Neural Image Abstraction Using Long Smoothing B-Splines

DANIEL BERIO, Goldsmiths, University of London, United Kingdom
MICHAEL STROH, University of Konstanz, Germany
SYLVAIN CALINON, Idiap Research Institute, Switzerland
FREDERIC FOL LEYMARIE, Goldsmiths, University of London, United Kingdom
OLIVER DEUSSEN, University of Konstanz, Germany
ARIEL SHAMIR, Reichman University, Israel

Fig. 1. Our method allows to optimize long and smooth B-Spline curves using DiffVG rasterization pipelines. Applications range from text stylization to image abstraction and vectorization.

We integrate smoothing B-splines into a standard differentiable vector graphics (DiffVG) pipeline through linear mapping, and show how this can be used to generate smooth and arbitrarily long paths within image-based deep learning systems. We take advantage of derivative-based smoothing costs for parametric control of fidelity vs. simplicity tradeoffs, while also enabling stylization control in geometric and image spaces. The proposed pipeline is compatible with recent vector graphics generation and vectorization methods. We demonstrate the versatility of our approach with four applications aimed at the generation of stylized vector graphics: stylized space-filling path generation, stroke-based image abstraction, closed-area image abstraction, and stylized text generation.

CCS Concepts: • **Computing methodologies** → **Non-photorealistic rendering**; *Rasterization*; **Parametric curve and surface models**; Neural networks; • **Applied computing** → *Fine arts*.

Additional Key Words and Phrases: Differentiable vector graphics, B-splines, Diffusion, CLIP, Long strokes

ACM Reference Format:
Daniel Berio, Michael Stroh, Sylvain Calinon, Frederic Fol Leymarie, Oliver Deussen, and Ariel Shamir. 2025. Neural Image Abstraction Using Long Smoothing B-Splines. *ACM Trans. Graph.* 44, 6, Article 225 (December 2025), 12 pages. https://doi.org/10.1145/3763345

Authors' Contact Information: Daniel Berio, Goldsmiths, University of London, London, United Kingdom, daniel.berio@gold.ac.uk; Michael Stroh, University of Konstanz, Konstanz, Germany, michael.stroh@uni-konstanz.de; Sylvain Calinon, Idiap Research Institute, Martigny, Switzerland, sylvain.calinon@idiap.ch; Frederic Fol Leymarie, Goldsmiths, University of London, London, United Kingdom, ffl@gold.ac.uk; Oliver Deussen, University of Konstanz, Konstanz, Germany, oliver.deussen@uni-konstanz.de; Ariel Shamir, Reichman University, Herzliya, Israel, arik@runi.ac.il.

ACM 1557-7368/2025/12-ART225
https://doi.org/10.1145/3763345

1 Introduction

The ability to produce long, smooth curves is central to a variety of design and artistic tasks. These include freehand drawing, sketching, calligraphy, typography, logo design as well as image abstractions into compositions of organic, flowing, or blobby shapes. Our aim is to enable the generation of vector graphic outputs that allow these types of designs, while taking advantage of recent advances in gradient-based image generation, stylization, and understanding.

Developments in differentiable vector graphics (DiffVG) rasterization have enabled gradient-based optimization methods that leverage complex image-space losses to drive image generation, stylization and abstraction methods. Most existing approaches rely on the method of Li et al. [2020], which implements differentiable rasterization for a large subset of elements of the Scalable Vector Graphics (SVG) standard, including piecewise cubic and quadratic Bézier curves. Most of these methods directly optimize Bézier curves, but even with additional smoothing penalties they do not provide guarantees of continuity across segments, which limits their ability to represent long, smooth and expressive strokes.

Our work is based on two observations. First, alternative spline parametrizations such as B-spline [De Boor 2001] or Catmull-Rom [DeRose and Barsky 1988] provide inherent continuity constraints in their definition. Second, the conversion of such curves to Bézier curves is a linear transformation, making their integration into existing DiffVG pipelines a matter of an additional matrix multiplication. Although these curve parameterizations are well established, to the

best of our knowledge, their integration into DiffVG remains largely unexplored.

In our work we focus on uniform B-splines for their simplicity, high-order continuity and analytic properties [Farin 2001]. This enables a straightforward implementation of derivative-based smoothing criteria that are well known in the fairing and motor-control/robotics domains, but most importantly support our goals of generating long stylized curves within a DiffVG pipeline.

We define B-splines with control-polygons consisting of series of "key-points" and convert these to piecewise cubic Bézier curves for rendering. Since this transformation is linear and rendering is differentiable, gradients from image-space losses can be back-propagated to the key-points. We treat stroke width as a third curve dimension, where each curve control point can be assigned an independent stroke radius, enabling smooth variations similar to that seen in physical brush strokes [Fujioka and Kano 2007]. Allowing the stroke width to vanish also presents an effective way to alter the number of visible strokes required for an image abstraction. Our method operates with both open and closed curves, supporting the generation of closed and organic-looking areas.

We present four different applications for our method: abstract space filling curves (Section 4.1), sketch-based stylization (Section 4.2), abstract image vectorization with color quantization (Section 4.3) as well as text stylization and calligram generation with a novel legibility cost (Section 4.4). We provide a practical implementation of smoothing B-splines that can be directly integrated into DiffVG pipelines and demonstrate how this enables long and expressive curves while maintaining flexible geometric and stylistic control. Working code and examples for our method are available at *github.com/colormotor/calligraph*.

2 Related work

2.1 Smooth and stylized curve generation

Long and stylized strokes have been explored used in the literature for applications including image stylization [Kaplan and Bosch 2005; Tong et al. 2025; Wong and Takahashi 2011], text-based stylization [Maharik et al. 2011], and fabrication [Liu et al. 2017; Yang et al. 2021]. To widen and enhance such applications, our method also enables the generation of long and smooth strokes through the use of neural-driven image-based costs. Smoothing is achieved by minimizing the squared magnitude of higher-order positional derivatives.

In the motor control literature, it is well established that the kinematics of hand and arm movements can be modeled by optimizing performance criteria [Flash and Hogan 1998]. The so-called minimum square derivative models have been successfully applied to handwriting and curved motion by minimizing third-order derivatives (jerk) [Flash and Hogan 1985] and fourth-order derivatives (snap) [Edelman and Flash 1987]. Similar minimum principles are widely employed for smooth motion control in drones [Mellinger and Kumar 2011; Ren and Kry 2019] and robots [Todorov 2004; Toussaint 2017], as well as in statistics for smoothing noisy data [Eilers and Marx 1996; Reinsch 1967].

Similar principles of smooth motion and continuity have also been used for curve fairing, where a "fair" curve is typically one that exhibits a smooth variation of the curvature [Farin 2001]. In this context, jerk has been adopted as an approximation for curvature variation [Lu 2015; Meier and Nowacki 1987; Pottmann 1990], while snap serves as an approximation for transverse distributed load [Meier and Nowacki 1987].

In an extensive body of work, Egerstedt and Martin [2009] develop "dynamic splines" that formulate polynomial splines through optimal control of linear systems. Berio et al. [2017] use similar principles for the interactive generation of stylized paths similar to the ones seen in graffiti art and calligraphy with applications similar to ours. Kano et al. [2003] study the relations between dynamic splines and B-splines and in a collection of work, they develop an optimal formulation of B-splines [Kano et al. 2005] applied to generate motion paths and curves similar to those found in Japanese calligraphy [Fujioka et al. 2006; Matsukida and Fujioka 2013]. Our approach is strongly inspired by the B-spline construction initially proposed by Kano et al. [2005], but we extend their formulation to support DiffVG and demonstrate its flexibility for generative and stylization settings.

2.2 DiffVG and applications

In recent years, differentiable rendering has enabled the use of large pretrained vision and generative imaging models with 3D [Kato et al. 2020; Tewari et al. 2020; Worchel and Alexa 2023] and 2D [Li et al. 2020; Mihai and Hare 2021; Worchel and Alexa 2023] parametric primitives . We adopt the method of Li et al. [2020], which supports a large subset of the SVG standard and cubic curves with varying width profiles. Our method leverages DiffVG's support for cubics with varying width profiles, a feature yet to be used comprehensively, likely due to limited support in mainstream vector graphics tools and standards.

CLIP-driven graphics. One of the first applications of DiffVG to large-pretrained models has been through the use of the Contrastive Language–Image Pretraining (CLIP) model [Radford et al. 2021], a multimodal model that has been trained to share an embedding space between images and their textual descriptions. Frans et al. [2022] demonstrate that together with DiffVG, the model is able to generate vector images guided by a text caption or "prompt". Ganz and Elad [2024] use an adversarial "robustification" method to fine-tune CLIP in order to enable gradients that are better aligned with human perception. Vinker et al. [2022] introduce the idea of using a loss on internal layers of CLIP to guide vector image abstraction. A similar approach, combined with DiffVG, has enabled the generation of stroke-based stylization methods [Schaldenbrand et al. 2023; Vinker et al. 2023; Xing et al. 2023]. Our method provides similar capacities, but we take advantage of the fine-tuned CLIPAG model of Ganz and Elad [2024] and support long smooth strokes, which was not possible with previous methods.

Diffusion-driven graphics. In the context of 3D asset generation, Poole et al. [2023] pioneered the so-called Score Distillation Sampling (SDS), which enables gradient propagation from pre-trained diffusion models to parametric representations. While effective, the original method relies on high classifier-free guidance (CFG) scales, often resulting in over-saturation and lack of detail [Katzir

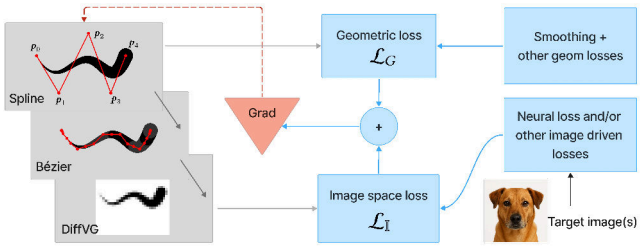


Fig. 2. Flow chart of our pipeline, all operations are differentiable.

Fig. 3. Optimization procedure. From left to right: input image with an initial spline (quintic with multiplicity 3 on all keypoints) and subsequent optimization steps 10, 150, 300.

et al. 2024]. Recent methods, including variational methods [Wang et al. 2023], DDIM inversion [Liang et al. 2024] and noise-free score distillation (NFSD) [Katzir et al. 2024], address these challenges, improving fidelity and control. Our method is compatible with all these techniques, and we specifically adopt the approach of Liang et al. [2024], which proved to be the most effective for us.

In the context of 2D asset generation, Jain et al. [2023b] pioneered the use of SDS in conjunction with the DiffVG method of Li et al. [2020], demonstrating the expressive potential of diffusion for vector graphics generation. Iluz et al. [2023] use SDS for stylizing vector font outlines to resemble user-defined semantics. To name a few, variants of SDS have been used for prompt-based sketch generation [Xing et al. 2023] and animation [Gal et al. 2024], 2D vector graphics [Xing et al. 2024; Zhang et al. 2024] as well as 3D line art [Qu et al. 2024; Tojo et al. 2024]. None of these methods support the creation of long, smooth strokes aligned with our objectives, except for Tojo et al. [2024], who also use B-splines to produce long single-stroke outputs. We incorporate their proposed repulsion loss in our method. However, their work does not cover high-order derivative smoothing, relies on curve discretization, uses a custom CUDA renderer, and does not support variable-width strokes.

3 Method

Our approach works as follows: we specify one or more B-splines through a series of 2D or 3D *keypoints*, where the third dimension can be used to describe width variation along a stroke. The B-splines are converted to cubic piecewise Bézier curves that are then rendered in a differentiable manner with the method of Li et al. [2020] (see also Figure 2). Similarly to conventional DiffVG pipelines, this enables gradient optimization of the key-points with costs that depend on curve geometry as well as on the rendered version of the curves. Figure 3 shows the process: first, some initial points and an image are given; then, during optimization a spline gradually represents the input image more explicitly, while the stroke widths are jointly adapted.

3.1 Uniform B-splines

We use normalized uniform or "cardinal" B-splines that have uniformly spaced integer knots [De Boor 2001], which simplifies computations and proves successful in our applications. A B-spline of degree p and order $k = p + 1$ is a linear combination

$$\boldsymbol{x}(u) = \sum_{i=0}^{n-1} \mathbf{c}_i N_k(u - t_i)$$

where n control-points $\boldsymbol{C} = [\mathbf{c}_0, \mathbf{c}_1, \dots \mathbf{c}_{n-1}]$ and shifted bases N_k are associated with a non-decreasing sequence of $m = n + k$ knots. In our formulation we keep these fixed to

$$\boldsymbol{t} = [t_0, \dots, t_{m-1}] = [-p, \dots, \underbrace{0, \dots, n-k}_{t_p, \dots, t_{m-k}}, \dots n].$$

The spline is defined by sampling u in the interval $[t_{k-1}, t_{m-k}]$. Increasing order derivatives $\boldsymbol{x}^{(d)}$ of a B-spline are easily computed as weighted combinations of lower order B-splines. We refer the reader to the supplement for details on the basis functions construction, but these are readily available in many modern scientific computation packages [Virtanen et al. 2020]. The number of curves and control points is predefined, so that basis functions and knot sequences can be precomputed and remain fixed during optimization.

3.2 Spline construction

B-splines are approximating curves, and both periodicity and clamping to endpoints require the repetition of either knots or control points. This is typically achieved with repeated knots, but we follow Fujioka et al. [2006] and use repeated control points. This maintains strict uniformity while enabling adaptive smoothing of corner-like features and simplifying integral computations, which is advantageous for our use-case. Instead of directly specifying control points, we let a user initially specify a spline through a series of M *key-points* $\boldsymbol{Q} = \boldsymbol{q}_1, \dots, \boldsymbol{q}_M$ and *optimize these rather than the spline control points directly*. The key-points are automatically adapted into a series of control points $\boldsymbol{C}$ depending on the curve's desired clamped or periodic behavior.

For a clamped (open) spline the control points are given by the key-points $\boldsymbol{Q}$ padded the first and last key-point repeated $k-1$ times. This results in a parametric motion that begins and ends with a rest. For periodic closed splines we construct $\boldsymbol{C}$ by appending the first $k-1$ keypoints to the initially specified key-point sequence $\boldsymbol{Q}$.

Key-points may optionally be repeated to create sharp corners, as each repetition initially reduces the continuity of the curve by one degree [Farin 2001]. This strategy is useful to produce additional degrees of freedom for the subsequent optimization, where the corners can be adaptively smoothed depending on the desired amount of smoothing.

3.3 Smoothing B-splines

B-splines of order k are by definition C^{k-2}-continuous, but more importantly their construction facilitates the formulation of smoothing criteria since they allow closed form computation of derivatives and

integrals. In our method, we adopt a smoothing cost based on the squared magnitude of the curve derivatives, which is standard in the smoothing literature and is also known for its utility in curve fairing [Pottmann 1990] and for modeling human arm movements [Todorov and Jordan 1998]. These methods typically trade off smoothness with a geometric accuracy term, but in our work we consider a variety of image-space objectives instead of geometry and define a smoothing cost:

$$\mathcal{L}^d_{\text{smooth}} = \frac{1}{T}\int_{t_{k-1}}^{t_{m-k}} \|\boldsymbol{x}^{(d)}(u)\|^2 \mathrm{d}u = \frac{1}{T}\boldsymbol{c}^\top \bar{\boldsymbol{G}}\boldsymbol{c} \tag{1}$$

where $T = t_{m-k} - t_{k-1}$ and $\boldsymbol{c}$ is a vector that concatenates all control points of the spline. The integral can be calculated exactly by setting $\bar{\boldsymbol{G}}$ to a block Gram-matrix constructed from the inner products of the basis function derivatives [Fujioka and Kano 2007; Vermeulen et al. 1992], resulting in the standard spline smoothing criterion. Alternatively, a finite difference approximation of $\bar{\boldsymbol{G}}$ results in the penalized-spline method of Eilers and Marx [1996]. Both methods have similar run-time performance because the matrix is precomputed for each stroke, and we refer the reader to the supplement for derivations. Most of our examples use quintic splines with a smoothing cost $\mathcal{L}^3_{\text{smooth}}$ on the third positional derivative (jerk). We do so on the basis that "minimum jerk" is a known criterion that has been used to model hand and arm movements [Flash and Hogan 1985; Todorov and Jordan 1998] as well as an approximant for curvature variation in curve fairing [Lu 2015]. Nevertheless, our method generalizes to different curve and smoothing orders (Fig. 4).

3.4 Conversion to Bézier and rendering

Our goal is to integrate smoothing B-splines into a DiffVG pipeline by taking advantage of the linear relationship between B-splines and Bézier curves. B-splines can be converted exactly to piecewise Bézier curves of the same degree. To do so we use the method of Romani and Sabin [2004], which reduces to a matrix multiplication between the flattened spline control points $\boldsymbol{c}$ and a block transformation matrix $\boldsymbol{S}$.

Our method also supports smoothing costs on higher-order positional derivatives such as jerk (third derivative) and snap (fourth derivative), which require polynomial curves of degree greater than three. Although native rendering of such higher-degree curves is not supported in DiffVG and remains a challenge, we observe that reducing the degree of B-splines to three introduces negligible geometric error (less than 0.3% of the curve's bounding box diagonal in all our experiments), making the optimization of higher-degree B-splines practical for image-based error calculations.

We perform a degree reduction of Bézier curves using the multi-reduction method of Sunwoo [2005], which involves a second block transformation matrix $\bar{\boldsymbol{R}}$. As a result, the control points for a cubic piecewise Bézier curve compatible with DiffVG are computed from the (flattened) control points $\boldsymbol{c}$ with the linear map $\bar{\boldsymbol{R}}\bar{\boldsymbol{S}}\boldsymbol{c}$. We refer the reader to the work of Romani and Sabin [2004] and Sunwoo [2005] for details; we include in the supplement details and matrices for quintic Bézier and their reduction to cubic.

DiffVG rendering and optimization. The conversion procedure results in a sequence of Bézier control points $\in \mathbb{R}^3$, where the third dimension represents the stroke radius. Control points and associated stroke and fill colors are all treated as differentiable parameters to be optimized. Rendering the scene results in an image $\mathbb{I}$, which is differentiable with respect to all the underlying parameters.

4 Applications

The proposed B-spline construction, smoothing and conversion to Bézier enables the optimization of long, expressive and optionally periodic curves, which would be challenging to produce with currently known methods leveraging DiffVG. All the results presented hereafter are produced using a combined cost:

$$\mathcal{L} = \mathcal{L}_{\mathbb{I}} + \mathcal{L}_{\text{G}} \tag{2}$$

consisting of an image-space term, $\mathcal{L}_{\mathbb{I}}$, and a geometric term, $\mathcal{L}_{\text{G}}$. We construct each term as a combination of losses depending on the application objective. $\mathcal{L}_{\mathbb{I}}$ relies on differentiable rasterization, which allows gradients to propagate from raster-based objectives to the geometry parameters. $\mathcal{L}_{\text{G}}$ leverages the properties of B-splines to enable smoothing, stylization objectives, and constraints while preserving continuity. We denote the relative weights of any loss $\mathcal{L}_\circ$ as $\lambda_\circ$, e.g. the weight of a smoothing loss on the third derivative is denoted as λ_{smooth}. If not specified, the weights are assumed to be 1. When also optimizing stroke widths, we clip these to a minimum and maximum value at each iteration.

We generate strokes using the Adam optimizer and use a cosine annealing schedule on the learning rates. We run our experiments on a single NVIDIA GeForce RTX 3060 with 12 Gb of memory. We run most of the presented applications for 300 steps, which approximately takes between 30 and 60 seconds on our system. One exception is using diffusion-guidance, which takes approximately 0.6 to 1.0 second per step depending on the method used, leading to an optimization time of up to 6 minutes.

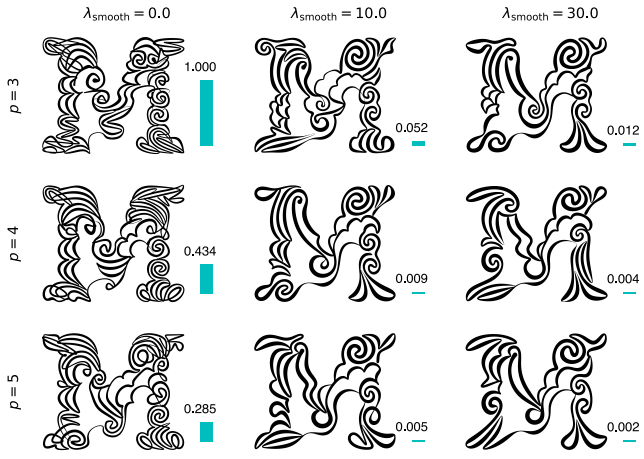


Fig. 4. Comparison of different spline degrees p (rows), smoothing derivative orders d and smoothing weight λ_{smooth} (columns). In each row, we let the smooting derivative to $p-1$. We quantify smoothness using the dimensionless jerk measure [Hogan and Sternad 2009]. Lower is smoother. We use the stylized area fill method in Section 4.1 using the style image in Fig. 6, left.

Fig. 5. Text combining areas generated with our stylized area filling method. Each letter is generated separately.

Fig. 6. Examples of stylized area filling for a letter "S". The images on the lower left are used to guide stylization.

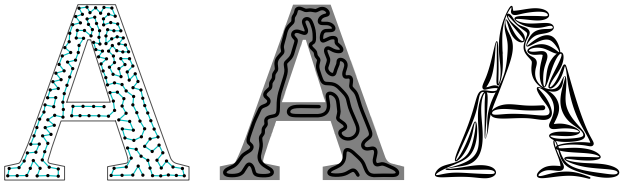


Fig. 7. Left, weighted Voronoi samples (black) for a bitmap of the letter 'A' and an open TSP path connecting the points (blue). Middle, initial quintic B-spline with key-points given by the Voronoi samples overlaid on the bitmap area with 50% opacity. Right, result of an optimization with $\mathcal{L}_G = \mathcal{L}^3_{\text{smooth}}$ and $\mathcal{L}_{\mathbb{I}}$ for the 50% opacity bitmap. Decreasing opacity results in sparser and thinner strokes.

4.1 Area fillings and pattern generation

As a baseline for our method we demonstrate how our pipeline can be used to create pattern fills of solid regions. It illustrates also how our approach can be flexibly used to control stylization while maintaining smoothness (Fig. 5 and 6).

Initialization. Stochastic gradient descent is well known to be sensitive to initialization due to its susceptibility to local minima. We find good points using an initialization strategy based on so-called weighted Voronoi stippling [Secord 2002]. For simplicity, we adopt this method for different applications presented in this paper. The input can be an arbitrary bitmap (Fig. 7, Left) or a saliency map (Fig. 13). To create a single stroke, we use a TSP route connecting the points in an open or looping path. This method is known in the literature as "TSP art" [Kaplan and Bosch 2005] (Fig. 7, Left). For open paths, we select the left-topmost point and the bottom-rightmost as initial and final points, respectively.

Image coverage loss. We find that setting $\mathcal{L}_{\mathbb{I}}$ as a multiscale mean squared error (MSE) loss works particularly well to fill an area or silhouette defined as an image. This loss is computed between the target and the rendered image, with each step corresponding to a progressively reduced scale and blurred version of the image. This approach is similar to the shape-based losses used by Iluz et al. [2023] and Tojo et al. [2024], but lower scales encourage alignment with broader intensity regions and faster convergence, while higher scales promote a more accurate silhouette reconstruction. Reducing the opacity of the target image directly decreases the density of curves used to cover it (Fig. 7-right), allowing control over the visual result.

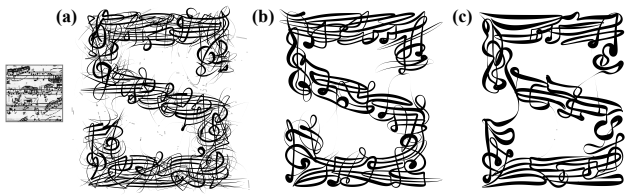


Fig. 8. Stylized coverage of a letter "S" using the image on the left as a target. **(a)** Optimization using Bézier curves. **(b)** Optimization using quintic splines and smoothing on jerk with $\lambda_{\text{smooth}} = 1$. **(c)**. Same procedure with $\lambda_{\text{smooth}} = 10$.

Fig. 9. Examples combining image coverage with a patch-wise loss on CLIP features derived from an example image (top left) and using the same initialization from Fig. 7. From the left, the first two examples use a quintic spline with a smoothing loss $\mathcal{L}_G = \mathcal{L}^3_{\text{smooth}}$ (jerk). For comparison, the right example uses a Catmull-Rom spline only enforcing C^1 continuity. Allowing zero stroke width results in the appearance of multiple strokes, but the optimization is still performed on a single curve (left, dotted cyan).

Bounding box loss. For some of our optimization procedures, it is useful to extend $\mathcal{L}_G$ with a bounding box loss that keeps curve key-points within the bounding box of a given image:

$$\mathcal{L}_{\text{box}} = \sum_i \mathbf{1}^\top \left[\varphi\left(\boldsymbol{b}_{\min} - \boldsymbol{p}_i\right) + \varphi\left(\boldsymbol{p}_i - \boldsymbol{b}_{\max}\right) \right]$$

where $\mathcal{L}_{\text{box}} > 0$ only if key-points fall outside of the bounding box $\boldsymbol{b}_{\min}, \boldsymbol{b}_{\max}$ and where φ can be either a Softplus or a ReLU function applied element-wise to the vectors.

Image-space semantic-driven stylization. Together with geometry-based stylization costs, we can add a semantic stylization term $\mathcal{L}_{\text{style}}$ to the image-space loss $\mathcal{L}_{\mathbb{I}}$, which enables stylization based on a text prompt or features extracted from an example image (Fig. 8 , 6 and 9). We apply the technique proposed by Kwon and Ye [2022] for semantic-driven image stylization and use a patch-wise directional loss between the encoded features of an example image and the encoded features of the rendered curves. We use the augmented CLIPAG [Ganz and Elad 2024] ViT-B/32 transformer architecture as we find it to be efficient while working well for our use vector stylization use-case.

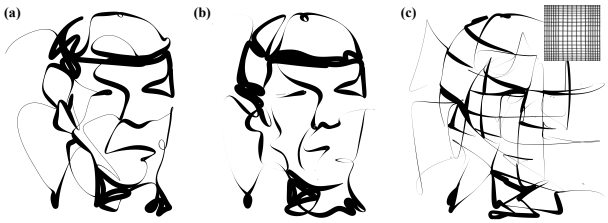


Fig. 10. Diffusion driven stroke abstraction with ControlNet and IP-adapter conditioning. **(a)** a variable width abstraction of "Spock". **(b)** Allowing a single strokes to reach zero width in regions results in an effective strategy for automatically determining the number of strokes for a multi-stroke abstraction. **(c)** combining the diffusion cost with a stylization term that favors horizontal and vertical orientations.

Fig. 11. Rendering stroke abstractions with a spray-like brush. The quintic B-splines together with smoothing on jerk produce smooth motions that tend to slow down where curvature is higher. This is a characteristic feature of human hand motions [Viviani, Paolo and Flash, Tamar 1995] and results in a lower deposition of paint particles where speed is higher.

4.2 Single-stroke image abstraction

Most existing DiffVG-based methods that work with diffusion models rely on variants of Score Distillation Sampling (SDS) together with a text caption to guide the generation of parametric vector primitives. We follow a similar approach but enable image-conditioned stylization by integrating ControlNet [Zhang et al. 2023] with Canny edge detection and IP-Adapter [Ye et al. 2023] into the diffusion pipeline. ControlNet helps to preserve structural cues from the input image, while IP-Adapter encourages the strokes to align with its global appearance and style (Fig. 10 and 11).

In our experiments, we find that using a generic text prompt such as "A black and white drawing" for stroke-based outputs is sufficient to generate recognizable abstractions and stylizations of an input image. For a given condition y, the gradient of the SDS-like loss with respect to the optimizated parameters θ has the form:

$$\nabla_\theta \mathcal{L}_{\text{SDS}} = \mathbb{E}_t \left[\omega(t) \left(\epsilon_\phi(x_t, t, y) - \epsilon \right) \frac{\partial g(\theta)}{\partial \theta} \right], \tag{3}$$

where $\epsilon_\phi(x_t, t, y)$ is the predicted denoising direction for a latent x_t at time step t, ϵ is the noise predicted by the model and $\omega(t)$ is a weighting function dependent on the time-step.

We employ the time-step schedule annealing procedure proposed by Liang et al. [2024] and use their Interval Score Matching (ISM) variant of SDS, which helps convergence in our experiments and enables a standard classifier-free guidance of 7.5.

It is known that for diffusion models, higher time steps during denoising typically produce coarser features, while lower time steps yield finer details [Hwang et al. 2023]. Given our goal of producing single stroke image abstractions, the curves lack sufficient degrees of freedom to capture these finer details, so we limit the time steps in the denoising process to a minimum of 500 (Fig. 12). With a similar motivation, we find that with diffusion-guided stroke abstraction it is useful to initialize the strokes with a multiplicity > 1 (we use 3 with quintic splines in our examples). Using a higher multiplicity results in smoother strokes, where fewer details are captured.

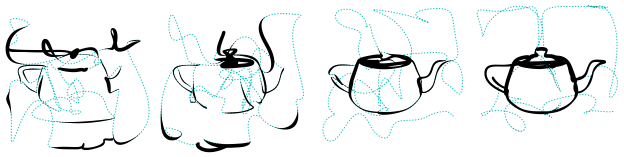


Fig. 12. From left to right: varying the minimum time-step (100, 300, 500, 700) for 300 timesteps of the ISM [Liang et al. 2024] variant of SDS and a single quintic stroke. The cyan line emphasizes the centerline of the stroke, which reaches zero width in certain regions.

Fig. 13. Left, initialization with a saliency map computed from the normalized logits of the last layer of the OneFormer panoptic segmentation model [Jain et al. 2023a]. Right, stroke optimization using ISM with diffusion conditioned on the edge map, using a minimum time step of 400 and with an additional stylization loss λ_{style} guided by the same image as Fig. 8.

4.3 Area-based image abstraction

Our method allows for the generation of smooth closed areas, and we observe that this is useful to generate image abstractions similar to what can be seen in certain designs consisting of overlapping smooth regions and a limited color palette. Examples include psychedelic designs, album covers, screen-printed graphics, or street-art inspired fashion and graphic designs. We are interested in generating outputs that aim to be printed or fabricated as collages with a limited number of regions and colors. To guide stylization, we use filled areas instead of strokes and set $\mathcal{L}_{\mathbb{I}}$ to a variant of the CLIP-driven geometric cost described by Vinker et al. [2022]

$$\mathcal{L}_{\text{CLIP}} = \sum_l \left\| \text{CLIP}_l(\hat{\mathbb{I}}) - \text{CLIP}_l(\mathbb{I}_\theta) \right\|_1, \tag{4}$$

using the L^1 norm instead of L^2 and omitting the semantic term originally proposed by the authors. We use layers 2 and 3 together with the CLIPAG [Ganz and Elad 2024] architecture. We use CLIP as opposed to diffusion because we find this to be significantly faster, while being effective for this kind of stylization task.

Repulsion loss. For applications using closed curves, we adopt the repulsion method for 3D wire fabrication proposed by Tojo

Fig. 14. Quantized color vectorizations using an additional image-driven stylization term $\mathcal{L}_{\text{style}}$. The palette is extracted from the style image.

Fig. 15. Increasing λ_{style} weight for an abstract vectorization of Bach. From left to right $\lambda_{\text{style}} = 0$, $\lambda_{\text{style}} = 0.06$, $\lambda_{\text{style}} = 0.1$

et al. [2024] to compute a geometric loss $\mathcal{L}_{\text{repul}}$, penalizing self-intersections and overlaps based on a tangent-point energy kernel for a set of sampled points along the spline. For this application, we compute the loss for each area separately, thus allowing overlaps and intersections among different areas.

Optimization with quantized coloring. Jang et al. [2017] use the Gumbel-Softmax trick to make discrete choices differentiable during training. We apply the same idea to assign colors to image regions, using soft selections from a fixed color palette that can be optimized with backpropagation. To progressively transit from soft to discrete assignments during the training process, we anneal the Gumbel-Softmax temperature using an exponential schedule.

Given a set of K palette colors organized as a matrix $\boldsymbol{V} \in \mathbb{R}^{K\times 3}$, we optimize the logits per area $\ell_i \in \mathbb{R}^K$ using a soft assignment

$$\boldsymbol{a}_i = \text{softmax}\left(\frac{\ell_i + \mathbf{g}_i}{\tau}\right) \quad \text{with} \quad \boldsymbol{g}_i \sim \text{Gumbel}(0, \beta)^K,$$

where β is a scale parameter that we empirically set to 0.15 to avoid excessive noise during optimization [Huijben et al. 2023] and τ is a temperature parameter that we anneal during optimization. We use these soft colors computed as $\mathbf{v}_i = \boldsymbol{a}_i^\top \boldsymbol{V}$ during the optimization. At the same time, for visualization, we obtain hard assignments by taking the argmax over the optimized logits and selecting the corresponding palette color. To encourage a balanced use of all the specified palette colors, we add a regularization term:

$$\lambda_K \left\| \mathbb{E}_i \left[\boldsymbol{a}_i \right] - K^{-1}\mathbf{1} \right\|^2$$

to $\mathcal{L}_{\mathbb{I}}$, which penalizes deviations from a uniform color assignment, encouraging a balanced use of the palette. Figures 14 and 15 show some results.

Area initialization and optimization. We initialize a user-defined number of areas using weighted Voronoi sampling on a saliency map of the input image and create an initial series of closed curves with keypoints given by the vertices of each resulting Voronoi regions. Each curve is then assigned random initial logit and the curves are sorted by increasing saliency of the covered area. Optimization proceeds with the inclusion of the repulsion loss in $\mathcal{L}_{\text{G}}$, which keeps the area outlines from intersecting.

4.4 Text stylization

In line with the smooth curve image abstractions, we aim to generate text abstractions made of smooth curves that fit inside a target area. Examples of this approach can be seen in posters, graphic designs, as well as in "calligrams": renditions of text that is arranged to fit a specific silhouette, such as those seen in the methods of Xu and Kaplan [2007] and Zou et al. [2016] (c.f. Figure 17). Our pipeline results in a simple way to generate calligrams, such as "blobby" texts (Fig. 16) and abstract monospace fonts (Fig. 18).

We tackle text stylization with the tools we have covered so far and start with a bitmap image $\hat{\mathbb{I}}$ representing the desired silhouette and an initial text layout rendered as a second image $\mathbb{I}_{\text{txt}}$. We uniformly sample the glyph outlines and produce key-point sequences used in optimization. The optimization deforms the outlines based on a loss that balances silhouette coverage, outline smoothness, and repulsion between outline points. This procedure alone smooths and fits the outlines into the target area, but this may compromise legibility (Figure 18c).

To preserve *legibility*, we introduce a perceptual loss based on the features of a pretrained vision encoder, which we use to compute the feature-space distance between the rendered deformed image $\mathbb{I}$ and the original layout $\mathbb{I}_{\text{txt}}$. We find that using the last-layer `[CLS]` token as feature of the TrOCR model [Li et al. 2023] and calculating a loss based on the L^1-norm of the embeddings produce robust results for this application (Figure 18).

The placement of glyph can be manual or automatic. In the automatic case, we optimize a similarity transform per glyph to maximize silhouette coverage while avoiding overlaps and maintaining a readable text layout. We first offset each glyph by a user-specified amount to encourage padding around the text. At each optimization step, we render both a morphologically opened version of the silhouette and glyphs into two images using white with 50% opacity on a black background. We minimize a loss that combines (i) a coverage term $\mathcal{L}_{\mathbb{I}}$ (Section 4.1), (ii) an overlap cost given by $\sum \text{ReLU}(v - 0.5)$ for each pixel intensity $v \in [0, 1]$ of the rendered image and (iii)

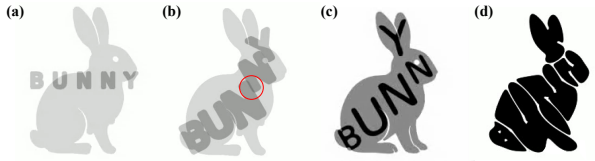


Fig. 16. Automatic calligram production for a silhouette generated with the prompt "Silhouette of a BUNNY". **(a)** initial text layout rendered with 50% opacity and overlayed on the silhouette. **(b)** intermediate step of the layout optimization displaying an image area that increases the overlap cost. **(c)** Sampled glyphs placed according to the layout. **(d)** Result of the optimization.

Fig. 17. Calligram generation: comparison of **(a)** an example from Zou et al. [2016] for a camel silhouette and **(b)** two runs of our method on the same silhouette with automatic initialization and two different fonts.

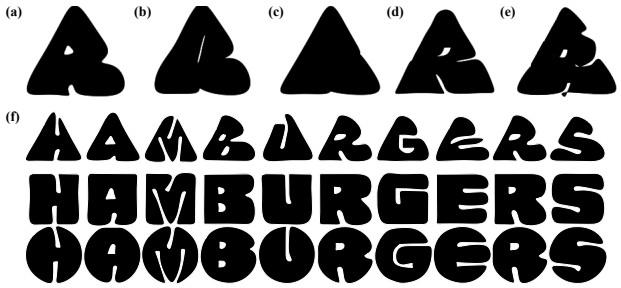


Fig. 18. Monospace font generation. **(a)** A letter "R" (quintic B-splines with jerk cost) adapted to a triangle, using $\lambda_{\text{repul}} = 666$, $\lambda_{\text{txt}} = 6.6$ and $\lambda_{\text{smooth}} = 200.0$. **(b)** Setting $\lambda_{\text{repul}} = 0$ (no repulsion), still results in a readable letter but **(c)** removing the legibility loss does not. (d) B-splines with legibility but no smoothing. **(e)** Catmull-Rom to enforce tangent continuity with legibility loss (for comparison). **(f)** Combining glyphs optimized to fit a triangle, a square and a circle.

Fig. 19. More calligrams generated with our system. The seagull silhouette is generated using the prompt "Silhouette of a SEAGULL".

Fig. 20. Stroke abstraction of Thelonious Monk. Left, using a single stroke and Voronoi with TSP initialzation. Middle, using multiple strokes with multiple key-points along vertical lines. Right, using facial features extracted with MediaPipe [Lugaresi et al. 2019].

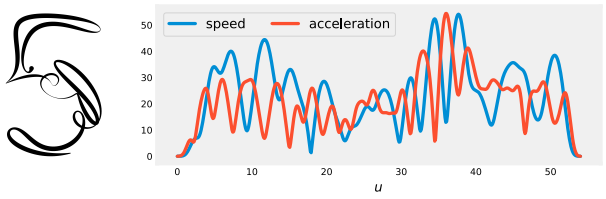


Fig. 21. Smooth speed and acceleration of a quntic spline covering a number "5" and optimized using the method of Section 4.1. With appropriate resampling the path kinematics can be safely tracked with a robot.

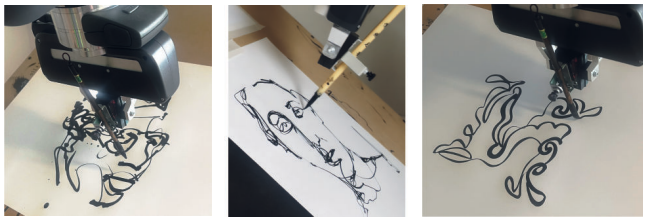


Fig. 22. Our method produces smooth kinematics that facilitate reproduction with a robot, and the varying width can be used to control brush pressure. Left and center, the robot reproducing portraits. Right, the robot reproducing a stylized area fill.

an alignment cost $\sum \|\theta\|$ that penalizes the absolute turning angles θ between consecutive glyph center-points and maintains text ordering. We note that image generation models such as DALL-E 3 [Betker et al. 2023] are particularly effective at generating silhouettes with a prompt, which finally results in a fully automatic calligram generation pipeline.

5 Discussion

In our example applications, we have seen how a B-spline reparametrization can be used to generate long and expressive strokes and curves in a DiffVG pipeline. B-splines enforce high-order continuity by design, which enables analytic smoothing losses that help producing more regular geometry when combined with different stylization losses. This offers a considerable advantage compared to using only Bézier curves or parametrizations with lower order continuity, especially for applications like the ones demonstrated in this paper. Qualitative examples of this can be seen in examples such as Figure 8 and Figure 18. For conciseness, we used a similar Voronoi-based initialization strategy in most of our examples. However, our method performs well with different initializations (Fig. 20), which can serve as an additional design parameter to be explored by users.

One common challenge in stroke-based abstraction pipelines is controlling the trade-off between visual fidelity and geometric simplicity. Previous methods typically address this by pre-determining the number of curves [Vinker et al. 2022] or by integrating a learned component into the optimization loop [Vinker et al. 2023]. We find that our use of smoothing, combined with optimizable stroke width, allows this trade-off to be controlled parametrically and with the number of strokes emerging from the optimization. This results in a solution that is significantly simpler than previous methods.

We investigate the utility of our representations and different loss terms in different examples of our applications. In Figure 18 we perform a small qualitative ablation showing the effectiveness

of the proposed legibility loss (Figure 18c) as well as the benefits of B-splines and smoothing compared to Catmull-Rom splines (Figure 18e), which only enforce C^1 continuity. In Figure 8a we can observe that directly optimizing Bézier curves effectively captures features of the example style image. However, higher degrees of freedom produce results that capture finer details at the expense of a clear stroke structure. Although this additional detail may be desirable in certain applications, it is not suitable for the applications considered in our work.

Interestingly, the computational overhead of the proposed B-spline to Bézier matrix conversion is lower than the one for the additional optimization parameters required for an equivalent multi-Bézier curve. We tested performance with a simple comparison where we cover an area by optimizing the 290 key-points of a single open cubic B-spline. We compared this to a similar setup directly optimizing the corresponding 874 Bézier control points. On our hardware setup, the Bézier case is 4.5 times slower. This shows that the additional cost of the proposed matrix conversion is negligible and suggests that our method is an efficient way to enforce output continuity in DiffVG settings.

Robotic reproduction. Optimizing splines with degree greater than three results in smooth acceleration profiles (Fig. 21). This enables a safe reproduction of the resulting trajectory kinematics with an articulated robot arm (Fig. 22), without requiring an intermediate reparameterization step. We tested this by reproducing the trajectories using a 7-axis Franka robot equipped with a brush. We first transformed the control points to a desired workspace coordinate system, treating the stroke widths as perpendicular distances to the drawing plane. We then sampled the trajectories at a resolution that produced a maximum speed and accelerations within the robot's mechanical limits. The inverse kinematics for the resulting trajectories are then computed with an iterative linear quadratic regulator (iLQR) [Li and Todorov 2004].

6 Conclusions and future work

We have presented a framework for integrating high-order B-splines into DiffVG pipelines together with minimum-square derivative-based smoothing costs. We have explored different applications and demonstrated how this enables the generation of long, smooth, and stylized strokes through a combination of geometric and image-space loss functions. While the combination of losses allows for a large variety of creative outputs, a practical challenge is the necessity to weigh different losses to achieve the desired result, which, given the iterative optimization procedure, can be slow and tedious.

Although our formulation draws on a large body of existing work on B-splines, an effective use of this tool together with DiffVG is novel, and we expect it to be a valuable tool for the community. We used uniform B-splines because of their simplicity and effectiveness for our use cases. However, exploring non-uniform parameterizations, such as NURBs, presents an interesting direction for further research, as it may unlock additional flexibility and control for stylized outputs.

Acknowledgments

This work was funded by the EACVA (Embodied Agents in Contemporary Visual Art) Project, led by Goldsmiths (UKRI/AHRC grant AH/X002241/1) and the University of Konstanz (grant 508324734, Deutsche Forschungsgemeinschaft/DFG). Special thanks to Guillaume Clivaz (Idiap Research Institute) for the technical support and useful discussions.

A B-Spline details

A B-spline (or basis-spline) or order k is a piecewise polynomial curve of degree $p = k-1$ defined by a linear combination of n weights or control points $\mathbf{c}_0, \mathbf{c}_1, \dots \mathbf{c}_{n-1}$ and a non-decrasing sequence of $m = n + k$ knots (or breakpoints) $t_0, t_1, t_2, \dots, t_{m-1}$.

$$\boldsymbol{x}(u) = \sum_{i=0}^{n-1} \mathbf{c}_i B_{i,k}(u)$$

Each basis function $B_{i,k}$ defines k polynomial segments spanning $k+1$ knots $t_i, t_{i+1}, \dots, t_{i+k}$ and is positive in the half-open domain $[t_i, t_i + k)$. The knots between t_p and t_{m-k} (not included) are called "internal" or "interior" knots. From here: For n control points we have $n + k$ knots and $n - k$ interior knots.

B-spline bases can be defined through the "Cox-de Boor" recursion starting from order 1 (degree 0):

$$B_{i,1}(u) = \begin{cases} 1 & \text{if } t_i \le u < t_{i+1} \\ 0 & \text{otherwise} \end{cases}$$

And with

$$B_{i,k}(u) = \frac{u - t_i}{t_{i+k-1} - t_i} B_{i,k-1}(u) + \frac{t_{i+k} - u}{t_{i+k} - t_{i+1}} B_{i+1,k-1}(u)$$

The number of control points n, order k and number of knots m are related by $n + k - m = 0$. For nonrepeating knot sequences, the curve will be C^{k-2} continuously differentiable.

A.1 Derivatives

The derivative of a B-spline basis function of order k is given by

$$\frac{\mathrm{d}}{\mathrm{d}u} B_{i,k}(u) = B'_{i,k}(u) = \frac{k-1}{t_{i+k-1} - t_i} B_{i,k-1}(u) - \frac{k-1}{t_{i+k} - t_{i+1}} B_{i+1,k-1}(u).$$

It is a linear combination of all the derivatives of the basis function. As a result, the derivative of a B-spline is equivalent to a B-spline of order $k-1$ with a new set of control points given by weighted differences of pairs of consecutive control points.

A.2 Cardinal B-splines

A cardinal B-spline (not to be confused with cardinal/Catmull-Rom splines) is a "normalized uniform B-spline". It has uniformly spaced knots, with $t_{i+1} - t_i = h$ (uniform) with $h = 1$ (normalized) so the knots are all integers (Fig. 23). Uniformity and normalization simplify the computations of a B-spline as all basis functions are translated versions of the same basis function that we denote as $N_k(u)$. We then have

$$\boldsymbol{x}(u) = \sum_{i=0}^{n-1} \mathbf{c}_i N_k(u - t_i)$$

and the B-spline derivatives simplify to

$$\frac{\mathrm{d}}{\mathrm{d}u}N_k(u) = N_k'(u) = N_{k-1}(u) - N_{k-1}(u-1)$$

so

$$\dot{\boldsymbol{x}}(u) = \sum_{i=0}^{n-1} \mathbf{c}_i \left(N_{k-1}(u-t_i) - N_{k-1}(u-t_i-1)\right)$$

A.3 Smoothing term

The smoothing term can be computed exactly and is considerably simplified for the case of cardinal B-splines [Schumaker 1981]. While different approaches exist to calculate this kind of integral [de Boor et al. 1976; Vermeulen et al. 1992] to calculate this kind of integral, we follow Fujioka and Kano [2007] and Fujioka et al. [2017] to have

$$\mathcal{L}^d_{\text{smooth}} = \int_{-\infty}^{\infty} D(u)\,\mathrm{d}u - \int_{-\infty}^{t_{k-1}} D(u)\,\mathrm{d}u - \int_{t_n}^{\infty} D(u)\,\mathrm{d}u$$

with $D(u) = \|\boldsymbol{x}^{(d)}(u)\|^2$

This can be computed explicitly by constructing a Gramian $\boldsymbol{G}$ with:

$$G_{i,j} = \begin{cases} \int_0^k N_{i,j}^{(d)}\,\mathrm{d}u - \int_0^{p-i} N_{i,j}^{(d)}\,\mathrm{d}u & \text{if } i < p \text{ and } j < p \\ \int_0^k N_{i,j}^{(d)}\,\mathrm{d}u - \int_0^{p-i} N_{n+p-i,n+p-j}^{(d)}\mathrm{d}u & \text{if } i \geq n \text{ and } j \geq n \\ \int_0^k N_{i,j}^{(d)}\,\mathrm{d}u & \text{otherwise} \end{cases}$$

and

$$N_{i,j}^{(d)} = N_k^{(d)}(u) N_k^{(d)}(u-j+i)$$

.

Then each $G_{i,j}$ can be computed exactly using quadrature [Vermeulen et al. 1992].

If we let $\boldsymbol{c} \in \mathbb{R}^{nD}$ be a vector that concatenates n control points, each of dimensions D we have

$$\mathcal{L}^d_{\text{smooth}} = \boldsymbol{c}^\top \bar{\boldsymbol{G}} \boldsymbol{c}, \quad \bar{\boldsymbol{G}} = \boldsymbol{G} \otimes \boldsymbol{I}_D$$

where $\otimes$ is the Kroenecker product and $\boldsymbol{I}_D$ is the identity matrix of dimensions D.

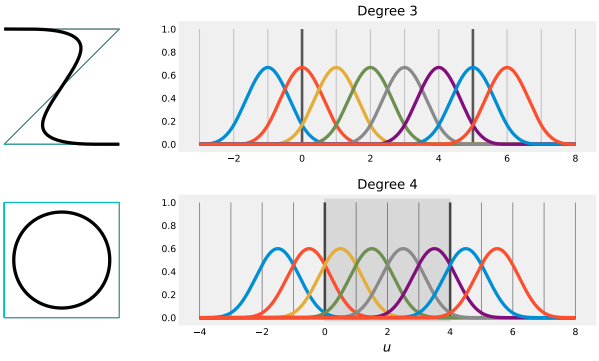
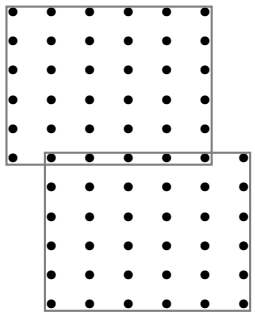


Fig. 23. B-Splines and their bases with degrees 3 and 4.

P-splines. A similar procedure can be efficiently approximated with discretization of the derivative cost, using penalized-splines (P-splines) as described by Eilers and Marx [1996]. To do this, we can simply use $\boldsymbol{G} = \boldsymbol{D}_{(d)}^\top \boldsymbol{D}_{(d)}$ with $\boldsymbol{D}_{(d)}$ a matrix representing the finite difference operator of order d. The advantage of this method is the simplicity of implementation and the possibility of achieving similar smoothing results. We can arbitrarily combine the degree of discrete differences with the degree of the curve. We expose both methods for completeness and to enable applications where the integral cost may be necessary (e.g., planning and robotics).

A.4 Conversion to Bézier

With the method of Romani and Sabin [2004], converting the $p+1$ control points of a quintic B-spline of degree p to single Bézier segment of the same degree, can be done with a $(p+1) \times (p+1)$ matrix that we denote as $\boldsymbol{S}^p$. To convert all the control points of a B-spline we stack multiple shifted and overlapping copies of $\boldsymbol{S}^p$ into a larger matrix $\boldsymbol{S}$, by shifting each copy by p rows and 1 column. For a quintic spline this can be visualized as:



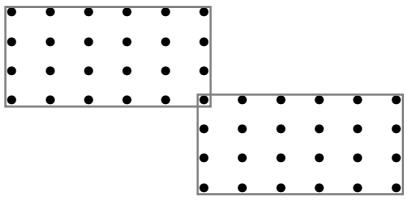
The blocks for a quintic spline are given by:

$$\boldsymbol{S}^5 = \frac{1}{120}\begin{bmatrix} 1 & 26 & 66 & 26 & 1 & 0 \\ 0 & 16 & 66 & 36 & 2 & 0 \\ 0 & 8 & 60 & 48 & 4 & 0 \\ 0 & 4 & 48 & 60 & 8 & 0 \\ 0 & 2 & 36 & 66 & 16 & 0 \\ 0 & 1 & 26 & 66 & 26 & 1 \end{bmatrix}.$$

The block matrix $\bar{\boldsymbol{S}}$ used to compute the Bézier control points from the flattend B-spline control points $\boldsymbol{c}$ is given by the Kroenecker product $\boldsymbol{S} \otimes \boldsymbol{I}_D$.

A.5 Degree reduction

With the method of Sunwoo [2005], reducing a Bézier curve of degree p to one of degree q can be done with a $(q+1) \times (p+1)$ matrix that we denote as $\boldsymbol{R}^{p,q}$. To reduce the degree of all the control points of a Bézier chain we stack multiple shifted and overlapping copies of $\boldsymbol{R}^{p,q}$ into a larger matrix $\boldsymbol{R}$ by shifting each copy by p rows and q columns. For a reduction from quintic to cubic this can be visualized as:



The blocks of the quintic to cubic reduction matrix are given by

$$\boldsymbol{R}^{5,3} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ -\frac{2}{3} & \frac{5}{3} & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & \frac{5}{3} & -\frac{2}{3} \\ 0 & 0 & 0 & 0 & 0 & 1 \end{bmatrix}$$

The block matrix $\bar{\boldsymbol{R}}$ used to compute the reduced Bézier control points from is given by the Kroenecker product $\boldsymbol{R} \otimes \boldsymbol{I}_D$.

References

Daniel Berio, Sylvain Calinon, and Frederic Fol Leymarie. 2017. Generating Calligraphic Trajectories with Model Predictive Control. In Proceedings of Graphics Interface. Canadian Human-Computer Communications Society, Edmonton, Canada.

James Betker, Gabriel Goh, Li Jing, Aditya Ramesh, et al. 2023. Improving Image Generation with Better Captions. Technical Report. OpenAI. https://cdn.openai.com/papers/dall-e-3.pdf

Carl De Boor. 2001. A Practical Guide to Splines (1 ed.). Springer, New York, NY.

Carl de Boor, Tom Lyche, and Larry L Schumaker. 1976. On Calculating with B-splines II. Integration. Numerische Methoden der Approximationstheorie/Numerical Methods of Approximation Theory: Vortragsauszüge der Tagung über numerische Methoden der Approximationstheorie vom 25. bis 31. Mai 1975 im Mathematischen Forschungsinstitut Oberwolfach (Schwarzwald) (1976), 123–146.

Tony D. DeRose and Brian A. Barsky. 1988. Geometric Continuity, Shape Parameters, and Geometric Constructions for Catmull-Rom Splines. ACM Transactions on Graphics 7, 1 (Jan. 1988), 1–41. doi:10.1145/42188.42265

Shimon Edelman and Tamar Flash. 1987. A Model of Handwriting. Biological Cybernetics 57, 1-2 (1987), 25–36.

Magnus Egerstedt and Clyde Martin. 2009. Control Theoretic Splines: Optimal Control, Statistics, and Path Planning. Princeton University Press, Princeton. doi:10.1515/9781400833870

Paul H. C. Eilers and Brian D. Marx. 1996. Flexible Smoothing with B-splines and Penalties. Statist. Sci. 11, 2 (May 1996). doi:10.1214/ss/1038425655

Gerald E. Farin. 2001. Curves and Surfaces for CAGD: A Practical Guide (5th ed ed.). Morgan Kaufmann, San Francisco, CA.

T Flash and N Hogan. 1985. The Coordination of Arm Movements: An Experimentally Confirmed Mathematical Model. The Journal of Neuroscience 5, 7 (July 1985), 1688–1703. doi:10.1523/JNEUROSCI.05-07-01688.1985

Tamar Flash and Neville Hogan. 1998. Optimization Principles in Motor Control. In The Handbook of Brain Theory and Neural Networks. MIT Press, Cambridge, MA, USA, 682–685.

Kevin Frans, Lisa Soros, and Olaf Witkowski. 2022. CLIPDraw: Exploring Text-to-Drawing Synthesis through Language-Image Encoders. In Advances in Neural Information Processing Systems, S. Koyejo, S. Mohamed, A. Agarwal, D. Belgrave, K. Cho, and A. Oh (Eds.), Vol. 35. Curran Associates, Inc., 5207–5218.

Hiroyuki Fujioka and Hiroyuki Kano. 2007. Constructing Character Font Models from Measured Human Handwriting Motion. In 2007 American Control Conference. IEEE, New York, NY, USA, 1467–1472. doi:10.1109/ACC.2007.4282784

H. Fujioka, H. Kano, H. Nakata, and H. Shinoda. 2006. Constructing and Reconstructing Characters, Words, and Sentences by Synthesizing Writing Motions. IEEE Transactions on Systems, Man, and Cybernetics - Part A: Systems and Humans 36, 4 (July 2006), 661–670. doi:10.1109/TSMCA.2005.851344

Hiroyuki Fujioka, Wenli Zhu, Akinori Hidaka, and Hiroyuki Kano. 2017. Reconstructing Dynamic Font-Based Chinese Characters Using Support Vector Machine. In 2017 IEEE International Conference on Systems, Man, and Cybernetics (SMC). 2408–2413. doi:10.1109/SMC.2017.8122983

Rinon Gal, Yael Vinker, Yuval Alaluf, Amit Bermano, Daniel Cohen-Or, Ariel Shamir, and Gal Chechik. 2024. Breathing Life into Sketches Using Text-to-Video Priors. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). 4325–4336.

Roy Ganz and Michael Elad. 2024. CLIPAG: Towards Generator-Free Text-to-Image Generation. In 2024 IEEE/CVF Winter Conference on Applications of Computer Vision (WACV). IEEE, Waikoloa, HI, USA, 3831–3841. doi:10.1109/WACV57701.2024.00380

Neville Hogan and Dagmar Sternad. 2009. Sensitivity of smoothness measures to movement duration, amplitude, and arrests. J. Mot. Behav. 41, 6 (Nov. 2009), 529–534.

Iris A. M. Huijben, Wouter Kool, Max B. Paulus, and Ruud J. G. van Sloun. 2023. A Review of the Gumbel-max Trick and its Extensions for Discrete Stochasticity in Machine Learning . IEEE Transactions on Pattern Analysis & Machine Intelligence 45, 02 (Feb. 2023), 1353–1371. doi:10.1109/TPAMI.2022.3157042

Juno Hwang, Yong-Hyun Park, and Junghyo Jo. 2023. Resolution Chromatography of Diffusion Models. arXiv:2401.10247 [cs] doi:10.48550/arXiv.2401.10247

Shir Iluz, Yael Vinker, Amir Hertz, Daniel Berio, Daniel Cohen-Or, and Ariel Shamir. 2023. Word-as-Image for Semantic Typography. ACM Transactions on Graphics 42, 4, Article 151 (July 2023). doi:10.1145/3592123

Ajay Jain, Amber Xie, and Pieter Abbeel. 2023b. VectorFusion: Text-to-SVG by Abstracting Pixel-Based Diffusion Models. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). 1911–1920.

Jitesh Jain, Jiachen Li, Mang Tik Chiu, Ali Hassani, Nikita Orlov, and Humphrey Shi. 2023a. OneFormer: One Transformer To Rule Universal Image Segmentation. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition. 2989–2998.

Eric Jang, Shixiang Gu, and Ben Poole. 2017. Categorical Reparametrization with Gumbel-Softmax. In Proceedings International Conference on Learning Representations (ICLR). https://openreview.net/pdf?id=rkE3y85ee

Hiroyuki Kano, Magnus Egerstedt, Hiroaki Nakata, and Clyde F. Martin. 2003. B-Splines and Control Theory. Appl. Math. Comput. 145, 2 (2003), 263–288. doi:10.1016/S0096-3003(02)00486-1

Hiroyuki Kano, Hiroaki Nakata, and Clyde F. Martin. 2005. Optimal Curve Fitting and Smoothing Using Normalized Uniform B-splines: A Tool for Studying Complex Systems. Appl. Math. Comput. 169, 1 (Oct. 2005), 96–128. doi:10.1016/j.amc.2004.10.034

Craig S Kaplan and Robert Bosch. 2005. TSP Art. In Renaissance Banff: Mathematics, Music, Art, Culture. 301–308.

Hiroharu Kato, Deniz Beker, Mihai Morariu, Takahiro Ando, Toru Matsuoka, Wadim Kehl, and Adrien Gaidon. 2020. Differentiable Rendering: A Survey. arXiv:2006.12057 [cs] doi:10.48550/arXiv.2006.12057

Oren Katzir, Or Patashnik, Daniel Cohen-Or, and Dani Lischinski. 2024. Noise-Free Score Distillation. In The Twelfth International Conference on Learning Representations.

Gihyun Kwon and Jong Chul Ye. 2022. CLIPstyler: Image Style Transfer with a Single Text Condition. In 2022 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, New Orleans, LA, USA, 18041–18050. doi:10.1109/CVPR52688.2022.01753

Minghao Li, Tengchao Lv, Jingye Chen, Lei Cui, Yijuan Lu, Dinei Florencio, Cha Zhang, Zhoujun Li, and Furu Wei. 2023. TrOCR: Transformer-based Optical Character Recognition with Pre-trained Models. In AAAI 2023.

Tzu-Mao Li, Michal Lukáč, Michaël Gharbi, and Jonathan Ragan-Kelley. 2020. Differentiable Vector Graphics Rasterization for Editing and Learning. ACM Transactions on Graphics 39, 6 (Dec. 2020), 1–15. doi:10.1145/3414685.3417871

W. Li and E. Todorov. 2004. Iterative Linear Quadratic Regulator Design for Nonlinear Biological Movement Systems. In Proc. Intl Conf. on Informatics in Control, Automation and Robotics (ICINCO). 222–229.

Yixun Liang, Xin Yang, Jiantao Lin, Haodong Li, Xiaogang Xu, and Ying-Cong Chen. 2024. LucidDreamer: Towards High-Fidelity Text-to-3D Generation via Interval Score Matching. In Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition.

Chenxi Liu, Jessica Hodgins, and James McCann. 2017. Whole-cloth quilting patterns from photographs. In Proceedings of the Symposium on Non-Photorealistic Animation and Rendering (Los Angeles, California) (NPAR '17). Association for Computing Machinery, New York, NY, USA, Article 7, 8 pages. doi:10.1145/3092919.3092925

Lizheng Lu. 2015. A Note on Curvature Variation Minimizing Cubic Hermite Interpolants. Appl. Math. Comput. 259 (May 2015), 596–599. doi:10.1016/j.amc.2014.11.113

Camillo Lugaresi, Jiuqiang Tang, Hadon Nash, Chris McClanahan, Esha Uboweja, Michael Hays, Fan Zhang, Chuo-Ling Chang, Ming Yong, Juhyun Lee, Wan-Teh Chang, Wei Hua, Manfred Georg, and Matthias Grundmann. 2019. MediaPipe: A Framework for Perceiving and Processing Reality. In Third Workshop on Computer Vision for AR/VR at IEEE Computer Vision and Pattern Recognition (CVPR) 2019. https://mixedreality.cs.cornell.edu/s/NewTitle_May1_MediaPipe_CVPR_CV4ARVR_Workshop_2019.pdf

Ron Maharik, Mikhail Bessmeltsev, Alla Sheffer, Ariel Shamir, and Nathan Carr. 2011. Digital micrography. ACM Transactions on Graphics 30, 4, Article 100 (July 2011), 12 pages. doi:10.1145/2010324.1964995

Hirotsugu Matsukida and Hiroyuki Fujioka. 2013. Modeling and Reshaping Handwritten Characters Based on Dynamic Font Model. In 2013 International Joint Conference on Awareness Science and Technology & Ubi-Media Computing (iCAST 2013 & UMEDIA 2013). IEEE, Aizuwakamatsu, Japan, 344–350. doi:10.1109/ICAwST.2013.6765463

H. Meier and H. Nowacki. 1987. Interpolating Curves with Gradual Changes in Curvature. Computer Aided Geometric Design 4, 4 (Dec. 1987), 297–305. doi:10.1016/0167-8396(87)90004-5

Daniel Mellinger and Vijay Kumar. 2011. Minimum Snap Trajectory Generation and Control for Quadrotors. In 2011 IEEE International Conference on Robotics and Automation. 2520–2525. doi:10.1109/ICRA.2011.5980409

Daniela Mihai and Jonathon Hare. 2021. Differentiable Drawing and Sketching. arXiv:2103.16194 [cs]

Ben Poole, Ajay Jain, Jonathan T. Barron, and Ben Mildenhall. 2023. DreamFusion: Text-to-3D Using 2D Diffusion. In The Eleventh International Conference on Learning Representations.

H. Pottmann. 1990. Smooth Curves under Tension. Computer-Aided Design 22, 4 (May 1990), 241–245. doi:10.1016/0010-4485(90)90053-F

Zhiyu Qu, Lan Yang, Honggang Zhang, Tao Xiang, Kaiyue Pang, and Yi-Zhe Song. 2024. Wired Perspectives: Multi-View Wire Art Embraces Generative AI. In 2024 IEEE/CVF Conference on Computer Vision and Pattern Recognition (CVPR). IEEE, Seattle, WA, USA, 6149–6158. doi:10.1109/CVPR52733.2024.00588

Alec Radford, Jong Wook Kim, Chris Hallacy, Aditya Ramesh, Gabriel Goh, Sandhini Agarwal, Girish Sastry, Amanda Askell, Pamela Mishkin, Jack Clark, Gretchen Krueger, and Ilya Sutskever. 2021. Learning Transferable Visual Models From Natural Language Supervision. In Proceedings of the 38th International Conference on Machine Learning. PMLR, 8748–8763.

Christian H. Reinsch. 1967. Smoothing by Spline Functions. Numer. Math. 10, 3 (Oct. 1967), 177–183. doi:10.1007/BF02162161

Kejia Ren and Paul G. Kry. 2019. Single Stroke Aerial Robot Light Painting. In Proceedings of the 8th ACM/Eurographics Expressive Symposium on Computational Aesthetics and Sketch Based Interfaces and Modeling and Non-Photorealistic Animation and Rendering (Expressive '19). Eurographics Association, Goslar, DEU, 61–67. doi:10.2312/exp.20191077

L Romani and M.A Sabin. 2004. The Conversion Matrix between Uniform B-spline and Bézier Representations. Computer Aided Geometric Design 21, 6 (July 2004), 549–560. doi:10.1016/j.cagd.2004.04.002

Peter Schaldenbrand, James McCann, and Jean Oh. 2023. FRIDA: A Collaborative Robot Painter with a Differentiable, Real2Sim2Real Planning Environment. In 2023 IEEE International Conference on Robotics and Automation (ICRA). 11712–11718. doi:10.1109/ICRA48891.2023.10160702

Larry L. Schumaker. 1981. Spline Functions: Basic Theory.

Adrian Secord. 2002. Weighted Voronoi Stippling. In Proceedings of the 2nd International Symposium on Non-photorealistic Animation and Rendering. ACM, Annecy France, 37–43. doi:10.1145/508530.508537

Hasik Sunwoo. 2005. Matrix Representation for Multi-Degree Reduction of Bézier Curves. Computer Aided Geometric Design 22, 3 (March 2005), 261–273. doi:10.1016/j.cagd.2004.12.002

A. Tewari, O. Fried, J. Thies, V. Sitzmann, S. Lombardi, K. Sunkavalli, R. Martin-Brualla, T. Simon, J. Saragih, M. Nießner, R. Pandey, S. Fanello, G. Wetzstein, J.-Y. Zhu, C. Theobalt, M. Agrawala, E. Shechtman, D. B Goldman, and M. Zollhöfer. 2020. State of the Art on Neural Rendering. Computer Graphics Forum 39, 2 (2020), 701–727. doi:10.1111/cgf.14022

E. Todorov. 2004. Optimality Principles in Sensorimotor Control. Nature Neuroscience 7, 9 (2004), 907–915.

Emanuel Todorov and Michael I. Jordan. 1998. Smoothness Maximization Along a Predefined Path Accurately Predicts the Speed Profiles of Complex Arm Movements. Journal of Neurophysiology 80, 2 (Aug. 1998), 696–714. doi:10.1152/jn.1998.80.2.696

Kenji Tojo, Ariel Shamir, Bernd Bickel, and Nobuyuki Umetani. 2024. Fabricable 3D Wire Art. In Special Interest Group on Computer Graphics and Interactive Techniques Conference Conference Papers '24. ACM, Denver CO USA, 1–11. doi:10.1145/3641519.3657453

Zhifang Tong, Bolei Zuo, Xiaoxia Yang, Shengjun Liu, and Xinru Liu. 2025. Continuous-Line Image Stylization Based on Hilbert Curve. Computer Graphics Forum (2025). doi:10.1111/cgf.70169

M. Toussaint. 2017. A Tutorial on Newton Methods for Constrained Trajectory Optimization and Relations to SLAM, Gaussian Process Smoothing, Optimal Control, and Probabilistic Inference. In Geometric and Numerical Foundations of Movements. Springer International Publishing, Cham, 361–392.

Alan H Vermeulen, Richard H Bartels, and Glenn R Heppler. 1992. Integrating Products of B-splines. SIAM journal on scientific and statistical computing 13, 4 (1992), 1025–1038.

Yael Vinker, Yuval Alaluf, Daniel Cohen-Or, and Ariel Shamir. 2023. CLIPascene: Scene Sketching with Different Types and Levels of Abstraction. In 2023 IEEE/CVF International Conference on Computer Vision (ICCV). IEEE, Paris, France, 4123–4133. doi:10.1109/ICCV51070.2023.00383

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 Transactions on Graphics 41, 4, Article 86 (July 2022). doi:10.1145/3528223.3530068

Pauli Virtanen et al. 2020. SciPy 1.0: Fundamental Algorithms for Scientific Computing in Python. Nature Methods 17 (2020), 261–272. doi:10.1038/s41592-019-0686-2

Viviani, Paolo and Flash, Tamar. 1995. Minimum-Jerk, Two-Third Power Law, and Isochrony, Converging Approaches to Motor Planning.

Zhengyi Wang, Cheng Lu, Yikai Wang, Fan Bao, Chongxuan LI, Hang Su, and Jun Zhu. 2023. ProlificDreamer: High-fidelity and Diverse Text-to-3D Generation with Variational Score Distillation. In Advances in Neural Information Processing Systems, A. Oh, T. Naumann, A. Globerson, K. Saenko, M. Hardt, and S. Levine (Eds.), Vol. 36. Curran Associates, Inc., 8406–8441.

Fernando J. Wong and Shigeo Takahashi. 2011. A Graph-based Approach to Continuous Line Illustrations with Variable Levels of Detail. Computer Graphics Forum 30, 7 (2011), 1931–1939. arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/j.1467-8659.2011.02040.x doi:10.1111/j.1467-8659.2011.02040.x

Markus Worchel and Marc Alexa. 2023. Differentiable Rendering of Parametric Geometry. ACM Transactions on Graphics 42, 6, Article 232 (Dec. 2023), 18 pages. doi:10.1145/3618387

XiMing Xing, Chuang Wang, Haitao Zhou, Jing Zhang, Qian Yu, and Dong Xu. 2023. DiffSketcher: Text Guided Vector Sketch Synthesis through Latent Diffusion Models. In Thirty-seventh Conference on Neural Information Processing Systems. https://openreview.net/forum?id=CY1xatvEQj

Ximing Xing, Haitao Zhou, Chuang Wang, Jing Zhang, Dong Xu, and Qian Yu. 2024. SVGDreamer: Text Guided SVG Generation with Diffusion Model. (June 2024), 4546–4555.

Jie Xu and Craig S. Kaplan. 2007. Calligraphic Packing. In Proceedings of Graphics Interface 2007 on - GI '07. ACM Press, Montreal, Canada, 43. doi:10.1145/1268517.1268527

Zhijin Yang, Pengfei Xu, Hongbo Fu, and Hui Huang. 2021. WireRoom: model-guided explorative design of abstract wire art. ACM Transactions on Graphics 40, 4, Article 128 (July 2021), 13 pages. doi:10.1145/3450626.3459796

Hu Ye, Jun Zhang, Sibo Liu, Xiao Han, and Wei Yang. 2023. IP-adapter: Text Compatible Image Prompt Adapter for Text-to-Image Diffusion Models. arXiv:2308.06721 [cs.CV]

Lvmin Zhang, Anyi Rao, and Maneesh Agrawala. 2023. Adding Conditional Control to Text-to-Image Diffusion Models. In 2023 IEEE/CVF International Conference on Computer Vision (ICCV). IEEE, Paris, France, 3813–3824. doi:10.1109/ICCV51070.2023.00355

Peiying Zhang, Nanxuan Zhao, and Jing Liao. 2024. Text-to-Vector Generation with Neural Path Representation. ACM Transactions on Graphics 43, 4 (July 2024), 1–13. doi:10.1145/3658204

Changqing Zou, Junjie Cao, Warunika Ranaweera, Ibraheem Alhashim, Ping Tan, Alla Sheffer, and Hao Zhang. 2016. Legible Compact Calligrams. ACM Transactions on Graphics 35, 4 (July 2016), 1–12. doi:10.1145/2897824.2925887