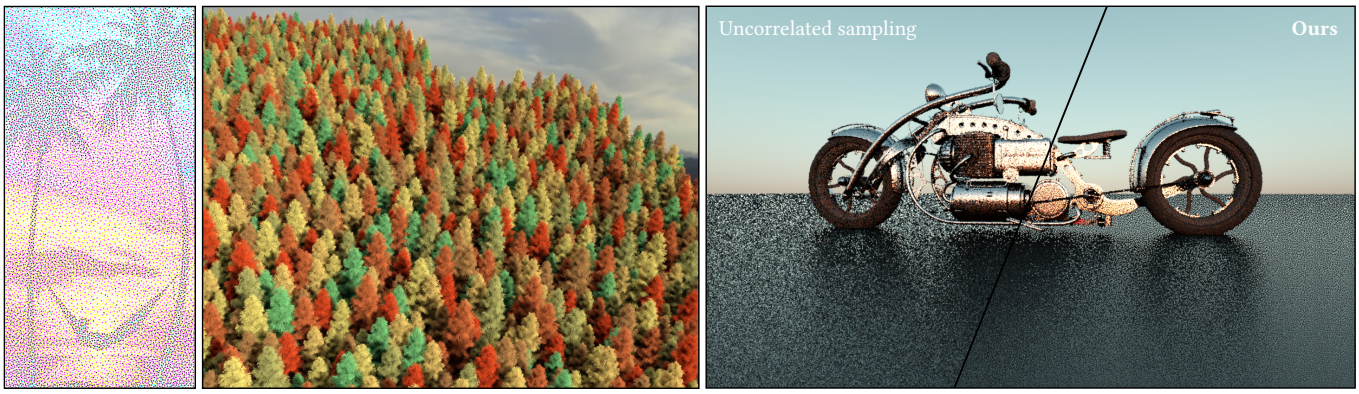
Scalable multi-class sampling via filtered sliced optimal transport

CORENTIN SALAÜN, Max-Planck-Institut für Informatik, Germany
ILIYAN GEORGIEV, Autodesk, United Kingdom
HANS-PETER SEIDEL, Max-Planck-Institut für Informatik, Germany
GURPRIT SINGH, Max-Planck-Institut für Informatik, Germany



Color stippling Object placement Perceptual error optimization

Fig. 1. Demonstration of our multi-class sampling framework on three applications. Left: CMYK color stippling involves optimizing for 15 classes, each following a different, non-uniform density. Middle: 7 colors of trees and their union optimized jointly. Right: Distributing rendering error as blue noise, cast as a multi-class problem (4096 classes), showing improved visual fidelity over traditional uncorrelated-pixel sampling.

We propose a multi-class point optimization formulation based on continuous Wasserstein barycenters. Our formulation is designed to handle hundreds to thousands of optimization objectives and comes with a practical optimization scheme. We demonstrate the effectiveness of our framework on various sampling applications like stippling, object placement, and Monte-Carlo integration. We a derive multi-class error bound for perceptual rendering error which can be minimized using our optimization. We provide source code at https://github.com/iribis/filtered-sliced-optimal-transport.

CCS Concepts: • **Computing methodologies** → **Ray tracing**.

Additional Key Words and Phrases: Multi-class sampling, blue noise, optimal transport, Monte Carlo, rendering, perceptual error

ACM Reference Format:
Corentin Salaün, Iliyan Georgiev, Hans-Peter Seidel, and Gurprit Singh. 2022. Scalable multi-class sampling via filtered sliced optimal transport. *ACM Trans. Graph.* 41, 6, Article 261 (December 2022), 15 pages. https://doi.org/10.1145/3550454.3555484

Authors' addresses: Corentin Salaün, Max-Planck-Institut für Informatik, Saarbrücken, Germany, csalaun@mpi-inf.mpg.de; Iliyan Georgiev, Autodesk, United Kingdom, me@iliyan.com; Hans-Peter Seidel, Max-Planck-Institut für Informatik, Saarbrücken, Germany, hpseidel@mpi-sb.mpg.de; Gurprit Singh, Max-Planck-Institut für Informatik, Saarbrücken, Germany, gsingh@mpi-inf.mpg.de.

0730-0301/2022/12-ART261 $15.00
https://doi.org/10.1145/3550454.3555484

1 INTRODUCTION

Multi-class sampling finds numerous applications in computer graphics, such as object placement [Wei 2010], visualization [Hu et al. 2020; Onzenoodt et al. 2021], and multi-tone image stippling [Secord 2002; Schulz et al. 2021]. The goal of multi-class sampling is to produce a point set that satisfies multiple objectives simultaneously. An objective is to optimize a specific subset of points to follow a given target distribution. When the subsets are mutually disjoint, the task is relatively easy since each objective can be optimized separately. The difficulty arises in applications where the subsets overlap. Overlaps introduce conflicts between optimization objectives. A classical example is multi-tone image stippling, where individual color channels—each represented by a point subset, or *class*—and their union(s) all have different target densities (Fig. 1a). Such problems call for formulating a global optimization problem that can encode all objectives with a desired balance between them.

Existing multi-class solutions [Wei 2010; Jiang et al. 2015; Qin et al. 2017] do not scale to large numbers objectives, both in terms of means to specify many objectives and ability to optimize them in reasonable time and/or memory footprint. We propose a formulation based on continuous Wasserstein barycenters to achieve such scalability. Our formulation provides a simple way to specify multiple objectives at once and the desired balance between them. It is complemented by a gradient-descent optimization scheme that is only weakly sensitive to the number of objectives.

We demonstrate the utility of our framework on diverse applications that involve a large number of objectives, including color stippling, object placement, (progressive) Monte-Carlo integration,

as well as blue-noise distribution of rendering error which we cast as a multi-class optimization problem. In summary:

- Our optimal-transport formulation allows specifying multiple optimization objectives at once via simple functions.
- Our stochastic optimization scheme scales to very large numbers of objectives.
- We derive an error bound for rendering error w.r.t. a given pixel-reconstruction kernel. When the kernel incorporates perceptual filtering, minimizing this bound yields point sets that distribute rendering error as blue noise over the image.

2 PRIOR WORK

In this section we review different sample correlations that are extensively studied in computer graphics.

2.1 Blue-noise sampling

Ulichney [1987] pointed out that isotropic point distributions with predominantly high-frequency spectral content, namely *blue noise*, cover the space evenly and look aesthetically pleasing. Since then blue-noise samples have been used for halftoning [Ulichney 1987], object placement [Kopf et al. 2006; Reinert et al. 2016], stippling [Secord 2002; Balzer et al. 2009] and visualization [Hu et al. 2020; Onzenoodt et al. 2021]. In rendering, Dippé and Wold [1985] and Cook [1986] also promoted samples with high-frequency content for anti-aliasing and image reconstruction. Various relaxation-based [Balzer et al. 2009; de Goes et al. 2012; Qin et al. 2017], tile-based [Ostromoukhov et al. 2004; Ostromoukhov 2007; Kopf et al. 2006; Wachtel et al. 2014] and number-theoretic-based methods [Keller 2013; Ahmed and Wonka 2021] have since been proposed to generate high-quality blue-noise samples in multiple dimensions.

Multi-class sampling. Considering only the spatial locations of the samples could severely limits their applicability in real-world scenarios. Multi-class sampling allows incorporating non-spatial features while maintaining the blue-noise property of the spatial coordinates. Wang and Parker [1999] first showed the impact of multi-class sampling on colored halftoning of images. They developed a sampling algorithm that generates blue-noise quality in combinations of the R,G, and B channels of an image. Wei [2010] proposed multi-class sampling algorithm based on dart throwing. Schmaltz et al. [2012] proposed electrostatic halftoning, whereas Jiang et al. [2015] used an SPH method to obtain multi-class samples. All these methods enforce multi-class blue noise through the use of an interaction matrix that encodes the spacing between class pairs. The matrix, however, can exhibit discontinuous changes in the off-diagonal entries, which represent the coupling between the different classes' distributions.Chen et al. [2012] proposed a two-step algorithm based on capacity-constrained Voronoi tessellation to obtain a multi-class property. In the first step, each class is individually optimized, and in the next step their unions are optimized. Chen et al. [2013] proposed a continuous multi-class sampling scheme limited to dart throwing and kernel-based optimization.

Qin et al. [2017] overcame these limitations via a multi-class framework based on optimal transport [Rabin et al. 2011; Rachev and Rüschendorf 1998; Agueh and Carlier 2011]. Classes and their unions each have a target distribution. By optimizing for the Wasserstein barycenter of these measures they obtain a multi-class blue-noise point set. To handle conflicts between classes and to avoid regularity, they leveraged entropic regularization [Cuturi 2013]. That method works well but lacks the flexibility to specify many objectives. Their optimization also does not scale well with the number of objectives. We propose a new formulation of multi-class sampling using sliced optimal transport which overcome these limitations and generalize multi-class sampling to different applications.

2.2 Monte-Carlo integration

In quasi-Monte Carlo literature, number-theoretic approaches are used to compute samples with good stratification, i.e., low discrepancy [Kuipers and Niederreiter 1974; Niederreiter 1992]. Discrepancy provides a measure of equidistribution and a bound for integration error via the Koksma-Hlawka inequality [Ermakov and Leora 2019]. Low-discrepancy point sets are widely used in image synthesis [Keller 2013; Pharr et al. 2016].

Following Durand [2011], a theoretical connection has been established between the error in Monte-Carlo integration and the sampling power spectra [Subr and Kautz 2013; Pilleboue et al. 2015; Singh and Jarosz 2017; Singh et al. 2019]. However, Fourier error remain insufficiently explored. Recently, Paulin et al. [2020] showen an error bound based on (sliced) optimal transport theory [Pitie et al. 2005; Villani 2008; Bonneel and Coeurjolly 2019; Julien et al. 2011]. That bound involves the Wasserstein distance [Kantorovich and Rubinstein 1958] which can be seen as the optimal-transport analog of the discrepancy metric. The samples obtained by minimizing this Wasserstein distance have blue-noise properties and compete with low-discrepancy distributions in terms of error.

2.3 Perceptual error optimization

Traditionally in Monte-Carlo rendering pixel values are estimated independently from one another, which yields white-noise distribution of error over the image. Mitchell [1991] noticed that error distributions with high-frequency power spectra give more pleasing appearance to noisy images. Following this observation, Georgiev and Fajardo [2016] proposed a dithering-inspired method to explicitly coordinate sampling across image pixels to achieve blue-noise error distribution. A number of follow-up works improved the quality and versatility of that basic approach Heitz et al. [2019]; Ahmed and Wonka [2020]; Belcour and Heitz [2021], but all these methods are heuristic in nature and lack a formal treatment.

Recently, Chizhov et al. [2022] adopted theory from halftoning to properly formalize perceptual error in rendering. We build on their formulation to derive a bound for this error; this bound can be minimized by our optimization scheme to produce sample sets that yield blue-noise error distribution.

3 PRELIMINARIES

We begin our exposition by reviewing basic concepts in optimal transport theory upon which we will build our multi-class point-sampling formulation in Section 4.

Optimal transport is concerned with moving the mass of one distribution to form another one [Villani 2008; Santambrogio 2015;

Peyré and Cuturi 2018]. Thinking of piles of sand, the question is what is the minimal cost (i.e., total mass displaced per unit distance) required to transport the sand from an initial pile to a target pile. This cost gives a notion of *distance* between two distributions.

Wasserstein distance. Formally, the optimal-transport distance between two measures (i.e., distributions) μ and ν is given by

$$W_p(\nu,\mu) = \left(\inf_{\gamma\in\Gamma(\nu,\mu)} \int_{\mathcal{X}^2} \|x-y\|^p \, \mathrm{d}\gamma(x,y) \right)^{1/p}, \tag{1}$$

which is called the p-order Wasserstein metric [Ollivier et al. 2014]. Here, $\|x-y\|$ denotes Euclidean distance on the domain $\mathcal{X}$. Intuitively, γ is a transport plan (formally, a joint measure with marginals μ and ν) such that $\mathrm{d}\gamma(x,y)$ gives the (differential) amount of mass to be transported between any two points x and y. The cost of doing so is thus $\|x-y\|^p \mathrm{d}\gamma(x,y)$. In the space $\Gamma(\nu,\mu)$ of all such plans, we are looking for one that minimizes the transportation cost over all pairs of points, i.e., the integral in Eq. (1).

Note that this formulation requires the two measures to have equal total mass, i.e., $\nu(\mathcal{X}) = \mu(\mathcal{X})$. However, they do not necessarily have to be probability measures, i.e., to have unit mass.

Sliced Wasserstein distance. Computing the optimal transport plan in the Wasserstein distance (1) can be very costly. A variant that is generally easier to solve involves computing only one-dimensional distances over all possible line projections of the two measures [Pitié et al. 2005; Rabin et al. 2011; Bonneel et al. 2015]:

$$SW_p(\nu,\mu) = \int_{\mathbb{S}^{d-1}} W_p\left(\nu^\theta,\mu^\theta\right) \mathrm{d}\theta \tag{2}$$

In this so-called sliced Wasserstein distance, $\theta \in \mathbb{S}^{d-1}$ is a point on the $(d-1)$-dimensional sphere, and ν^θ and μ^θ are the orthogonal projections onto the line through θ of the two measures.

(Sliced) Wasserstein barycenter. The Wasserstein distance provides an intuitive means to construct weighted averages of distributions, beyond simple mixtures (i.e., density averages). Generalizing the notion of barycentric interpolation between points, the Wasserstein barycenter ν interpolates between several measures μ_i [Agueh and Carlier 2011; Rabin et al. 2011]:

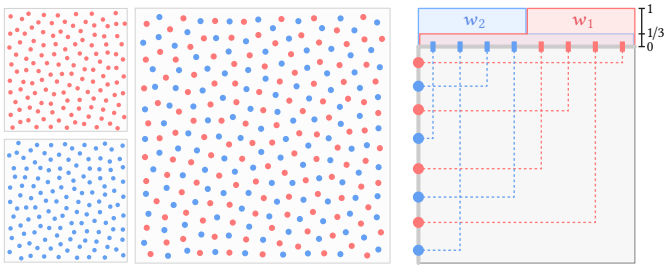
$$\nu = \arg\min_{\nu} \sum_i \lambda_i W_p^p(\nu,\mu_i), \tag{3}$$

where the scalar weights λ_i sum up to one. The Wasserstein barycenter can be seen as a means to compromise between the various objectives, here finding a distribution that minimizes the (weighted) distance to several targets. Replacing W_p by its sliced variant SW_p enables the practical computation of the barycenter through repeated 1D optimizations [Rabin et al. 2011; Bonneel et al. 2015].

4 MULTI-CLASS OPTIMAL TRANSPORT

Multi-class sampling involves producing a point set $X = \{x_i\}_{i=1}^n \subset \mathcal{X}$ with a unique optimization objective for each of multiple subsets. For example, in Fig. 2a the goal is to achieve high-quality isotropic uniform distribution for each point color and their union. We thus have three optimization objectives.

Typically, objectives are specified individually, which is reasonable when their number is small as in the applications considered by prior works [Wei 2010; Jiang et al. 2015; Qin et al. 2017]. However, this approach does not scale to our goal of handling large numbers of objectives. We want a principled and convenient means to specify objectives in bulk and to manage the conflicts between them. We also seek a more abstract way to specify point subsets beyond simple indices, to enable applications where points are more naturally grouped by other attributes.



(a) Classical 3-class optimization: red, blue, union (b) Our representation

Fig. 2. (a) A classical 3-class example, where the red and blue point subsets represent one uniform-density objective (class) each and their union is another objective, all three with equal optimization priority. (b) We represent this three-objective problem using two staircase functions, one per color, defined on an extra dimension (e.g., the point indices). The overlap between the functions implicitly specifies the third (union) objective.

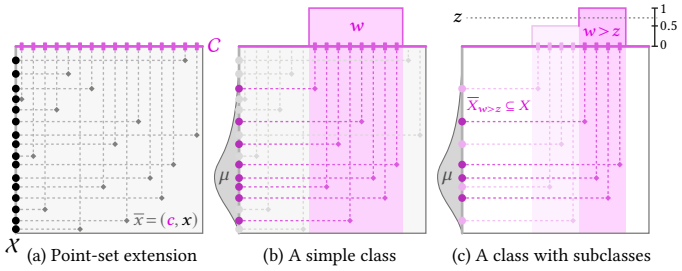
In this section, we propose a novel formulation of the multi-class sampling problem based on optimal transport, to achieve the aforementioned goals. We operate on an extended space where the extra dimension is used for point classification and the remaining dimensions are optimized. Figure 2b shows a simple example where we represent the three optimization objectives in Fig. 2a using *two* functions on that extra dimension. In the remainder of this section we introduce each component of this figure. Table 1 summarizes the most important notations we use.

4.1 Classes and subclasses

A class is an optimization objective specified by subset of points and a target distribution. The subset is typically given as a range of indices. We begin by generalizing the index space. Specifically, we extend the optimization space $\mathcal{X}$ by a *classification dimension* C. Given a point set X, a corresponding extended point set $\overline{X} = \{\overline{x}_i\}_{i=1}^n$ is created by distributing n class coordinates c_i uniformly in C and assigning them arbitrarily to the optimization points: $\overline{x}_i = (c_i, x_i)$.

Figure 3a illustrates this setup. The best choice for C depends on the application. C can be multi-dimensional but most often it will simply be the unit line, i.e., $C = [0,1]$, and the class coordinates will be the normalized indices in the base point set: $\overline{x}_i = (i/n, x_i)$. This normalization will allow us to define classes directly on the (fixed) extended space $C \times \mathcal{X}$, independently of (the size of) any particular point set. Sometimes a different class dimension is more natural, e.g., in rendering-error minimization C will be the 2D image plane (see Section 6).

Classes. The classification dimension C is not part of the optimization and is invariant to the number of points to be optimized. This allows to isolate point subsets by taking subregions of C, as



(a) Point-set extension (b) A simple class (c) A class with subclasses

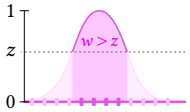
Fig. 3. (a) We extend the dimension of a point set by assigning to each point x a unique class coordinate $c \in C$ that remains fixed during optimization. This coordinate will often be the (normalized) index of the point. (b) A subregion in dimension C isolates a subset of points that can be optimized to follow a target distribution μ. The subregion can be given by the support of a function w on C, here a box function. We define a class as a pair (w, μ). (c) A non-trivial class function yields multiple subclasses (sharing a target distribution) enumerated by slicing w. The staircase function here yields two subclasses, selecting all ($\overline{X}_{w>0}$) and half ($\overline{X}_{w>0.5}$) the class' points.

illustrated in Fig. 3b. One way to specify a subregion is through the support of some function w on C; the figure shows the simplest case of a box function. The isolated points can then be optimized toward a target distribution μ.

We can now give a concrete definition of a *class* as a pair (w, μ). For a specific point set $\overline{X}$, the optimization objective is to have all points within the support of w follow the distribution μ.

Subclasses. The expressiveness of our formulation comes from the use of non-trivial (i.e., non-box) *class functions* w. Such a function can specify multiple optimization objectives. Figure 3c shows a simple staircase function. Two unique intervals on C can be extracted by thresholding that function, selecting all ($w > 0$) and half ($w > 0.5$) of the class' points, respectively. Each such interval represents a distinct optimization objective.

Extending the example to a more complex staircase—or even smooth—function w, allows us to specify an arbitrary number of sub-objectives, or *subclasses*, all sharing the target distribution μ. Each subclass selects a point subset within the support of the thresholded, or *filtered*, class function w at a value $z \geq 0$ (see the inline figure). Formally, the support of the filtered function is the set $\{c\colon w(c) > z\} \subseteq C$. A subclass is then defined by a tuple (w, μ, z).



Point-set filtering. A class is defined on a continuous extended space $\overline{X}$. A smooth class function defined in $\overline{X}$ thus specifies an entire continuum of subclasses. For a specific point set $\overline{X}$, their effective count is capped by the number of points. The subset of points selected by a subclass (w, μ, z) is obtained via a filtering operation: $\overline{X}_{w>z} \subseteq X$ contains all points $x_i \in X$ for which $w(c_i) > z$ (see Fig. 3c). Note that this also strips the class coordinate, producing a subset of the original point set X. A threshold value $z = 0$ selects all points in a class, and larger z values yield smaller subsets. The example in Fig. 2b demonstrates one such example with two staircase functions. For each staircase function, when $w > 1/3$ it selects half of the points, otherwise for $w > 0$ all points are selected.

Table 1. List of notations used throughout the document, which are also illustrated graphically in Figs. 3 to 5. For simplicity, we denote point sets and their corresponding Dirac point-mass measures using the same symbol X.

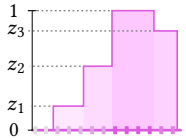
Symbol	Description
$\mathcal{X}, C, \overline{\mathcal{X}}$	Sampling domain, class domain, $\overline{\mathcal{X}} = C \times \mathcal{X}$
$x, c, \overline{x}$	Sample point, class coordinate, extended point $\overline{x} = (c, x)$
$X, \overline{X}$	Point set $\{x_i\}_{i=1}^n \subset \mathcal{X}$, extended point set $\{\overline{x}_i\}_{i=1}^n \subset \overline{\mathcal{X}}$
w, μ	Class function on C, target distribution on $\mathcal{X}$
$\mathcal{T}, w_t, \mu_t$	Space $\mathcal{T} = [0, 1] \ni t$ of classes (w_t, μ_t)
$\overline{X}_{w>z}$	Subset extraction via filteringc: $\{x\colon (c, x) \in \overline{X}, w(c) > z\} \subseteq X$
$\mu_{w>z}$	Scaling μ to match its mass to that of subset $\overline{X}_{w>z}$
X^θ, μ^θ	Projections of distributions onto axis θ

4.2 Subclass barycenter

The objective specified by one subclass can be satisfied by minimizing the Wasserstein distance between the corresponding point subset and the target. However, subclasses overlap, meaning that a point can be subject to multiple objectives. A compromise between all subclass objectives can be achieved via a barycenter that minimizes all involved Wasserstein distances simultaneously. For a continuous class function w, the barycenter takes an integral form:

$$\overline{X} = \arg\min_{\overline{X}} \underbrace{\int_{\mathbb{R}} W_p^p\left(\overline{X}_{w>z}, \mu_{w>z}\right) \mathrm{d}z}_{B_p(\overline{X}, w, \mu)}, \tag{4}$$

which is a continuous variant of Eq. (3), with the difference that we have one target distribution μ and multiple optimization (point) distributions. We scale the target distribution, $\mu_{w>z}$, to match the total mass of the subclass $\overline{X}_{w>z}$. The scaling factor is the relative number of points in the subclass compared to the size of the entire point set. When the class function w is piecewise constant, with levels $0 = z_0, z_1, \ldots, z_s = 1$ (see inline figure), the integral becomes a sum, turning the problem into a discrete barycenter:



$$\overline{X} = \arg\min_{\overline{X}} \sum_{j=1}^{s} \lambda_j W_p^p\left(\overline{X}_{w>z_j}, \mu_{w>z}\right), \quad \text{with} \quad \lambda_j = z_j - z_{j-1}. \tag{5}$$

We enforce w to have a maximum value of one to ensure that the weights λ_j sum up to unity. For a trivial (box-function) class, the barycenter simplifies to the single objective of minimizing the Wasserstein distance between all class points and the target μ.

Note that the class function w encodes both the shape and the relative importance of each subclass (i.e., its weight λ_i in the discrete case). A class (w, μ) thus completely describes an entire optimization problem (4), independently of the point-set size.

4.3 Multi-class barycenter

While a single subclass barycenter can completely describe some optimization tasks, it is not sufficiently expressive to model many practical problems. For example, having overlapping point subsets follow different target distributions. Even with one target, multiple subsets can be assembled into a single class only if they are nested

into one another. The example in Fig. 2 cannot be modelled with a single class as the red and blue subsets are disjoint. Such problems require specifying and optimizing across multiple classes.

Continuous case. To specify multiple classes, we add one more dimension to our representation from Fig. 3, illustrated in Fig. 4a. Each point t on the $\mathcal{T}$ axis identifies a class (w_t, μ_t). $\mathcal{T}$ can be multi-dimensional but for simplicity we use the unit line: $\mathcal{T} = [0, 1]$. The different classes generally have conflicting objectives due to overlaps in their associated functions w_t. As discussed in Section 4.2, the compromise between these objectives can be expressed as the Wasserstein barycenter

$$\overline{X} = \arg\min_{\overline{X}} \underbrace{\int_0^1 \int_0^1 W_p^p\left(\overline{X}_{w_t>z}, \mu_{t,w_t>z}\right) \mathrm{d}z \, \mathrm{d}t}_{B_p(\overline{X}, w_t, \mu_t) \quad \text{Eq. (4)}} \tag{6}$$

across all classes (outer integral) and their subclasses (inner integral), recalling that our class functions have a maximum value of one.

Discrete case. Not every identifier t has to map to a unique class. When the classes are a finite number n, the mapping is piecewise constant: $0 = t_0, t_1, \ldots, t_n = 1$, and every $t \in [t_{i-1}, t_i)$ maps to the class (w_i, μ_i). In the fully discrete case, where each class has a staircase-like function, the optimization problem (6) becomes a sum:

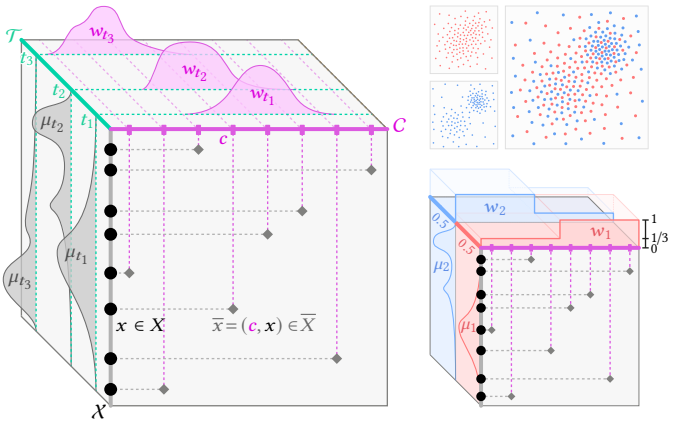
$$\overline{X} = \arg\min_{\overline{X}} \sum_{i=1}^{n} \sum_{j=1}^{s_i} \kappa_i \lambda_{i,j} W_p^p\left(\overline{X}_{w_i>z_{i,j}}, \mu_{i,w_i>z_{i,j}}\right), \tag{7}$$

where $\kappa_i = t_i - t_{i-1}$ are the class weights, and $\lambda_{i,j} = z_{i,j} - z_{i,j-1}$ are the subclass weights as in Eq. (5).

Figure 4b extends the example from Fig. 2 to non-uniform target distributions. We formalize this optimization problem using only two classes: (w_1, μ_1) and (w_2, μ_2). Filtering the point set using these class functions give four subsets: $\{\overline{X}_{w_1>0}, \overline{X}_{w_1>1/3}, \overline{X}_{w_2>0}, \overline{X}_{w_2>1/3}\}$. Equation (7) aims to find the barycenter defined by the Wasserstein distance wrt each subset. It is important to note that the target distribution is only defined for the red (μ_1) and the blue points (μ_2) and not their union. The union will be aiming towards a barycenter of μ_1 and μ_2 since each class function considers all the points (the union) when $w_i > 0$.

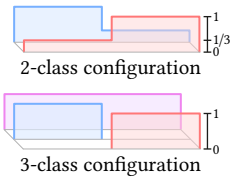
Discussion. Note that the class functions w (defined along the magenta axis in Fig. 4) can overlap. Overlaps allow increasing the "footprints" of individual objectives to target more points than would be otherwise possible; however, they also introduce conflicts. In regions of overlap, the values of each function indicate its class' relative local optimization priority. Consequently, using functions with smooth falloffs allows us to precisely control the barycentric trade-off between class objectives in such regions. In the general case of diverse targets μ_1 and μ_2, the class overlaps can make it difficult to satisfy simultaneously the objectives. Generally, we want to avoid having two classes assign high priority to the same region. Class functions should ideally be arranged to overlap only in their tails; this helps better satisfy each class' objective by minimizing conflicts and reducing optimization pressure. Class functions can be designed depending on the application.

Our more traditional-looking discrete barycenter (7) makes it clear that the atomic optimization objective in our framework is the subclass. A subclass is equivalent to a trivial, box-function class. The right inline figure at the bottom shows 3 such box functions representing the classical 3-class example (red, blue and their union).The formulation of Qin et al. [2017] supports only such classes. It is still as expressive as ours (2-class) but does not provide means to easily specify trade-offs between many objectives as it is not designed to scale to large number of objectives. Finally, we do not need to explicitly specify a target distribution for the union, which ends up being optimized toward a barycenter of the two targets. Figure 4b shows one such example point set where the target distributions are only defined for the red and blue points. The union is optimized towards their barycenter following Eq. (7).



(a) Optimization parameters (b) Two-class example

Fig. 4. (a) Our continuous optimization formulation yields a barycenter between classes (w_t, μ_t), each identified by point t on the unit line $\mathcal{T}$; we show three classes here. Smooth class-function falloffs allow for accurate control over conflicts resulting from overlaps. (b) A classical, fully discrete example with non-uniform target distributions. Each class function assigns equal optimization priority to each half of the points and their union. The top shows an optimized 256-point set.



5 STOCHASTIC GRADIENT-DESCENT OPTIMIZATION

In its most general form, the multi-class barycenter problem (6) is continuous. For a finite number of optimization points, the effective number of subclasses within any class is finite too. However, the use of continuous class functions makes this number very large, far beyond the few objectives that state-of-the-art multi-class methods [Wei 2010; Jiang et al. 2015; Qin et al. 2017] can scale to, in terms of both memory and computation time. This is because these iterative methods optimize for all objectives at every step.

Taking cues from stochastic gradient-descent methods [Bottou 1998], our approach is to optimize one objective at each of many iterations. Such optimization trivially scales to arbitrarily many objectives, although with potentially reduced convergence speed. Another advantage of this approach is that memory consumption does not directly depend on the objective count.

Sliced multi-class barycenter. Our multi-class barycenter formulation (6) computing optimal transport plans and minimizing Wasserstein distances, which can be very costly. For practical efficiency, we turn to sliced optimal transport, replacing the Wasserstein distance W_p by its sliced approximation SW_p (2). This adds another dimension to the integral in Eq. (6), over the projection axis θ:

$$\overline{X} = \arg\min_{\overline{X}} \int_0^1 \int_0^1 \int_{\mathbb{S}^{d-1}} W_p^p \left(\overline{X}^\theta_{w_t > z}, \mu^\theta_{t, w_t > z} \right) \mathrm{d}\theta \mathrm{d}z \mathrm{d}t. \quad (8)$$

Since the sliced Wasserstein distance (2) bounds the regular Wasserstein distance (1) [Bonnotte 2013], the resulting optimization problem (8) is an upper bound for the one in Eq. (6). During optimization, we use this Eq. (8) as our cost function which involves *filtering* point set for each slice θ. For brevity, we refer to this as filtered sliced optimal transport (FSOT) in the rest of the paper.

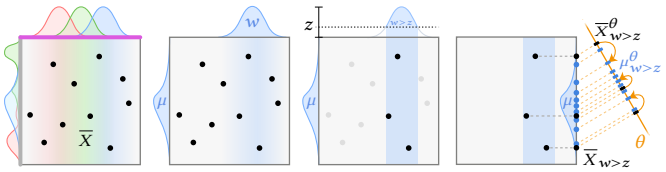
Iterative minimization. The 1D subclass Wasserstein distance in Eq. (8) has a known solution whose derivative w.r.t. an optimization point x_i we derive in Appendix A. The derivative of the entire barycenter is then a nested integral of such 1D derivatives. This enables an iterative stochastic minimization scheme which performs repeated 1D gradient-optimization steps by randomly sampling the multi-dimensional integral in Eq. (8).

Figure 5 illustrates one step of our optimization procedure. Given an extended point set $\overline{X}$ and a class configuration, we first select a class (w, μ), then threshold its function w with a random value z to choose a subclass that isolates a fraction $\overline{X}_{w>z}$ of the points. Finally, we sample an axis θ and perform one step of gradient-descent optimization on the 1D Wasserstein distance between the projected points $\overline{X}^\theta_{w>z}$ and the projected (scaled) target distribution $\mu^\theta_{w>z}$ along the axis.

We repeat this entire process multiple times to obtain many offset vectors for every point. Appendix B describes the computation of these offsets which are balanced across subclasses with varying sizes. We average these offsets, update the point's position, and begin a new iteration on the result. This is similar to the method of Paulin et al. [2020] but simultaneously considering multiple optimization targets. Another difference is that we consider arbitrary target distributions on a general Euclidean domain $\mathcal{X}$.

Since 1D projections of target distributions cannot always be analytically represented, we point-sample them at a rate 3–5× higher than the n points being optimized. The optimization still solves a balanced (i.e., discrete one-to-one) optimal-transport problem. This is done by binning the target points across n adaptive bins that follow the target distribution. Points within each bin are then averaged. Appendix B provides more details.

Offset correction. Projecting a target distribution μ along an arbitrary axis generally yields a different, non-uniform distribution μ^θ for each axis, even when the target is uniform [Paulin et al. 2020]. Stochastic gradient descent on such different distributions can produce point offsets that are highly anisotropic in the (full-dimensional) optimization domain. The anisotropy is aligned with the density/domain boundaries and is susceptible to causing point alignments, as seen in Fig. 7c. We avoid this problem by scaling the gradient of each projected point by the projected target density at that point. We estimate this density using the projected target samples. For single-class sampling, this gradient correction is the major change between our method and Paulin et al. [2020] which leads to the quality improvement shown in Fig. 7f. More details on the computation of this factor can be found in Appendix B.



(a) Configuration (b) Class selection (c) Subclass selection (d) 1D optimal transport

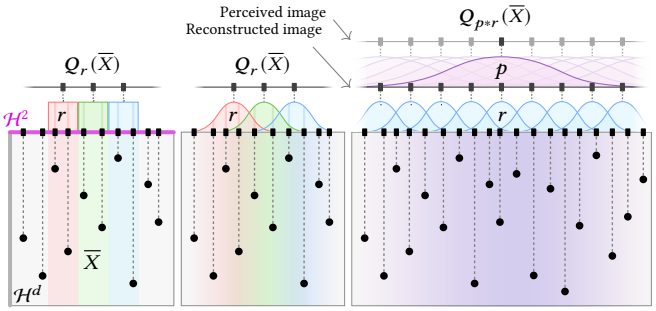
Fig. 5. One step of our stochastic gradient-descent minimization of the sliced multi-class barycenter (8). Given the optimization parameters and an extended point set (a), we first select a class (b) and then randomly threshold) the class function to sample a subclass (c). Finally, we project the filtered set and the class' target distribution onto a sampled axis and perform one step of 1D gradient-descent optimization (d). We handle arbitrary target distributions by point-sampling them before projection.

Discussion. The sliced Wasserstein distance is only an approximation to the regular distance, and can yield suboptimal barycenters [Bonneel and Pfister 2013]. However, in our experience it is a practical option for optimizing many points for many targets and produces satisfactory results even with highly non-uniform target distributions. Other approaches such as entropic regularization [Cuturi 2013], stochastic barycenters [Claici et al. 2018] or neural solvers [Korotin et al. 2022] can also be employed but we leave that for future work. Another consequence of using sliced optimal transport is that it increases the effective number of objectives, by adding an extra (spherical) dimension to the barycentric integral (8). Thankfully, the individual "sliced" 1D objectives are simple, and stochastic optimization scales to the added complexity.

6 PERCEPTUAL ERROR OPTIMIZATION

In this section we describe how to use our multi-class framework for perceptual optimization of image error in Monte-Carlo rendering.

In rendering, the value of every pixel is a light-transport integral. In practice pixel integrals are estimated via point sampling, and the resulting error manifests itself as image noise. Research efforts in sampling have traditionally focused on reducing the *magnitude* of the error, i.e., the accuracy of individual pixel estimates. Recently, it has been recognized that the *distribution* of this error over the image plays an important perceptual role, and that visual fidelity can be drastically improved when this distribution is isotropic and high-frequency [Georgiev and Fajardo 2016]. Achieving such a blue-noise distribution requires carefully coordinating the samples *across pixels*. We show that this problem can be cast as a multi-class optimization. We derive an image-error bound which can be minimized using our multi-class barycenter (6). The resulting formulation provides a principled way to minimize error in Monte Carlo rendering w.r.t. given perceptual and/or pixel-reconstruction kernels.



(a) Box reconst. (b) Gaussian reconst. (c) Gaussian reconst. + percept. filter

Fig. 6. Illustration of image synthesis where the grey box represents the sampling space, i.e. the unit hypercube $\mathcal{H}^{2+d}$; the horizontal axis represents the image subspace where reconstruction from the samples $\overline{X}$ is performed. (a) When using a box reconstruction kernel r, the sample sets estimating different pixels are disjoint. (b) A Gaussian kernel introduces overlaps, making each sample contribute to multiple pixel estimates. (c) The human visual system applies additional filtering on the reconstructed image with a generally wider kernel p. The convolution $p * r$ acts as an effective reconstruction kernel for the *perceived* image, and introduces even more overlaps.

6.1 Problem statement

Given a point set $X = \{x_i\}_{i=1}^n$, the value I_r of an image pixel is estimated by point-sampling its associated integral:

$$Q_r(X) = \frac{1}{n}\sum_{i=1}^{n} r(x_i)f(x_i) \;\approx\; I_r = \int_{\mathcal{H}^{2+d}} r(x)f(x)\,\mathrm{d}\rho(x) = \langle r, f\rangle. \quad (9)$$

Here r is a pixel-reconstruction kernel, $f(x)$ is the illumination carried by a light-transport path corresponding to the point x in the unit hypercube $\mathcal{H}^{2+d}$ with Lebesgue measure ρ. The first two dimensions are image space (where r operates), and d is the path-space dimension. The variance of an estimate $Q_r(X)$ is reduced when the samples in X within the kernel support are well-stratified.

When using box-kernel reconstruction (Fig. 6a), every sample falls within the kernel of a single pixel, which allows stratifying the samples independently per pixel. Non-box kernels, e.g., Gaussians, generally overlap in image space, making each sample contribute to the estimates of several pixels (Fig. 6b). This case calls for coordinating the stratification of samples across pixels.

Moreover, our eyes do not perceive individual pixels but rather process the image as a whole. One type of processing that occurs is pre-filtering the input visual signal to avoid aliasing. That is, we perceive a version of the image that is blurred by an amount dependent on the observing distance. This filtering can be modeled as a discrete convolution of the (r-reconstructed) pixels with a perceptual filter p [González et al. 2006; Näsänen 1984] that can be well approximated by a Gaussian [Pappas and Neuhoff 1999]. Every pixel in the *perceived ground-truth image* thus takes the form $p * I_r = p * \langle r, f\rangle = \langle p * r, f\rangle = I_{p*r}$. Analogously, pixels in the *perceived estimated image* can be written as $Q_{p*r}(X)$, which we illustrate in Fig. 6c. That image can thus be computed by convolving the samples with a combined reconstruction kernel $p * r$ centered at every pixel. The difference between the two images can be viewed as a measure of perceptual error [Chizhov et al. 2022]. We can then formulate our problem as minimizing reconstruction w.r.t. a given (combined) kernel by optimizing the distribution of the samples X.

Note that in reality pixel reconstruction is performed by the renderer—to compute pixel estimates, while perceptual filtering occurs in the human visual system upon perceiving these estimates.

6.2 Multi-class image-error bound

Figure 6 illustrates graphically how image-error minimization can be viewed as a multi-class optimization problem. Mapping the problem to the language of Section 4, the optimization domain is the d-dimensional unit hypercube, $\mathcal{X} = \mathcal{H}^d$, and, notably, the class dimension is not the unit line (e.g., as in Fig. 2b) but the unit square, $\mathcal{C} = \mathcal{H}^2$. The regular and extended point sets are identical, $\overline{X} = X$. Every pixel has an associated reconstruction kernel and defines a distinct class, all sharing the Lebesgue measure ρ as their (uniform) target distribution. Next we show that the barycenter between these classes provides a bound for the (perceptual) error of the image.

Pixel-error bound. The error of a pixel w.r.t. some given kernel w is the difference between the estimated value and the ground truth: $\epsilon_w(X) = \left|Q_w(X) - I_w\right|$. This becomes a *perceptual error* when the kernel $w := p * r$ incorporates perceptual filtering. Paulin et al. [2020] recently showed that optimal transport can provide a bound on the estimation error of pixel during Monte Carlo integration. In Appendix D we provide a simple proof for this bound which for a pixel in our setting reads $\epsilon_w(\overline{X}) \le L_{w\cdot f} W(\overline{X}, \rho)$, which is the product of the Lipschitz constant of the integrand $w \cdot f$ and the 1-Wasserstein distance between the point set and the uniform distribution. Unfortunately, this bound is not immediately useful: it measures the deviation of the entire point set $\overline{X}$ from uniformity and does not capture the strong effect of the narrow-support kernel w on each pixel estimate. We instead desire a bound tailored to the estimation of weighted integrals of the form $\int w \cdot f$. We derive such a bound for the pixel error in Appendix D:

$$\epsilon_w(\overline{X}) \;\le\; L_f \int_{\mathbb{R}} W(\overline{X}_{w>z}, \rho_{w>z})\,\mathrm{d}z \;=\; L_f B_1(\overline{X}, w, \rho), \quad (10)$$

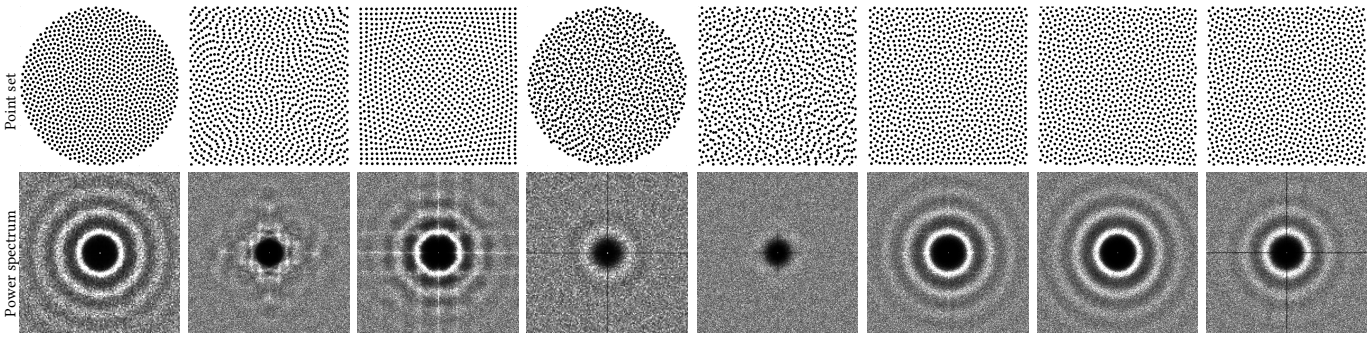
where B_1 is the minimization objective of the 1-Wasserstein subclass barycenter (4). Note that the kernel w has moved from the Lipschitz constant to the Wasserstein distance.

Image-error bound. Our end goal is to minimize the total image error. Applying the bound from Eq. (10) to each of M pixels yields a bound for the L_1 error:

$$\sum_{i=1}^{M} \epsilon_{w_i}(\overline{X}) \;\le\; L_f \sum_{i=1}^{M} B_1(\overline{X}, w_i, \rho). \quad (11)$$

This bound is a product of the Lipschitz constant of f and a (discrete) M-class barycenter (6). It postulates that to reduce the image error, we need to increase the uniformity of all subsets of $\overline{X} = X$ given by the z-filtering of every kernel (i.e., class function) w_i.

Equation (11) is based on the 1-Wasserstein distance W_1, but in practice we use our W_2-based optimization scheme from Section 5 to minimize a sliced variant of the bound. This works because SW_1 is bounded by SW_2. We provide a derivation of the SW_1 gradient in supplemental Section S1. Note that we do not optimize the image-space dimensions of the points which are fixed and used for classification.



(a) Paulin et al. [2020] (circle) (b) Paulin et al. [2020] (circle+warp) (c) Paulin et al. [2020] (square) (d) Paulin et al. [2020] (circle)+projections (e) Paulin et al. [2020] (circle+warp)+projections (f) **Ours FSOT** (square) (g) **Ours FSOT** (toroidal square) (h) **Ours FSOT** (square) + projections

Fig. 7. Comparison between different variants of our optimization and that of Paulin et al. [2020] which we build upon. All point sets are of size 1024 and the Fourier power spectra are averaged over 10 realizations. For our method we show realizations constructed with and without toroidality. Paulin et al. optimize on the unit circle (a) and then warp the resulting point set to the unit square (b); they also show direct unit-square optimization (c). Both their variants yield alignments that our method avoids (f-h), largely thanks to our offset correction (described in Section 5). One can also prioritize certain projections which can be beneficial for Monte-Carlo integration (see Fig. 14); here we choose the x- or y-axis in 30% of the optimization steps (d, e, h).

7 EXPERIMENTS

To demonstrate the utility of our multi-class framework, we show the results of several experiments from CPU (C++) and GPU (CUDA) implementations. The C++ implementation of our stochastic gradient-descent optimization is parallelizable across the projections within each iteration. The CUDA one parallelizes over different operations (projections, sorting, averaging). The different point sets presented below have been generated on an NVIDIA Quadro RTX 8000 and Intel® Core™ i9-8950HK CPU @ 2.90GHz. All rendering results have been generated using PBRT-v3 [Pharr et al. 2016]. The supplemental material includes an HTML viewer with more results.

7.1 Blue-noise sampling

Single-class blue noise. Figure 7 compares the blue-noise quality for a single-class point set and its power spectrum averaged over 10 realizations. Paulin et al. [2020] perform the optimization on the unit circle, achieving high quality (Fig. 7a) which, however, deteriorates after warping the points to the unit square (Fig. 7b); this is also reflected in the power spectrum. Paulin et al. also show direct optimization on the unit square, which yields strong alignments along the domain boundaries (Fig. 7c). In contrast, our unit-square optimization produces a high-quality blue-noise distribution, without any alignments (see Fig. 7f). This quality improvement is mostly due to our offset correction (Section 5) which avoids alignments. Our optimization can also operate on a toroidal domain (Fig. 7g).

Prioritizing certain projection directions can be beneficial in Monte-Carlo integration as we will demonstrate below; Figure 7h shows an example where we choose the x- or y-aixs with 30% probability, creating a cross in the power spectrum. While Paulin et al. [2020] can also prioritize these projections on the unit circle (Fig. 7d), the achieved quality is not maintained after mapping the points to the unit square (Fig. 7e).

Multi-class sampling with uniform density. In Fig. 8, we compare our method to that of Qin et al. [2017] on the two-color problem from Fig. 2. The spectra obtained by Qin et al. [2017] and our method without toroidality show some artefacts due to natural point alignments near the domain boundaries. Our method shows same quality for the single colors and slightly better for the complete set. In Fig. 9, we extend the problem to 3 colors, i.e., 7 classes. The overall distribution quality is good for all classes. The spectral distributions of the three color pairs RG, RB, GB exhibit double peaks, which has also been observed by Qin et al. [2017, Fig. 7]. The reason for this double peak is that the improvement of these particular two-color classes has a strong impact on the other classes. Improving two-color classes would reduce the quality of the other classes too much.

Color stippling. Figure 10 shows a CMYK image stippled with 20,000 points. The four individual colors and their various 2- and 3-color combinations each represent a class with a different target density, for a total of 15 classes. We show five of these classes. The combinations have weighted-average densities based on the respective energy of the channels. Unlike prior work [Qin et al. 2017], our stochastic gradient-descent optimization scales to this many classes with a negligible memory footprint. The supplemental document shows another color-stippling result with 40,000 points.

Figure 12 compares our stippling to that of de Goes et al. [2012] on a greyscale image using 15,000 points. Although our method is not tuned for single-class problems, we achieve competitive quality. The supplemental document includes a result with 100,000 points.

Our method also be used for animation stippling where consecutive frames share a fraction of the points. We include an example in the supplemental material.

Continuous class extraction. To demonstrate the scalability of our optimization to a large number of objectives, in Fig. 11 we consider a non-traditional multi-class problem. We define two classes with linear-ramp functions on the index space of points, as illustrated in the inline figure. The target density is uniform. This construction allows us to split the optimized set at any point index, so that the subsets on the left and right of it (and their union) have

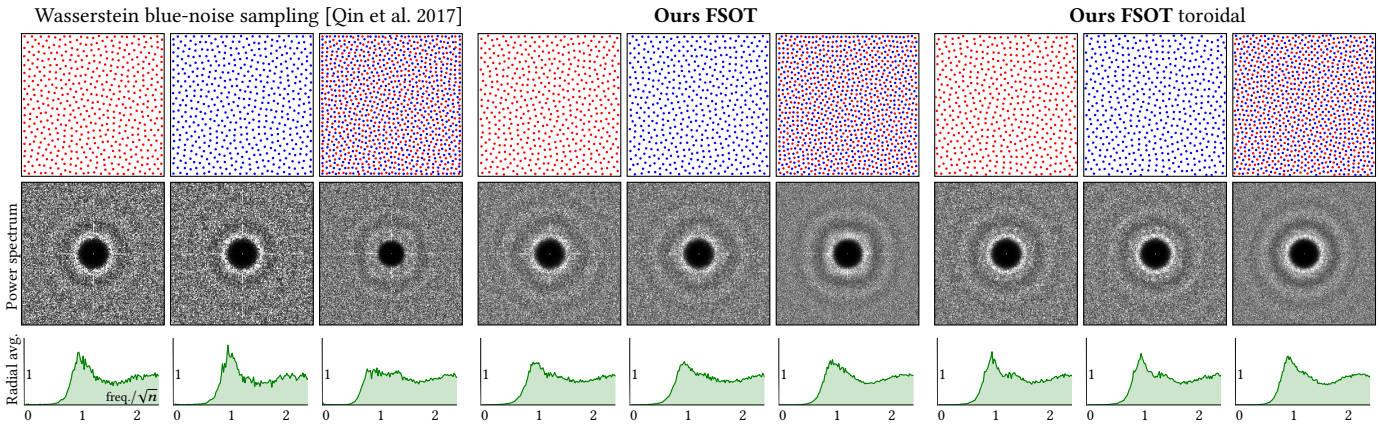


Fig. 8. 3-class (red, blue, red & blue) optimization of 2048 points (top row), along with the corresponding expected power spectra (middle row) and their radial averages (bottom row). Our optimization (bounded and toroidal) achieves similar quality to that of Qin et al. [2017] (bounded); ours takes 38 sec on GPU and theirs takes about 1 hour on CPU. The spectral anisotropy in the left two results is due to point alignments near the boundaries.

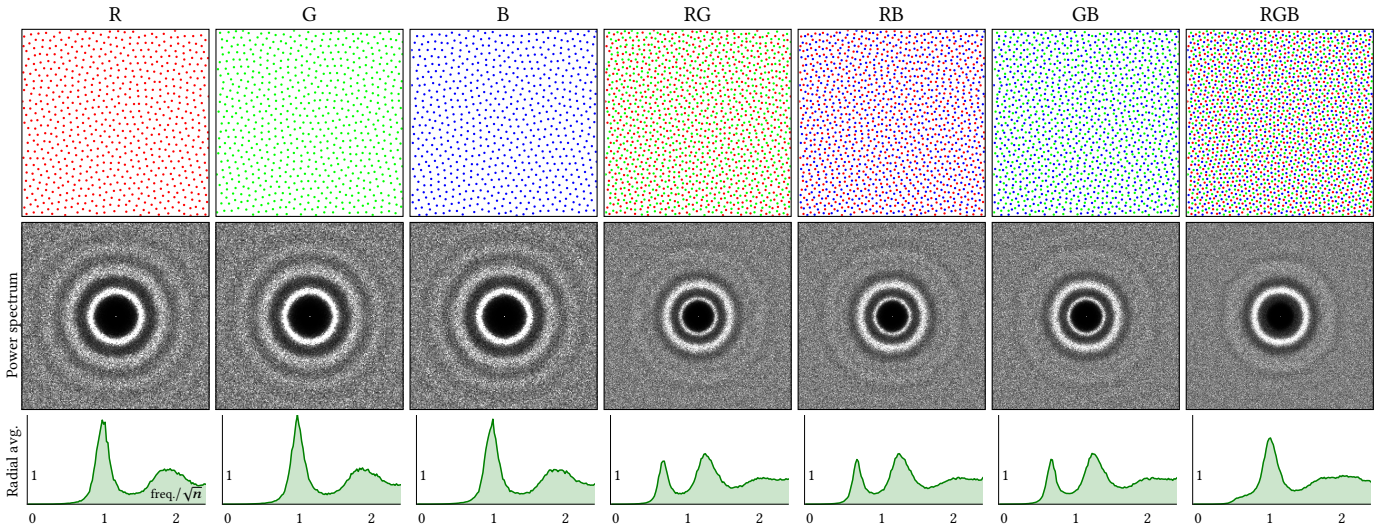


Fig. 9. Extending the problem in Fig. 8 to three colors (i.e., 7 classes), using 2049 points (683 points per color). Achieving uniform blue-noise quality across all classes is more difficult in this case due to higher, contention between the objectives.

good-quality distribution. We optimize 2048 points for an effective total number of 4096 targets (i.e., subclasses). In the figure we include a few example splits; the corresponding subset power spectra show reasonable blue-noise quality considering the large number of optimization objectives. An animation showing the evolution of the visualization according to the choice of splitting index can be found in the supplemental material.

Object placement. Multi-class sampling can also be used to place objects in an environment. Figure 1 middle shows an example distribution of trees, each taking one of 7 colors. We also optimize for the union, for a total of 8 classes. Two other results are displayed in Fig. 13. The point set used (in the lower left corner) was produced using the optimization configuration from the continuous class extraction problem presented above. In the left image, the point color guides the tree color, and in the right image it guides the tree height.

7.2 Monte-Carlo integration

We also evaluate our approach on Monte-Carlo integration. In Fig. 14 we analyze the convergence behavior of our optimized point sets against the method of Paulin et al. [2020] on two simple integrands. We generate two types of point sets using each method: one with axis-aligned 1D projections prioritized with 30% probability (as in Fig. 7h) and the other without prioritization (as in Fig. 7g). For the isotropic integrand on the left the four variants give similar results. On the other hand, on the right integrand with axis-aligned variation, our projections yield lower integration error. Axis prioritization

Fig. 10. CMYK color stippling involves optimizing 15 classes—four base colors and their various 2- and 3-color combinations, each targeting a different density. In this example we use 20,000 points and show five of these classes.

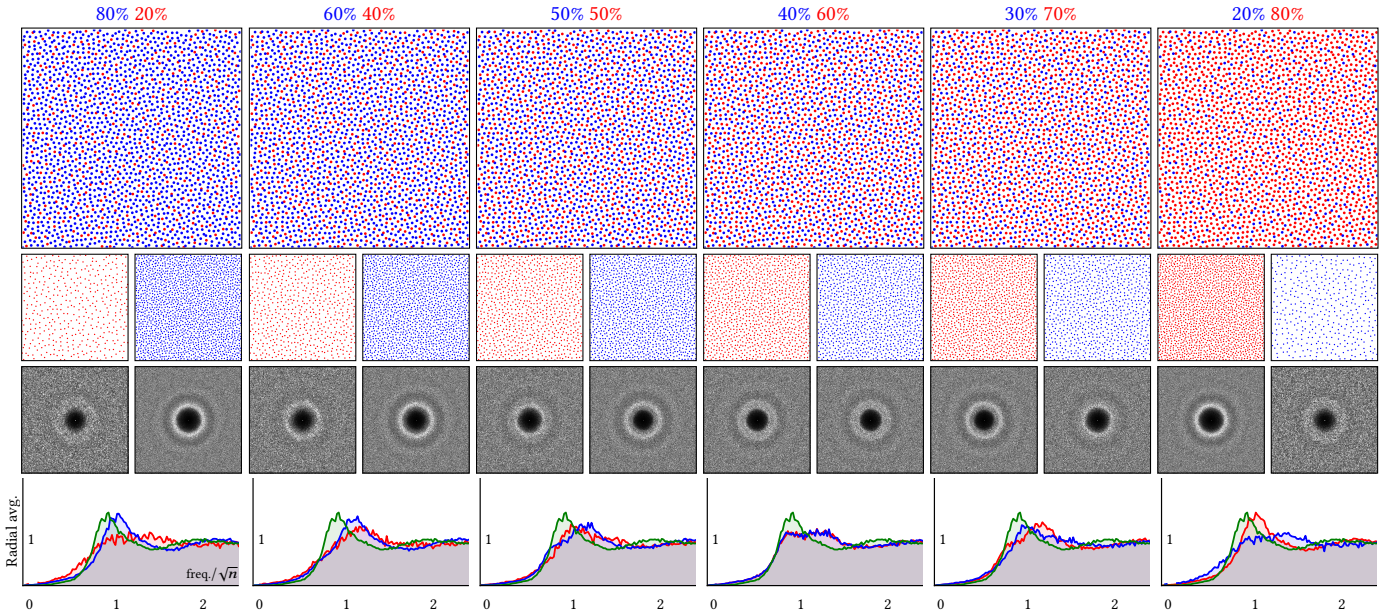
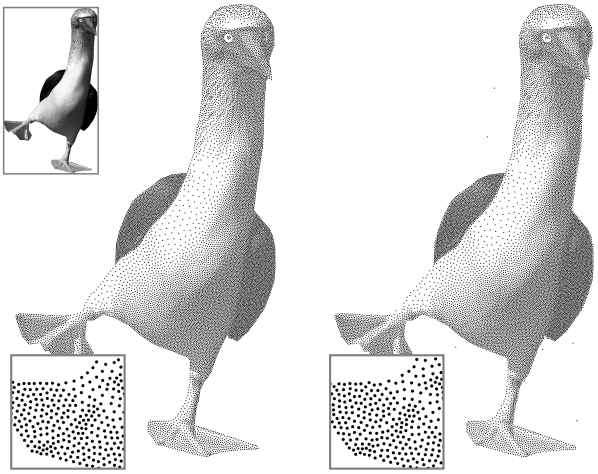


Fig. 11. A set of 2048 points optimized according to a configuration with two linear-ramp-function classes. At every point index we can split the set into two subsets with good-quality distribution each, for an effective number of 4096 optimization targets (i.e., subclasses). We show six such (color-coded) splits and the 2D Fourier power spectra of the two extracted subsets. The last row shows the radially averaged spectra of the subsets and the entire point set (in green).

using Paulin et al.'s method is ineffective since the post-optimization point warping to the unit square ruins the point-set properties.

Progressive sampling. Our multi-class formulation allows constructing progressive point patterns with controlled granularity. We can use a single, staircase-function class where the number of steps (i.e., subclasses) dictates the number of prefix subsets (i.e., progressive levels) to optimize for. A constant class function corresponds to optimizing only the full set of points for uniformity; in the other extreme of a linear-ramp class function every prefix of points is optimized. Figure 15 shows progressive error-convergence plots for 5 such variants using 16,384 points. The steps have equal lengths in power-of-2 scale. The 1-subclass red curve behaves almost like a random one for all sample counts except for the strong dip at the end. Only when all samples are used is the integration error low; in fact, this is the lowest error achieved by any point set in the plot. Increasing the number of subclasses increases the number of dips but also shortens each. This result clearly illustrates that finer progressive granularity comes at the cost of increased error due to the larger number of objectives the optimization needs to balance. In the extreme case of 16,384 subclasses, the point set is fully progressive and shows uniform error behavior.



BNOT [de Goes et al. 2012] **Ours FSOT**

Fig. 12. Comparison of monochrome image stippling using 15,000 points.

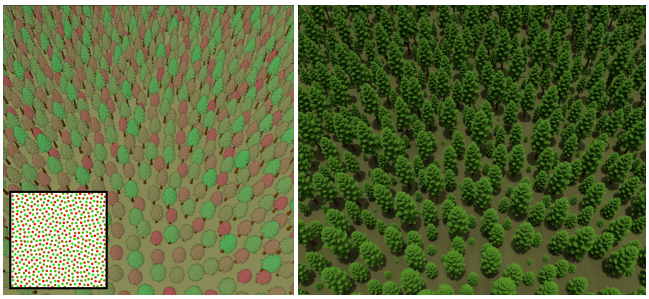


Fig. 13. Application of our continuous-class optimization to placement of objects with continuous variation in color (left) and size (right).

Rendering. For rendering applications, we optimize a point set covering 128×128 pixels that is toroidally tiled over the image. In Fig. 16, we compare our point sets against those from prior work on perceptual (i.e., blue-noise) error optimization [Ahmed and Wonka 2020; Belcour and Heitz 2021]; we use box reconstruction for a fair comparison. Both scenes are rendered with 1 sample/pixel under direct lighting. The benefit of our approach (rightmost column) is most visible in the top scene, where the specular regions show a much improved error distribution. In the bottom row scene, we use a finite-aperture camera. The zoom-ins show better perceived quality achieved by our method over the state of the art. We provide more comparisons on different scenes in the supplemental material.

While traditionally point sets are optimized assuming a box pixel-reconstruction kernel, our framework allows optimizing for arbitrary kernels. Figure 17 shows the impact on error distribution while taking into account the reconstruction kernel. On the right, note the substantial improvement in Fig. 17c over Fig. 17b, due to specially optimizing for the Gaussian reconstruction kernel used. Additionally accounting for perceptual blur further pushes the error distribution toward high frequencies (Fig. 17d). Another comparison against the method of Belcour and Heitz [2021] using box and Gaussian

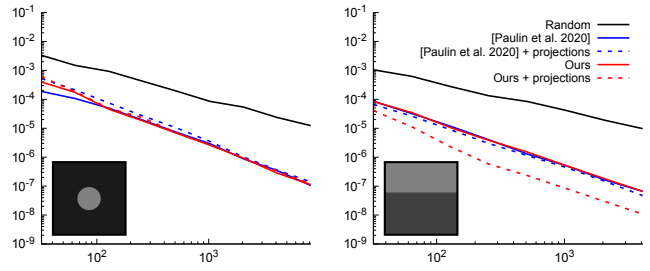


Fig. 14. Comparison of the Monte-Carlo variance convergence of our optimized point sets against those of Paulin et al. [2020]. We average variance over 10 realizations of each method and 40 variations of each function. Our axis-aligned projection prioritization is more effective than theirs.

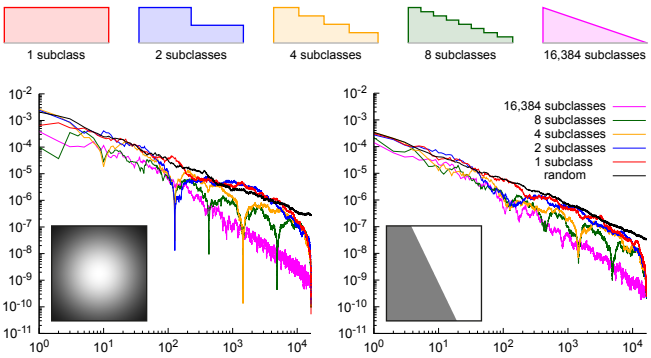


Fig. 15. Progressive point-set optimization using a single, staircase-function class. Increasing the number of steps (with equal lengths in power-of-2 scale) increases the number of prefix subsets to optimize; we show 5 examples. The graphs plot integration-error behavior with increasing number of points taken, up to 16,384, averaged over 30 integrand variations and 20 point-set realisations. We see that finer progressive granularity yields a larger number of error dips, but each is shorter. The fully progressive (pink) point set exhibits uniform error behavior.

pixel reconstruction can be found in the supplemental document. It shows the importance of optimization not only for perceptual blur but also for the pixel-reconstruction kernel.

7.3 Algorithmic complexity and performance

The bottleneck of our optimization is the sorting of m projected optimization points and $c \cdot m$ density-sample points (where $c = \text{const}$), with complexity $O(m \log(m))$ per iteration. The number of classes and subclasses has no direct impact on complexity, although in practice increasing the number of optimization objectives can impact the convergence speed of gradient descent. The memory consumption of our algorithm is linear in the total number of optimization points n.

For 4096 points, single-class GPU optimization takes 40 sec, 3-class takes 59 sec, and 7-class takes 71 sec. For 262,144 points, single-class takes 3840 sec (2000 iterations), 3-class takes 4325 sec (2500 iterations), and 7-class takes 4370 sec (3000 iterations). The added cost of increasing the number of classes is moderate. The reason is that, while more classes require more optimization iterations to obtain high quality, the time per iteration is lower as fewer points

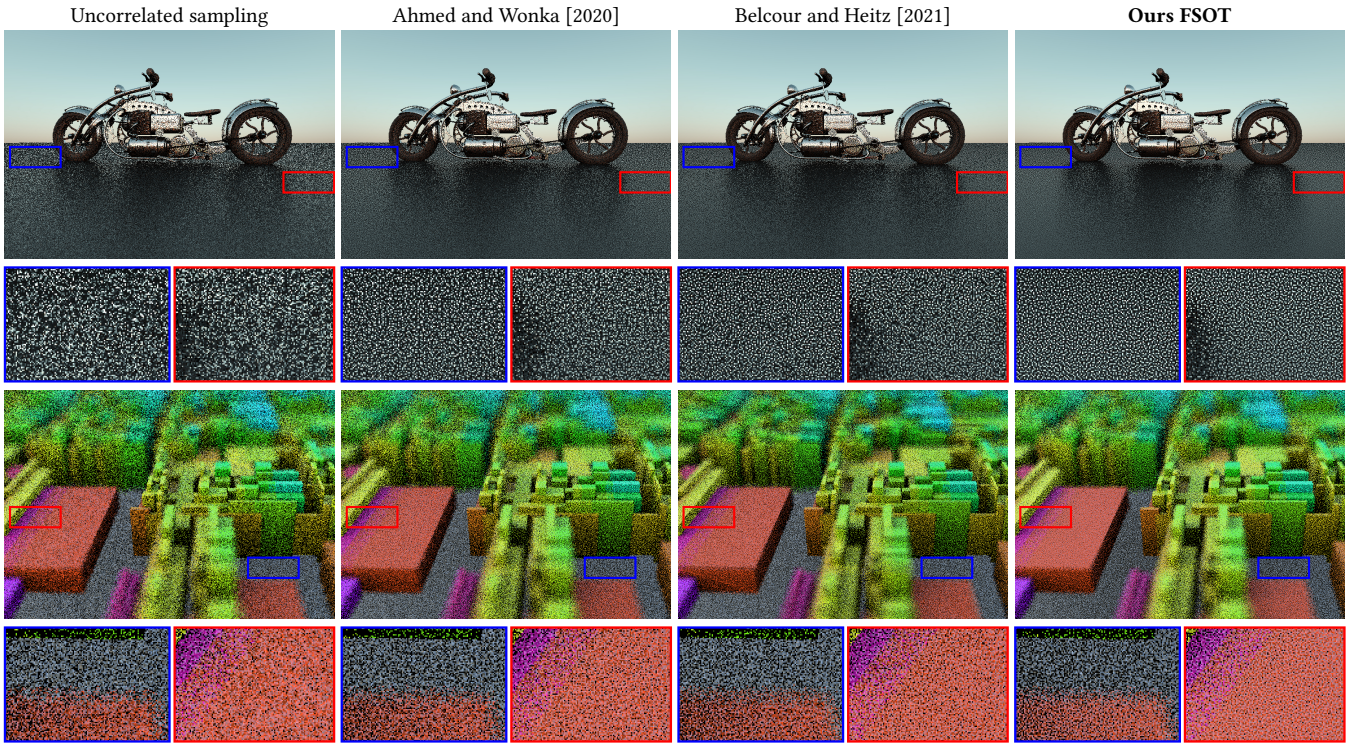


Fig. 16. Comparison of our perceptual-error optimization against classical uncorrelated pixel sampling and state-of-the-art blue-noise error distribution methods. The top scene is directly lit by an environment map, and the bottom scene has defocus blur that increases the sampling dimensions to four.

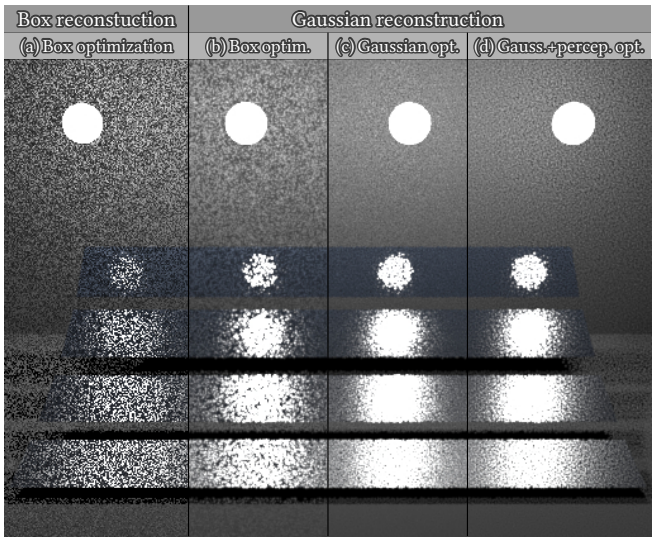


Fig. 17. Optimizing pixel samples for different reconstruction kernels. (a) When using box reconstruction, the samples for individual pixels can be optimized separately. (b) Traditionally this optimization is used also when the reconstruction is non-box. (c) Our framework allows optimizing for the specific kernel used, taking into account overlaps between pixels and showing substantial error reduction. (d) Additionally taking into account perceptual blur achieves blue-noise error distribution over the image..

are optimized at once (since one subclass it optimized per iteration). With this in mind, it is possible to imagine a more efficient optimization, e.g., utilizing a data structure to pre-order the points before projection and then using a sorting algorithm that takes advantage of this pre-ordering. One can also imagine relaxing the constraints on the Wasserstein equations to perform local rather than global optimizations. By computing several Wasserstein distances on subsets of the domain, it is possible to approximate the full distance with fewer points in each "sub-distance". Because of the complexity of these operations, reducing the number of points would speed up the computation at the cost of a looser error bound.

8 CONCLUSION

We develop a theoretical point optimization framework designed for handling large numbers of objectives. Specifying these objectives for a given application can be tedious if done manually. Prior methods [Wei 2010; Qin et al. 2017] have overlooked this issue as they target applications with fewer objectives.

We devise a principled framework for point optimization that can handle large numbers of objectives. We introduce the notion of a subclass which adds a level of granularity by specifying an objective for a subset of points in a class. Our framework scales to such a large number of objectives because, theoretically, the achievable quality does not depend on the number of objectives but on the amount of overlap between them and the difference in target distributions. The memory footprint of our stochastic gradient-descent optimization

is negligible as we optimize a single subclass per iteration. We demonstrate a variety of applications, also formalizing perceptual-error optimization as a multi-class problem.

Limitations. Our multi-class Wasserstein barycenter objective has a fully integral form, which allows us to leverage stochastic optimization and achieve scalability. However, optimizing for a single objective per iteration can yield noisy gradients and slow down convergence toward the sought barycenter. Our point-sampling of non-uniform distributions exacerbates the issue by adding more noise to the gradients.

Wherever functions of different classes overlap, points are implicitly optimized toward a barycenter of the corresponding target distributions. Some applications require a union of point subsets to follow a *mixture* of the targets instead. A notable example is color stippling where the base targets are the distributions of the individual color channels. Our framework requires specifying mixture targets explicitly via dedicated classes.

Future work. Our optimization can benefit from analytic target-distribution projection and informed choices of projection axes that allows tailoring application-specific samplers. A more advanced optimizer could achieve better local minima than stochastic gradient descent. While enabling efficient optimization, the sliced Wasserstein barycenter we use may not yield a good distribution interpolation [Bonneel et al. 2015; Bonneel and Pfister 2013]. Efficient optimization of the regular Wasserstein barycenter is an interesting direction for future investigation.

The Wasserstein distance provides a convenient integration-error bound as it is amenable to gradient-based minimization. However, the tightness of that bound is not well understood, especially in relation to the discrepancy-based bound given by the Koksma-Hlawka inequality. Exploring this relation could help better understand the optimization manifolds for future sampling patterns. Another interesting investigation would be the efficient minimization of discrepancy metrics.

ACKNOWLEDGMENTS

We thank all the anonymous reviewers for their helpful comments in shaping the final version of the paper. We thank the following for scenes used in our experiments: julioras3d (CHOPPER-TITAN), Mikael Hvidtfeldt Christensen (STRUCTURESYNTH), Greyscalegorilla (VW-VAN) and Eric Veach (VEACH-MIS). We also thanks Sponchia for the elephants image.

REFERENCES

Martial Agueh and Guillaume Carlier. 2011. Barycenters in the Wasserstein Space. *SIAM Journal on Mathematical Analysis* 43, 2 (2011), 904–924. https://doi.org/10.1137/100805741

Abdalla G. M. Ahmed and Peter Wonka. 2020. Screen-space blue-noise diffusion of Monte Carlo sampling error via hierarchical ordering of pixels. *ACM Trans. Graph.* 39, 6 (2020), 244:1–244:15. https://doi.org/10.1145/3414685.3417881

Abdalla G. M. Ahmed and Peter Wonka. 2021. Optimizing Dyadic Nets. *ACM Trans. Graph.* 40, 4, Article 141 (jul 2021), 17 pages. https://doi.org/10.1145/3450626.3459880

Michael Balzer, Thomas Schlömer, and Oliver Deussen. 2009. Capacity-Constrained Point Distributions: A Variant of Lloyd's Method. 28, 3, Article 86 (July 2009), 8 pages.

Laurent Belcour and Eric Heitz. 2021. Lessons Learned and Improvements When Building Screen-Space Samplers with Blue-Noise Error Distribution. In *ACM SIGGRAPH 2021 Talks* (Virtual Event, USA) *(SIGGRAPH '21)*. Association for Computing Machinery, New York, NY, USA, Article 9, 2 pages. https://doi.org/10.1145/3450623.3464645

Nicolas Bonneel and David Coeurjolly. 2019. SPOT: Sliced Partial Optimal Transport. *ACM Trans. Graph.* 38, 4, Article 89 (July 2019), 13 pages. https://doi.org/10.1145/3306346.3323021

Nicolas Bonneel and Hanspeter Pfister. 2013. *Sliced Wasserstein Barycenter of Multiple Densities.* Technical Report. Harvard Technical Report TR-02-13.

Nicolas Bonneel, Julien Rabin, Gabriel Peyré, and Hanspeter Pfister. 2015. Sliced and Radon Wasserstein Barycenters of Measures. *J. Math. Imaging Vis.* 51, 1 (2015). https://doi.org/10.1007/s10851-014-0506-3

Nicolas Bonnotte. 2013. *Unidimensional and evolution methods for optimal transportation.* Ph.D. Dissertation. Paris 11.

Léon Bottou. 1998. Online Algorithms and Stochastic Approximations. In *Online Learning and Neural Networks*, David Saad (Ed.). Cambridge University Press, Cambridge, UK.

Jiating Chen, Xiaoyin Ge, Li-Yi Wei, Bin Wang, Yusu Wang, Huamin Wang, Yun Fei, Kang-Lai Qian, Jun-Hai Yong, and Wenping Wang. 2013. Bilateral Blue Noise Sampling. *ACM Trans. Graph.* 32, 6, Article 216 (nov 2013), 11 pages. https://doi.org/10.1145/2508363.2508375

Zhonggui Chen, Zhan Yuan, Yi-King Choi, Ligang Liu, and Wenping Wang. 2012. Variational Blue Noise Sampling. *IEEE Transactions on Visualization and Computer Graphics* 18, 10 (2012), 1784–1796. https://doi.org/10.1109/TVCG.2012.94

Vassillen Chizhov, Iliyan Georgiev, Karol Myszkowski, and Gurprit Singh. 2022. Perceptual Error Optimization for Monte Carlo Rendering. *ACM Trans. Graph.* 41, 3, Article 26 (mar 2022), 17 pages. https://doi.org/10.1145/3504002

Sebastian Claici, Edward Chien, and Justin Solomon. 2018. Stochastic wasserstein barycenters. In *International Conference on Machine Learning*. PMLR, 999–1008.

Robert L. Cook. 1986. Stochastic sampling in computer graphics. 5, 1 (1986), 51–72.

Marco Cuturi. 2013. Sinkhorn Distances: Lightspeed Computation of Optimal Transport. In *Advances in Neural Information Processing Systems*, C. J. C. Burges, L. Bottou, M. Welling, Z. Ghahramani, and K. Q. Weinberger (Eds.), Vol. 26. Curran Associates, Inc. https://proceedings.neurips.cc/paper/2013/file/af21d0c97db2e27e13572cbf59eb343d-Paper.pdf

Fernando de Goes, Katherine Breeden, Victor Ostromoukhov, and Mathieu Desbrun. 2012. Blue Noise Through Optimal Transport. 31, 6, Article 171 (Nov. 2012), 11 pages.

Josef Dick and Friedrich Pillichshammer. 2010. *Digital Nets and Sequences: Discrepancy Theory and Quasi–Monte Carlo Integration.* Cambridge University Press. https://doi.org/10.1017/CBO9780511761188

Mark A. Z. Dippé and Erling Henry Wold. 1985. Antialiasing Through Stochastic Sampling. 19, 3 (July 1985), 69–78. https://doi.org/10.1145/325165.325182

Fredo Durand. 2011. *A frequency analysis of Monte-Carlo and other numerical integration schemes.* Technical Report MIT-CSAILTR-2011-052. CSAIL, MIT,.

Sergey Ermakov and Svetlana Leora. 2019. Monte Carlo Methods and the Koksma-Hlawka Inequality. *Mathematics* 7, 8 (2019). https://doi.org/10.3390/math7080725

Iliyan Georgiev and Marcos Fajardo. 2016. Blue-Noise Dithered Sampling. In *ACM SIGGRAPH 2016 Talks* (Anaheim, California) *(SIGGRAPH '16)*. Association for Computing Machinery, New York, NY, USA, Article 35, 1 pages. https://doi.org/10.1145/2897839.2927430

Alvaro J. González, Jan Bacca Rodríguez, Gonzalo R. Arce, and Daniel Leo Lau. 2006. Alpha stable human visual system models for digital halftoning. In *Electronic Imaging*.

Eric Heitz and Laurent Belcour. 2019. Distributing Monte Carlo Errors as a Blue Noise in Screen Space by Permuting Pixel Seeds Between Frames. *Computer Graphics Forum* (2019). https://doi.org/10.1111/cgf.13778

Eric Heitz, Laurent Belcour, Victor Ostromoukhov, David Coeurjolly, and Jean-Claude Iehl. 2019. A low-discrepancy sampler that distributes Monte Carlo errors as a blue noise in screen space. 1–2. https://doi.org/10.1145/3306307.3328191

Hu, Sha Ruizhen, van Kaick Tingkai, Deussen Oliver, Huang Oliver, and Hui. 2020. Data Sampling in Multi-view and Multi-class Scatterplots via Set Cover Optimization. *IEEE Transactions on Visualization and Computer Graphics (Proceedings of InfoVis 2019)* 26, 1 (2020), 739–748.

Min Jiang, Yahan Zhou, Rui Wang, Richard Southern, and Jian Jun Zhang. 2015. Blue Noise Sampling Using an SPH-Based Method. *ACM Trans. Graph.* 34, 6, Article 211 (oct 2015), 11 pages. https://doi.org/10.1145/2816795.2818102

Rabin Julien, Gabriel Peyré, Julie Delon, and Bernot Marc. 2011. Wasserstein Barycenter and its Application to Texture Mixing. In *SSVM'11*. Springer, Israel, 435–446. https://hal.archives-ouvertes.fr/hal-00476064

Leonid V. Kantorovich and Gennady S. Rubinstein. 1958. On a space of completely additive functions. *Vestnik Leningrad Univ 13* 7 (1958), 52–59.

Alexander Keller. 2013. Quasi-Monte Carlo Image Synthesis in a Nutshell. In *Monte Carlo and Quasi-Monte Carlo Methods 2012*, Josef Dick, Frances Y. Kuo, Gareth W. Peters, and Ian H. Sloan (Eds.). Springer Berlin Heidelberg, 213–249.

Johannes Kopf, Daniel Cohen-Or, Oliver Deussen, and Dani Lischinski. 2006. Recursive Wang tiles for real-time blue noise. *ACM Trans. Graph. (Proc. SIGGRAPH)* 25, 3 (2006).

Alexander Korotin, Daniil Selikhanovych, and Evgeny Burnaev. 2022. Neural optimal transport. *arXiv preprint arXiv:2201.12220* (2022).
Lauwerens Kuipers and Harald Niederreiter. 1974. *Uniform Distribution of Sequences*. Wiley, New York, USA.
C. Lemieux. 2009. *Monte Carlo and Quasi-Monte Carlo Sampling*. Springer New York. https://books.google.fr/books?id=wj5OyydZ5bkC
Don P. Mitchell. 1991. Spectrally Optimal Sampling for Distribution Ray Tracing. *SIGGRAPH Computer Graphics* 25, 4 (July 1991), 157–164.
R. Näsänen. 1984. Visibility of halftone dot textures. *IEEE Transactions on Systems, Man, and Cybernetics* SMC-14, 6 (1984), 920–924.
Harald Niederreiter. 1992. *Random Number Generation and quasi-Monte Carlo Methods*. Society for Industrial and Applied Mathematics, Philadelphia, PA, USA.
Yann Ollivier, Herve Pajot, and Cédric Villani (Eds.). 2014. *Optimal Transport - Theory and Applications*. London Mathematical Society lecture note series, Vol. 413. Cambridge University Press.
Christian van Onzenoodt, Gurprit Singh, Timo Ropinski, and Tobias Ritschel. 2021. Blue Noise Plots. *Computer Graphics Forum* (2021). https://doi.org/10.1111/cgf.142644
Victor Ostromoukhov. 2007. Sampling with Polyominoes. In *ACM SIGGRAPH 2007 Papers* (San Diego, California) *(SIGGRAPH '07)*. Association for Computing Machinery, New York, NY, USA, 78–es. https://doi.org/10.1145/1275808.1276475
Victor Ostromoukhov, Charles Donohue, and Pierre-Marc Jodoin. 2004. Fast Hierarchical Importance Sampling with Blue Noise Properties. 23, 3 (aug 2004), 488–495. https://doi.org/10.1145/1015706.1015750
Thrasyvoulos N. Pappas and David L. Neuhoff. 1999. Least-squares model-based halftoning. *IEEE Transactions on Image Processing* 8, 8 (Aug 1999), 1102–1116. https://doi.org/10.1109/83.777090
Lois Paulin, Nicolas Bonneel, David Coeurjolly, Jean-Claude Iehl, Antoine Webanck, Mathieu Desbrun, and Victor Ostromoukhov. 2020. Sliced Optimal Transport Sampling. *ACM Trans. Graph.* 39, 4, Article 99 (July 2020), 17 pages. https://doi.org/10.1145/3386569.3392395
Gabriel Peyré and Marco Cuturi. 2018. Computational Optimal Transport. (2018). https://doi.org/10.48550/ARXIV.1803.00567
Matt Pharr, Wenzel Jakob, and Greg Humphreys. 2016. *Physically based rendering: From theory to implementation*. Morgan Kaufmann.
Adrien Pilleboue, Gurprit Singh, David Coeurjolly, Michael Kazhdan, and Victor Ostromoukhov. 2015. Variance Analysis for Monte Carlo Integration. 34, 4, Article 124 (July 2015), 14 pages.
F. Pitie, A.C. Kokaram, and R. Dahyot. 2005. N-dimensional probability density function transfer and its application to color transfer. In *Tenth IEEE International Conference on Computer Vision (ICCV'05) Volume 1*, Vol. 2. 1434–1439 Vol. 2. https://doi.org/10.1109/ICCV.2005.166
François Pitié, Anil C. Kokaram, and Rozenn Dahyot. 2005. N-Dimensional Probablility Density Function Transfer and its Application to Colour Transfer. In *10th IEEE International Conference on Computer Vision*. IEEE Computer Society. https://doi.org/10.1109/ICCV.2005.166
Hongxing Qin, Yi Chen, Jinlong He, and Baoquan Chen. 2017. Wasserstein Blue Noise Sampling. *ACM Trans. Graph.* 36, 5, Article 168 (Oct. 2017), 13 pages. https://doi.org/10.1145/3119910
Julien Rabin, Gabriel Peyré, Julie Delon, and Marc Bernot. 2011. Wasserstein Barycenter and Its Application to Texture Mixing. In *Scale Space and Variational Methods in Computer Vision - Third International Conference, SSVM 2011, Ein-Gedi, Israel, May 29 - June 2, 2011, Revised Selected Papers (Lecture Notes in Computer Science, Vol. 6667)*, Alfred M. Bruckstein, Bart M. ter Haar Romeny, Alexander M. Bronstein, and Michael M. Bronstein (Eds.). https://doi.org/10.1007/978-3-642-24785-9_37
Svetlozar Rachev and Ludger Rüschendorf. 1998. *Mass Transportation Problems: Volume I: Theory*. Springer.
Bernhard Reinert, Tobias Ritschel, Hans-Peter Seidel, and Iliyan Georgiev. 2016. Projective Blue-Noise Sampling. *Comp. Graph. Forum* 35, 1 (2016).
F. Santambrogio. 2015. *Optimal Transport for Applied Mathematicians: Calculus of Variations, PDEs, and Modeling*. Springer International Publishing.
C. Schmaltz, P. Gwosdek, and J. Weickert. 2012. Multi-Class Anisotropic Electrostatic Halftoning. *Comput. Graph. Forum* 31, 6 (sep 2012), 1924–1935. https://doi.org/10.1111/j.1467-8659.2012.03072.x
Christoph Schulz, Kin Chung Kwan, Michael Becher, Daniel Baumgartner, Guido Reina, Oliver Deussen, and Daniel Weiskopf. 2021. Multi-Class Inverted Stippling. *ACM Trans. Graph.* 40, 6, Article 245 (dec 2021), 12 pages. https://doi.org/10.1145/3478513.3480534
Adrian Secord. 2002. Weighted Voronoi stippling. In *Proc. NPAR*.
Gurprit Singh and Wojciech Jarosz. 2017. Convergence Analysis for Anisotropic Monte Carlo Sampling Spectra. 36, 4, Article 137 (July 2017), 14 pages. https://doi.org/10.1145/3072959.3073656
Gurprit Singh, Cengiz Öztireli, Abdalla G.M. Ahmed, David Coeurjolly, Kartic Subr, Oliver Deussen, Victor Ostromoukhov, Ravi Ramamoorthi, and Wojciech Jarosz. 2019. Analysis of Sample Correlations for Monte Carlo Rendering. *Computer Graphics Forum* 38, 2 (2019), 473–491. https://doi.org/10.1111/cgf.13653 arXiv:https://onlinelibrary.wiley.com/doi/pdf/10.1111/cgf.13653
Kartic Subr and Jan Kautz. 2013. Fourier Analysis of Stochastic Sampling Strategies for Assessing Bias and Variance in Integration. 32, 4, Article 128 (July 2013), 12 pages.
Robert Ulichney. 1987. *Digital Halftoning*. MIT Press.
C. Villani. 2008. *Optimal Transport: Old and New*. Springer Berlin Heidelberg. https://books.google.fr/books?id=hV8o5R7_5tkC
Florent Wachtel, Adrien Pilleboue, David Coeurjolly, Katherine Breeden, Gurprit Singh, Gaël Cathelin, Fernando de Goes, Mathieu Desbrun, and Victor Ostromoukhov. 2014. Fast Tile-based Adaptive Sampling with User-specified Fourier Spectra. 33, 4, Article 56 (July 2014), 11 pages.
Muge Wang and Kevin J. Parker. 1999. Properties of combined blue noise patterns. *Proceedings 1999 International Conference on Image Processing (Cat. 99CH36348)* 4 (1999), 328–332 vol.4.
Li-Yi Wei. 2010. Multi-Class Blue Noise Sampling. *ACM Trans. Graph.* 29, 4, Article 79 (July 2010), 8 pages. https://doi.org/10.1145/1778765.1778816

A 1D WASSERSTEIN DISTANCE DERIVATIVE

Here we derive the derivative of the 1D Wasserstein distance which has an analytic solution [Rachev and Rüschendorf 1998]:

$$W_p^p(\nu,\mu) \;=\; \int_0^\infty \left| F_\nu^{-1}(x) - F_\mu^{-1}(x) \right|^p \mathrm{d}x, \tag{12}$$

where F_ν^{-1} and F_μ^{-1} are the measures' inverse cumulative distribution functions (CDFs). We are specifically interested in the case where $p = 2$ and one of the measures represents a 1D point set $X = \{x_i\}_{i=1}^n$. For this case we have

$$W_2^2(X,\mu) = \int_0^1 \left[F_X^{-1}(x) - F_\mu^{-1}(x)\right]^2 \mathrm{d}x = \sum_{i=1}^n \int_{\frac{i-1}{n}}^{\frac{i}{n}} \left(x_i - F_\mu^{-1}(x)\right)^2 \mathrm{d}x.$$

We want to differentiate this distance w.r.t. every point x_i. Only one of the integrals depends on each x_i, making the differentiation of its convex integrand easy:

$$\begin{aligned}\frac{\mathrm{d}}{\mathrm{d}x_i} W_2^2(X,\mu) &= \int_{\frac{i-1}{n}}^{\frac{i}{n}} \frac{\mathrm{d}}{\mathrm{d}x_i}\left(x_i^2 - 2x_i F_\mu^{-1}(x) + (F_\mu^{-1}(x))^2\right) \mathrm{d}x \qquad (13)\\ &= \int_{\frac{i-1}{n}}^{\frac{i}{n}} 2(x_i - F_\mu^{-1}(x))\,\mathrm{d}x = 2\frac{x_i}{n} - 2\int_{\frac{i-1}{n}}^{\frac{i}{n}} F_\mu^{-1}(x)\mathrm{d}x.\end{aligned}$$

The integral of the inverse target CDF is simply the average point inside the i^{th} region of the target density with mass $1/n$, multiplied by $1/n$. The resulting derivative is thus similar to offset used by Paulin et al. [2020]; the difference is the above scaling factor of $2/n$ and that they take the median target point (instead of the mean).

B GRADIENT ESTIMATION AND POINT OFFSETS

The 1D optimization step involves offsetting each projected point x_i^θ along the (negative) derivative of the Wasserstein distance w.r.t. the point's position:

$$x_i^\theta \;=\; x_i^\theta - \eta \cdot \gamma_i^\theta \cdot \underbrace{\frac{\mathrm{d}}{\mathrm{d}x_i} W_p^p\left(\overline{X}_{w>z}^\theta, \mu_{w>z}^\theta\right)}_{\Delta_i^\theta}, \tag{14}$$

where η is a step-size parameter (a.k.a. learning rate) and γ the offset scaling factor (Section 5). In Appendix A above we derive the derivative for the semi-discrete 2-Wasserstein distance for the case where both distributions are normalized. In our case they have reduced mass due to filtering, which can be compensated by simply scaling the derivative by the relative number of selected optimization

points $m/n = \left|\overline{X}_{w>z}\right|/n$:

$$\Delta_i^\theta \;=\; \frac{m}{n}\left[2\frac{x_i^\theta}{m} - 2\int_{\frac{i-1}{m}}^{\frac{i}{m}} F^{-1}_{\mu^\theta}(x)\mathrm{d}x\right], \tag{15}$$

where the semi-discrete derivative in the parentheses is computed w.r.t. normalized distributions.

Numerical gradient estimation. The partial derivative step (14) requires computing the inverse CDF of the projected target distribution μ^θ. In practice, we use $c \times n$ points to better approximate the target distribution. First, all points are uniformly binned in n bins. The inverse CDF then adaptively changes the bin length according to the target distribution.

The integral term in Eq. (15) corresponds to the average location of the points within a certain interval of the inverse CDF: $b_i^\theta = {}^1\!/_c \sum_{j=(i-1)c}^{ic} y_j^\theta$, where the projected target samples y_j^θ are sorted. This gives the *average* location per i-th bin. Computing the offset Δ_i^θ then involves sorting x_i^θ and pairwise matching them with the bin values b_i^θ.

Gradient scaling factor. The scaling factor γ_i (14) is simply the relative change in the length of the i-th bin:

$$\gamma_i^\theta \;=\; \frac{\text{Average bin length}}{\text{Length of bin } i} = \frac{(F^{-1}_{\mu^\theta}(1) - F^{-1}_{\mu^\theta}(0))/m}{F^{-1}_{\mu^\theta}(i/m) - F^{-1}_{\mu^\theta}((i-1)/m)} \tag{16}$$

This scaling factor helps avoid the alignments shown in Fig. 7c by scaling the gradients (offsets) for the projected non-uniform target density.

C WASSERSTEIN INTEGRATION-ERROR BOUND

Here we provide a derivation of the integration error bound shown by Paulin et al. [2020]. Consider a continuous function $f : \mathcal{H} \to \mathbb{R}^+$ on the hypercube $\mathcal{H}$ with Lipschitz constant L_f such that, $\forall x, y \in \mathcal{H}$,

$$|f(x) - f(y)| \le L_f \|x - y\|. \tag{17}$$

Let $\gamma \in \Gamma(\nu, \mu)$ be a joint measure whose marginals ν and μ are measures on the unit hypercube $\mathcal{H}$. Integrating both sides of Eq. (17) w.r.t. γ, and then using $\left|\int g\right| \le \int |g|$, yields

$$\int_{\mathcal{H}^2} |f(x) - f(y)|\,\mathrm{d}\gamma(x,y) \le L_f \int_{\mathcal{H}^2} \|x - y\|\,\mathrm{d}\gamma(x,y) \tag{18}$$

$$\left|\int_{\mathcal{H}^2} \big[f(x) - f(y)\big]\,\mathrm{d}\gamma(x,y)\right| \le L_f \int_{\mathcal{H}^2} \|x - y\|\,\mathrm{d}\gamma(x,y). \tag{19}$$

We expand the left side of Eq. (19) into two integrals and simplify each by marginalizing the product measure; the bound on the right is tightened by taking the infimum over all valid joint measures γ:

$$\left|\int_{\mathcal{H}} f(x)\,\mathrm{d}\nu(x) - \int_{\mathcal{H}} f(x)\,\mathrm{d}\mu(x)\right| \le L_f \underbrace{\inf_{\gamma\in\Gamma(\nu,\mu)} \int_{\mathcal{H}^2} \|x - y\|\,\mathrm{d}\gamma(x,y)}_{W(\nu,\mu)}, \tag{20}$$

where $W(\nu, \mu)$ is the Wasserstein distance between ν and μ. The resulting inequality provides a numerical integration bound when ν is a Dirac point-mass measure, i.e., a point set.

D RECONSTRUCTION-ERROR BOUND

We build on Appendix C to derive an error bound for integrands of the form $w(x)f(x)$, where w is an analytically known function. As in Appendix C, our derivations use general probability measures ν and μ, but for our application we are specifically interested in the case where ν is a Dirac point-mass measure, i.e., a point set.

We begin by expressing $w(x)$ and $w(y)$ in the error as integrals over corresponding indicator functions, then swap the integration order using Fubini's theorem:

$$\left|\int_{\mathcal{H}} w(x)f(x)\,\mathrm{d}\nu(x) - \int_{\mathcal{H}} w(y)f(y)\,\mathrm{d}\mu(y)\right| \tag{21}$$

$$= \left|\int_{\mathcal{H}} \underbrace{\int_{\mathbb{R}} \mathbf{1}_{[0,w(x)]}(z)\mathrm{d}z}_{w(x)} f(x)\,\mathrm{d}\nu(x) - \int_{\mathcal{H}} \underbrace{\int_{\mathbb{R}} \mathbf{1}_{[0,w(y)]}(z)\mathrm{d}z}_{w(y)} f(y)\,\mathrm{d}\mu(y)\right|$$

$$= \left|\int_{\mathbb{R}} \left[\int_{\mathcal{H}} \mathbf{1}_{[0,w(x)]}(z)f(x)\,\mathrm{d}\nu(x) - \int_{\mathcal{H}} \mathbf{1}_{[0,w(y)]}(z)f(y)\,\mathrm{d}\mu(y)\right]\mathrm{d}z\right|. \tag{22}$$

Next, note that due the following identity for any $x \in \mathcal{H}$ and $z \in \mathbb{R}$:

$$\mathbf{1}_{[0,w(x)]}(z) = \mathbf{1}_{[z,\infty]}(w(x)) = \mathbf{1}_{\{x'\in\mathcal{H} | w(x')>z\}}(x) =: \mathbf{1}_{\mathcal{H}_{w>z}}(x), \tag{23}$$

the indicator function effectively restricts the integration to the region $\mathcal{H}_{w>z}$ where $w(\cdot) > z$. Plugging this identity into Eq. (22) and then using $\left|\int g\right| \le \int |g|$, we get

$$= \left|\int_{\mathbb{R}} \left[\int_{\mathcal{H}} \mathbf{1}_{\mathcal{H}_{w>z}}(x)f(x)\,\mathrm{d}\nu(x) - \int_{\mathcal{H}} \mathbf{1}_{\mathcal{H}_{w>z}}(y)f(y)\,\mathrm{d}\mu(y)\right]\mathrm{d}z\right| \tag{24}$$

$$\le \int_{\mathbb{R}} \left|\int_{\mathcal{H}} \mathbf{1}_{\mathcal{H}_{w>z}}(x)f(x)\,\mathrm{d}\nu(x) - \int_{\mathcal{H}} \mathbf{1}_{\mathcal{H}_{w>z}}(y)f(y)\,\mathrm{d}\mu(y)\right|\mathrm{d}z \tag{25}$$

$$= \int_{\mathbb{R}} \left|\int_{\mathcal{H}} f(x)\,\mathrm{d}\nu_{w>z}(x) - \int_{\mathcal{H}} f(y)\,\mathrm{d}\mu_{w>z}(y)\right|\mathrm{d}z, \tag{26}$$

where $\nu_{w>z}$ and $\mu_{w>z}$ are the measures ν and μ restricted to the region $\mathcal{H}_{w>z}$. We can now apply Eq. (20) to the absolute error in the outer integral, obtaining a bound for the expression in Eq. (21):

$$\left|\int_{\mathcal{H}} w(x)f(x)\mathrm{d}\nu(x) - \int_{\mathcal{H}} w(y)f(y)\mathrm{d}\mu(y)\right| \le L_f \int_{\mathbb{R}} W(\nu_{w>z},\mu_{w>z})\mathrm{d}z. \tag{27}$$

It is important to note that for the Wasserstein distance to work, the measures $\nu_{w>z}$ and $\mu_{w>z}$ must have equal masses in the hypercube subset corresponding to each valid slicing $w > z$. In other words, we need $\nu(\mathcal{H}_{w>z}) = \mu(\mathcal{H}_{w>z})$, or equivalently, $\nu_{w>z}(\mathcal{H}) = \mu_{w>z}(\mathcal{H})$, for all $z \in [0, \max w(.)]$.