Deep Parametric Shape Predictions using Distance Fields

Dmitriy Smirnov[1] Matthew Fisher[2] Vladimir G. Kim[2] Richard Zhang[2] Justin Solomon[1]
[1]Massachusetts Institute of Technology [2]Adobe Research

Abstract

Many tasks in graphics and vision demand machinery for converting shapes into consistent representations with sparse sets of parameters; these representations facilitate rendering, editing, and storage. When the source data is noisy or ambiguous, however, artists and engineers often manually construct such representations, a tedious and potentially time-consuming process. While advances in deep learning have been successfully applied to noisy geometric data, the task of generating parametric shapes has so far been difficult for these methods. Hence, we propose a new framework for predicting parametric shape primitives using deep learning. We use distance fields to transition between shape parameters like control points and input data on a pixel grid. We demonstrate efficacy on 2D and 3D tasks, including font vectorization and surface abstraction.

1. Introduction

The creation, modification, and rendering of parametric shapes, such as in vector graphics, is a fundamental problem of interest to engineers, artists, animators, and designers. Such representations offer distinct advantages. By expressing shapes as collections of primitives, we can easily apply transformations and render at arbitrary resolution while storing only a sparse representation. Moreover, generating parametric representations that are *consistent* across inputs enables us to learn common underlying structure and estimate correspondences between shapes, facilitating tools for retrieval, exploration, style/structure transfer, and so on.

It is often useful to generate parametric models from data that do not directly correspond to the target geometry and contain imperfections or missing parts. This can be an artifact of noise, corruption, or human-generated input; often, an artist intends to create a precise geometric object but produces one that is "sketchy" and ambiguous. Hence, we turn to machine learning methods, which have shown success in inferring structure from noisy data.

Convolutional neural networks (CNNs) achieve state-of-the-art results in vision tasks such as image classification [22], segmentation [25], and image-to-image translation [19]. CNNs, however, operate on *raster* representations.

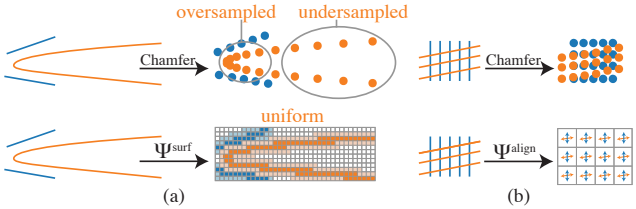


Figure 1: Drawbacks of Chamfer distance (above) fixed by our losses (below). In (a), sampling uniformly in the parameter space of a Bèzier curve (orange) yields oversampling at the high-curvature area, resulting in a low Chamfer distance to the segments (blue). Our method yields a spatially uniform representation. In (b), two sets of nearly-orthogonal line segments have near-zero Chamfer distance despite misaligned normals. We explicitly measure normal alignment.

Grid structure is fundamentally built into convolution as a mechanism for information to travel between network layers. This structure is leveraged to optimize GPU performance. Recent deep learning pipelines that output vector shape primitives have been significantly less successful than pipelines for analogous tasks on raster images or voxelized volumes.

A challenge in applying deep learning to parametric geometry is the combination of Eulerian and Lagrangian representations. CNNs process data in an *Eulerian* fashion, applying fixed operations to a dense grid; Eulerian shape representations like indicator functions come as values on a fixed grid. Parametric shapes, on the other hand, use sparse sets of parameters like control points to express geometry. In contrast to stationary Eulerian grids, this *Lagrangian* representation moves with the shape. Mediating between Eulerian and Lagrangian geometry is key to any learning pipeline for the problems above, a task we consider in detail.

We propose a learning framework for predicting parametric shapes, addressing the aforementioned issues. By analytically computing a distance field to the primitives during training, we formulate an Eulerian version of Chamfer distance, a common metric for geometric similarity [41, 12, 24, 17]. Our metric does not require samples from the predicted or target shapes, eliminating artifacts that emerge due to nonuniform sampling. Additionally, our distance field enables alternative loss functions that are sensitive to specific geometric qualities like alignment. We illustrate the advantages of our

method over Chamfer distance in Figure 1.

We apply our new framework in 2D to a diverse dataset of fonts, training a network that takes in a raster image of a glyph and outputs a collection of Bézier curves. This effectively maps glyphs onto a common set of parameters that can be traversed intuitively. We use this embedding for font exploration and retrieval, correspondence, and interpolation in a completely self-supervised setting, without need for human labelling or annotation.

We also show that our approach works in 3D. With surface primitives in place of curves, we perform abstraction on ShapeNet [7], outputting parametric primitives to approximate each input. Our method can produce consistent shape segmentations, outperforming state-of-the-art deep cuboid fitting of Tulsiani et al. [41] on semantic segmentation.

Contributions. We present a technique for predicting parametric shapes from 2D and 3D raster data, including:

- a *general distance field loss function* motivating several self-supervised losses based on a common formulation;
- application to 2D *font glyph vectorization*, with application to correspondence, exploration, retrieval, and repair;
- application to 3D *surface abstraction*, with results for different primitives and constructive solid geometry (CSG) as well as application to segmentation.

2. Related Work

Deep shape reconstruction. Reconstructing geometry from one or more viewpoints is crucial in applications like robotics and autonomous driving [13, 35, 38]. Recent deep networks can produce point clouds or voxel occupancy grids given a single image [12, 8], but their output suffers from fixed resolution.

Learning signed distance fields defined on a voxel grid [9, 37] or directly [30] allows high-resolution rendering but requires surface extraction; this representation is neither sparse nor modular. Liao et al. address the rendering issue by incorporating marching cubes into a differentiable pipeline, but the lack of sparsity remains problematic, and predicted shapes are still on a voxel grid [23].

Parametric shapes offer a sparse, non-voxelized solution. Methods for converting point clouds to geometric primitives achieve high-quality results but require supervision, either relying on existing labeled data [27, 26, 15] or prescribed templates [14]. Groueix et al. output primitives at any resolution, but their primitives are not naturally parameterized or sparsely represented [17]. Genova et. al. propose to represent geometry as isosurfaces of axis-aligned Gaussians [16]. Others [17, 39, 31] develop tailored primitives but use standard Chamfer distance as the loss objective. We demonstrate and address the issues inherent in Chamfer distance.

Font exploration and manipulation. Designing or even finding a font can be tedious using generic vector graphics tools. Certain geometric features distinguish letters from one another across fonts, while others distinguish fonts from one another. Due to these difficulties and the presence of large font datasets, font exploration, design, and retrieval have emerged as challenging problems in graphics and learning.

Previous exploration methods categorize and organize fonts via crowdsourced attributes [28] or embed fonts on a manifold using purely geometric features [6, 2]. Instead, we leverage deep vectorization to automatically generate a sparse representation for each glyph. This enables exploration on the basis of general shape rather than fine detail.

Automatic font generation methods usually fall into two categories. Rule-based methods [40, 32] use engineered decomposition and reassembly of glyphs into parts. Deep learning approaches [1, 43] produce raster images, with limited resolution and potential for image-based artifacts, making them unfit for use as glyphs. We apply our method to edit existing fonts while retaining vector structure and demonstrate vectorization of glyphs from noisy partial data.

Parametric shape collections. As the number of publicly-available 3D models grows, methods for organizing, classifying, and exploring models become crucial. Many approaches decompose models into modular parametric components, commonly relying on prespecified templates or labeled collections of specific parts [20, 36, 29]. Such shape collections prove useful in domain-specific applications in design and manufacturing [34, 42]. Our deep learning pipeline allows generation of parametric shapes to perform these tasks. It works quickly on new inputs at test time and is generic, handling a variety of modalities without supervision and producing different output types.

3. Preliminaries

Let $A, B \subset \mathbb{R}^n$ be two measurable shapes. Let X and Y be two point sets sampled uniformly from A and B. The *directed Chamfer distance* between X and Y is

$$\mathrm{Ch}_{\mathrm{dir}}(X, Y) = \frac{1}{|X|} \sum_{x \in X} \min_{y \in Y} \|x - y\|_2^2, \tag{1}$$

and the *symmetric Chamfer distance* is defined as

$$\mathrm{Ch}(X, Y) = \mathrm{Ch}_{\mathrm{dir}}(X, Y) + \mathrm{Ch}_{\mathrm{dir}}(Y, X). \tag{2}$$

These were proposed for computational applications in [5] and have been used as a loss function assessing similarity of a learned shape to ground truth in learning [41, 12, 24, 17].

To relate our proposed loss to Chamfer distance, we define *variational directed Chamfer distance* as

$$\mathrm{Ch}_{\mathrm{dir}}^{\mathrm{var}}(A, B) = \frac{1}{\mathrm{Vol(A)}} \int_A \inf_{y \in B} \|x - y\|_2^2 \, \mathrm{d}V(x), \tag{3}$$

with *variational symmetric Chamfer distance* $\mathrm{Ch}(A, B)^{\mathrm{var}}$ defined analogously, extending (1) and (2) to smooth objects.

If points are sampled uniformly, under relatively weak assumptions, $\mathrm{Ch}(X,Y)\to 0$ iff $A=B$ as the number of samples grows, making it a reasonable shape matching metric. Chamfer distance, however, has fundamental drawbacks:

- It is highly dependent on the sampled points and sensitive to non-uniform sampling, as in Figure 1a.
- It is agnostic to normal alignment. As in Figure 1b, Chamfer distance between a dense set of vertical lines and a dense set of horizontal lines approaches zero.
- It is slow to compute. For each x sampled from A, it is necessary to find the closest y sampled from B, a quadratic-time operation when implemented naïvely. Efficient structures like k-d trees are not well-suited to GPUs.

Our method does not suffer from these disadvantages.

4. Method

We introduce a framework for formulating loss functions suitable for learning parametric shapes in 2D and 3D; our formulation not only generalizes Chamfer distance but also leads to stronger loss functions that improve performance on a variety of tasks. We start by defining a general loss on distance fields and propose two specific losses.

4.1. General Distance Field Loss

Given $A, B \subseteq \mathbb{R}^n$, let $\mathrm{d}_A, \mathrm{d}_B : \mathbb{R}^n \to \mathbb{R}_+$ measure distance from each point in $\mathbb{R}^n$ to A and B, respectively, $\mathrm{d}_A(x) := \inf_{y\in A} \|x-y\|_2$. In our experiments, $n \in \{2,3\}$. Let $S \subseteq \mathbb{R}^n$ be a bounded set with $A, B \subseteq S$. We define a *general distance field loss* as

$$\mathcal{L}_\Psi[A,B] = \frac{1}{\mathrm{Vol}(S)} \int_{x\in S} \Psi_{A,B}(x)\,\mathrm{d}V(x), \tag{4}$$

for some measure of discrepancy Ψ. Note that we represent A and B only by their respective distance functions, and the loss is computed over S.

Let $\Phi \in \mathbb{R}^p$ be a collection of parameters defining a shape $S_\Phi \subseteq \mathbb{R}^n$. For instance, if S_Φ consists of Bézier curves, Φ contains a list of control points. Given a target shape $T \subseteq \mathbb{R}^n$, we formulate fitting a parametric shape to approximate T w.r.t. Ψ as minimizing

$$f_\Psi(\Phi) = \mathcal{L}_\Psi[S_\Phi, T]. \tag{5}$$

For optimal shape parameters, $\hat{\Phi} := \arg\min_\Phi f_\Psi(\Phi)$. We propose two discrepancy measures, providing loss functions that capture different geometric features.

4.2. Surface Loss

We define surface discrepancy to be

$$\Psi^{\mathrm{surf}}_{A,B}(x) = \delta\{\ker \mathrm{d}_A^2\}(x)\mathrm{d}_B^2(x) + \delta\{\ker \mathrm{d}_B^2\}(x)\mathrm{d}_A^2(x) \tag{6}$$

where $\delta\{X\}$ is the Dirac delta defined uniformly on X, and $\ker f$ denotes the zero level-set of f. $\Psi^{\mathrm{surf}} > 0$ iff the shapes do not match, making it sensitive to local geometry:

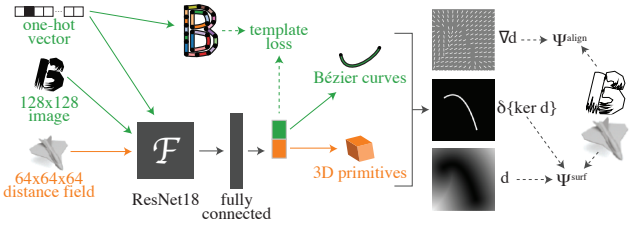


Figure 2: An overview of our pipelines—font vectorization (green) and 3D abstraction (orange).

Proposition 1 *The symmetric variational Chamfer distance between $A, B \subseteq \mathbb{R}^n$ is equal to the surface loss between A and B, i.e.,* $\mathrm{Ch}^{\mathrm{var}}(A,B) = \mathcal{L}_{\Psi^{surf}_{A,B}}$.

Unlike Chamfer distance, the discrete version of our surface loss can be approximated efficiently without sampling points from either the parametric or target shape via evaluation over a regular grid, as we show in §4.4.

4.3. Normal Alignment Loss

We define normal alignment discrepancy to be

$$\Psi^{\mathrm{align}}_{A,B}(x) = 1 - \langle \nabla\,\mathrm{d}_A(x), \nabla\,\mathrm{d}_B(x)\rangle^2. \tag{7}$$

Minimizing $f_{\Psi^{\mathrm{align}}}$ aligns normals of the predicted primitives to those of the target. Following Figure 1b, if A contains dense vertical lines and B contains horizontal lines, $\mathcal{L}_{\Psi^{\mathrm{align}}_{A,B}} \gg 0$ while $\mathrm{Ch}(A,B) \approx 0$.

4.4. Final Loss Function

The general distance field loss and proposed discrepancy measures are differentiable w.r.t. the shape parameters Φ, as long as d_{S_Φ} is differentiable w.r.t. Φ. Thus, they are well-suited to be optimized by a deep network predicting parametric shapes. We approximate (4) via Monte Carlo integration:

$$\mathcal{L}_\Psi[A,B] \approx \frac{1}{|G|}\sum_{x\in G} \Psi_{A,B}(x), \tag{8}$$

where G is a 2D or 3D grid.

While we use a voxel grid grid to compute the integrals in our loss function, the resolution of the voxel grid only affects quadrature without limiting the resolution of our representation. The grid dictates how we sample distance values; the values themselves are derived from a continuous parametric representation. A small subvoxel change in the geometry will affect the distance value at multiple discrete voxels. This property is in distinct contrast to representations that only consider the occupancy grid of a shape—the resolution of such representations is strictly limited by the grid resolution.

For Ψ^{surf}, we use $\mathrm{Smootherstep}(1 - \mathrm{d}_A^2/\gamma^2)$ (with Smootherstep defined as in [11]) as a smooth version of

$\delta\{\ker \mathrm{d}_A^2\}$ to evaluate the expression on a grid and to avoid discontinuities, enabling smooth gradients in our optimization. We set γ to twice the diameter of a voxel. For Ψ^{align}, we approximate gradients using finite differences.

We minimize $f_\Psi = f_{\Psi^{\text{surf}}} + \alpha^{\text{align}} f_{\Psi^{\text{align}}}$, determining $\alpha^{\text{align}} = 0.01$ for all experiments using cross-validation.

4.5. Network Architecture and Training

The network takes a 128×128 image or a $64 \times 64 \times 64$ distance field as input and outputs a parametric shape. We encode our input to a $\mathbb{R}^{256}$ latent space using a ResNet-18 [18] architecture. We then use a fully connected layer with 256 units and ReLU nonlinearity followed by a fully connected layer with number of units equal to the dimension of the target parameterization. We pass the output through a sigmoid and rescale it depending on the parameters being predicted. Our pipeline is illustrated in Figure 2. We train each network on a single Tesla GeForce GTX Titan X GPU for approximately one day, using Adam [21] with learning rate 10^{-4} and batch size 32 for 2D and 16 for 3D.

5. 2D: Font Exploration and Manipulation

We demonstrate our method in 2D for font glyph vectorization. Given a raster image of a glyph, our network outputs control points defining a collection of quadratic Bézier curves that approximate its outline. We produce nearly exact vector representations of glyphs from simple (non-decorative) fonts. From a decorative glyph with fine-grained detail, however, we recover a good approximation of the glyph's shape using a small number of Bézier primitives and a consistent structure. This process can be interpreted as projection onto a common latent space of control points.

We first describe our choice of primitives as well as the computation of their distance fields. We introduce a template-based approach to allow our network to better handle multi-modal data (different letters) and test several applications.

5.1. Approach

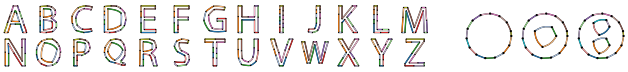
Primitives. We wish to use a 2D parametric shape primitive that is sparse and expressive and admits an analytic distance field. Our choice is the *quadratic Bèzier curve* (which we refer to as *curve*), parameterized by control points $a, b, c \in \mathbb{R}^2$ and defined by $\gamma(t) = (1-t)^2 a + 2(1-t)tb + t^2 c$, for $0 \le t \le 1$. We represent 2D shapes as the union of n curves parameterized by $\Phi = \{a_i, b_i, c_i\}_{i=1}^n \subseteq \mathbb{R}^{3n}$.

Proposition 2 *Given a curve γ parameterized by $a, b, c \in \mathbb{R}^2$ and a point $p \in \mathbb{R}^2$, the $\hat{t} \in \mathbb{R}$ such that $\gamma(\hat{t})$ is the closest point on the curve to p satisfies the following:*

$$\begin{aligned}\langle B, B\rangle \hat{t}^3 + 3\langle A, B\rangle \hat{t}^2 + (2\langle A, A\rangle + \langle B, a-p\rangle)\hat{t} \\ + \langle A, a-p\rangle = 0,\end{aligned} \tag{9}$$

where $A = b - a$ and $B = c - 2b + a$.

Figure 3: Glyphs with predicted boundary curves rendered with predicted stroke thickness. The network thickens curves to account for stylistic details at the glyph boundaries.



(a) Lettter templates (a) Simple templates

Figure 4: Font glyph templates. These determine the connectivity and initialize the placement of the predicted curves.

Thus, evaluating the distance to a single curve $\mathrm{d}_{\gamma_i}(p) = \|p - \gamma_i(\hat{t})\|_2$ requires finding the roots of a cubic [33], which we can do analytically. To compute distance to the union of the curves, we take a minimum: $\mathrm{d}_\Phi(p) = \min_{i=1}^n \mathrm{d}_{\gamma_i}(p)$.

In addition to the control points, we predict a stroke thickness for each curve. We use this parameter when computing the loss by "lifting" the predicted distance field, thus thickening the curve—if curve γ has thickness s, we set $d_\gamma^s(p) = \min(d_\gamma(p) - s, 0)$. While we do not visualize stroke thickness in our experiments, this approach allows the network to thicken curves to better match high-frequency filigree (see Figure 3). This thickening is a simple operation in our distance field representation; sampling-based methods do not provide a natural way to thicken predicted geometry.

Templates. Our training procedure is self-supervised, as we do not have ground truth curve annotations. To better handle the multimodal nature of our entire dataset with a single network, we label each training example with its letter, passed as additional input. This allows us to condition on input class by concatenating a 26-dimensional one-hot vector to the input, a common technique for conditioning [44].

We choose a "standard" curve representation per letter, capturing each letter's distinct geometric and topological features, by designing 26 templates from a shared set of control points. A *template* of type $\ell \in \{\mathrm{A}, \ldots, \mathrm{Z}\}$ is a collection of points $T_\ell = \{p_1, \ldots, p_n\} \subseteq \mathbb{R}^{2n}$ with corresponding *connectivity* determining how the points define curves. Since our curves form closed loops, we reuse endpoints.

For glyph boundaries of uppercase English letters, there are three connectivity types—one loop (e.g., "C"), two loops (e.g., "A"), and three loops ("B"). In our templates, the first loop has 15 curves and the other loops have 4 curves each. We will show that while letter templates (Figure 4a) better specialize to the boundaries of each glyph, we still achieve good results with simple templates (Figure 4b). Even without letter-specific templates, our system learns a consistent geometric representation, establishing cross-glyph correspondences purely using self-supervision.

We use predefined templates together with our labeling of each training example for two purposes. First, connectivity

is used to compute curve control points from the network output. Second, they provide a *template loss*:

$$\mathcal{L}^{\text{template}}(\ell, x) = \alpha^{\text{template}} e^{(t/s)} \|T_\ell - h^t(x)\|_2^2, \qquad (10)$$

where $s \in \mathbb{Z}_+$, $\gamma \in (0, 1)$, t is the iteration number, x is the input image, and $h^t(x)$ is the network output at iteration t. This initializes the network output, such that an input of type ℓ initially maps to template ℓ. As this term decays, the other loss terms take over. We set $\alpha^{\text{template}} = 10$ and $s = 500$, though other choices of parameters for which the template term initially overpowers the rest of the loss also work.

5.2. Experiments

We train our network on the 26 uppercase English letters extracted from nearly 10,000 fonts. The input is a raster image of a letter, and the target distance field to the boundary of the original vector representation is precomputed.

Ablation study. We demonstrate the benefit of our loss over Chamfer distance as well as the contribution of each of our loss terms. While having 26 unique templates helps achieve better results, it is not crucial—we evaluate a network trained with three "simple templates" (Figure 4b), which capture the three topology classes of our data.

Model	Average error
Full model (ours)	**0.509**
No surface term (ours)	1.613
No alignment term (ours)	0.642
Simple templates (ours)	0.641
Chamfer (with letter templates)	0.623
AtlasNet [17]	5.154

Table 1: Comparison between subsets of our full model as well as standard Chamfer distance and AtlasNet. Average error is Chamfer distance (in pixels on a 128×128 image) between ground truth and uniformly sampled predicted curves.

For the Chamfer loss experiment, we use the same hyperparameters as for our method and sample 5,000 points from the source and target geometry. We initialize the model output to the full letter templates, like in our full model.

We also evaluate on 20 sans-serif fonts, computing Chamfer distance between our predicted curves and ground truth geometry, sampling uniformly (average error in Table 1). Uniform sampling is a computationally-expensive and non-differentiable procedure only for evaluation *a posteriori*—not suitable for training. While it does not correct all of Chamfer distance's shortcomings, we use it as a baseline to evaluate quality. We limit to sans-serif fonts since we do not expect to faithfully recover local geometry. Our full loss outperforms Chamfer loss, and both our loss terms are necessary. Figure 5 shows qualitative results on test set glyphs; see supplementary material for additional results.

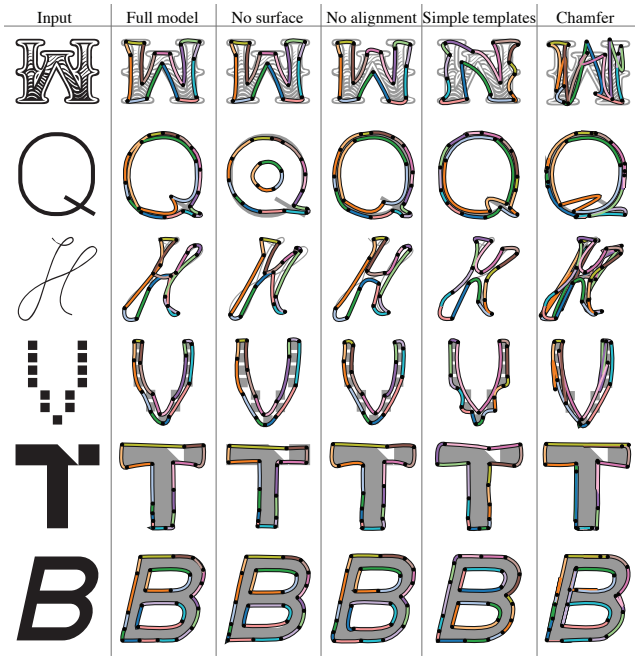


Figure 5: Ablation study and comparison to Chamfer.

We demonstrate robustness in Figure 6 by quantizing our loss values and plotting the number of examples for each value. High loss outliers are generally caused by noisy data—they are either not uppercase English letters or have fundamentally uncommon structure.

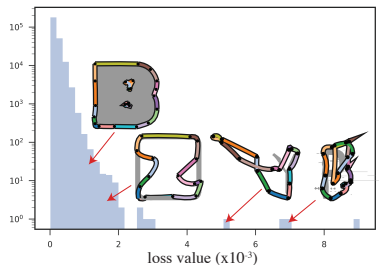
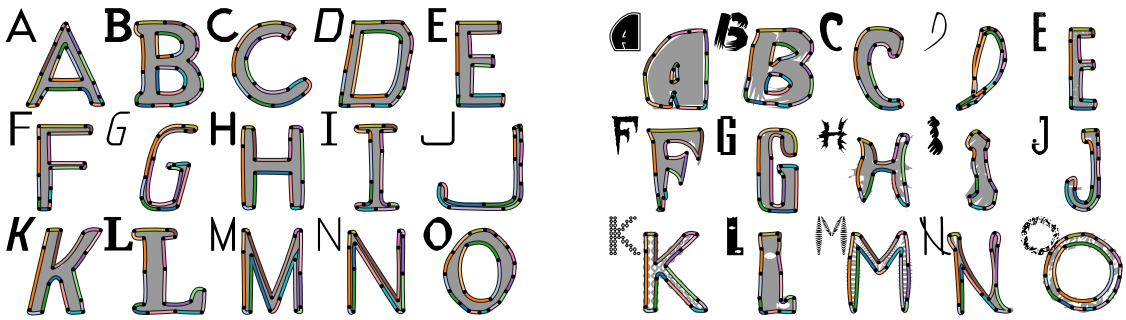


Figure 6: Number of examples per quantized loss value. We visualize the input and predicted curves for several outliers.

Comparison to AtlasNet. In AtlasNet [17], geometry is reconstructed by training implicit decoders, which map a point in the unit square to a point on the target surface, optimizing Chamfer distance. We modify the AtlasNet system to our task and demonstrate that our method method proposes a more effective geometry representation and loss.

AtlasNet represents shapes as points in a learned high dimensional space, which does not obviously correlate to geometric features. Thus, in contrast to our explicit representation as a collection of control points, it does not facilitate geometric interpretability. Additionally, this makes it difficult to impose geometric priors—it is unclear how to initialize AtlasNet to predefined templates, as we do in §5.1.



(a) Plain font glyphs (b) Decorative font glyphs

Figure 7: Vectorization of various glyphs. For each we show the raster input (top left,black) along with the vectorization (colored curves) superimposed. When the input has simple structure (a), we recover an accurate vectorization. For fonts with decorative details (b), our method places curves to capture overall structure. Results are taken from the test dataset.

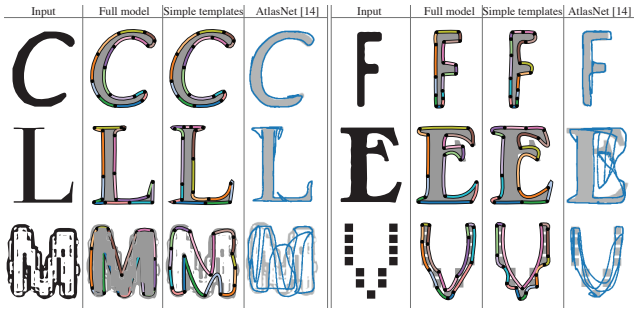


Figure 8: Comparison to AtlasNet [17] with a closed loop start shape to our simple templates and full models. We only train (and test) AtlasNet on letters with a single loop.

For a fair comparison, we train an AtlasNet model that maps points from the boundary of a circle (rather than the interior of a square) into 2D. We only train on letters with single loop topology (*C, E, F*, etc.) and sample 5,000 points. Thus, this setting is comparable to the *simple templates* experiment from our ablation.

We show results in Figure 8. Although AtlasNet recovers overall structure of the input, it suffers from artifacts, self-intersections, and imprecision not exhibited by our method, even with simple templates. Likely, this is due to the fact that AtlasNet exhibits the drawbacks of Chamfer distance identified in §3, i.e., non-uniform sampling and lack of sensitivity to normal alignment. We include a quantitative comparison in Table 1. Our method outperforms AtlasNet even based on on a uniformly-sampled Chamfer distance metric.

Vectorization. For any font glyph, our method generates a consistent sparse vector representation, robustly and accurately describing the glyph's structure while ignoring decorative and noisy details. For simple fonts, our representation is a near-perfect vectorization, as in Figure 7a. For decorative glyphs, our method produces a meaningful abstraction. While a true vectorization would contain many curves with a large number of connected components, we succinctly capture the glyph's overall structure (Figure 7b).

Our method preserves semantic correspondences. The same curve is consistently used for the boundary of, e.g., the top of an "I". These correspondences persist *across* letters with both full and simple templates—see, e.g., the "E" and "F" in Figure 7a and 7b and "simple templates" in Figure 5.

Retrieval and exploration. Our sparse representation can be used to explore the space of glyphs, useful for artists and designers, without the need for manual labelling or annotation. Treating control points as a metric space, we can perform Euclidean nearest-neighbor lookups for font retrieval.

In Figure 9, for each query glyph, we compute its curve representation and retrieve seven nearest neighbors in curve space. Because our representation captures geometric structure, we find glyphs that are similar structurally, despite decorative and stylistic differences.

We can also consider a path in curve space starting at the curves for one glyph and ending at those for another. By sampling nearest neighbors along this trajectory, we "interpolate" between glyphs. As in Figure 10, this produces meaningful collections of fonts for the same letter and reasonable results when the start and end glyphs are different letters. Additional results are in supplementary material.

Figure 9: Nearest neighbors for a glyph in curve space, sorted by proximity. The query glyph is in orange.

Nearest-neighbor lookups in curve space also can help find a font matching desired geometric characteristics. A possible workflow is in Figure 11—through incremental refinements of the curves the user can quickly find a font.

Style and structure mixing. Our sparse curve representation describes geometric structure, ignoring stylistic and

Figure 10: Interpolating between fonts in curve space. The start and end are in orange and blue, respectively, and the nearest glyphs to linear interpolants are shown in order.

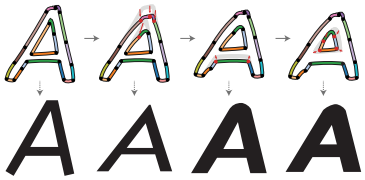


Figure 11: User-guided font exploration. At each edit, the nearest glyph is displayed below. This lets the user explore the dataset through geometric refinements.

Figure 12: Mixing of style (columns) and structure (rows) of the *A* glyph from different fonts. We deform each starting glyph (orange) into the structure of each target glyph (blue).

decorative details. We leverage this to warp a glyph with desired style to the structure of another glyph (Figure 12).

We first generate the sparse curve representation for source and target glyphs. Since our representation uses the same set of curves, we can estimate dense correspondences and use them to warp original vectors of the source glyph to conform to the shape of the target. For each point on the source, we apply a translation that is a weighted sum of the translations from the sparse curve control points in the source glyph to those in the target glyph.

Repair. Our system learns a strong prior on glyph shape, allowing us to robustly handle noisy input. In [1], a generative adversarial network (GAN) generates novel glyphs. The outputs, however, are raster images, often with noise and missing parts. Figure 13 shows how our method can simultaneously vectorize and repair GAN-generated glyphs. Compared to a vectorization tool like Adobe Illustrator Live Trace, we infer missing data based on learned priors, making the glyphs usable starting points for font design.

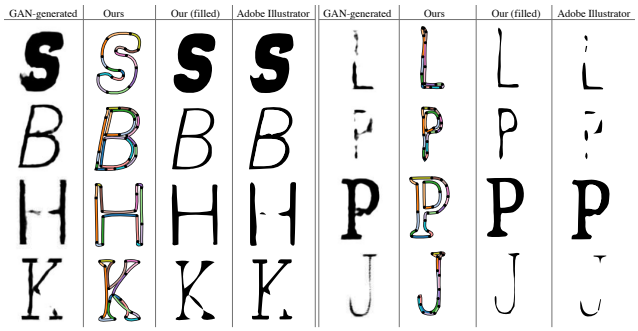


Figure 13: Vectorized GAN-generated fonts from [1].

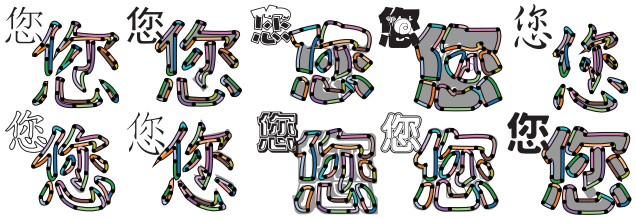


Figure 14: Vectorization of Chinese character 您.

Other glyphs. Our method generalizes to more complex input than uppercase English glyphs. We demonstrate this by training a model to vectorize the Chinese character 您, which has significant geometric and topological complexity. We use a template that roughly captures the structure of the character. Results on several fonts are shown in Figure 14.

6. 3D: Volumetric Primitive Prediction

We reconstruct 3D surfaces out of various primitives, which allow our model to be expressive, sparse, and abstract.

6.1. Approach

Our first primitive is a *cuboid*, parameterized by $\{b, t, q\}$, where $b = (w, h, d)$, $t \in \mathbb{R}^3$ and $q \in \mathbb{S}^4$ a quaternion, i.e., an origin-centered (hollow) rectangular prism with dimensions $2b$ to which we apply rotation q and then translation t.

Proposition 3 *Let C be a cuboid with parameters $\{b, t, q\}$ and $p \in \mathbb{R}^3$ a point. Then, the signed distance between p and C is*

$$\mathrm{d}_C(p) = \|\max(d, 0)\|_2 + \min(\max(d_x, d_y, d_z), 0), \quad (11)$$

where $p' = q^{-1}(p - t)q$ using the Hamilton product and $d = (|p'_x|, |p'_y|, |p'_z|) - b$.

Inspired by [31], we additionally use a *rounded cuboid* primitive by introducing a radius parameter r and computing the signed distance by $\mathrm{d}_{RC}(p) = \mathrm{d}_C(p) - r$.

A unique advantage of our distance field representation is the ability to perform CSG boolean operations. Since our distances are *signed*, we can compute the distance to the union of n primitives by taking a minimum over distance

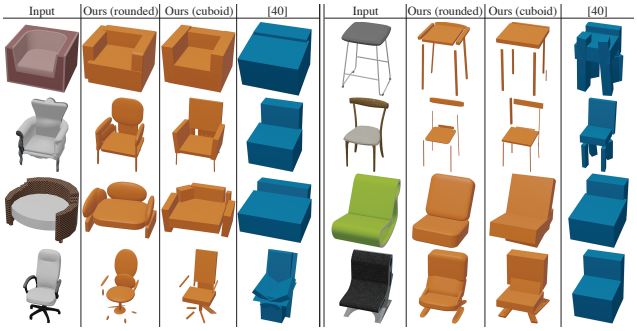


Figure 15: Abstractions of test set chairs using our method and the method of [41].

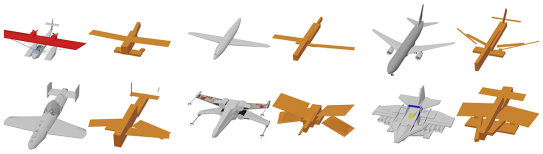


Figure 16: Cuboid abstractions of test set airplanes.

fields. With sampling-based methods such as Chamfer distance optimization, care must be taken to avoid sampling interior faces that are not part of the outer surface.

6.2. Experiments

We train on the airplane and chair categories of ShapeNet Core V2 [7], taking as input a distance field. Thus, our method is fully self-supervised.

Surface abstraction. In Figure 15, for each ShapeNet chair, we show the our cuboid abstraction, our rounded cuboid abstraction, and the abstraction of [41]. We show our cuboid abstractions of ShapeNet airplanes in Figure 16. Each of our networks outputs 16 primitives, and we discard cuboids with high overlap using the method of [41]. The resulting abstractions capture high-level structures of the input. See supplementary material for additional results.

Segmentation. Because we place cuboids consistently, we can use them for segmentation. Following [41], we demonstrate on the COSEG chair dataset. We first label each cuboid predicted by our network (trained on ShapeNet chairs) with a segmentation class (seat, back, legs). Then, we generate a cuboid decomposition of each chair in the dataset and segment according to the nearest cuboid. We achieve a mean accuracy of 94.6%, exceeding the 89.0% accuracy of [41].

CSG operations. In Figure 17, we show results of a network that outputs parameters for the union of eight rounded cuboids minus eight rounded cuboids. For inputs compatible with this template, we get good results. It is unclear how to achieve unsupervised CSG predictions using Chamfer loss.

7. Conclusion

Representation is a key theme in deep learning—and machine learning more broadly—applied to geometry. Assorted

Figure 17: Test set chair CSG abstractions. We predict eight rounded cuboids minus eight other rounded cuboids.

means of communicating a shape to and from a deep network present varying tradeoffs between efficiency, quality, and applicability. While considerable effort has been put into choosing representations for certain tasks, the tasks we consider have *fixed* representations for the input and output: They take in a shape as a function on a grid and output a sparse set of parameters. Using distance fields and derived functions as intermediate representations is natural and effective, not only performing well empirically but also providing a simple way to describe geometric loss functions.

Our learning procedure is applicable to many additional tasks. A natural next step is to incorporate our network into more complex pipelines for tasks like vectorization of complex drawings [3], for which the output of a learning procedure needs to be combined with classical techniques to ensure smooth, topologically valid output. A challenging direction might be to incorporate user guidance into training or evaluation, developing the algorithm as a partner in shape reconstruction rather than generating a deterministic output.

Our experiments suggest several extensions for future work. The key drawback of our approach is the requirement of closed-form distances for the primitives. While there are many primitives that could be incorporated this way, a fruitful direction might be to alleviate this requirement, e.g. by including flexible implicit primitives like metaballs [4]. We could also incorporate more boolean operations into our pipeline, which easily supports them using algebraic operations on signed distances, in analogy to the CAD pipeline, to generate complex topologies and geometries with few primitives. The combinatorial problem of determining the best sequence of boolean operations for a given input would be particularly challenging even for clean data [10]. Finally, it may be possible to incorporate our network into *generative* algorithms to create new unseen shapes.

8. Acknowledgement

The authors acknowledge the generous support of Army Research Office grant W911NF1710068, Air Force Office of Scientific Research award FA9550-19-1-031, of National Science Foundation grant IIS-1838071, National Science Foundation Graduate Research Fellowship under Grant No. 1122374, from an Amazon Research Award, from the MIT-IBM Watson AI Laboratory, from the Toyota-CSAIL Joint Research Center, from a gift from Adobe Systems, and from the Skoltech-MIT Next Generation Program.

References

[1] Samaneh Azadi, Matthew Fisher, Vladimir Kim, Zhaowen Wang, Eli Shechtman, and Trevor Darrell. Multi-content gan for few-shot font style transfer. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, volume 11, page 13, 2018. 2, 7

[2] Elena Balashova, Amit Bermano, Vladimir G. Kim, Stephen DiVerdi, Aaron Hertzmann, and Thomas Funkhouser. Learning a stroke-based representation for fonts. *CGF*, 2018. 2

[3] Mikhail Bessmeltsev and Justin Solomon. Vectorization of line drawings via polyvector fields. *ACM Transactions on Graphics (TOG)*, 2019. 8

[4] James F Blinn. A generalization of algebraic surface drawing. *ACM Transactions on Graphics (TOG)*, 1(3):235–256, 1982. 8

[5] Gunilla Borgefors. Distance transformations in arbitrary dimensions. *Computer vision, graphics, and image processing*, 27(3):321–345, 1984. 2

[6] Neill DF Campbell and Jan Kautz. Learning a manifold of fonts. *ACM Transactions on Graphics (TOG)*, 33(4):91, 2014. 2

[7] Angel X. Chang, Thomas Funkhouser, Leonidas Guibas, Pat Hanrahan, Qixing Huang, Zimo Li, Silvio Savarese, Manolis Savva, Shuran Song, Hao Su, Jianxiong Xiao, Li Yi, and Fisher Yu. ShapeNet: An Information-Rich 3D Model Repository. Technical Report arXiv:1512.03012 [cs.GR], Stanford University — Princeton University — Toyota Technological Institute at Chicago, 2015. 2, 8

[8] Christopher B Choy, Danfei Xu, JunYoung Gwak, Kevin Chen, and Silvio Savarese. 3d-r2n2: A unified approach for single and multi-view 3d object reconstruction. In *European conference on computer vision*, pages 628–644. Springer, 2016. 2

[9] Angela Dai, Charles Ruizhongtai Qi, and Matthias Nießner. Shape completion using 3d-encoder-predictor cnns and shape synthesis. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, volume 3, 2017. 2

[10] Tao Du, Jeevana Priya Inala, Yewen Pu, Andrew Spielberg, Adriana Schulz, Daniela Rus, Armando Solar-Lezama, and Wojciech Matusik. Inversecsg: Automatic conversion of 3d models to csg trees. In *SIGGRAPH Asia 2018 Technical Papers*, page 213. ACM, 2018. 8

[11] David S Ebert and F Kenton Musgrave. *Texturing & modeling: a procedural approach*. Morgan Kaufmann, 2003. 3

[12] Haoqiang Fan, Hao Su, and Leonidas J Guibas. A point set generation network for 3d object reconstruction from a single image. In *CVPR*, volume 2, page 6, 2017. 1, 2

[13] Jorge Fuentes-Pacheco, José Ruiz-Ascencio, and Juan Manuel Rendón-Mancha. Visual simultaneous localization and mapping: a survey. *Artificial Intelligence Review*, 43(1):55–81, 2015. 2

[14] Vignesh Ganapathi-Subramanian, Olga Diamanti, Soeren Pirk, Chengcheng Tang, Matthias Niessner, and Leonidas Guibas. Parsing geometry using structure-aware shape templates. In *2018 International Conference on 3D Vision (3DV)*, pages 672–681. IEEE, 2018. 2

[15] Jun Gao, Chengcheng Tang, Vignesh Ganapathi-Subramanian, Jiahui Huang, Hao Su, and Leonidas J Guibas. Deepspline: Data-driven reconstruction of parametric curves and surfaces. *arXiv preprint arXiv:1901.03781*, 2019. 2

[16] Kyle Genova, Forrester Cole, Daniel Vlasic, Aaron Sarna, William T. Freeman, and Thomas Funkhouser. Learning shape templates with structured implicit functions. In *Proc. IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019. 2

[17] Thibault Groueix, Matthew Fisher, Vladimir G. Kim, Bryan Russell, and Mathieu Aubry. AtlasNet: A Papier-Mâché Approach to Learning 3D Surface Generation. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2018. 1, 2, 5, 6

[18] Kaiming He, Xiangyu Zhang, Shaoqing Ren, and Jian Sun. Deep residual learning for image recognition. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 770–778, 2016. 4

[19] Phillip Isola, Jun-Yan Zhu, Tinghui Zhou, and Alexei A Efros. Image-to-image translation with conditional adversarial networks. *CVPR*, 2017. 1

[20] Vladimir G Kim, Wilmot Li, Niloy J Mitra, Siddhartha Chaudhuri, Stephen DiVerdi, and Thomas Funkhouser. Learning part-based templates from large collections of 3d shapes. *ACM Transactions on Graphics (TOG)*, 32(4):70, 2013. 2

[21] Diederik Kingma and Jimmy Ba. Adam: A method for stochastic optimization. *International Conference on Learning Representations*, 12 2014. 4

[22] Alex Krizhevsky, Ilya Sutskever, and Geoffrey E Hinton. Imagenet classification with deep convolutional neural networks. In *Advances in neural information processing systems*, pages 1097–1105, 2012. 1

[23] Yiyi Liao, Simon Donné, and Andreas Geiger. Deep marching cubes: Learning explicit surface representations. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 2916–2925, 2018. 2

[24] Xia Liu and Kikuo Fujimura. Hand gesture recognition using depth data. In *Proc. 6th IEEE Int. Conf. Automatic Face Gesture Recog.*, page 529. IEEE, 2004. 1, 2

[25] Jonathan Long, Evan Shelhamer, and Trevor Darrell. Fully convolutional networks for semantic segmentation. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 3431–3440, 2015. 1

[26] Kaichun Mo, Paul Guerrero, Li Yi, Hao Su, Peter Wonka, Niloy Mitra, and Leonidas Guibas. Structurenet: Hierarchical graph networks for 3d shape generation. *ACM Transactions on Graphics (TOG), Siggraph Asia 2019*, 38(6):Article 242, 2019. 2

[27] Chengjie Niu, Jun Li, and Kai Xu. Im2struct: Recovering 3d shape structure from a single rgb image. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*, pages 4521–4529, 2018. 2

[28] Peter O'Donovan, Jānis Lībeks, Aseem Agarwala, and Aaron Hertzmann. Exploratory font selection using crowdsourced attributes. *ACM Transactions on Graphics (TOG)*, 33(4):92, 2014. 2

[29] Maks Ovsjanikov, Wilmot Li, Leonidas Guibas, and Niloy J Mitra. Exploration of continuous variability in collections of 3d shapes. In *ACM Transactions on Graphics (TOG)*, volume 30, page 33. ACM, 2011. 2

[30] Jeong Joon Park, Peter Florence, Julian Straub, Richard Newcombe, and Steven Lovegrove. Deepsdf: Learning continuous signed distance functions for shape representation. In *The IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, June 2019. 2

[31] Despoina Paschalidou, Ali Osman Ulusoy, and Andreas Geiger. Superquadrics revisited: Learning 3d shape parsing beyond cuboids. In *Proceedings IEEE Conf. on Computer Vision and Pattern Recognition (CVPR)*, 2019. 2, 7

[32] Huy Quoc Phan, Hongbo Fu, and Antoni B Chan. Flexyfont: Learning transferring rules for flexible typeface synthesis. In *Computer Graphics Forum*, volume 34, pages 245–256. Wiley Online Library, 2015. 2

[33] Zheng Qin, Michael D McCool, and Craig S Kaplan. Real-time texture-mapped vector glyphs. In *Proceedings of the 2006 symposium on Interactive 3D graphics and games*, pages 125–132. ACM, 2006. 4

[34] Adriana Schulz, Ariel Shamir, Ilya Baran, David IW Levin, Pitchaya Sitthi-Amorn, and Wojciech Matusik. Retrieval on parametric shape collections. *ACM Transactions on Graphics (TOG)*, 36(1):11, 2017. 2

[35] Steven M Seitz, Brian Curless, James Diebel, Daniel Scharstein, and Richard Szeliski. A comparison and evaluation of multi-view stereo reconstruction algorithms. In *CVPR*, 2006. 2

[36] Chao-Hui Shen, Hongbo Fu, Kang Chen, and Shi-Min Hu. Structure recovery by part assembly. *ACM Transactions on Graphics (TOG)*, 31(6):180, 2012. 2

[37] David Stutz and Andreas Geiger. Learning 3d shape completion under weak supervision. *International Journal of Computer Vision*, pages 1–20, 2018. 2

[38] Hang Su, Subhransu Maji, Evangelos Kalogerakis, and Erik Learned-Miller. Multi-view convolutional neural networks for 3d shape recognition. In *Proceedings of the IEEE international conference on computer vision*, pages 945–953, 2015. 2

[39] Chunyu Sun, Qianfang Zou, Xin Tong, and Yang Liu. Learning adaptive hierarchical cuboid abstractions of 3d shape collections. *ACM Transactions on Graphics (SIGGRAPH Asia)*, 38(6), 2019. 2

[40] Rapee Suveeranont and Takeo Igarashi. Example-based automatic font generation. In *International Symposium on Smart Graphics*, pages 127–138. Springer, 2010. 2

[41] Shubham Tulsiani, Hao Su, Leonidas J Guibas, Alexei A Efros, and Jitendra Malik. Learning shape abstractions by assembling volumetric primitives. In *Proc. CVPR*, volume 2, 2017. 1, 2, 8

[42] Nobuyuki Umetani, Takeo Igarashi, and Niloy J Mitra. Guided exploration of physically valid shapes for furniture design. *ACM Trans. Graph.*, 31(4):86–1, 2012. 2

[43] Paul Upchurch, Noah Snavely, and Kavita Bala. From a to z: supervised transfer of style and content using deep neural network generators. *arXiv preprint arXiv:1603.02003*, 2016. 2

[44] Jun-Yan Zhu, Richard Zhang, Deepak Pathak, Trevor Darrell, Alexei A Efros, Oliver Wang, and Eli Shechtman. Toward multimodal image-to-image translation. In *Advances in Neural Information Processing Systems*, pages 465–476, 2017. 4