Efficient Graph Field Integrators Meet Point Clouds

Krzysztof Choromanski, Arijit Sehanobish*, Han Lin*, Yunfan Zhao*, Eli Berger, Tetiana Parshakova, Alvin Pan, David Watkins, Tianyi Zhang, Valerii Likhosherstov, Somnath Basu Roy Chowdhury, Kumar Avinava Dubey, Deepali Jain, Tamas Sarlos, Snigdha Chaturvedi, Adrian Weller*

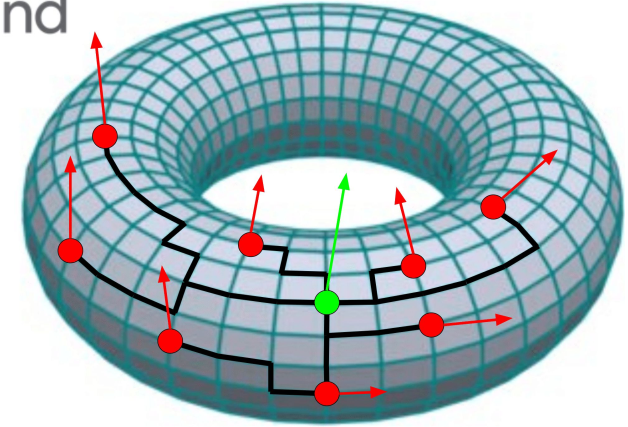
Google DeepMind

Columbia University

University of Haifa

Google Research

University of North Carolina Chapel Hill



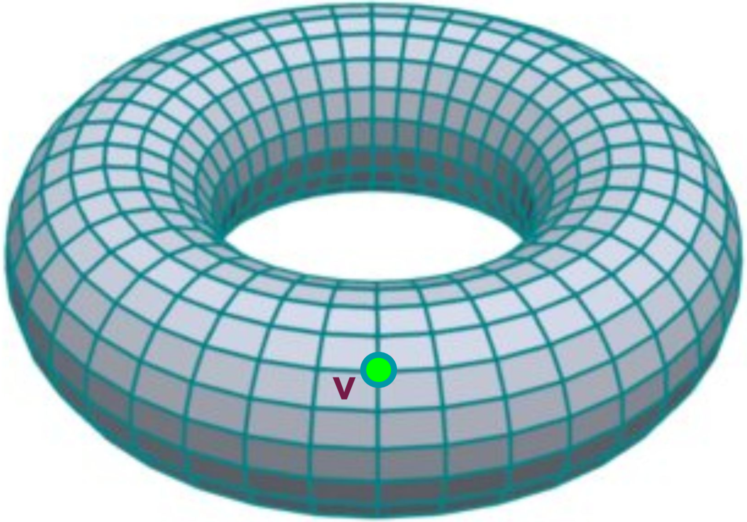
* equal contribution

ICML 40 Years International Conference On Machine Learning

Problem formulation: Efficient Graph Field Integration

Compute efficiently (in the sub-quadratic time in the number of nodes **N** of the graph) the following expressions **for every node v** of the given graph G

$$i(v) := \sum_{w \in \mathrm{V}} \mathrm{K}(w, v)\mathcal{F}(w)$$

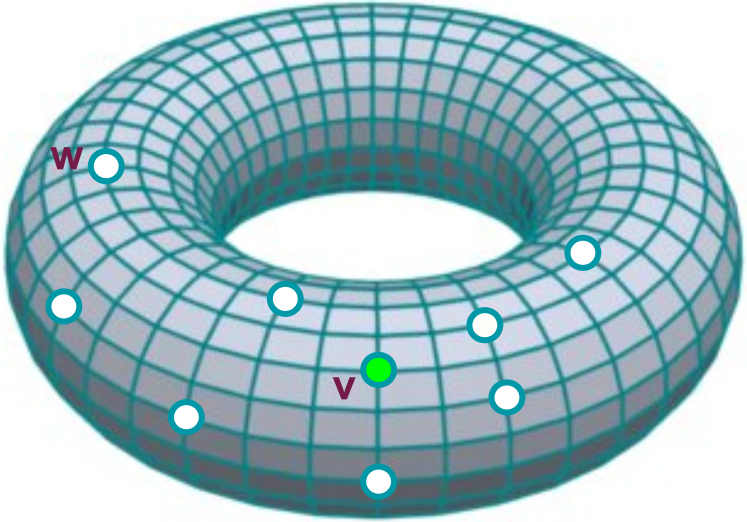


Problem formulation: Efficient Graph Field Integration

Compute efficiently (in the sub-quadratic time in the number of nodes **N** of the graph) the following expressions **for every node v** of the given graph G

$$i(v) := \sum_{w \in \mathrm{V}} \mathrm{K}(w, v)\mathcal{F}(w)$$

integration over all the nodes



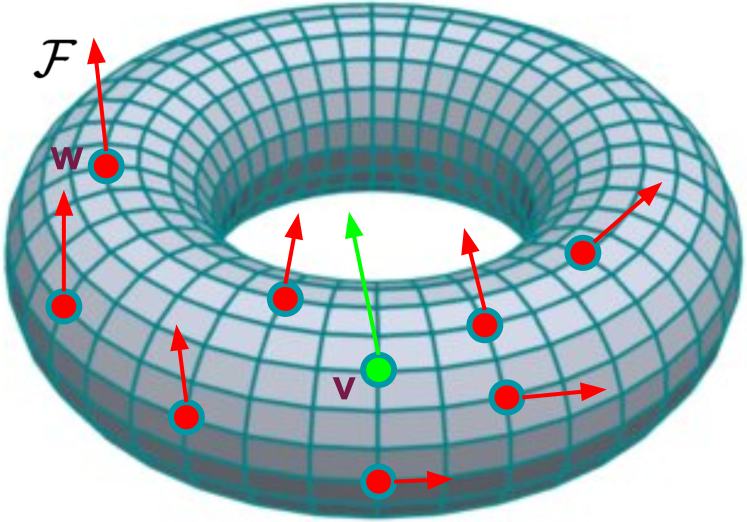
Problem formulation: Efficient Graph Field Integration

Compute efficiently (in the sub-quadratic time in the number of nodes **N** of the graph) the following expressions **for every node v** of the given graph G

$$i(v) := \sum_{w \in \mathrm{V}} \mathrm{K}(w, v)\mathcal{F}(w)$$

tensor field defined on the graph

integration over all the nodes



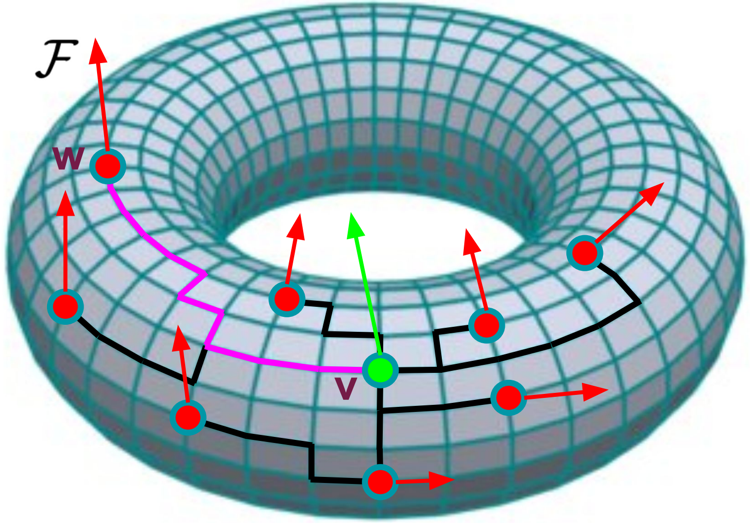
Problem formulation: Efficient Graph Field Integration

Compute efficiently (in the sub-quadratic time in the number of nodes **N** of the graph) the following expressions **for every node v** of the given graph G

$$i(v) := \sum_{w \in \mathrm{V}} \mathrm{K}(w, v) \mathcal{F}(w)$$

integration over all the nodes

similarity between two nodes (e.g. a function of the **shortest-path distance** between them)



Problem formulation: Efficient Graph Field Integration

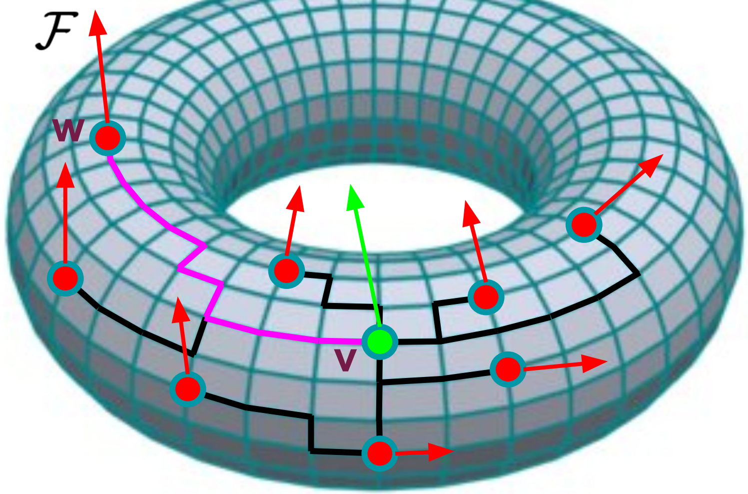
Compute efficiently (in the sub-quadratic time in the number of nodes **N** of the graph) the following expressions **for every node v** of the given graph G

$$i(v) := \sum_{w \in \mathrm{V}} \mathrm{K}(w, v)\mathcal{F}(w)$$

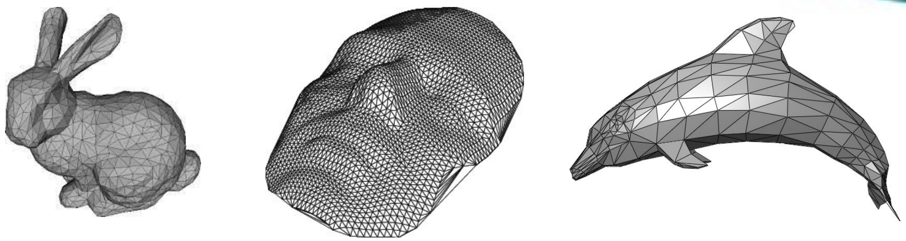
tensor field defined on the graph

integration over all the nodes

similarity between two nodes (e.g. a function of the **shortest-path distance** between them)



Graph as a discretization of the 2-dim manifold:



Problem formulation: Efficient Graph Field Integration

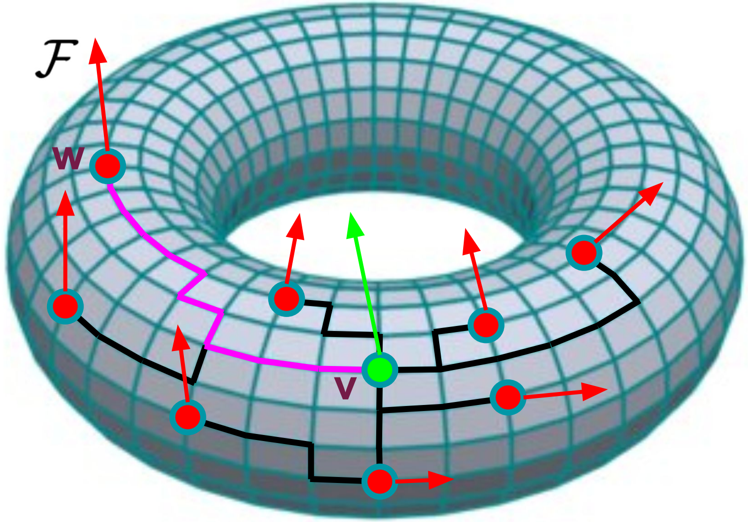
Compute efficiently (in the sub-quadratic time in the number of nodes **N** of the graph) the following expressions **for every node v** of the given graph G

$$i(v) := \sum_{w \in \mathrm{V}} \mathrm{K}(w, v)\mathcal{F}(w)$$

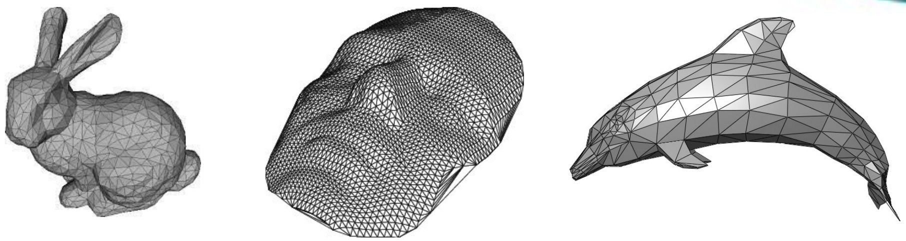
tensor field defined on the graph

integration over all the nodes

similarity between two nodes (e.g. a function of the **shortest-path distance** between them)



Graph as a discretization of the 2-dim manifold:

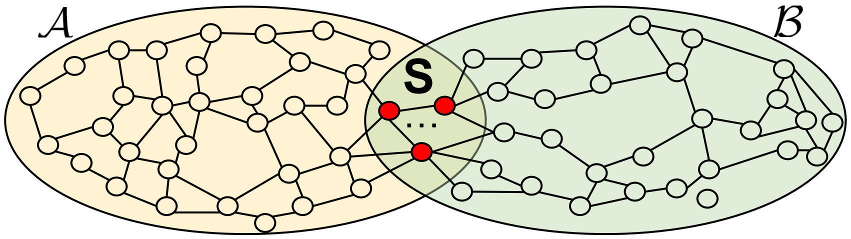


Applications: interpolation on manifolds, topological masking mechanisms for Transformers with structural inputs, physics simulations in curved spaces, Wasserstein barycenter, (Fused) Gromov Wasserstein, …

Our contributions: SeparatorFactorization (SF) and RFDiffusion (RFD)

SF

- works with input mesh-graphs
- leverages their **low-genus** structure (-> small-size separators)
- applies our new results in structural graph theory on fast graph field integration via separator-based divide-and-conquer methods and Fast Fourier Transform
- T = $O(N \log^2(N))$ time complexity for general K, T = $O(N \log^{1.383...}(N))$ if K := exp(-λ*shortest-path distance)

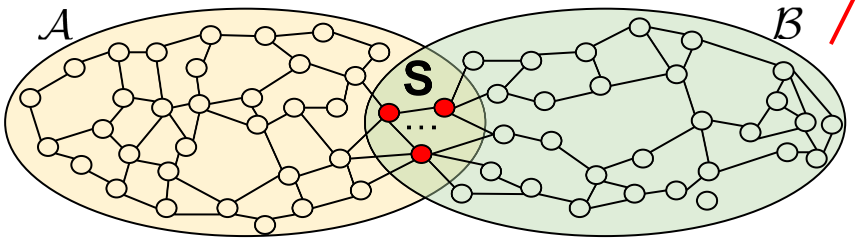


Our contributions: SeparatorFactorization (SF) and RFDiffusion (RFD)

SF

- works with input mesh-graphs
- leverages their **low-genus** structure (-> small-size separators)
- applies our new results in structural graph theory on fast graph field integration via separator-based divide-and-conquer methods and Fast Fourier Transform
- T = $O(N \log^2(N))$ time complexity for general K, T = $O(N \log^{1.383...}(N))$ if K := exp(-λ*shortest-path distance)

most expensive to compute **cross-term contributions** to the integration successfully handled by two signals:



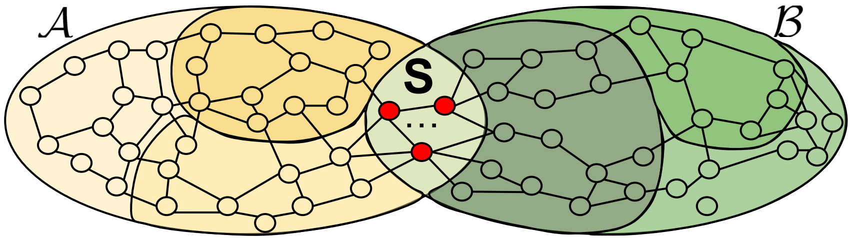
Our contributions: SeparatorFactorization (SF) and RFDiffusion (RFD)

SF

- works with input mesh-graphs
- leverages their **low-genus** structure (-> small-size separators)
- applies our new results in structural graph theory on fast graph field integration via separator-based divide-and-conquer methods and Fast Fourier Transform
- T = $O(N\log^2(N))$ time complexity for general K, T = $O(N\log^{1.383\ldots}(N))$ if K := exp(-λ*shortest-path distance)

most expensive to compute **cross-term contributions** to the integration successfully handled by two signals:

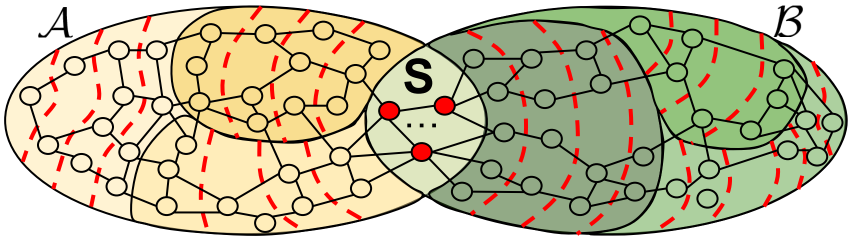
➢ residual vectors (the so-called *signature vectors*) giving corrections needed to reach particular nodes of the separator (different-color regions)



Our contributions: SeparatorFactorization (SF) and RFDiffusion (RFD)

SF

- works with input mesh-graphs
- leverages their **low-genus** structure (-> small-size separators)
- applies our new results in structural graph theory on fast graph field integration via separator-based divide-and-conquer methods and Fast Fourier Transform
- T = $O(N \log^2(N))$ time complexity for general K, T = $O(N \log^{1.383...}(N))$ if K := exp(-λ*shortest-path distance)



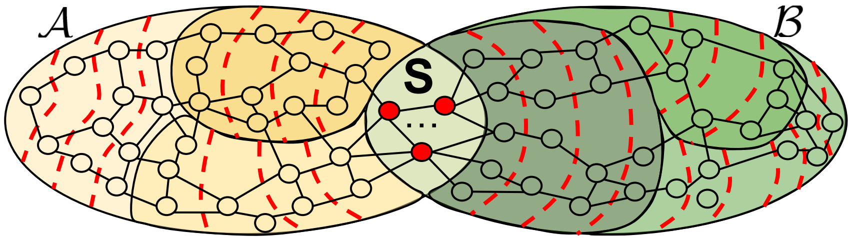
most expensive to compute **cross-term contributions** to the integration successfully handled by two signals:

- ➢ residual vectors (the so-called *signature vectors*) giving corrections needed to reach particular nodes of the separator (different-color regions)
- ➢ distance from the separator **S** (red dotted lines)

Our contributions: SeparatorFactorization (SF) and RFDiffusion (RFD)

SF

- works with input mesh-graphs
- leverages their **low-genus** structure (-> small-size separators)
- applies our new results in structural graph theory on fast graph field integration via separator-based divide-and-conquer methods and Fast Fourier Transform
- T = $O(N \log^2(N))$ time complexity for general K, T = $O(N \log^{1.383...}(N))$ if K := exp(-λ*shortest-path distance)



RFD

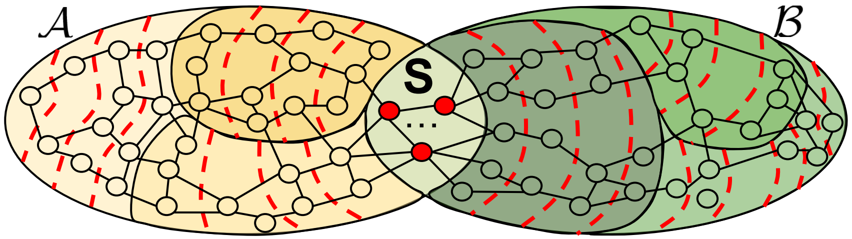
- works with point cloud (no mesh needed)
- leverages the implicit graph structure given by the following adjacency matrix:

$$\mathbf{W}_{\mathrm{G}}(i,j) = f(\mathbf{n}_i - \mathbf{n}_j)$$

Our contributions: SeparatorFactorization (SF) and RFDiffusion (RFD)

SF

- works with input mesh-graphs
- leverages their **low-genus** structure (-> small-size separators)
- applies our new results in structural graph theory on fast graph field integration via separator-based divide-and-conquer methods and Fast Fourier Transform
- T = $O(N\log^2(N))$ time complexity for general K, T = $O(N\log^{1.383\ldots}(N))$ if K := exp(-λ*shortest-path distance)



RFD

- works with point cloud (no mesh needed)
- leverages the implicit graph structure given by the following adjacency matrix:

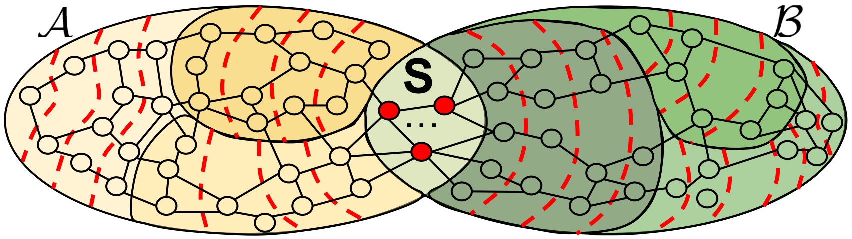
$$\mathbf{W}_{\mathrm{G}}(i,j) = f(\mathbf{n}_i - \mathbf{n}_j)$$

node representations (e.g. 3d-coords)

Our contributions: SeparatorFactorization (SF) and RFDiffusion (RFD)

SF

- works with input mesh-graphs
- leverages their **low-genus** structure (-> small-size separators)
- applies our new results in structural graph theory on fast graph field integration via separator-based divide-and-conquer methods and Fast Fourier Transform
- T = $O(N\log^2(N))$ time complexity for general K, T = $O(N\log^{1.383...}(N))$ if K := exp(-λ*shortest-path distance)



RFD

- works with point cloud (no mesh needed)
- leverages the implicit graph structure given by the following adjacency matrix:

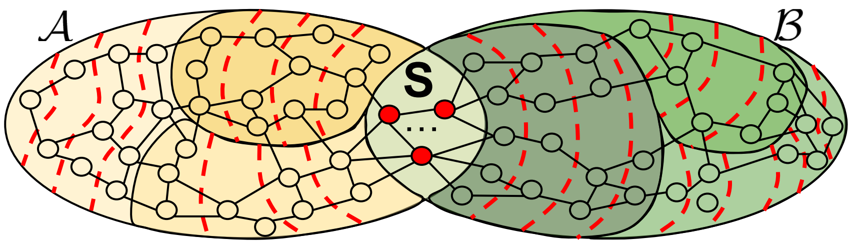
$$\mathbf{W}_{\mathrm{G}}(i,j) = f(\mathbf{n}_i - \mathbf{n}_j)$$

e.g. eps-neighborhood indicator in the particular norm

Our contributions: SeparatorFactorization (SF) and RFDiffusion (RFD)

SF

- works with input mesh-graphs
- leverages their **low-genus** structure (-> small-size separators)
- applies our new results in structural graph theory on fast graph field integration via separator-based divide-and-conquer methods and Fast Fourier Transform
- T = $O(N \log^2(N))$ time complexity for general K, T = $O(N \log^{1.383...}(N))$ if K := exp(-λ*shortest-path distance)

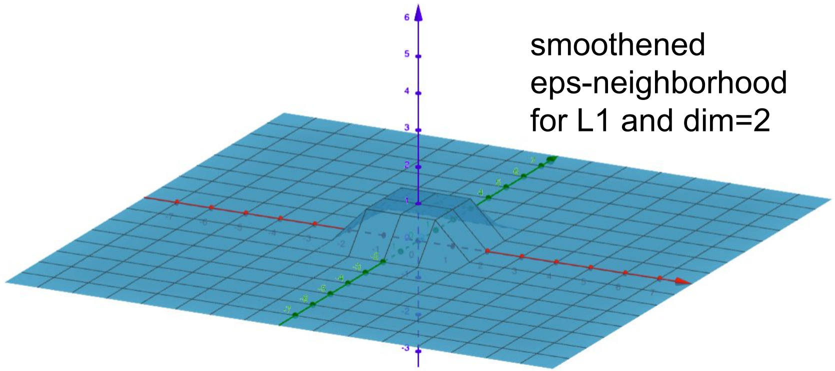


RFD

- works with point cloud (no mesh needed)
- leverages the implicit graph structure given by the following adjacency matrix:

$$\mathbf{W}_{\mathrm{G}}(i,j) = f(\mathbf{n}_i - \mathbf{n}_j)$$

e.g. eps-neighborhood indicator in the particular norm

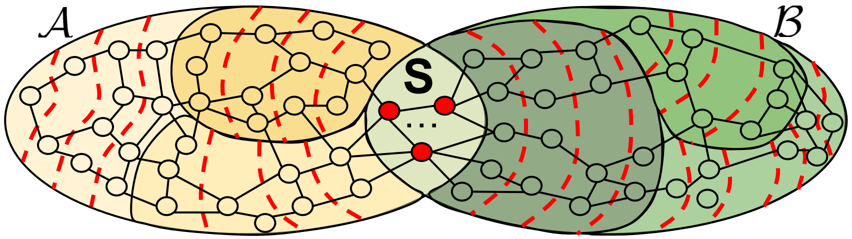


smoothened eps-neighborhood for L1 and dim=2

Our contributions: SeparatorFactorization (SF) and RFDiffusion (RFD)

SF

- works with input mesh-graphs
- leverages their **low-genus** structure (-> small-size separators)
- applies our new results in structural graph theory on fast graph field integration via separator-based divide-and-conquer methods and Fast Fourier Transform
- T = $O(N \log^2(N))$ time complexity for general K, T = $O(N \log^{1.383...}(N))$ if K := exp(-λ*shortest-path distance)



RFD

- works with point cloud (no mesh needed)
- leverages the implicit graph structure given by the following adjacency matrix:

$$\mathbf{W}_{\mathrm{G}}(i,j) = f(\mathbf{n}_i - \mathbf{n}_j)$$

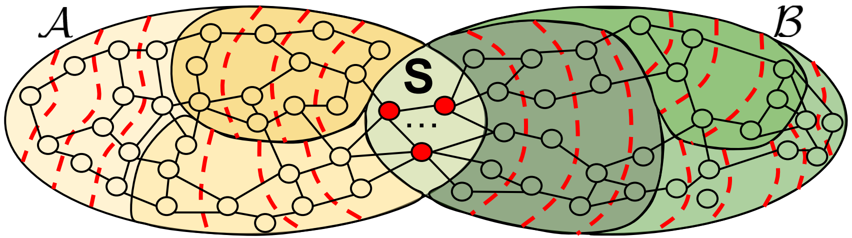
- linearizes the adjacency matrix via Fourier-Transform based random feature map mechanism:

$$f(\mathbf{z}) = \int_{\mathbb{R}^d} \exp(2\pi \mathbf{i} \omega^\top \mathbf{z}) \tau(\omega) d(\omega)$$
$$= \int_{\mathbb{R}^d} \exp(2\pi \mathbf{i} \omega^\top \mathbf{z}) \frac{\tau(\omega)}{p(\omega)} p(\omega) d(\omega)$$

Our contributions: SeparatorFactorization (SF) and RFDiffusion (RFD)

SF

- works with input mesh-graphs
- leverages their **low-genus** structure (-> small-size separators)
- applies our new results in structural graph theory on fast graph field integration via separator-based divide-and-conquer methods and Fast Fourier Transform
- T = $O(N \log^2(N))$ time complexity for general K, T = $O(N \log^{1.383...}(N))$ if K := exp(-λ*shortest-path distance)



RFD

- works with point cloud (no mesh needed)
- leverages the implicit graph structure given by the following adjacency matrix:

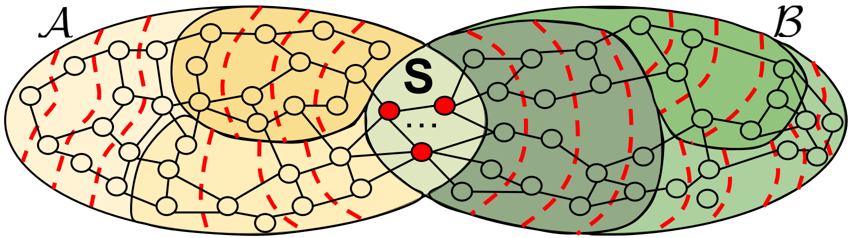
$$\mathbf{W}_{\mathrm{G}}(i,j) = f(\mathbf{n}_i - \mathbf{n}_j)$$

- linearizes the adjacency matrix via Fourier-Transform based random feature map mechanism
- O(N) time complexity, but for a specific class of **graph diffusion kernels**, leveraging our novel decomposition of the exponentials of low-rank matrices:

Our contributions: SeparatorFactorization (SF) and RFDiffusion (RFD)

SF

- works with input mesh-graphs
- leverages their **low-genus** structure (-> small-size separators)
- applies our new results in structural graph theory on fast graph field integration via separator-based divide-and-conquer methods and Fast Fourier Transform
- T = $O(N\log^2(N))$ time complexity for general K, T = $O(N\log^{1.383\ldots}(N))$ if K := exp(-λ*shortest-path distance)



RFD

- works with point cloud (no mesh needed)
- leverages the implicit graph structure given by the following adjacency matrix:

$$\mathbf{W}_{\mathrm{G}}(i,j) = f(\mathbf{n}_i - \mathbf{n}_j)$$

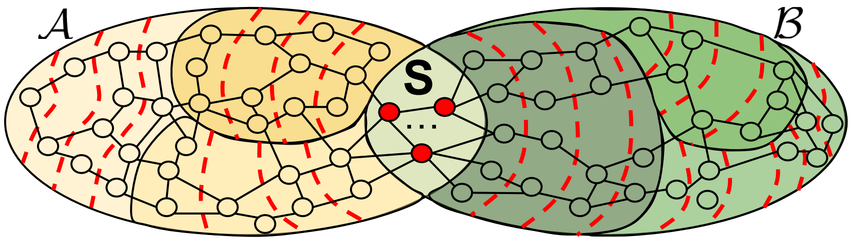
- linearizes the adjacency matrix via Fourier-Transform based random feature map mechanism
- O(N) time complexity, but for a specific class of **graph diffusion kernels**, leveraging our novel decomposition of the exponentials of low-rank matrices:

$$\exp(\Lambda \cdot \mathbf{A}\mathbf{B}^\top) = \sum_{i=0}^{\infty} \frac{1}{i!}(\Lambda \mathbf{A}\mathbf{B}^\top)^i$$
$$= \mathbf{I} + \sum_{i=0}^{\infty} \frac{1}{(i+1)!}\mathbf{A}(\Lambda \mathbf{B}^\top \mathbf{A})^{i+1}\mathbf{A}^{-1}$$
$$= \mathbf{I} + \mathbf{A}[\exp(\Lambda \mathbf{B}^\top \mathbf{A}) - \mathbf{I}](\mathbf{B}^\top \mathbf{A})^{-1}\mathbf{B}^\top$$

Our contributions: SeparatorFactorization (SF) and RFDiffusion (RFD)

SF

- works with input mesh-graphs
- leverages their **low-genus** structure (-> small-size separators)
- applies our new results in structural graph theory on fast graph field integration via separator-based divide-and-conquer methods and Fast Fourier Transform
- T = $O(N\log^2(N))$ time complexity for general K, T = $O(N\log^{1.383\ldots}(N))$ if K := exp(-λ*shortest-path distance)



RFD

- works with point cloud (no mesh needed)
- leverages the implicit graph structure given by the following adjacency matrix:

$$\mathbf{W}_{\mathrm{G}}(i,j) = f(\mathbf{n}_i - \mathbf{n}_j)$$

- linearizes the adjacency matrix via Fourier-Transform based random feature map mechanism
- O(N) time complexity, but for a specific class of **graph diffusion kernels**, leveraging our novel decomposition of the exponentials of low-rank matrices:

$$\exp(\Lambda \cdot \mathbf{A}\mathbf{B}^\top) = \sum_{i=0}^{\infty} \frac{1}{i!}(\Lambda \mathbf{A}\mathbf{B}^\top)^i$$

$$= \mathbf{I} + \sum_{i=0}^{\infty} \frac{1}{(i+1)!}\mathbf{A}(\Lambda \mathbf{B}^\top \mathbf{A})^{i+1}\mathbf{A}^{-1}$$

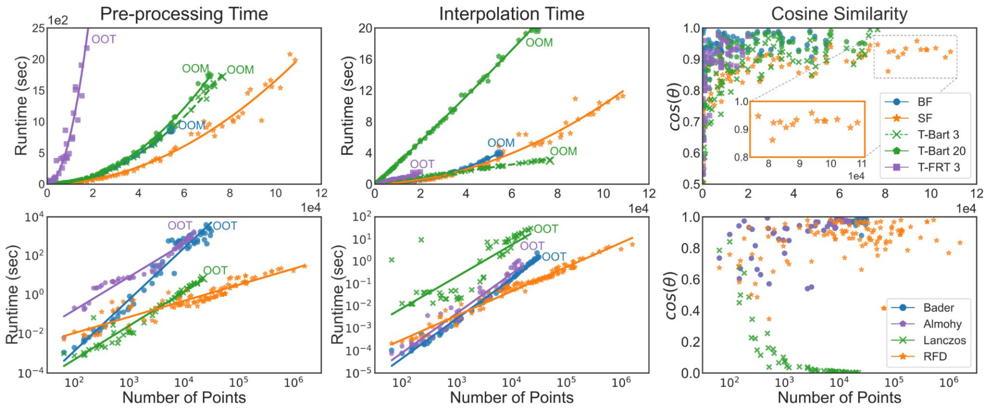
$$= \mathbf{I} + \mathbf{A}[\exp(\Lambda \mathbf{B}^\top \mathbf{A}) - \mathbf{I}](\mathbf{B}^\top \mathbf{A})^{-1}\mathbf{B}^\top$$

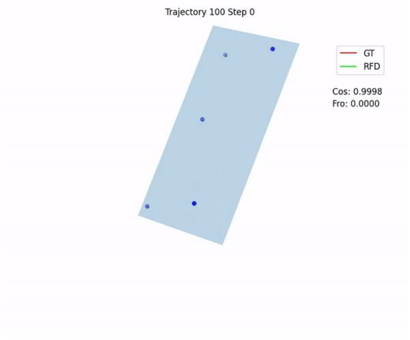
low-rank decomposition of **W** via random features

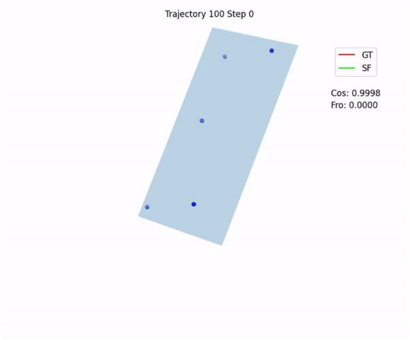
Experiments

Vertex normal and velocity prediction

- Benchmark on **120** meshes for 3D-printed objects from Thinki10k.
- Compare SF with a naive brute force method (GT) as well as various low distortion tree methods.
- Compare RFD with various algorithms that efficiently compute the action of matrix exponentials.







Wasserstein Distances and Barycenters

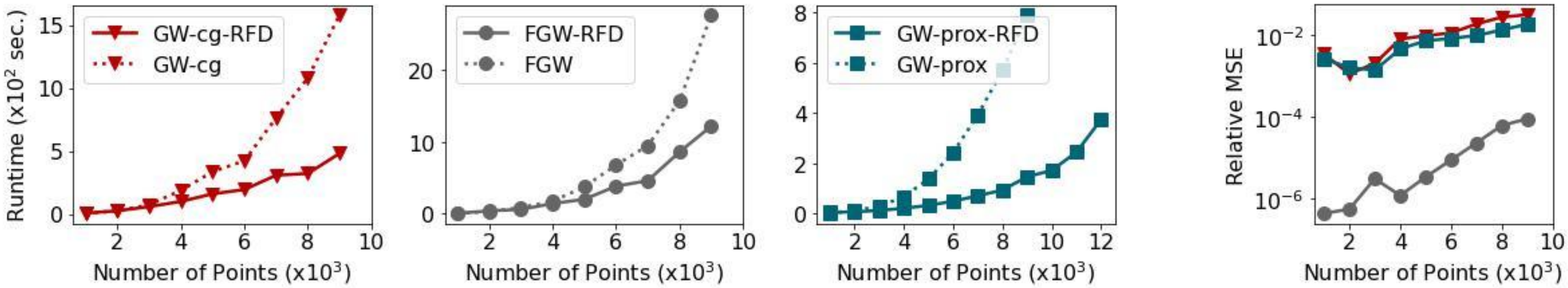
- Integrate our GFI methods into the OT problem of moving masses on a surface mesh, particularly computation of Wasserstein barycenters.
- Geodesic distance on a surface is intractable, so use 2 approximations of this metric:
    - shortest-path distance (SF)
    - distance coming from an ϵ-NN graph approximating the surface (RFD)

Mesh	|V|	**Total Runtime**		**MSE**
		BF	RFD	
Alien	5212	8.06	**0.39**	0.041
Duck	9862	45.36	**1.10**	0.002
Land	14738	147.64	**2.17**	0.017
Octocat	18944	302.84	**3.36**	0.027

Mesh	|V|	**Total Runtime**		**MSE**
		BF	SF	
Dice	4468	6.8	**4.9**	0.063
Duck	9862	39.2	**19.4**	0.002
Land	14738	90.7	**38.9**	0.015
bubblepot2	18633	113.2	**48.3**	0.081

(Fused) Gromov Wasserstein distances

- Integrate RFD method in the computation of (Fused) Gromov Wasserstein discrepancy.
- Benchmark it by running extensive speed/accuracy tests on synthetic 3D distributions.



Point Cloud Classification

- Compute the eigendecomposition of the approximated RFD kernel matrix.
- Use 16 smallest eigenvalues for classification on ModelNet10 and Cubes datasets using a random forest.

Dataset	# Graphs	# Classes	Baseline	RFD
ModelNet10	3991/908	10	43.0	**70.1**
Cubes	3759/659	23	39.3	**44.6**

- Methods like SPH and LFD on ModelNet achieves about 79%.
- Cubes is challenging and PointNet achieves only 55% accuracy.

Thank You