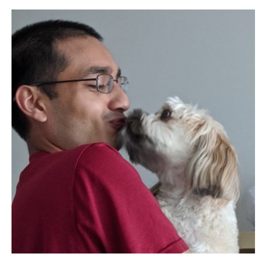
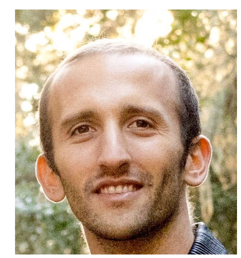
Fast Approximate Algorithms for Chamfer Distance

NeurIPS 2023, to appear

Ainesh Bakshi	Piotr Indyk	Raj Jayaram	Sandeep Silwal	Erik Waingarten
FODSI MIT	FODSI MIT	Google	MIT	U Penn



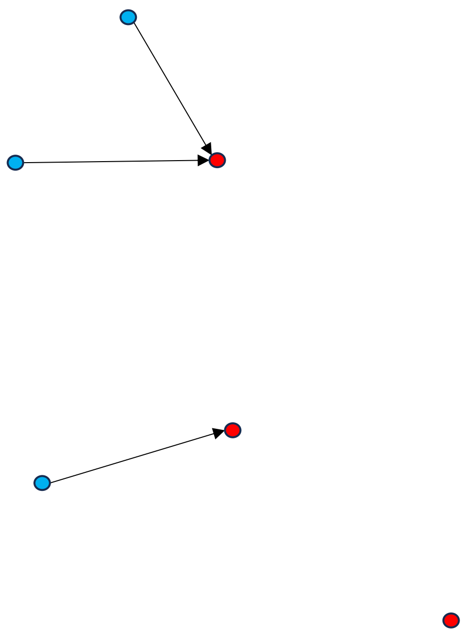
What is Chamfer distance?

- A distance between two point clouds A and B:

$$\text{CD}(A,B)=\sum_{a\in A} min_{b\in B}\ dist(a,b)$$

where dist(a,b) is e.g., the Euclidean distance

- Not a metric:
  - Not symmetric
    - Typically addressed by taking CD(A,B)+CD(B,A)
  - No triangle inequality
    - Typically addressed by not worrying about it



Chamfer distance = Relaxed Earth-Mover Distance

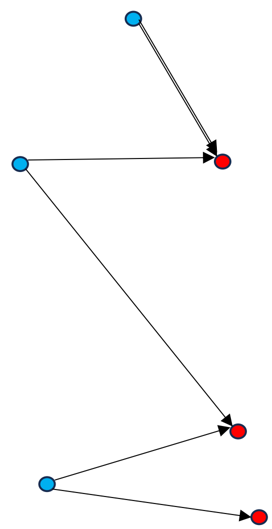
- Alternative definition of Chamfer distance:

  $$\text{CD(A,B)}=\min_{f:A\to B} \sum_{a\in A} dist(a,f(b))$$

- Earth-Mover Distance*:

  $$\text{EMD(A,B)}=\min_{f:A\xrightarrow{1:1}B} \sum_{a\in A} dist(a,f(b))$$

- CD is computationally more efficient than EMD
  - Frequently used as a cheaper proxy for EMD
  - "Relaxed EMD" (Kusner et al'15, Atasu et al'19)

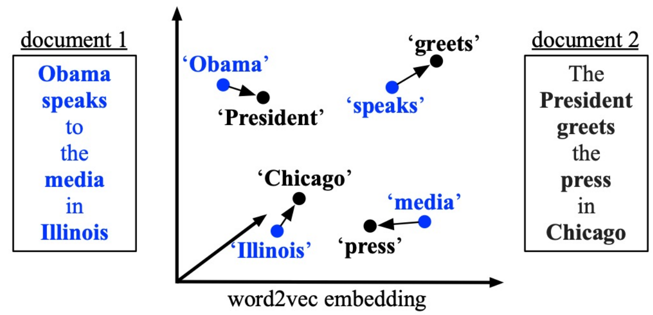


*A.k.a. Wasserstein distance, Mallows distance, optimal transport distance

Chamfer distance: applications

- Distance between shapes (in 2D, 3D)

- Distance between bags of words (in high D)



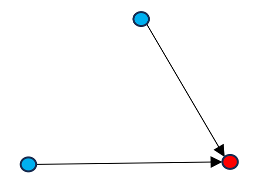
- Loss function for deep learning (as above)

- Implemented in multiple libraries

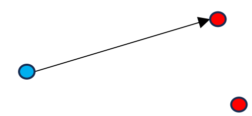
How quickly can we compute CD(A,B) ?

- Recall

$$CD(A,B)=\sum_{a\in A} \min_{b\in B} \; dist(a,b)$$



- Assume $A,B \subseteq R^d$, $|A|=|B|=n$, dist=Euclidean distance
- Naive algorithm: dn^2
- Accelerated algorithm: n nearest neighbor queries
[Sudderth-Mandel-Freeman-Willsky'04]
  - (1+ε)-approximate, low d: $n\,(1/\varepsilon)^{d/2}\log n$ [Clarkson'94]
  - (1+ε)-approximate, high d: $O\sim(dn^{1+1/(2(1+\varepsilon)^2-1)})$ [Andoni-Razenshteyn'15]



Our results

- Best prior algorithms: $n\,(1/\varepsilon)^{O(d)} \log n$, $dn^{1+1/2(1+\varepsilon)^2-1}$
- **Our result I:** can $(1+\varepsilon)$-approximate the value of CD(A,B) in time

$$d/\varepsilon^2 \; n \; \log n$$

  - Easily parallelizable, "clean"
  - Empirically fast

- **Our result II:** such a running time is impossible to achieve if we want to output a $(1+\varepsilon)$-approximate mapping $f: A \rightarrow B$
  - Assuming Hitting Set Conjecture

- **Intuition:** Our algorithm computes f(a) for only a small sample of as from A

Algorithm

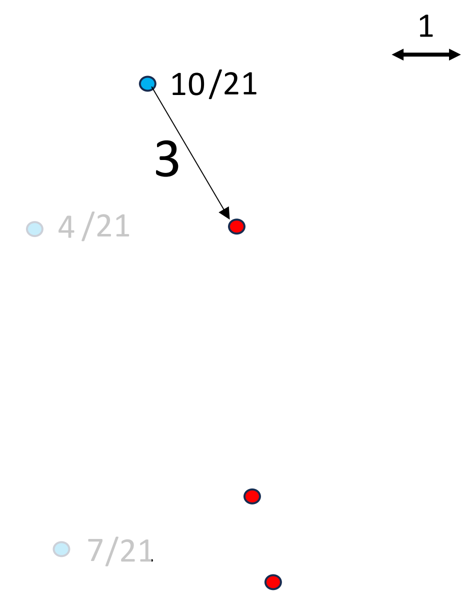
1. Execute CrudeNN(A, B), which for each $a \in A$ outputs D_a such that
   - $D_a \geqslant \min_{b \in B} dist(a,b)$, and
   - $D = \sum_{a \in A} D_a = O(\log n)\ CD(A,B)$
2. Construct a probability distribution, supported on the set A, such that for each $a \in A$,

$$Pr[x=a] = D_a/D$$

3. Let $T = O(1/\varepsilon^2 \log n)$. For $i=1\ldots T$, sample a_i and compute

$$\eta_i := \min_{b \in B} dist(a,b)$$

4. Output $|A|/T\ \sum_i \eta_i\ D/D_{ai}$



Output = 3* 21/10 = 6.3

Truth = 3+4+4 = 11

Analysis

Time ?

1. Execute CrudeNN(A, B), which for each $a \in A$ outputs D_a such that
   - $D_a \geqslant \min_{b \in B} dist(a,b)$, and
   - $D = \sum_{a \in A} D_a = O(\log n)\ CD(A,B)$

dn log n

2. Construct a probability distribution, supported on the set A, such that for each $a \in A$,

$$Pr[x=a] = D_a/D$$

3. Let $T = O(1/\varepsilon^2 \log n)$. For $i=1\ldots T$, sample a_i and compute

$$\eta_i := \min_{b \in B} dist(a,b)\ \ D/D_a$$

4. Output $|A|/T\ \sum_i \eta_i$

dn/ε^2 log n

- **Correctness ?** $E[\eta_i] = CD(A,B)/|A|$

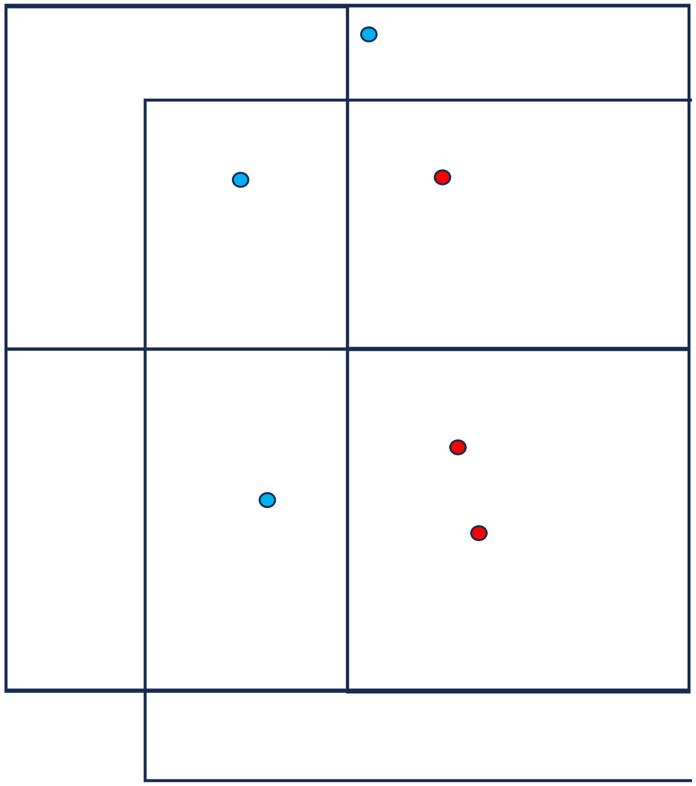
  Variance can be bounded as well

CrudeNN(A,B) (described for $dist(a,b)=||a-b||_1$)

- Goal: For each $a \in A$ output D_a such that:
  - $D_a \geqslant \min_{b \in B} dist(a,b)$, and
  - $D = \sum_{a \in A} D_a = O(\log n) \; CD(A,B)$
- One way to achieve this:
  - Build a $O(\log n)$-approximate NN data structure for B
  - For each $a \in A$, query the data structure; D_a = distance from a to returned point
  - Query time $O^{\sim}(dn^{1/c})$ for $c=O(\log n)$, but $O^{\sim}()$ hides some $\log n$ factors
- We go back to "first principles" instead…
- …and obtain a weaker guarantee:
  - The **expectation** of D is $O(\log n)$ $CD(A,B)$

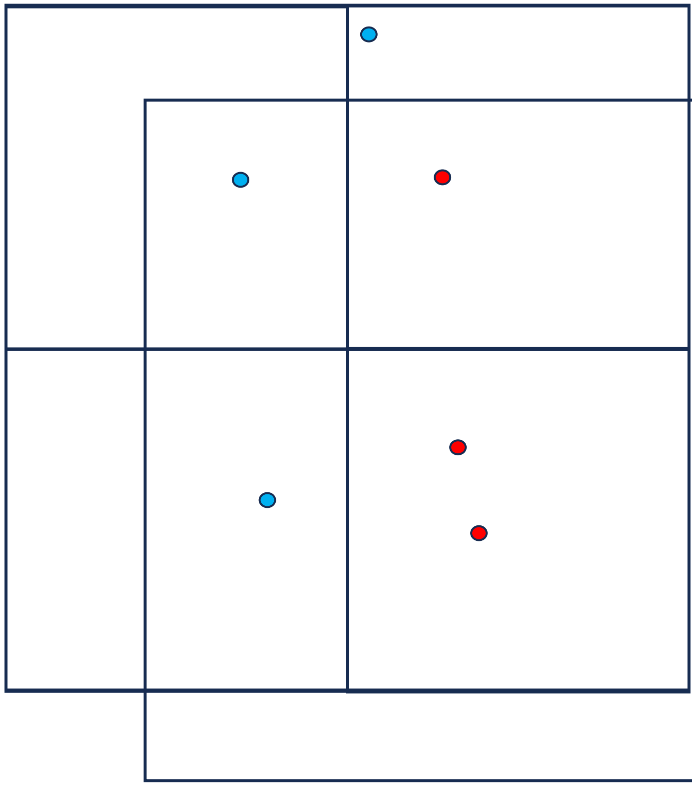
CrudeNN(A,B) (described for $dist(a,b)=||a-b||_1$)

- Similar to embedding into HSTs [Bartal'96]:
  - Build a quadtree* for B
  - For each $a \in A$, find the lowest level such that a's cell contains a point $b \in B$. Set $D_a=dist(a,b)$.
- One difference: each level is **independently** shifted by a random translation
- Not a tree, but the algorithm still well-defined

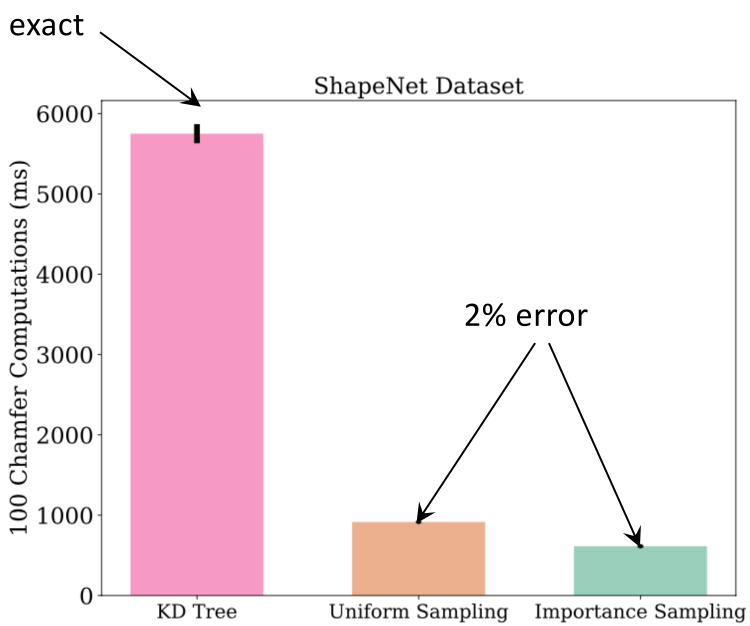


CrudeNN(A,B)

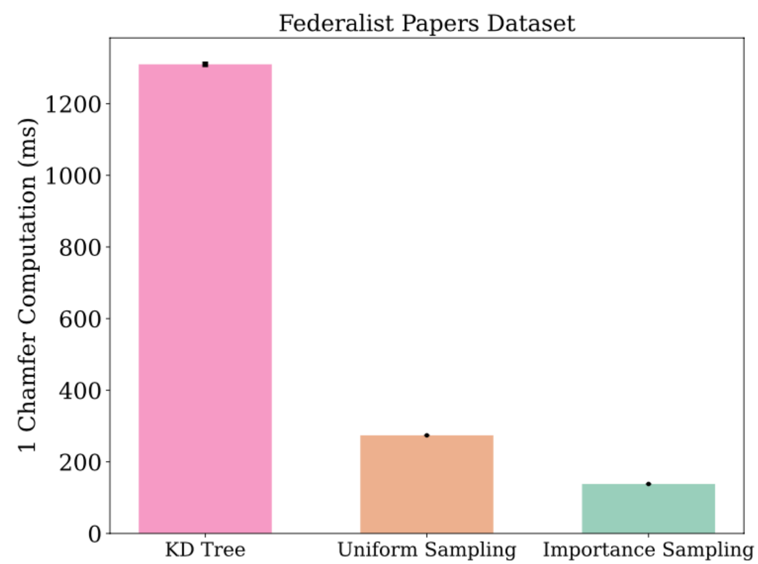
- For each $a \in A$, find the lowest level such that a's cell contains a point $b \in B$. Set $D_a = dist(a,b)$.
- Argument intuition:
  - Consider a level of "scale" r; let $h_r(x)$ be the grid cell containing x. We have:
    - $Pr[h_r(x) \neq h_r(y)] < ||x-y||_1/r$ (scale>>distance)
    - $Pr[h_r(x)=h_r(y)] < \exp(-||x-y||_1/r)$ (distance >>scale)
  - Let b be the NN of a in B
  - "Typically", for $r=O(||a-b||_1)$ we have:
    - $h_r(a) = h_r(b)$
    - $h_{r'}(a) \neq h_{r'}(b)$ for all smaller scales $r' < r$ and all b' such that $||a-b'||_1 > \log n \, ||a-b||_1$
  - But we also need to consider "untypical" cases where $h_r(a) \neq h_r(b)$ for $r=O(||a-b||_1)$
  - This is where the independence of the levels helps



Sample Experiments



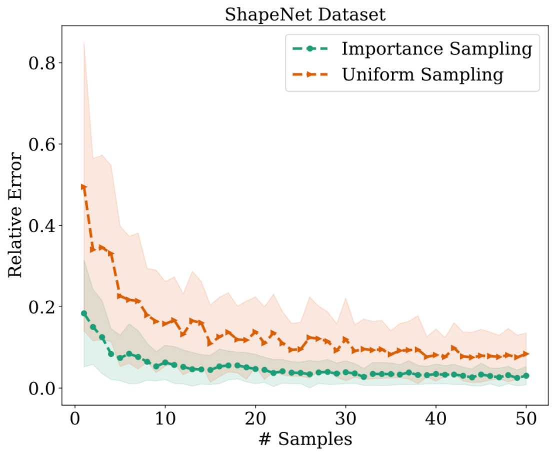
(a) ShapeNet 3D



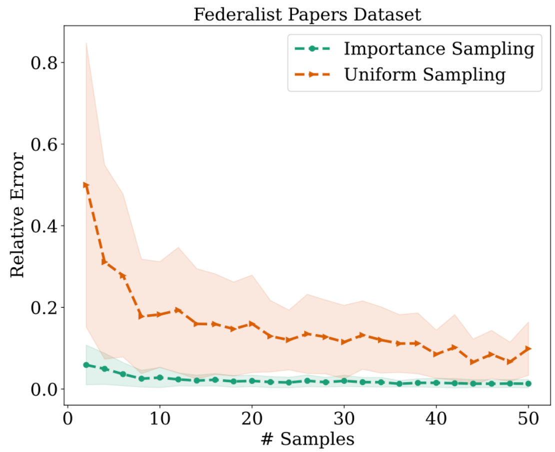
(b) Federalist Papers high D

Our algorithm is fast, accurate and robust (provably)

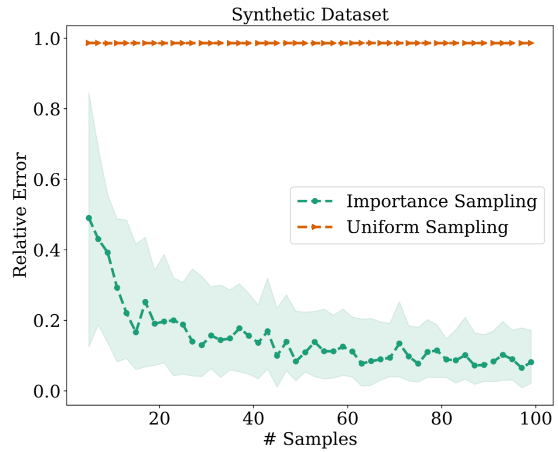
Uniform vs. Importance sampling



(a) ShapeNet



(b) Federalist Papers



(c) Gaussian Points

Conclusions

- Fast algorithm for Chamfer distance
- Generalizes to weighted pointsets, other *dist*(.,.), etc
- Could be the algorithm of choice for comparing point clouds