Mode-dependent Rate-distortion Optimized Transforms

Using Graph Signal Processing Methods

Keng-Shih Lu and Antonio Ortega

October 22, 2019

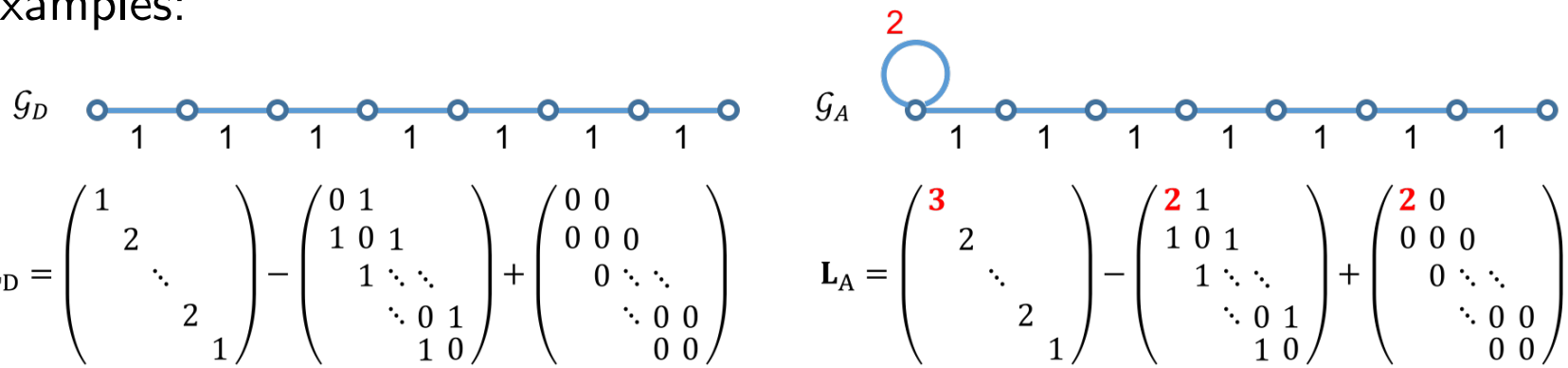
Outline

1. Background: Graph Signal Processing
2. Mode-dependent Data-driven Transforms
3. Fast GFTs based on Graph Symmetries
4. Efficient RD Approximation using Laplacian Operators
5. Conclusion

Graph Signal Processing (GSP)

Graph Fourier Transform (GFT) (a.k.a. graph-based transforms)

- Laplacian matrix $\mathbf{L} = \mathbf{D} - \mathbf{W} + \mathbf{S}$
- Examples:

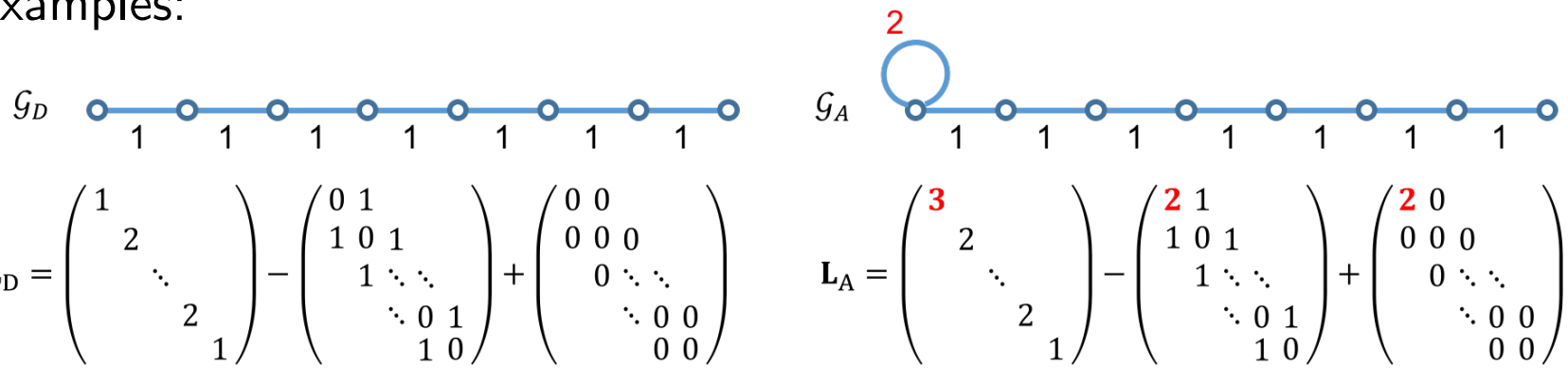


- GFT basis functions $\mathbf{U}$: eigenvectors of $\mathbf{L}$ ($\mathbf{L} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top$)
  - GFTs of $\mathcal{G}_D$ and $\mathcal{G}_A$ are DCT and ADST

Graph Signal Processing (GSP)

Graph Fourier Transform (GFT) (a.k.a. graph-based transforms)

- Laplacian matrix $\mathbf{L} = \mathbf{D} - \mathbf{W} + \mathbf{S}$
- Examples:



- GFT basis functions $\mathbf{U}$: eigenvectors of $\mathbf{L}$ ($\mathbf{L} = \mathbf{U}\mathbf{\Lambda}\mathbf{U}^\top$)
  - GFTs of $\mathcal{G}_D$ and $\mathcal{G}_A$ are DCT and ADST

Probabilistic interpretations:

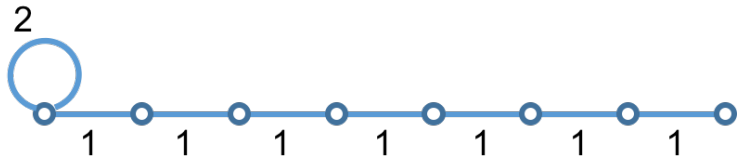
- Graph $\longleftrightarrow$ Gaussian Markov Random Field (GMRF)
- Large edge weight $\longleftrightarrow$ high correlation
- GFT on graph signal $\longleftrightarrow$ decorrelation (PCA) of GMRF data
- Designing graph weights $\longleftrightarrow$ parameter estimation for a GMRF

DCT and ADST

Discrete cosine transform (DCT)

(a) Graph

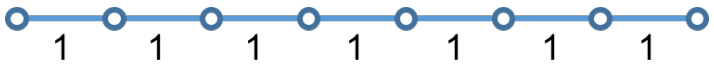
Asymmetric discrete sine transform (ADST)



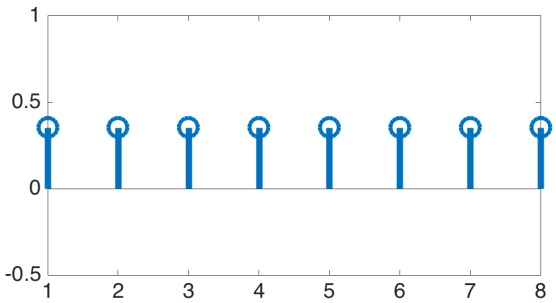
(a) Graph

DCT and ADST

Discrete cosine transform (DCT)

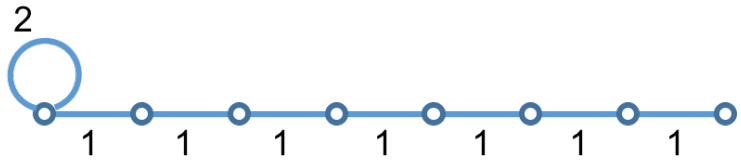


(a) Graph

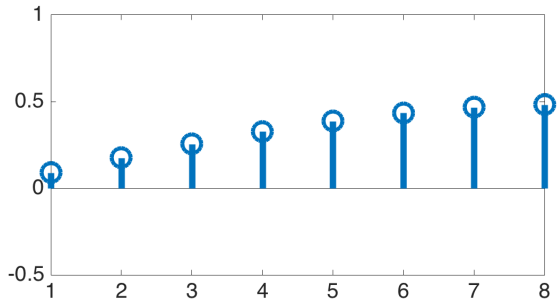


(b) $\mathbf{u}_1$

Asymmetric discrete sine transform (ADST)



(a) Graph



(b) $\mathbf{u}_1$

- Each node corresponds to one pixel
- Large self-loop $\longleftrightarrow$ small value in $\mathbf{u}_1$

GSP for Image and video compression

Prior work

- Graph template transforms for texture images [Pavez et. al. 2015]
- Piecewise smooth image compression [Hu et. al. 2015]
- Generalized GFTs for intra predicted video coding [Hu et. al. 2015]
- Edge-adaptive GFTs for inter predicted video coding [Egilmez et. al. 2015]

In this talk: graph-based methods for AV1/AV2

- Rate-distortion optimized transforms (with graph-based regularizations)
  - Transforms obtained are mode-dependent
  - Achieved compression gains on AV1/AV2
- Fast GFT designs
- Fast RD approximation
  - Achieved speedup in transform type search

Outline

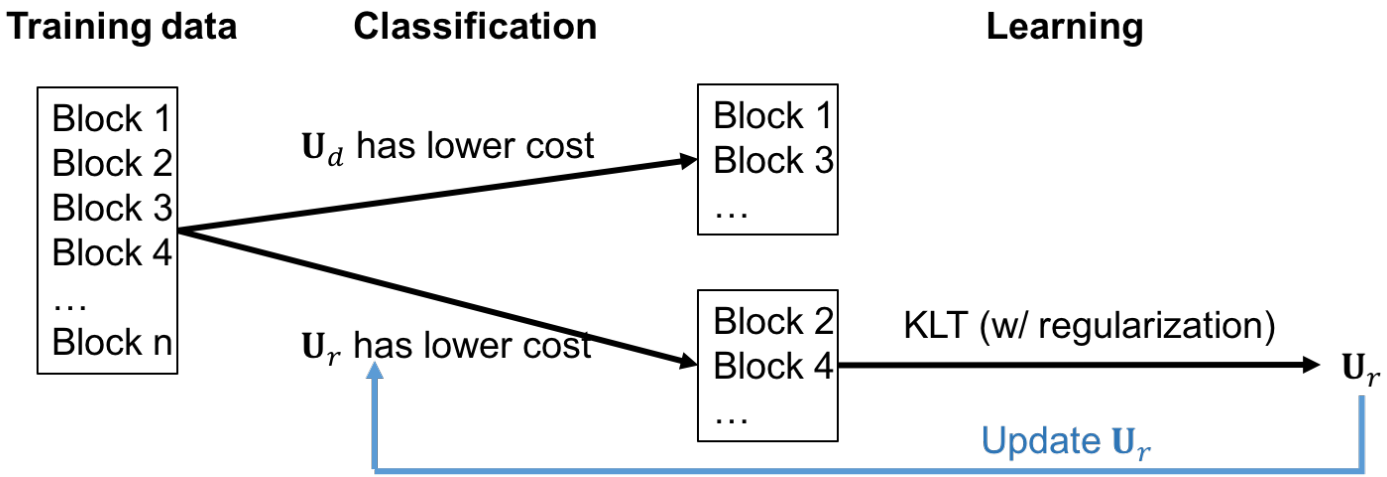
1. Background: Graph Signal Processing
2. Mode-dependent Data-driven Transforms
3. Fast GFTs based on Graph Symmetries
4. Efficient RD Approximation using Laplacian Operators
5. Conclusion

Rate-Distortion Optimized Transforms (RDOT)

RDOT [Effros et. al., 1999], [Zhao et. al. 2012], [Zou et. al. 2013],

- Goal: learn a transform in a system using multiple transforms (e.g. AV1)
- Main idea: use RD-based transform selection during learning

Procedure: for each iteration



Note

- Can be easily extended to multiple learned transforms
- Lloyd-like algorithm $\longrightarrow$ solution depends on the initialization

Training RDOT for AV1

Goal: introduce a new 1D transform for each inter/intra block

- Intra–block statistics are highly mode-dependent
  - We train MD-RDOT: mode-dependent RDOTs
- Inter–block statistics are symmetric
  - Learn RDOT and FLIPRDOT together

New transform types: 2D combinations of

- Each intra mode: MD-RDOT & DCT
- Inter: RDOT, FLIPRDOT, and DCT

Implementation details

- Training data: 2D residues extracted from AV1
- We use weighted sum of squared transform coefficients for classification
  - Proxy of the RD cost

Graph-based Regularizations

Idea: force the RDOT to be a GFT

- Learning a graph from data (covariance matrix $\mathbf{S}$)

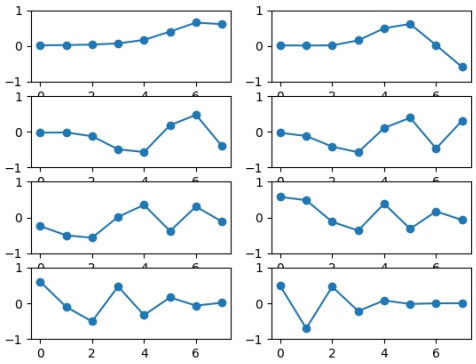
$$\underset{\mathbf{L} \text{ is a Laplacian}}{\text{minimize}} \quad -\log\det(\mathbf{L}) + \text{trace}(\mathbf{LS})$$

  - Convex problem with iterative solver [Egilmez et. al. 2018]

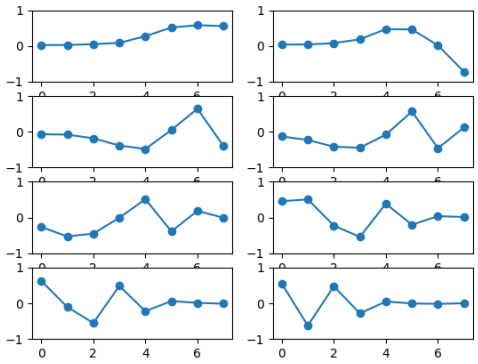
Transforms with different regularization settings

- KLT: no regularization
- GFT: with graph Laplacian constraints
- LGT: line graph transform (graph Laplacian with line graph topology)

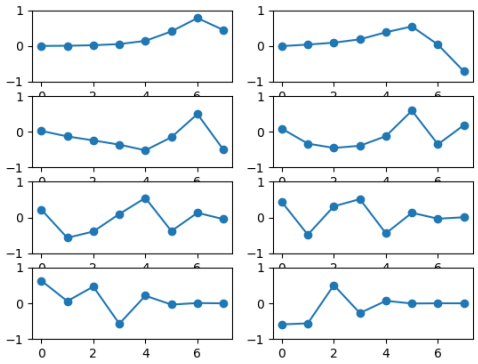
Resulting Transform Bases



(a) KLT for inter



(b) GFT for inter



(c) LGT for inter

Observations: when using regularization constraints

- Similar shape to KLT
- But more localized basis functions with sharper transitions
- Fewer parameters to choose

Experimental Results

Graph-based regularization

- Compression gain on AV1 w.r.t. training set size

	Training set size per mode			
	12500	25000	50000	100000
KLT	0.7317%	0.6922%	**0.8476%**	**0.7749%**
GFT	**0.7480%**	**0.6935%**	0.6235%	0.4233%
LGT	0.5527%	0.5401%	0.7235%	0.5698%

- Graph-based transforms may outperform KLT when training set is small

Experimental Results

Graph-based regularization

- Compression gain on AV1 w.r.t. training set size

	Training set size per mode			
	12500	25000	50000	100000
KLT	0.7317%	0.6922%	**0.8476%**	**0.7749%**
GFT	**0.7480%**	**0.6935%**	0.6235%	0.4233%
LGT	0.5527%	0.5401%	0.7235%	0.5698%

- Graph-based transforms may outperform KLT when training set is small

AV2 Experiment–CONFIG_MODE_DEP_TX

- RDOT with KLT applied
- Compression gains on AOM lowres test set

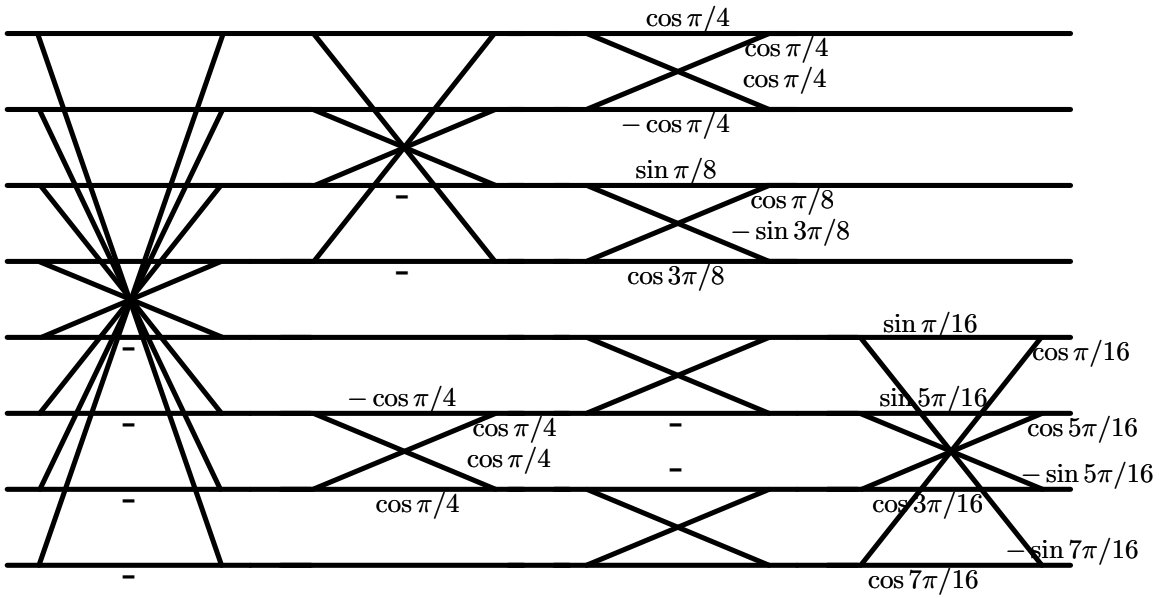
	Overall	Key frames
With sep. KLT	0.70%	0.64%
With sep. & non-sep. KLTs	0.79%	1.09%

Outline

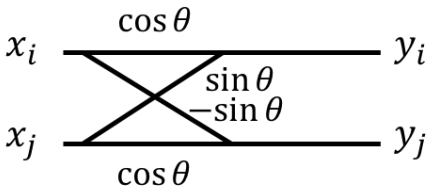
1. Background: Graph Signal Processing
2. Mode-dependent Data-driven Transforms
3. Fast GFTs based on Graph Symmetries
4. Efficient RD Approximation using Laplacian Operators
5. Conclusion

Fast GFT?

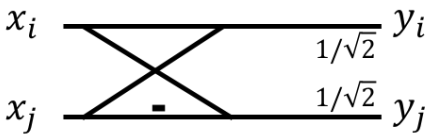
Example: DCT



Key components



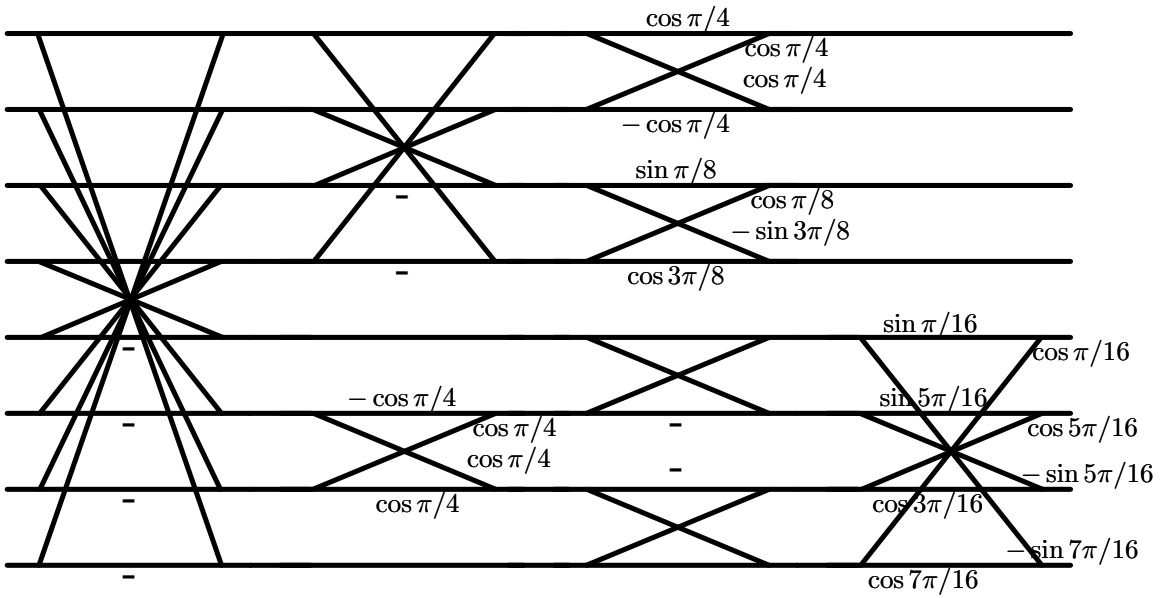
(a) Givens rotation



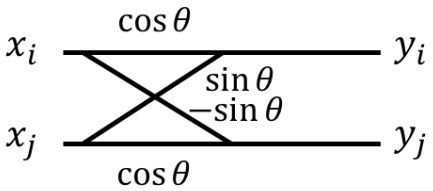
(b) Haar unit

Fast GFT?

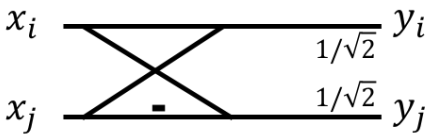
Example: DCT



Key components



(a) Givens rotation

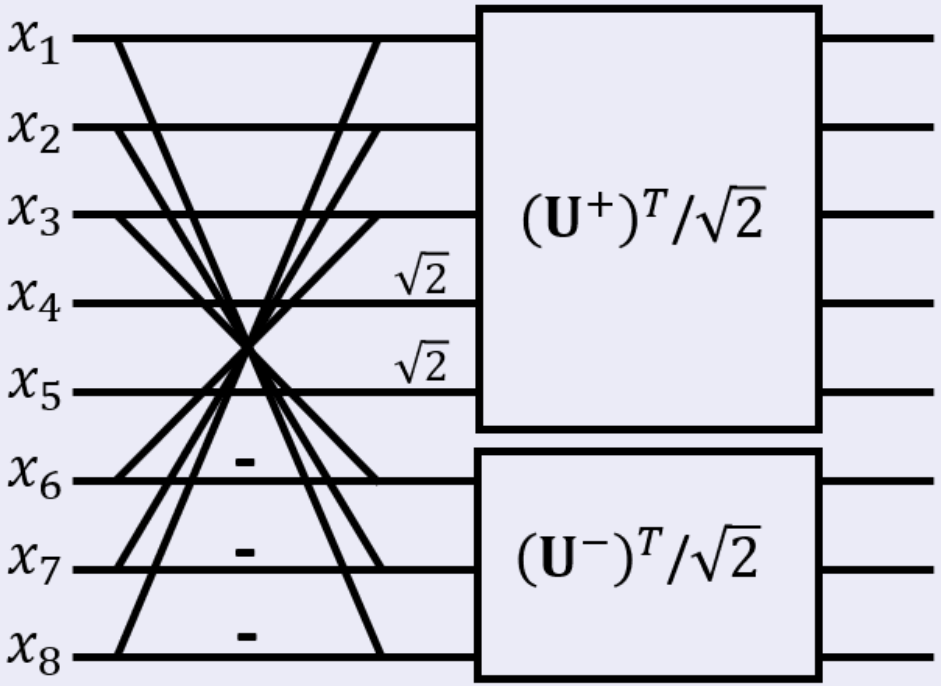


(b) Haar unit

- Graph structural property $\longrightarrow$ fast GFT?
- We will focus on butterfly stages with Haar units

GFTs with Haar Units

Theorem

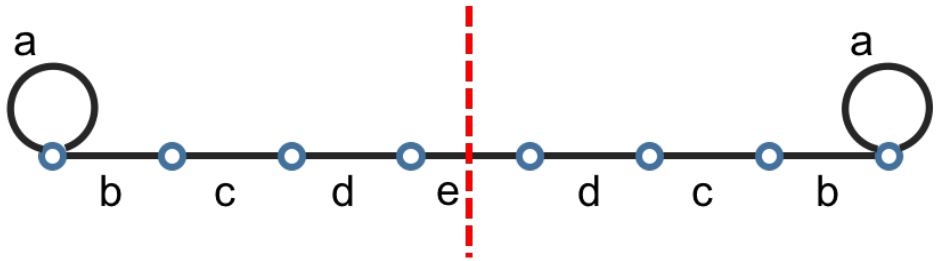


GFT has a left butterfly stage $\Longleftrightarrow$ graph is symmetric

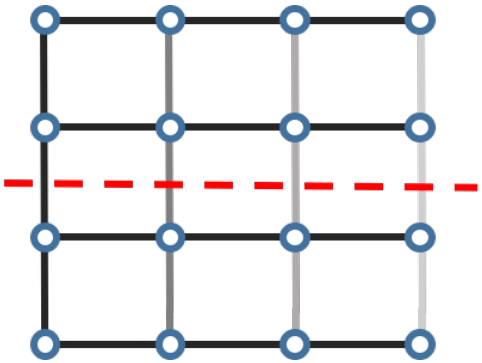
- See [Lu and Ortega, TSP 2019] for formal definition of symmetry (node pairing)

Examples of Fast GFTs

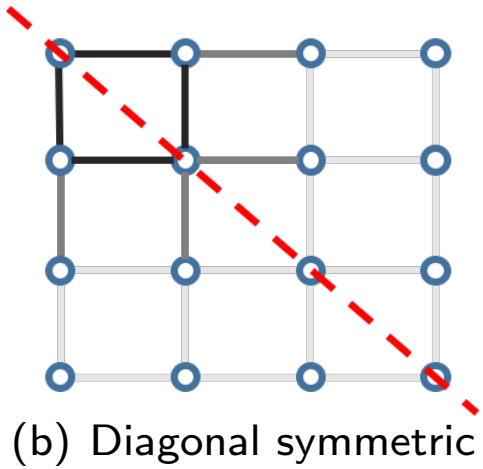
Fast GFTs on 1D blocks: symmetric line graph



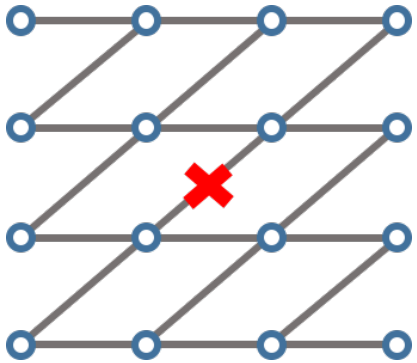
Fast GFTs on 2D blocks: symmetric grid graph



(a) Up-down symmetric



(b) Diagonal symmetric



(c) Centrosymmetric

- Each symmetry $\Rightarrow$ multiplications reduced by half
- Leads to fast separable & non-separable transforms
  - Coding gain achieved in [Gnutti et. al., 2018]

Outline

1. Background: Graph Signal Processing
2. Mode-dependent Data-driven Transforms
3. Fast GFTs based on Graph Symmetries
4. Efficient RD Approximation using Laplacian Operators
5. Conclusion

Rate-distortion Optimization

RD cost evaluation: $D + \lambda \times R$

- For each (partition, mode, tx_type), we need
  - transform & quantization & entropy coding

$\Rightarrow$ Brute force is very computationally expensive

Rate-distortion Optimization

RD cost evaluation: $D + \lambda \times R$

- For each (partition, mode, tx_type), we need
  - transform & quantization & entropy coding

$\Rightarrow$ Brute force is very computationally expensive

Motivation: can we estimate RD cost in the pixel domain?

- No need to compute transform & quantization & entropy coding

Approximation with Sparse Graph Laplacians

Idea: graph Laplacians associated to DCT/ADST are sparse

Example: DCT

$$\underbrace{\sum_{i=1}^{n-1}(x_i - x_{i+1})^2}_{\text{(A) pixel domain}} = \underbrace{\sum_{l=1}^{n}\lambda_l(\boldsymbol{\phi}_l^\top \mathbf{x})^2}_{\text{(B) transform (GFT) domain}}$$

(A) Simple computation

(B) Weighted sum of squared GFT coefficients (approximate RD cost)

Approximation with Sparse Graph Laplacians

Idea: graph Laplacians associated to DCT/ADST are sparse

Example: DCT

$$\underbrace{\sum_{i=1}^{n-1}(x_i - x_{i+1})^2}_{\text{(A) pixel domain}} = \underbrace{\sum_{l=1}^{n}\lambda_l(\boldsymbol{\phi}_l^\top \mathbf{x})^2}_{\text{(B) transform (GFT) domain}}$$

(A) Simple computation

(B) Weighted sum of squared GFT coefficients (approximate RD cost)

Can we do this for general weights? (not λ_l)

- Eigenvalues may not ideally reflect the RD cost

$\rightarrow$ idea: use other graphs associated to DCT/ADST

Sparse Laplacian Operators for DCT/ADST

How to find sparse Laplacians?

- We extend the derivations in [Strang 1999]

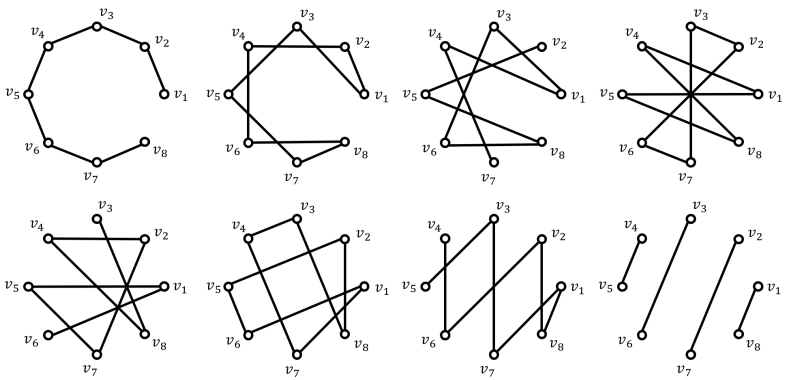
Sparse Laplacian Operators for DCT/ADST

How to find sparse Laplacians?

- We extend the derivations in [Strang 1999]

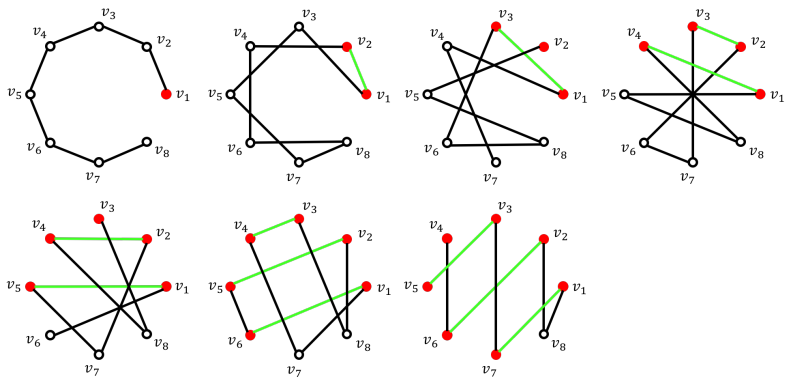
Results:

DCT $N = 8$



(a) Graphs (with $\mathbf{L}_{\mathrm{D}}^{(1)}$ to $\mathbf{L}_{\mathrm{D}}^{(8)}$)

ADST $N = 8$



(b) Graphs (with $\mathbf{L}_{\mathrm{A}}^{(1)}$ to $\mathbf{L}_{\mathrm{A}}^{(7)}$)

- Red: self-loop
- Green: negative edge

RD Cost Approximation

Approach: use linear combination of a few among $\mathbf{L}_{\mathrm{D}}^{(\ell)}$

Procedure

1. Design weights w_i s.t.

$$\text{RD cost} \approx \sum_i w_i (\tilde{\mathbf{x}}_i)^2$$

2. Find linear combination of k graphs s.t. eigenvalues $\approx w_i$
   - k-sparse representation (can be solved offline)

RD Cost Approximation

Approach: use linear combination of a few among $\mathbf{L}_{\mathrm{D}}^{(\ell)}$

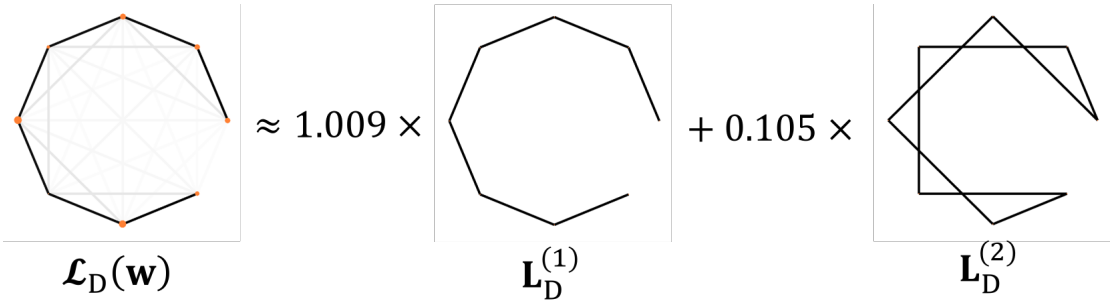
Procedure

1. Design weights w_i s.t.

$$\text{RD cost} \approx \sum_i w_i (\tilde{\mathbf{x}}_i)^2$$

2. Find linear combination of k graphs s.t. eigenvalues $\approx w_i$
   - k-sparse representation (can be solved offline)

Example: for $w_i = 2 - 2\cos((j-1/2)\pi/N)$, $k = 2$



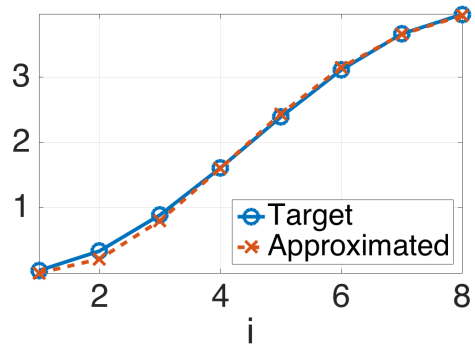


Figure: Eigenvalues

Experiment: Fast Transform Type Selection in AV1

Transform types in AV1

- 1D: DCT (D), ADST (A), FLIPADST (F), IDTX (I)
- 2D: all 16 combinations 1D transforms
- Our goal: apply pruning to transform type search

Experiment: Fast Transform Type Selection in AV1

Transform types in AV1

- 1D: DCT (D), ADST (A), FLIPADST (F), IDTX (I)
- 2D: all 16 combinations 1D transforms
- Our goal: apply pruning to transform type search

Transform type pruning (details in [Lu et. al., PCS 2018])

- Use 3 sparse Laplacians and sinusoidal increasing weights
- Evaluate approximate costs $\mathcal{Q}_{\mathsf{D}}$, $\mathcal{Q}_{\mathsf{A}}$, $\mathcal{Q}_{\mathsf{F}}$, $\mathcal{Q}_{\mathsf{I}}$

$$\text{Prune} \begin{matrix} \text{DCT} \\ \text{ADST} \\ \text{FLIPADST} \\ \text{IDTX} \end{matrix} \text{ if } \begin{matrix} \mathcal{Q}_{\mathsf{D}} \\ \mathcal{Q}_{\mathsf{A}} \\ \mathcal{Q}_{\mathsf{F}} \\ \mathcal{Q}_{\mathsf{I}} \end{matrix} > \tau(\mathcal{Q}_{\mathsf{D}} + \mathcal{Q}_{\mathsf{A}} + \mathcal{Q}_{\mathsf{F}} + \mathcal{Q}_{\mathsf{I}})$$

Results

- Small test set (5 videos and 7 target bitrate levels)

	Encoding time	Bitrate loss
Baseline	100%	0.00%
PRUNE_ONE	81%	0.22%
PRUNE_2D_ACCURATE	83%	-0.04%
PRUNE_LAPLACIAN	81%	0.18%

Results

- Small test set (5 videos and 7 target bitrate levels)

	Encoding time	Bitrate loss
Baseline	100%	0.00%
PRUNE_ONE	81%	0.22%
PRUNE_2D_ACCURATE	83%	-0.04%
PRUNE_LAPLACIAN	81%	0.18%

Our method provides

- Smaller loss than PRUNE_ONE (thresholding of empirical correlation)
- Higher loss than PRUNE_2D_ACCURATE (neural network)
  - Easier to train (not data-driven vs. 9 neural networks)
  - Easier to interpret (62 vs >5000 parameters)

Results

- Small test set (5 videos and 7 target bitrate levels)

	Encoding time	Bitrate loss
Baseline	100%	0.00%
PRUNE_ONE	81%	0.22%
PRUNE_2D_ACCURATE	83%	-0.04%
PRUNE_LAPLACIAN	81%	0.18%

Our method provides

- Smaller loss than PRUNE_ONE (thresholding of empirical correlation)
- Higher loss than PRUNE_2D_ACCURATE (neural network)
  - Easier to train (not data-driven vs. 9 neural networks)
  - Easier to interpret (62 vs >5000 parameters)

Outline

1. Background: Graph Signal Processing
2. Mode-dependent Data-driven Transforms
3. Fast GFTs based on Graph Symmetries
4. Efficient RD Approximation using Laplacian Operators
5. Conclusion

Summary

Mode-dependent data-driven transforms

- Demonstrated results with graph-based regularizations
- AV2 experiment–CONFIG_MODE_DEP_TX)
  - 0.7% gain achieved by introduced separable MD-RDOTs
  - 0.1% additional gain achieved by non-separable MD-RDOTs

Fast GFT

- Symmetric graph $\longleftrightarrow$ butterfly stage

Fast RD approximation

- Sparse Laplacian operators
- 19% encoder speedup