g)S)] \\
&= \mathbb{E}_{g'}[\mathcal{L}'(\rho(g'g)I, \rho(g'g)S)] \\
&= \mathbb{E}_{g''}[\mathcal{L}'(\rho(g'')I, \rho(g'')S)] \\
&= \mathcal{L}(I,S).
\end{aligned} \tag{18}$$

□

Lemma 1 implies that optimal $[r\circ f]$ which minimizes the metric function $\mathcal{L}$ that satisfies the conditions in Lemma 1 can make sketch $S = [r\circ f](I)$ as a geometry-aware representation w.r.t arbitrary geometric transformations. We now show that a set of strokes $\boldsymbol{p} = f(I)$ can also be a geometry-aware representation w.r.t arbitrary affine transformations $\mathcal{A}$.

Lemma 2. *If the following condition holds:*

(i) S_I is a geometry-aware representation w.r.t $\mathcal{A}$.

(ii) The rendering process $r : \mathcal{P}\to\mathcal{S}$ is a $\mathcal{A}$-equivariant map.

(iii) There exists a function $f^* : \mathcal{I}\to\boldsymbol{p}$ s.t. $[r\circ f^*] = [r\circ f]^*$ and $r(f^*(I)) = r(f^*(I')) \Rightarrow f^*(I) = f^*(I')$.

Then $f^*(I)$ with given $\mathcal{L}$ is a geometry-aware representation with respect to $\mathcal{A}$.

Proof. Since r is a $\mathcal{A}$-equivariant map, there exists a ρ' s.t. $\rho(a)r(\boldsymbol{p}) = r(\rho'(a)\boldsymbol{p})$ for $\forall a\in\mathcal{A}, \boldsymbol{p}\in\mathcal{P}$. Then,

$$r(f^*(\rho(a)I)) = [r\circ f]^*(\rho(a)I) = \rho(a)[r\circ f]^*(I) = \rho(a)(r(f^*(I))) = r(\rho'(a)f^*(I)). \tag{19}$$

$$\therefore\ f^*(\rho(a)I) = \rho'(a)f^*(I),\ f^* \text{ is a geometry-aware representation w.r.t } \mathcal{A}.$$

□

Lemma 2 requires additional conditions: r must be $\mathcal{A}$-equivariant map and must be injective in the range of f^*. By using the rendering process as a differentiable rasterizer with parameterized Bézier curves described in Appendix D, conditions to be $\mathcal{A}$-equivariance map can be satisfied.

Lemma 3. *Let $C(s,p) = (1-s)^3 t_1 + s(1-s)^2 t_2 + s^2(1-s)t_3 + s^3 t_4$ a cubic Bézier curve where $p = (t_1, t_2, t_3, t_4, c, w)$, $u \in \Omega$, and $s \in [0,1]$. Then a rendering process with a single stroke $[r'(p)](u) = \exp(-\min_s ||C(s,p) - u||_2^2 / w^2) \cdot c$ is a $\mathcal{A}$-equivariant map, where ρ' is defined as $\rho'(a)(t_1, t_2, t_3, t_4, c, w) = (a.t_1, a.t_2, a.t_3, a.t_4, c, w)$.*

Proof.

$$
\begin{aligned}
\rho(a)[r'(p)](u) &= [r'(p)](a^{-1}.u) \\
&= \exp(-\min_s ||C(t,p) - a^{-1}u||_2^2 / w^2) \cdot c \\
&= \exp(-\min_s ||a.[(1-s)^3 t_1 + s(1-s)^2 t_2 + s^2(1-s)t_3 + s^3 t_4] - u||_2^2 / w^2) \cdot c \\
&= \exp(-\min_s ||[(1-s)^3 a.t_1 + s(1-s)^2 a.t_2 + s^2(1-s)a.t_3 + s^3 a.t_4] - u||_2^2 / w^2) \cdot c \\
&= \exp(-\min_s ||C(s, \rho'(a)p) - u||_2^2 / w^2) \cdot c \\
&= [r'(\rho'(a)p)](u).
\end{aligned}
\tag{20}
$$

Therefore, the rendering process with a single stroke $r' : p \to \mathcal{S}$ is a $\mathcal{A}$-equivariant map. □

Assuming that strokes do not overlap with each other, we can easily expand Lemma 3 with the process of a set of strokes $r : \boldsymbol{p} \to \mathcal{S}$ by composing $r'(p)$ of each stroke into a single image using the composition function described in Appendix D.

To make $r : \boldsymbol{p} \to \mathcal{S}$ injective, we must restrict the range of f^*. For example, the direction of each stroke does not change the rasterized sketch, i.e., $r'(t_1, t_2, t_3, t_4, c, w) = r'(t_4, t_3, t_2, t_1, c, w)$. Moreover, the order between each stroke is also invariant for sketches if the strokes do not overlap each other, i.e., $r(\{p^{(1)}, p^{(2)}\}) = r(\{p^{(2)}, p^{(1)}\})$. Thus, we treat the strokes generated by Stroke Generator in Section 4.1 as a permutation invariant set, forcing the order of each stroke to be left to right with aligning $\boldsymbol{p}_{gt}$ as described in Appendix C.

Combining Lemma 1, Lemma 2, Lemma 3, we obtain Prop. 1.

F. Ablation on integrated representation

Table 7. Quantitative results on ablating the integrated representation z_{LBS+}, using InfoNCE loss as $\mathcal{L}_{embed}$. CLEVR-C refers to the classification of the colors of the rightmost object on the CLEVR dataset, while CLEVR-S refers to the classification of the shapes.

	Geoclidean	CLEVR-C	CLEVR-S	STL-10
z_e	24.55±1.05	74.35±0.82	64.99±0.49	77.09±0.22
z_p	50.60±2.12	**84.70**±0.52	63.74±3.02	72.52±0.44
z_h	50.68±2.77	71.32±0.56	52.35±1.36	73.29±0.42
(z_p, z_h)	**50.97**±1.85	78.47±0.18	62.48±2.05	75.01±0.29
full	50.01±1.58	83.76±0.54	**70.89**±1.56	**79.15**±0.22

We perform the ablation study on LBS (InfoNCE), which uses InfoNCE loss as $\mathcal{L}_{embed}$. As summarized in Table 7, utilizing both z_p and z_h improves performance in various tasks compared to using only z_e. The interesting point is that the concatenated stroke z_p is competitive with conventional features from the CNN encoder with fully aggregated representations. Since z_h seems less important than z_p or z_e for tasks on CLEVR and Geoclidean, we observe that z_h can improve classification accuracy on STL-10 by providing aggregated information about the strokes.

G. Ablation on model architecture

Since the authors of CLIP provided pre-trained models with multiple backbone architectures[3], we conducted an ablation study on the architecture of the CLIP model used to measure $\mathcal{L}_{percept}$. Table 8 shows that the performance does not vary

[3]https://github.com/openai/CLIP

Table 8. Quantitative results on ablating CLIP model. The model is trained on the STL-10 dataset and evaluated for tasks on STL-10 (Class) and CLEVR (Color, Size, and Shape).

Architecture	Class	Color	Size	Shape
RN101	79.15	**53.87**	**92.46**	**59.52**
RN50	79.56	47.96	89.76	55.73
RN50x4	**79.88**	49.25	90.07	56.70
ViT-B/32	79.78	50.55	90.63	57.46
ViT-B/16	79.06	47.99	90.08	57.73

significantly across different designs of the CLIP model, with the use of the ResNet101 architecture resulting in better performance on tasks involving the CLEVR dataset.

For the architecture for the image encoder, we observed that training features with the ResNet architecture are substantially better than using ViT [17] in terms of both overall sketch quality and quantitative metrics.

H. Ablation study on loss function

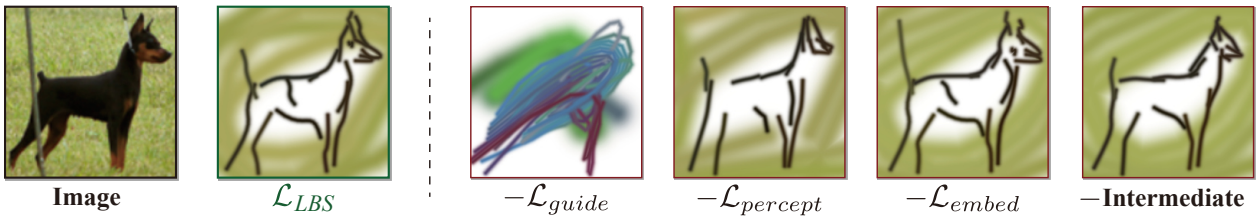


Figure 9. Qualitative results on ablating loss terms in $\mathcal{L}_{LBS}$ trained on STL-10 dataset. The importance of $\mathcal{L}_{guide}$ for training LBS is highlighted. The visualization shows the qualitative effect of removing each loss component.

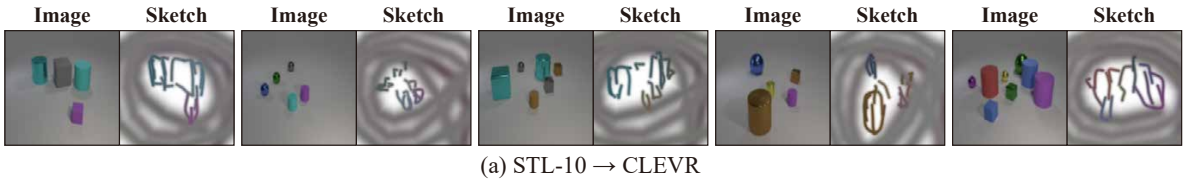
Table 9. Quantitative results on ablating loss terms in $\mathcal{L}_{LBS}$ trained on STL-10 dataset. The classification result in STL-10 is denoted as Class, while other results are evaluated on the CLEVR dataset. LBS is trained with $\lambda_e = 0$, while LBS (InfoNCE) is trained with $\lambda_e \neq 0$. The results demonstrate the impact of the proposed loss function on the model's performance.

	Class	RC	BC	Size	Shape	Material
LBS	78.07	53.87	61.84	**92.46**	59.52	80.06
LBS (InfoNCE)	**79.15**	**54.53**	62.26	91.81	**59.62**	**80.44**
$-\mathcal{L}_{guide}$	74.70	33.00	45.24	83.26	50.26	70.01
$-\mathcal{L}_{percept}$	78.88	53.79	**62.77**	91.49	59.00	79.96
$-$Intermediate	78.13	50.64	58.70	91.17	57.71	77.74

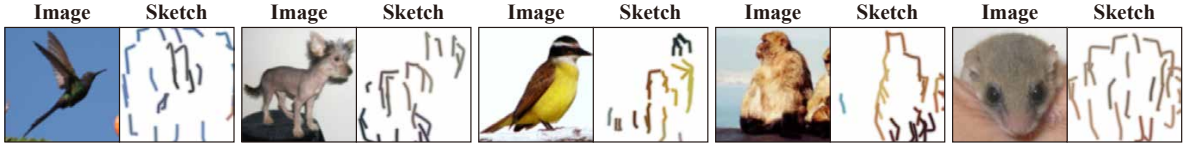
To investigate the effect of each proposed loss term on the learned features, we conducted experiments by ablating $\mathcal{L}_{guide}$ and $\mathcal{L}_{percept}$, and disabling the progressive optimization process for intermediate layers (*i.e.*, using $\mathcal{L}_{guide}$ with only the final layer's output). Additionally, we compare the results of LBS (InfoNCE) to examine the impact of z_h on the learned features. The results presented in Tab. 9 and Fig. 9 show that all loss terms contribute to the final performance of the model in terms of generating informative and high-quality sketches. By including $\mathcal{L}_{embed}$, we observed a slight improvement in performance on the CLEVR dataset. Additionally, we observe a meaningful improvement in high-level inference tasks, such as classification on the STL-10 dataset, while maintaining the quality of the generated sketches. The loss term $\mathcal{L}_{guide}$ is crucial for training LBS, and guiding the intermediate layers also improves the quantitative results while generating higher-quality sketches. The inclusion of $\mathcal{L}_{percept}$ in the loss function resulted in a discernible improvement in the quality of the generated sketches with a measurable performance boost in the quantitative comparison.

I. Qualitative results

Figure 10. Qualitative results on all datasets. We evaluate LBS against various vision datasets and visualize example sketches for each dataset. LBS can effectively represent images' geometric information across various datasets, including rotMNIST, Geoclidean, CLEVR, STL-10, and QMUL-ShoeV2.



(a) STL-10 → CLEVR



(b) CLEVR → STL-10

Figure 11. Qualitative results of the domain transfer experiment, demonstrating the ability of LBS to transfer learned geometric information across different domains.

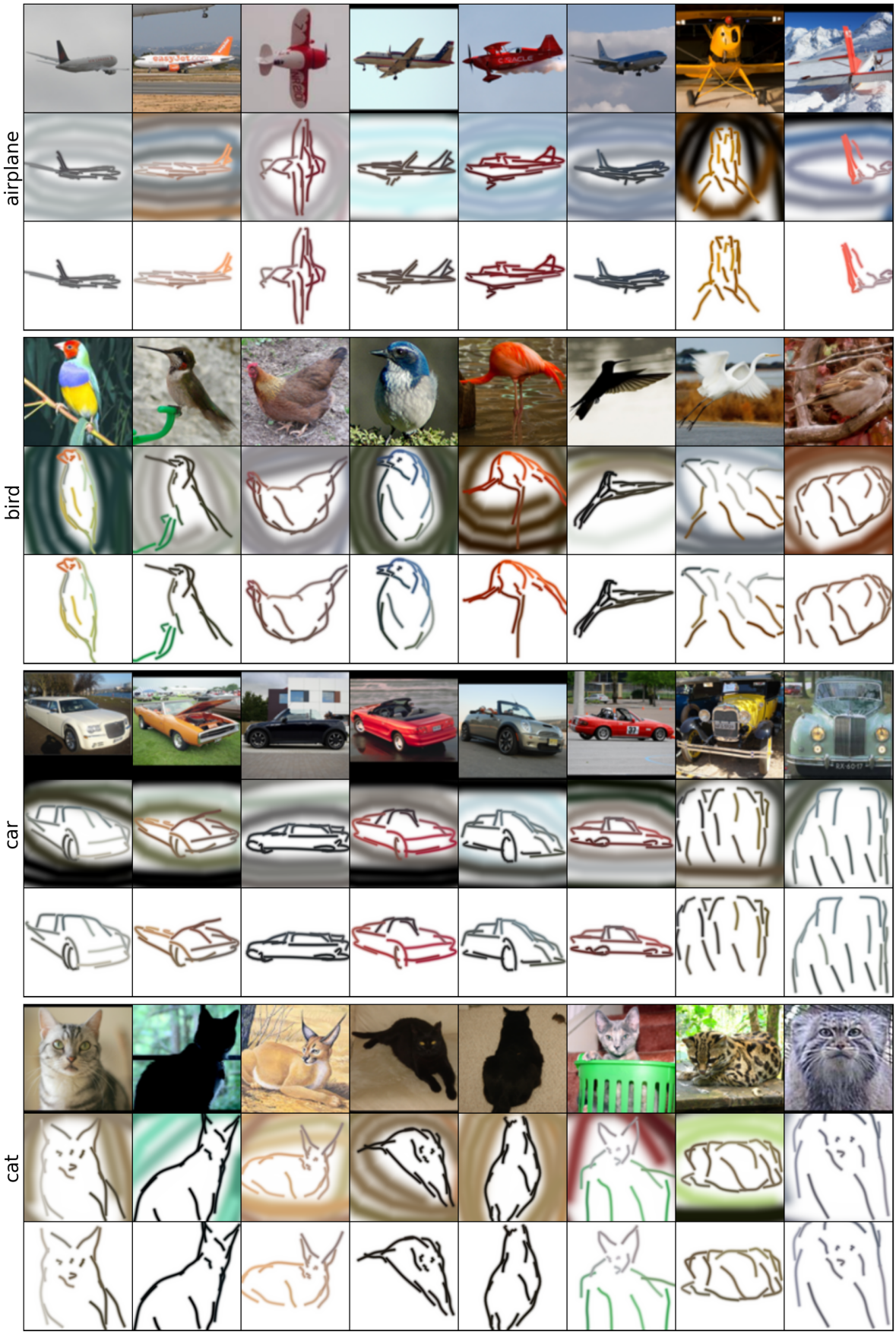


Figure 12. Qualitative results in STL-10. Results for airplane, bird, car, and cat categories on STL-10. Two samples on the right are examples of low-quality generation.

Figure 13. Qualitative results in STL-10. Results for deer, dog, horse, and monkey categories on STL-10. Two samples on the right are examples of low-quality generation.

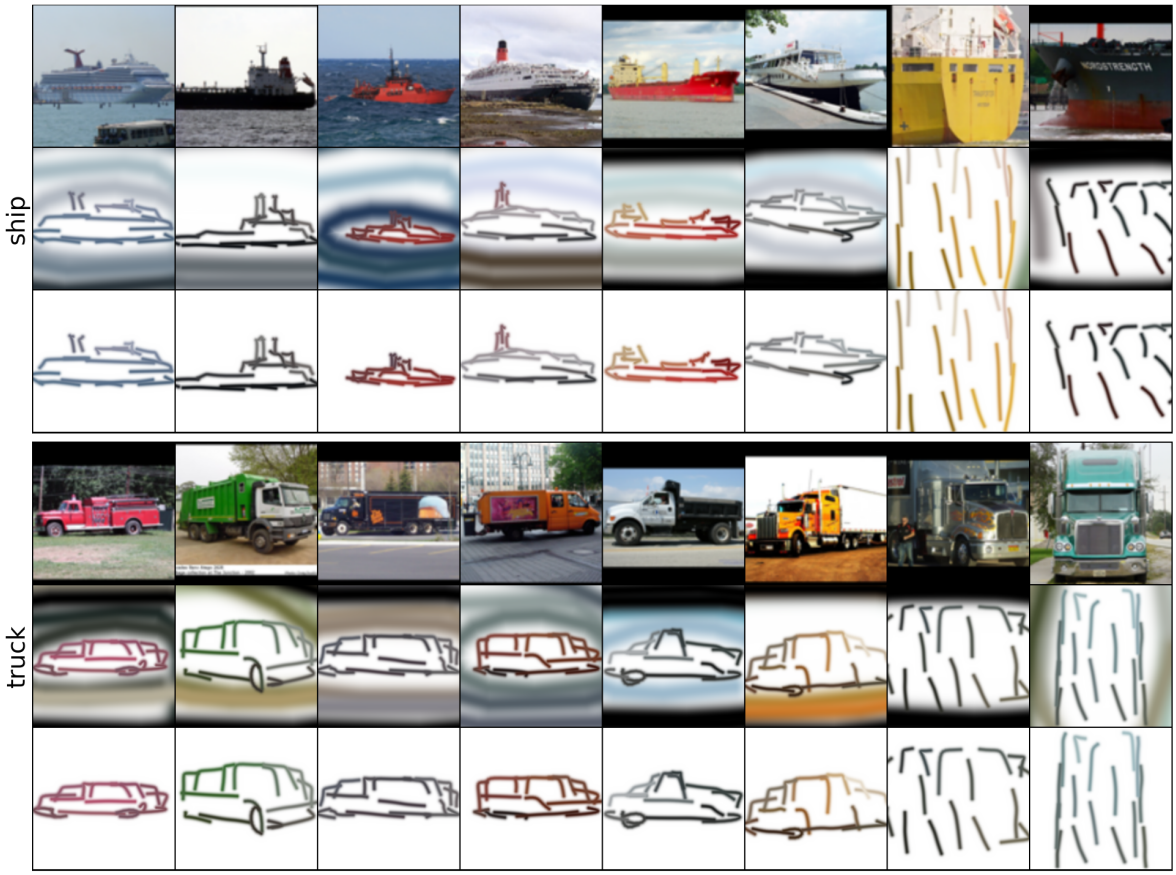


Figure 14. Qualitative results in STL-10. Results for ship and truck categories on STL-10. Two samples on the right are examples of low-quality generation.