Eurographics Symposium on Rendering (2022), pp. 1–1
A. Ghosh and L.-Y. Wei (Guest Editors)

Controlling Material Appearance by Examples — Supplemental Material

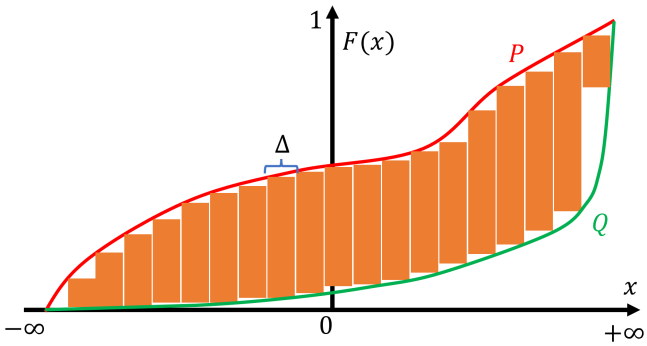

Figure 1: Practically, we should numerically evalute discrete CDF of P and Q from their samples and compute the integral of their difference (orange regions).

1. Sliced Cramér Loss

In our main paper, we introduce a resampling method to compute the Sliced Wasserstein loss between different number of samples. However, this loss can be accurately computed without resampling. We propose a sliced Cramér loss based on 1D Cramér distance [Cra28]. Given two distributions P and Q over R, their Cumulative Distribution Functions (CDFs) are respectively F_P and F_Q. The p-Cramér distance between P and Q is defined as :

$$l_p(P,Q) = \left(\int_{-\infty}^{+\infty} (F_P(x) - F_Q(x))^p \, dx \right)^{\frac{1}{p}} \tag{1}$$

when $p = 1$, the Cramér distance is equivalent to the first order Wasserstein distance. Practically, we have sample points drawn from P and Q (pixels on labeled regions in our case). We need to construct their empirical CDFs and numerically compute the integral shown in Fig. 1. In Listing 1, we show a Pytorch implementation, based on sorting with a time complexity of $O((M+N)log(M+N))$ where M and N are numbers of sample points for P and Q respectively. The implementation supports auto-differentiation with batch training.

Theoretically, a sliced Cramér loss should be a more precise comparison of two distributions than resampled Sliced Wasserstein. However, while evaluating the Cramér loss, we do not observe significant improvement over our resampling strategy. As the Cramér loss requires an additional sorting operation (line 10 in Listing 1), its evaluation is slightly slower. In Fig. 2, we show a few results generated using the sliced Wasserstein loss with resampling and the sliced Cramér loss respectively.

```
1  import torch as T
2
3  def cramer_loss(u, v):
4      # u is an array with shape [B, M] and v is an
        array with shape [B, N];
5      # B is batch size, M and N are number of
       sample points
6
7      # compute empirical CDFs of u and v
8      u_ = T.sort(u, dim=1)
9      v_ = T.sort(v, dim=1)
10     w = T.cat((u, v), dim=1).sort(dim=1)
11     u_cdf = T.searchsorted(u_, w[:, :-1], right=
       True)
12     v_cdf = T.searchsorted(v_, w[:, :-1], right=
       True)
13     u_cdf = u_cdf / u.shape[1]
14     v_cdf = v_cdf / v.shape[1]
15
16     # compute delta
17     deltas = torch.diff(w, dim=1)
18
19     # compute integral
20     return ((u_cdf - v_cdf) * deltas).abs().sum(
       dim=1).mean()
```

Listing 1: Implementation of 1D Cramér loss

References

[Cra28] CRAMÉR H.: *On the composition of elementary errors: Statistical applications*. Almqvist and Wiksell, 1928. 1