Gene Cheung
Associate Professor, York University
16th November, 2018

Graph Spectral Image Processing
Acknowledgement

Collaborators:

- Y. Nakatsukasa (NII, Japan)
- S. Muramatsu (Niigata, Japan)
- A. Ortega (USC, USA)
- D. Florencio (MSR, USA)
- P. Frossard (EPFL, Switzerland)
- J. Liang, I. Bajic (SFU, Canada)
- X. Wu (McMaster U, Canada)
- V. Stankovic (U of Strathclyde, UK)
- P. Le Callet (U of Nantes, France)
- X. Liu (HIT, China)
- W. Hu, J. Liu, Z. Guo, W. Gao (Peking U., China)
- X. Ji, L. Fang (Tsinghua, China)
- Y. Zhao (BJTU, China)
- C.-W. Lin (National Tsing Hua University, Taiwan)
- E. Peixoto, B. Macchiavello, E. M. Hung (U. Brasilia, Brazil)
Outline

• GSP Fundamentals

• GSP for Image Compression
  • Optimality of GFT

• GSP for Inverse Imaging
  • Graph Laplacian Regularizer (GLR)
  • Reweighted Graph TV

• Deep GLR

• Ongoing & Future Work
Outline

• GSP Fundamentals

• GSP for Image Compression
  • Optimality of GFT

• GSP for Inverse Imaging
  • Graph Laplacian Regularizer
  • Reweighted Graph TV

• Deep GLR

• Ongoing & Future Work
Digital Signal Processing

• Discrete signals on **regular** data kernels.
  • Ex.1: audio on regularly sampled timeline.
  • Ex.2: image on 2D grid.

• **Harmonic analysis** tools (transforms, wavelets) for diff. tasks:
  • Compression.
  • Restoration.
  • Segmentation, classification.
Smoothness of Signals

- Signals are often **smooth**.
- Notion of **frequency, band-limited**.
- Ex.: **DCT**:
  \[
  X_k = \sum_{n=0}^{N-1} x_n \cos\left(\frac{\pi}{N} \left(n + \frac{1}{2}\right)k\right)
  \]

2D DCT basis

\[
\mathbf{a} = \Phi \mathbf{x}
\]

desired signal
transform
transform coeff.

\[
\mathbf{a} = \begin{bmatrix}
  a_0 \\
  a_1 \\
  0 \\
  \vdots \\
  0
\end{bmatrix}
\]
Smoothness of Signals

• Signals are often smooth.
• Notion of frequency, band-limited.
• Ex.: DCT:

\[ X_k = \sum_{n=0}^{N-1} x_n \cos \left( \frac{\pi}{N} \left( n + \frac{1}{2} \right) k \right) \]

2D DCT basis

\[ a = \Phi x \]

transform coeff.

\[ a = \begin{bmatrix} a_0 \\ a_1 \\ 0 \\ \vdots \\ 0 \end{bmatrix} \]

desired signal transform

Typical pixel blocks have almost no high frequency components.
Smoothness of Signals

- Signals are often **smooth**.
- Notion of **frequency, band-limited**.
- Ex.: **DCT**:
  \[ X_k = \sum_{n=0}^{N-1} x_n \cos \left( \frac{\pi}{N} \left( n + \frac{1}{2} \right) k \right) \]

Typical pixel blocks have almost no high frequency components.

2D DCT basis

\[ \mathbf{a} = \Phi \mathbf{x} \]

**Transform coeff.**

| \[ a_0 \] |
| \[ a_1 \] |
| \[ 0 \] |
| \[ \vdots \] |
| \[ 0 \] |

**Compact signal representation**

**Desired signal transform**
Graph Signal Processing

• Signals on *irregular* data kernels described by graphs.
  • Graph: nodes and edges.
  • Edges reveals *node-to-node relationships*.

1. Data domain is naturally a graph.
   • Ex: ages of users on social networks.

2. Underlying data structure unknown.
   • Ex: images: 2D grid → structured graph.

Graph Signal Processing

- Signals on *irregular* data kernels described by graphs.
  - Graph: nodes and edges.
  - Edges reveals *node-to-node relationships*.

1. Data domain is naturally a graph.
   - Ex: ages of users on social networks.

2. Underlying data structure unknown.
   - Ex: images: 2D grid → structured graph.

Graph Signal Processing (GSP) addresses the problem of processing signals that live on graphs.

Graph Signal Processing

- Signals on *irregular* data kernels described by graphs.
  - Graph: nodes and edges.
  - Edges reveals *node-to-node relationships*.

1. Data domain is naturally a graph.
   - **Ex:** ages of users on social networks.

2. Underlying data structure unknown.
   - **Ex:** images: 2D grid → structured graph.

---

Graph Signal Processing (GSP) addresses the problem of processing signals that live on graphs.

---

Graph Signal Processing

- Signals on *irregular* data kernels described by graphs.
  - Graph: nodes and edges.
  - Edges reveals *node-to-node relationships*.

1. Data domain is naturally a graph.
   - *Ex*: ages of users on social networks.

2. Underlying data structure unknown.
   - *Ex*: images: 2D grid → structured graph.

Graph Signal Processing (GSP) addresses the problem of processing signals that live on graphs.

Graph Signal Processing

• Signals on *irregular* data kernels described by graphs.
  • Graph: nodes and edges.
  • Edges reveals *node-to-node relationships*.

1. Data domain is naturally a graph.
  • *Ex*: ages of users on social networks.

2. Underlying data structure unknown.
  • *Ex*: images: 2D grid → structured graph.

Graph Signal Processing (GSP) addresses the problem of processing signals that live on graphs.

Graph Signal Processing

Research questions*:  

**Sampling**: how to efficiently acquire / sense a graph-signal?  
  • Graph sampling theorems.

**Representation**: Given graph-signal, how to compactly represent it?  
  • Transforms, wavelets, dictionaries.

**Signal restoration**: Given noisy and/or partial graph-signal, how to recover it?  
  • Graph-signal priors.

Graph Signal Processing

Research questions*:

• **Sampling**: how to efficiently acquire / sense a graph-signal?
  • Graph sampling theorems.

• **Representation**: Given graph-signal, how to compactly represent it?
  • Transforms, wavelets, dictionaries.

• **Signal restoration**: Given noisy and/or partial graph-signal, how to recover it?
  • Graph-signal priors.

Graph Fourier Transform (GFT)

Graph Laplacian:

- **Adjacency Matrix** $A$: entry $A_{i,j}$ has *non-negative* edge weight $w_{i,j}$ connecting nodes $i$ and $j$.

- **Degree Matrix** $D$: diagonal matrix w/ entry $D_{i,i}$ being sum of column entries in row $i$ of $A$.
  \[ D_{i,i} = \sum_j A_{i,j} \]

- **Combinatorial Graph Laplacian** $L$: $L = D - A$
  - $L$ is *symmetric* (graph undirected).
  - $L$ is a *high-pass* filter.
  - $L$ is related to *2nd derivative*.

\[ L_{3,:}x = -x_2 + 2x_3 - x_4 \]
\[ f''(x) = \lim_{h \to 0} \frac{f(x+h)-2f(x)+f(x-h)}{h^2} \]

*https://en.wikipedia.org/wiki/Second_derivative*
Graph Fourier Transform (GFT)

Graph Laplacian:

• **Adjacency Matrix** $A$: entry $A_{i,j}$ has *non-negative* edge weight $w_{i,j}$ connecting nodes $i$ and $j$.

• **Degree Matrix** $D$: diagonal matrix w/ entry $D_{i,i}$ being sum of column entries in row $i$ of $A$.

  \[ D_{i,i} = \sum_j A_{i,j} \]

• **Combinatorial Graph Laplacian** $L$: \[ L = D - A \]
  - $L$ is *symmetric* (graph undirected).
  - $L$ is a *high-pass* filter.
  - $L$ is related to $2^{nd}$ derivative.

\[ L_{3,:}x = -x_2 + 2x_3 - x_4 \]

\[ f''(x) = \lim_{h \to 0} \frac{f(x+h) - 2f(x) + f(x-h)}{h^2} \]

*https://en.wikipedia.org/wiki/Second_derivative*
Graph Spectrum from GFT

- **Graph Fourier Transform** (GFT) is eigen-matrix of graph Laplacian $L$.

$$L = V \sum V^T$$

1. Edge weights affect shapes of eigenvectors.
2. Eigenvalues ($\geq 0$) as *graph frequencies*.
   - Constant eigenvector is DC.
   - # *zero-crossings* increases as $\lambda$ increases.

- GFT defaults to **DCT** for un-weighted connected line.
- GFT defaults to **DFT** for un-weighted connected circle.
Graph Spectrum from GFT

- **Graph Fourier Transform** (GFT) is eigen-matrix of graph Laplacian $L$.

\[
L = V \Sigma V^T
\]

- Eigenvectors in columns
- Eigenvalues along diagonal

\[
\tilde{x} = V^T x
\]

GFT coefficients

1. Edge weights affect shapes of eigenvectors.

2. Eigenvalues ($\geq 0$) as *graph frequencies*.
   - Constant eigenvector is DC.
   - # *zero-crossings* increases as $\lambda$ increases.

- GFT defaults to **DCT** for un-weighted connected line.
- GFT defaults to **DFT** for un-weighted connected circle.
Graph Frequency Examples (US Temperature)

- Weather stations from 100 most populated cities.
- Graph connections from Delaunay Triangulation*.
- Edge weights inverse proportion to distance.

\[ w_{i,j} = \exp \left( \frac{-\|t_i - t_j\|^2}{\sigma^2} \right) \]

*https://en.wikipedia.org/wiki/Delaunay_triangulation
Graph Frequency Examples (US Temperature)

- Weather stations from 100 most populated cities.
- Graph connections from Delaunay Triangulation*. 
  \[
  w_{i,j} = \exp\left(\frac{-\|t_i - t_j\|^2}{\sigma^2}\right)
  \]
  
  Edge weights

V2: 1\textsuperscript{st} AC component

*https://en.wikipedia.org/wiki/Delaunay triangulation
Graph Frequency Examples (US Temperature)

- Weather stations from 100 most populated cities.
- Graph connections from Delaunay Triangulation*.

\[ w_{i,j} = \exp \left( -\frac{\|I_i - I_j\|^2}{\sigma^2} \right) \]

Location diff.

Edge weights

V3: 2\textsuperscript{nd} AC component

*https://en.wikipedia.org/wiki/Delaunay triangulation
Graph Frequency Examples (US Temperature)

- Weather stations from 100 most populated cities.
- Graph connections from Delaunay Triangulation.

\[ w_{i,j} = \exp \left( -\frac{\|t_i - t_j\|_2^2}{\sigma^2} \right) \]

edge weights

V4: 9th AC component
Variants of Graph Laplacians

- **Graph Fourier Transform (GFT)** is eigen-matrix of graph Laplacian $L$.

  \[
  L = V \Sigma V^T
  \]

- Other definitions of graph Laplacians:
  - **Normalized** graph Laplacian:
    \[
    L_n = D^{-1/2} LD^{-1/2} = I - D^{-1/2} AD^{-1/2}
    \]
  - **Random walk** graph Laplacian:
    \[
    L_{rw} = D^{-1} L = I - D^{-1} A
    \]
  - **Generalized** graph Laplacian [1]:
    \[
    L_g = L + D^*
    \]

**Characteristics:**
- Normalized.
- Symmetric.
- No DC component.
- Asymmetric.
- Eigenvectors not orthog.
- Symmetric.
- $L$ plus self loops.
- Defaults to DST, ADST.

---

**GSP:** SP framework that unifies concepts from multiple fields.

**Graph Signal Processing** (GSP)

- **Partial Differential Eq’ns**
- **Machine Learning**
- **Combinatorial Graph Theory**
  - graphical model, manifold learning, classifier learning
  - Max cut, graph transformation

- **Spectral Graph Theory**
  - eigen-analysis of graph Laplacian, adjacency matrices

- **Computer Graphics**
  - Laplace equation
  - Laplace-Beltrami operator

- **Computer Vision**
  - spectral clustering

- **DSP**
  - eigen-analysis of graph Laplacian, adjacency matrices
Outline

• GSP Fundamentals

• GSP for Image Compression
  • Optimality of GFT

• GSP for Inverse Imaging
  • Graph Laplacian Regularizer
  • Reweighted Graph TV

• Deep GLR

• Ongoing & Future Work
GFT for Image Compression

- DCT are **fixed** basis. Can we do better?
- **Idea:** use *adaptive* GFT to improve sparsity [1].

1. Assign edge weight 1 to adjacent pixel pairs.
2. Assign edge weight 0 to sharp signal discontinuity.
3. Compute GFT for transform coding, transmit coeff. \[
\tilde{x} = V^T x
\]
4. Transmit bits (*contour*) to identify chosen GFT to decoder (*overhead of GFT*).

---


GFT for Image Compression

• DCT are *fixed* basis. Can we do better?
• **Idea**: use *adaptive* GFT to improve sparsity [1].

1. Assign edge weight 1 to adjacent pixel pairs.
2. Assign edge weight 0 to sharp signal discontinuity.
3. Compute GFT for transform coding, transmit coeff. 
   \[ \tilde{x} = V^T x \]
4. Transmit bits (*contour*) to identify chosen GFT to decoder (*overhead of GFT*).


GFT: Derivation of Optimal Edge Weights

- Assume a 1D 1st-order \textit{autoregressive (AR) process} \( x = [x_1, \ldots, x_N]^T \) where,

\[
\begin{align*}
x_k &= \begin{cases} 
\eta & k = 1 \\
x_{k-1} + e_k & 1 < k \leq N
\end{cases}
\end{align*}
\]

0-mean r.v. with large var. \( \sigma^2 \)

0-mean r.v. with var. \( \sigma_k^2 \)

\[
x_1 = \eta \\
x_2 - x_1 = e_2 \\
\vdots \\
x_N - x_{N-1} = e_N
\]

\[
F = \begin{bmatrix}
1 & 0 & 0 & 0 & 0 & 0 & 0 \\
-1 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & \cdots & \cdots & 0 & 0 & 0 & 0 \\
0 & 0 & -1 & 1 & 0 & 0 & 0 \\
0 & 0 & 0 & \cdots & \cdots & 0 & 0 \\
0 & 0 & 0 & 0 & -1 & 1 & 0 \\
0 & 0 & 0 & 0 & 0 & -1 & 1
\end{bmatrix}, \quad b = \begin{bmatrix}
\eta \\
e_2 \\
\vdots \\
e_N
\end{bmatrix}
\]

\[
Fx = b, \quad x = F^{-1}b
\]
GFT: Derivation of Optimal Edge Weights

- Covariance matrix

\[
C = E[(x - \mu)(x - \mu)^T] \\
= E[x x^T] = E[F^{-1}bb^T(F^{-1})^T] \\
= F^{-1}E[bb^T](F^{-1})^T
\]

- Precision matrix (tri-diagonal)

\[
Q = C^{-1} = \begin{bmatrix}
\frac{1}{\sigma_2^2} + \frac{1}{\sigma_2^2} & -\frac{1}{\sigma_2^2} & 0 & \cdots & 0 \\
-\frac{1}{\sigma_2^2} & \frac{1}{\sigma_2^2} + \frac{1}{\sigma_3^2} & -\frac{1}{\sigma_2^2} & \cdots & 0 \\
0 & \cdots & \frac{1}{\sigma_N^2} & \cdots & 0 \\
0 & \cdots & -\frac{1}{\sigma_N^2} & \frac{1}{\sigma_N^2} + \frac{1}{\sigma_N^2} & -\frac{1}{\sigma_N^2} \\
0 & \cdots & 0 & -\frac{1}{\sigma_N^2} & \frac{1}{\sigma_N^2}
\end{bmatrix}
\]

\[
E[bb^T] = \begin{bmatrix}
\sigma_2^2 & 0 & \cdots & 0 \\
0 & \sigma_2^2 & \cdots & \vdots \\
\vdots & \ddots & \ddots & 0 \\
0 & \cdots & 0 & \sigma_N^2
\end{bmatrix}
\]

Graph Laplacian matrix!

\[
\approx L
\]

\[
1 \leftrightarrow 2 \leftrightarrow 3 \rightarrow \ldots \rightarrow N
\]
Multi-resolution-GFT Implementation

Fig. 2. MR-GFT coding system for PWS images.

Experimentation

• Setup
  - Test images: depth maps of *Teddy* and *Cones*, and graphics images of *Dude* and *Tsukuba*.
  - Compare against: HR-DCT, HR-SGFT, SAW, MR-SGFT in H.264.

• Results

- **Teddy**
  - HR-DCT: 6.8dB
  - HR-SGFT: 5.9dB
  - SAW: 2.5dB
  - MR-SGFT: 1.2dB

- **Cones**

- **Dude**

- **Tsukuba**

- **Legend**
  - MR-GFT
  - MR-UGFT
  - SAW
  - HR-UGFT
  - HR-DCT
Subjective Results

HR-DCT  

HR-SGFT  

MR-GFT
Summary of GFT for Image Coding

• Optimality of GFT for AR model.

• Variants of GFT for prediction residuals, anti-correlated pixels.

• Fast implementation (w/o eigen-decomposition) via Graph Lifting Transform (GLT) [1] or Fast Graph Fourier Transform (FGFT) [2].


Graph-Signal Sampling / Encoding for 3D Point Cloud

• **Problem**: Point clouds require encoding specific 3D coordinates.

• **Assumption**: smooth 2D manifold in 3D space.

• **Proposal**: progressive 3D geometry rep. as series of graph-signals.
  1. adaptively identifies new samples on the manifold surface, and
  2. encodes them efficiently as graph-signals.

• **Example**:
  1. Interpolate \( i^{th} \) iteration samples (black circles) to a **continuous kernel** (mesh), an approximation of the target surface \( S \).
  2. New sample locations, **knots** (squares), are located on the kernel surface.
  3. **Signed distances** between knots and \( S \) are recorded as sample values.
  4. **Sample values** (green circles) are encoded as a **graph-signal via GFT**.
Graph-Signal Sampling / Encoding for 3D Point Cloud

• **Experimental Results:**

![Graphs and charts showing experimental results](image1)

Pre-Demosiac Light Field Image Compression Using Graph Lifting Transform

**Problem**: Sub-aperture images in Light field data are huge.

**Proposal**: postpone demosiacking to decoder.

- **Raw Lenselet Image**
- **Demosaicked Image**
  - Demosaicing
  - \( \times 3 \)
  - Calibration (Scaling, Transition, Rotation)
  - \( \times \approx 1.5 \)
- **Calibrated Color Image**
- **Sub-aperture Images**
- **Image Coding**

**Raw Lenselet Image**

**Calibrated Lenselet Image**

**Sub-aperture space**

**Re-arranged on Calibrated Image**

**Re-arranged in 4D space**

**Graph-based lifting transform**

24
Pre-Demosiac Light Field Image Compression Using
Graph Lifting Transform

• Experimental Results:

Dataset: EPFL light field image dataset
Baseline: All-intra HEVC coding in YUV4:2:0 and RGB 4:4:4

Outline

• GSP Fundamentals

• GSP for Image Compression
  • Optimality of GFT

• GSP for Inverse Imaging
  • Graph Laplacian Regularizer
    • Image denoising, contrast enhancement
  • Reweighted Graph TV

• Deep GLR

• Ongoing & Future Work
Graph Laplacian Regularizer

- $x^T L x$ (graph Laplacian regularizer) [1]) is one smoothness measure.

$$x^T L x = \frac{1}{2} \sum_{i,j} w_{i,j} (x_i - x_j)^2 = \sum_k \lambda_k \tilde{x}_k^2$$

- **Signal Denoising**: signal smooth in nodal domain
  
  observation
  
  desired signal

- **MAP Formulation**: signal contains mostly low graph freq.
  
  noise

$$y = x + v$$

$$\min_x \|y - x\|_2^2 + \mu x^T L x$$

$$(I + \mu L)x^* = y$$

Graph Laplacian Regularizer

- \( x^T L x \) (graph Laplacian regularizer) [1]) is one smoothness measure.

\[
x^T L x = \frac{1}{2} \sum_{i,j} w_{i,j} (x_i - x_j)^2 = \sum_k \lambda_k \tilde{x}_k^2
\]

- **Signal Denoising:** signal smooth in nodal domain

- **MAP Formulation:**

\[
y = x + v
\]

\[
\min_{x} \|y - x\|_2^2 + \mu x^T L x
\]

\[
(I + \mu L) x^* = y
\]

update edge weights

Bilateral filter weights

\[
w_{i,j} = \exp \left( -\frac{\|x_i - x_j\|_2^2}{\sigma_1^2} \right) \exp \left( -\frac{\|l_i - l_j\|_2^2}{\sigma_2^2} \right)
\]

pixel intensity diff.

pixel location diff.

linear system of eqn’s w/ sparse, symmetric PD matrix

---

Optimal Graph Laplacian Regularization for Denoising

• Adopt a patch-based recovery framework, for a noisy patch $p_0$

1. Find $K-1$ patches similar to $p_0$ in terms of Euclidean distance.
2. Compute feature functions, leading to edge weights and Laplacian.
3. Solve the unconstrained quadratic optimization:

$$q^* = \arg \min_q \|p_0 - q\|^2_2 + \lambda q^T L q \quad \Rightarrow \quad q = (I + \lambda L)^{-1} p_0$$

to obtain the denoised patch.

• Aggregate denoised patches to form an updated image.

• Denoise the image iteratively to gradually enhance its quality.

• **Optimal Graph Laplacian Regularization for Denoising (OGLRD).**

---

Denoising Experiments (natural images)

- Subjective comparisons ($\sigma_1 = 40$)

Original

Noisy, 16.48 dB

K-SVD, 26.84 dB

BM3D, 27.99 dB

PLOW, 28.11 dB

OGLR, 28.35 dB
Denoising Experiments (depth images)

- Subjective comparisons ($\sigma = 30$)

Original  | Noisy, 18.66 dB  | BM3D, 33.26 dB  | NLGBT, 33.41dB  | OGLR, 34.32 dB

GLR for Joint Dequantization / Contrast Enhancement

- **Retinex decomposition model:**
  \[ y = T \circ l + z \]
  - \( y \) = reflectance
  - \( l \) = scalar
  - \( r \) = illumination
  - \( z \) = noise

- **Objective:** general smoothness for luminance, smoothness w/ negative edges for reflectance.

- **Constraints:** quantization bin constraints

- **Solution:** Alternating accelerated proximal gradient alg [1].

---

Experimental Results
Experimental Results
Experimental Results
Outline

• GSP Fundamentals

• GSP for Image Compression
  • Optimality of GFT

• GSP for Inverse Imaging
  • Graph Laplacian Regularizer
  • Reweighted Graph TV
    • 3D point cloud denoising, image deblurring

• Deep GLR

• Ongoing & Future Work
GTV for Point Cloud Denoising

• Acquisition of point cloud introduces noise.
• Point cloud is irregularly sampled 2D manifold in 3D space.
• Not appropriate to apply GTV directly on 3D coordinates \([1]\).
  • only a singular 3D point has zero GTV value.

\[
\sum_i |f(x_i) - f(x_{i-1})| \\
\sum_i |y_i - y_{i-1}|
\]

• Proposal: Apply GTV is to the surface normals of 3D point cloud—a generalization of TV to 3D geometry.

GTV for Point Cloud Denoising

- Acquisition of point cloud introduces noise.
- Point cloud is irregularly sampled 2D manifold in 3D space.
- Not appropriate to apply GTV directly on 3D coordinates [1].
  - only a singular 3D point has zero GTV value.

• Proposal: Apply GTV is to the surface normals of 3D point cloud—a generalization of TV to 3D geometry.

GTV for Point Cloud Denoising

• Acquisition of point cloud introduces noise.
• Point cloud is irregularly sampled 2D manifold in 3D space.
• Not appropriate to apply GTV directly on 3D coordinates [1].
  • only a singular 3D point has zero GTV value.

• Proposal: Apply GTV is to the surface normals of 3D point cloud—a generalization of TV to 3D geometry.

Algorithm Overview

• Use graph total variation (GTV) of surface normals over the K-NN graph:

$||n||_{GTV} = \sum_{i, j \in E} w_{i, j} ||n_i - n_j||_1$

• Denoising problem as $l_2$-norm fidelity plus GTV of surface normals:

$$\min_{p, n} \left( \frac{1}{2} ||q - p||_2^2 + \gamma \sum_{i, j \in E} w_{i, j} ||n_i - n_j||_1 \right)$$

• Surface normal estimation of $n_i$ is a nonlinear function of $p_i$ and neighbors.

Proposal:
1. Partition point cloud into two independent classes (say red and blue).
2. When computing surface normal for a red node, use only neighboring blue points.
3. Solve convex optimization for red (blue) nodes alternately.

Experimental Results – Visual Comparison

Anchor model ($\sigma=0.3$)

(a) ground truth  (b) noisy input  (c) APSS

(d) RIMLS  (e) MRPCA  (f) proposed
Experimental Results – Visual Comparison

Daratech model ($\sigma=0.3$)
Reweighted Graph Total Variation

• TV on graphs.

Gradient of nodes on the graph:

\[(\nabla_i x)_j \triangleq x_j - x_i,\]

Reweighted Graph TV:

\[
\|x\|_{RGTV} = \sum_{i \in V} \| \text{diag}(W_{i,\cdot}(x)) \nabla_i x \|_1 \\
= \sum_{i=1}^{N} \sum_{j=1}^{N} w_{i,j}(x_i, x_j)|x_j - x_i|,
\]

where

\[
w_{i,j} = \exp \left( \frac{-\|x_i - x_j\|^2}{\sigma_1^2} \right).
\]


Background for Image Deblurring

• Image blur is a common image degradation.
• Typically, blur process is modeled:

\[ y = k \circledast x \]

where \( y \) is the blurry image, \( k \) is the blur kernel, \( x \) is the original sharp image.

• **Blind-image deblurring** focuses on estimating blur kernel \( k \).
• Given \( k \), problem becomes *de-convolution*. 
Observation

- **Skeleton image**:
  - PWS image keeping only structural edges.
  - Proxy to estimate blur kernel $k$. 

![Images](image1.png)  
(a) ![Images](image2.png)  
(b) ![Images](image3.png)  
(c) 

![Images](image4.png)  
(d) ![Images](image5.png)  
(e) ![Images](image6.png)  
(f)
Observation

- **Skeleton image**:  
  - PWS image keeping only structural edges.  
  - Proxy to estimate blur kernel $k$. 

(a) (b) (c) (d) (e) (f)
Observation

- **Skeleton image**:
  - PWS image keeping only structural edges.
  - Proxy to estimate blur kernel $k$. 

![Skeleton image](image)
Our algorithm

• The optimization function can be written as follows,
  \[ \hat{x}, \hat{k} = \arg\min_{x,k} \varphi(x \otimes k - b) + \mu_1 \cdot \theta_x(x) + \mu_2 \cdot \theta_k(k) \]

• Assume \( L_2 \) norm for fidelity term \( \varphi(\cdot) \).

• \( \theta_x(\cdot) = \text{RGTV}(\cdot) \).

• \( \theta_k(\cdot) = \| \cdot \|_2 \), assuming zero mean Gaussian distribution of \( k \).

• RGTV is non-differentiable and non-convex.

Solution:

• Solve \( x \) and \( k \) alternatingly.

• For \( x \), spectral interpretation of GTV, fast spectral filter.
Workflow

Blurry Image

Skeleton Image Reconstruction

Kernel Estimation

Reconstruction
Experimental Results

Fig. 8. Real Blind Motion Deblurring Example. Image size: 618 x 464, kernel size: 69 x 69. (a) Blurry image. (b) Krishnan et al. [33], (c) Levin et al. [32], (d) Michaeli & Irani [35], (e) Pan et al. [4]. (f) The proposed Algorithm 1. The images are better viewed in full size on computer screen.
Experimental Results
Experiment Results
Outline

• GSP Fundamentals

• GSP for Image Compression
  • Optimality of GFT

• GSP for Inverse Imaging
  • Graph Laplacian Regularizer
  • Reweighted Graph TV

• Deep GLR

• Ongoing & Future Work
Unrolling Graph Laplacian Regularizer

- Recall MAP formulation of denoising problem with quadratic graph Laplacian regularizer:
  \[
  \min_x \| y - x \|^2_2 + \mu x^T L x
  \]

  fidelity term  \rightarrow  \text{smoothness prior}

- Solution is system of linear equations:
  \[
  (I + \mu L)x^* = y
  \]

  linear system of eqn’s w/ sparse, symmetric PD matrix

Unrolling Graph Laplacian Regularizer

• Recall MAP formulation of denoising problem with quadratic graph Laplacian regularizer:

\[
\min_x \|y - x\|_2^2 + \mu x^T L x
\]

fidelity term \hspace{1cm} \text{smoothness prior}

• Solution is system of linear equations:

\[
(I + \mu L)x^* = y
\]

linear system of eqn’s w/ sparse, symmetric PD matrix

Q: what is the “most appropriate” graph?

Unrolling Graph Laplacian Regularizer

• Recall MAP formulation of denoising problem with quadratic graph Laplacian regularizer:

\[
\min_x \|y - x\|_2^2 + \mu x^T L x
\]

fidelity term \hspace{1cm} \text{smoothness prior}

• Solution is system of linear equations:

\[
(I + \mu L) x^* = y
\]

linear system of eqn’s w/ sparse, symmetric PD matrix

Q: what is the “most appropriate” graph?

Bilateral weights:

\[
w_{i,j} = \exp\left(\frac{-\|x_i - x_j\|_2^2}{\sigma_1^2}\right) \exp\left(\frac{-\|l_i - l_j\|_2^2}{\sigma_2^2}\right)
\]

Unrolling Graph Laplacian Regularizer

- **Deep Graph Laplacian Regularization:**
  1. Learn features $f$'s using CNN.
  2. Compute distance from features.
  3. Compute edge weights using Gaussian kernel.
  4. Construct graph, solve QP.

$$w_{ij} = \exp\left(-\frac{\text{dist}(i, j)}{2\epsilon^2}\right),$$

$$\text{dist}(i, j) = \sum_{n=1}^{N} (f_n(i) - f_n(j))^2.$$

---

**Fig. 1.** Block diagram of the proposed GLRNet which employs a graph Laplacian regularization layer for image denoising.


Unrolling Graph Laplacian Regularizer

Fig. 3. Network architectures of CNN$_F$, CNN$_\tilde{\gamma}$ and CNN$_\mu$ in the experiments. Data produced by the decoder of CNN$_F$ is colored in orange.

Unrolling Graph Laplacian Regularizer

![Diagram](image)

**Fig. 2.** Block diagram of the overall DeepGLR framework.

- **Graph Model** guarantees numerical stability of solution:

\[
(I + \mu L) x^* = y
\]

- **Thm 1**: condition number \( \kappa \) of matrix satisfies [1]:

\[
\kappa \leq 1 + 2 \mu d_{\text{max}},
\]

- **Observation**: By restricting search space of CNN to degree-bounded graphs, we achieve robust learning.

Experimental Results – Numerical Comparison

- Trained on AWGN on 5 images, patches of size 26-by-26.
- Batch size is 4, model is trained for 200 epochs.
- Trained for both known and blind noise variance.

Table 1. Average PSNR (dB) and SSIM values of different methods for Gaussian noise removal. The best results for each metric is highlighted in boldface.

<table>
<thead>
<tr>
<th>Noise</th>
<th>Metric</th>
<th>BM3D</th>
<th>WNNM</th>
<th>OGLR</th>
<th>DnCNN-S</th>
<th>DnCNN-B</th>
<th>DeepGLR-S</th>
<th>DeepGLR-B</th>
</tr>
</thead>
<tbody>
<tr>
<td>25</td>
<td>PSNR</td>
<td>29.95</td>
<td>30.28</td>
<td>29.78</td>
<td>30.41</td>
<td>30.33</td>
<td>30.26</td>
<td>30.21</td>
</tr>
<tr>
<td></td>
<td>SSIM</td>
<td>0.8496</td>
<td>0.8554</td>
<td>0.8463</td>
<td>0.8609</td>
<td>0.8594</td>
<td>0.8599</td>
<td>0.8557</td>
</tr>
<tr>
<td></td>
<td>SSIM</td>
<td>0.7920</td>
<td>0.8018</td>
<td>0.7949</td>
<td>0.8080</td>
<td>0.8091</td>
<td>0.8125</td>
<td>0.8063</td>
</tr>
<tr>
<td>50</td>
<td>PSNR</td>
<td>26.69</td>
<td>27.08</td>
<td>26.58</td>
<td>27.15</td>
<td>27.18</td>
<td>27.25</td>
<td>27.12</td>
</tr>
<tr>
<td></td>
<td>SSIM</td>
<td>0.7651</td>
<td>0.7769</td>
<td>0.7539</td>
<td>0.7809</td>
<td>0.7811</td>
<td>0.7852</td>
<td>0.7807</td>
</tr>
</tbody>
</table>

Experimental Results – Numerical Comparison

- DeepGLR has average PSNR of 0.34 dB higher than CDnCNN [1].
- Model-based provides robustness against overfitting.

**Table 2.** Evaluation of different methods for low-light image denoising. The best results for each metric, except for those tested on the training set, are highlighted in boldface.

<table>
<thead>
<tr>
<th>Metric</th>
<th>Noisy</th>
<th>CBM3D</th>
<th>MC-WNNM</th>
<th>CDnCNN (train)</th>
<th>CDnCNN</th>
<th>CDDeepGLR (train)</th>
<th>CDDeepGLR</th>
</tr>
</thead>
<tbody>
<tr>
<td>PSNR</td>
<td>20.36</td>
<td>26.08</td>
<td>26.23</td>
<td>33.43</td>
<td>31.26</td>
<td>32.31</td>
<td>31.60</td>
</tr>
<tr>
<td>SSIM Y</td>
<td>0.5198</td>
<td>0.8698</td>
<td>0.8531</td>
<td>0.9138</td>
<td>0.8978</td>
<td>0.9013</td>
<td>0.9028</td>
</tr>
<tr>
<td>SSIM R</td>
<td>0.2270</td>
<td>0.6293</td>
<td>0.5746</td>
<td>0.8538</td>
<td>0.8218</td>
<td>0.8372</td>
<td>0.8297</td>
</tr>
<tr>
<td>SSIM G</td>
<td>0.4073</td>
<td>0.8252</td>
<td>0.7566</td>
<td>0.8979</td>
<td>0.8828</td>
<td>0.8840</td>
<td>0.8854</td>
</tr>
<tr>
<td>SSIM B</td>
<td>0.1823</td>
<td>0.5633</td>
<td>0.5570</td>
<td>0.8294</td>
<td>0.7812</td>
<td>0.8138</td>
<td>0.7997</td>
</tr>
</tbody>
</table>


Experimental Results – Visual Comparison

- trained on Gaussian noise, tested on low-light images in (RENOIR).
- Competing methods: DnCNN [1], noise clinic [2].
- outperformed DnCNN by 5.52 dB, and noise clinic by 1.87 dB.

Experimental Results – Visual Comparison

- trained on Gaussian noise, tested on low-light images in (RENOIR).
- Competing methods: DnCNN [1], noise clinic [2].
- outperformed DnCNN by 5.52 dB, and noise clinic by 1.87 dB.

Experimental Results – Visual Comparison

- trained on Gaussian noise, tested on low-light images in (RENOIR).
- Competing methods: DnCNN [1], noise clinic [2].
- outperformed DnCNN by 5.52 dB, and noise clinic by 1.87 dB.

Outline

• GSP Fundamentals

• GSP for Image Compression
  • Optimality of GFT

• GSP for Inverse Imaging
  • Graph Laplacian Regularizer
  • Reweighted Graph TV: 3D Point Cloud Denoising

• Deep GLR

• Ongoing & Future Work
Summary

• Frequencies for Graphs
  • Optimality of GFT for signal decorrelation
  • Compression for PWS images, point cloud, light field images

• GSP for Inverse Imaging
  • PWS-promoting Graph Laplacian Regularizer, RGTV
  • Image / point cloud denoising, deblurring, contrast enhancement

• Hybrid graph-based / data-driven approach
  • Robustness against CNN overfitting
Ongoing & Future Work

• Graph learning given small data
  • Bayesian networks, DAG

• Unrolling of graph-based convex optimization
  • Unrolling of ADMM, proximal gradient with GTV prior, convex set constraints.

• Graph sampling
  • Fast sampling w/o eigen-decomposition, matrix inverse [1]
  • Reconstruction-cognizant sampling [2]

Q&A

• Email: genec@yorku.ca
• Homepage: https://www.eecs.yorku.ca/~genec/index.html