# Power Watersheds and Contrast Invariance [1]

Sravan Danda*

*Computer Science and Information Systems,
BITS-Pilani, Goa, India

March 31, 2021

[1] Joint work with Aditya Challa (Indian Institute of Science, Bangalore), B S Daya Sagar (Indian Statistical Institute, Bangalore Centre), and Laurent Najman (ESIEE, France)

# Outline

# Translation and Rotation Invariances

Translation of object does not affect the image class:

*Convolutional Neural Nets* are designed to utilize this inductive bias with shared parameters!

Image Source [2]

[2] On Translation Invariance in CNNs: Convolutional Layers can Exploit Absolute Spatial Location, CVPR 2020

# Contrast Invariance

Increasing/Decreasing contrast should not affect the object boundaries

Image Source [3]

[3] Alpert et al. Image segmentation by probabilistic bottom-up aggregation and cue integration. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, June 2007

# Edge-Weighted Graph Models for Images

$\mathcal{G} = (V, E, W)$ : 4-adjacency edge-weighted graph

1. $V$:- Pixels
2. $W : E \rightarrow \mathbb{R}$ :- Dissimilarity/Similarity between adjacent pixels[4]

[4] depending on application at hand

# Contrast Invariance [6]

$\mathcal{G}' = (V, E, W') \leftrightarrow \mathcal{G} = (V, E, W)$: Same segmentation results

where $W' = T \circ W$ with $T' > 0$ [5]

$\Rightarrow$ Segmentation should depend on relative ordering of edge weights alone and not the actual weights!

[5] $T' > 1$ denotes increase in contrast, $0 < T' < 1$ denotes decrease in contrast

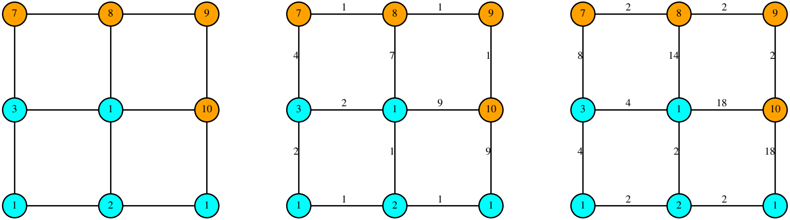
[6] this is contrast defined using graph edge-based image gradients and hence is slightly different from classic notion that uses border pixels as gradients

# Contrast Invariance: Illustration

**Left**: Image with two objects

**Middle**: Edge-weighted graph with weights denoting dissimilarity

**Right**: Doubled edge weights ≈ increase in contrast



# What is Power Watershed Framework?

Cost minimization problems on finite graphs [7]

Minimize

$$Q(\mathbf{x}) = \sum_{e_{ij} \in E} f(w_{ij}) Q_{ij}(x_i, x_j) \tag{1}$$

$\mathbf{x}$: target labels of the nodes

$w_{ij}$: weight of edge $e_{ij}$

$Q_{ij}$: real-valued smooth function in two variables

$f$: increasing function

[7] also works for a more general cost form but combination of pairwise costs has some special properties

# Power Watershed Optimization Framework [8]

Recast an optimization problem into Nested minimization problems

$$\mathbf{x}^{(p)} \rightarrow \mathbf{x}^* \quad (?)$$

where

$$\mathbf{x}^{(p)} = \arg\min_{\mathbf{x}} Q^{(p)}(\mathbf{x})$$

and

$$Q^{(p)}(\mathbf{x}) = \sum_{e_{ij} \in E} f(w_{ij})^p Q_{ij}(x_i, x_j) \tag{2}$$

[8] Laurent Najman. Extending the Power Watershed framework thanks to Γ-convergence. SIAM Journal on Imaging Sciences, 10(4):2275–2292, November 2017.

# Why is Power Watershed Framework Useful?

1. **Contrast Invariance**:

   $lim_{p \to \infty} \mathbf{x}^{(p)}$ invariant to relative ordering of edge weights.

2. **Empirically Similar Results**:

   $\arg\min_{\mathbf{x}} Q^{(p)}(\mathbf{x}) \approx \arg\min_{\mathbf{x}} Q(\mathbf{x})$

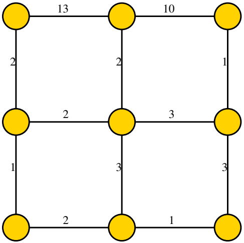
3. **Empirically Faster Computation**:

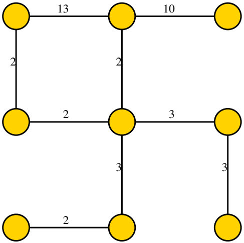
   The answer depends on a substructure of the image graph: Union of Maximum/Minimum Spanning Trees (Similarity/Dissimilarity) of the image graph

# What is a UMaxST/UMinST?

**Left**: Image Similarity Graph

**Right**: UMaxST: Induced subgraph with edges of all MaxSTs



# Computing the Power Watershed Solution

Rearrange

$$Q^{(p)}(\mathbf{x}) = \sum_{e_{ij} \in E} f(w_{ij})^p Q_{ij}(x_i, x_j) \quad (3)$$

as

$$Q^{(p)}(\mathbf{x}) = \sum_{i=1}^{l} f(w_i)^p Q_i(\mathbf{x}) \quad (4)$$

such that $f(w_i) < f(w_j)$ if $i < j$ and $l$: number of distinct weights

# Computing the Power Watershed Solution

**Intuition**: Higher weights dominate the cost in the limiting case

1. Set $i = I$ and $M_i$ is the entire solution space.
2. while $i > 1$
   - Compute the set of minimizers $M_{i-1} = \arg\min_{\mathbf{x} \in M_i} Q_i(\mathbf{x})$
3. Return arbitrary $\mathbf{x} \in M_1$.

# Intuition Behind the Algorithm

$$Q^{(p)}(x_1, x_2) = w_1^p \left[(x_1 - 1)^2 + x_2^2\right] + w_2^p \left[(x_1 - x_2)^2\right] \qquad (5)$$

where $w_1 = w$ and $w_2 = 2w$

Direct computation:

$$\hat{x_1}^{(p)} = \frac{2^p}{2^{p+1} + 1} \qquad (6)$$

$$\hat{x_2}^{(p)} = \frac{2^p + 1}{2^{p+1} + 1} \qquad (7)$$

$lim_{p \to \infty} \hat{\mathbf{x}}^p = (\frac{1}{2}, \frac{1}{2})$

# Intuition Behind the Algorithm

At first pass

$$\frac{Q^{(p)}(x_1, x_2)}{w_2^p} = \frac{((x_1 - 1)^2 + x_2^2)}{2^p} + (x_1 - x_2)^2 \tag{8}$$

subspace $M_2 = \{(x_1, x_2) \in \mathbb{R}^2 \mid x_1 = x_2\}$.

At second pass:

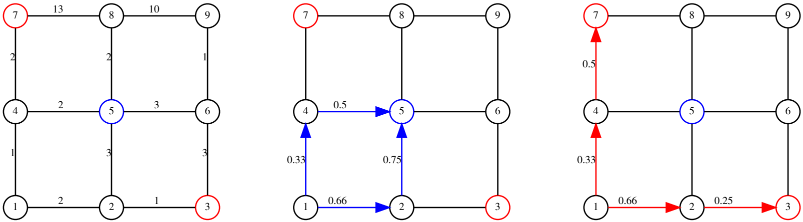
$M_1 = (\frac{1}{2}, \frac{1}{2})$

# Random Walker [9]

**Left**: Image with two labels

**Middle**: Probability that node 1 is labelled blue

**Right**: Probability that node 1 is labelled red




[9] Leo Grady. Random walks for image segmentation. IEEE PAMI, 28(11):1768–1783, 2006

# Random Walker as a Cost Minimization

$$RWCost(\mathbf{x}) = \frac{1}{2} \sum_{e_{ij} \in E} w_{ij}(x_i - x_j)^2, \quad (9)$$

$$subject\ to\ \mathbf{x}_{seed} = \mathbf{f}_{seed}, \quad (10)$$

# Random Walker: Matrix Solver

$$L = \begin{pmatrix} L_{seed} & B \\ B^T & L_U \end{pmatrix} \tag{11}$$

$$RWCost(\mathbf{x}) = \frac{1}{2}(\mathbf{x}_{seed}^T L_{seed} \mathbf{x}_{seed} + 2\mathbf{x}_U^T B^T \mathbf{x}_{seed} + \mathbf{x}_U^T L_U \mathbf{x}_U), \tag{12}$$

Solution satisfies:

$$L_U \mathbf{x}_U = -B^T \mathbf{x}_{seed} \tag{13}$$

$$L_U X = -B^T S \tag{14}$$

# Power Watershed-based Random Walker [10]

$$RWCost^{(p)}(\mathbf{x}) = \frac{1}{2} \sum_{e_{ij} \in E} w_{ij}^{p} (x_i - x_j)^2, \quad (15)$$

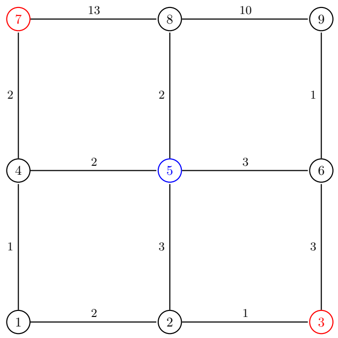
$$subject\ to\ \mathbf{x}_{seed} = \mathbf{f}_{seed}, \quad (16)$$

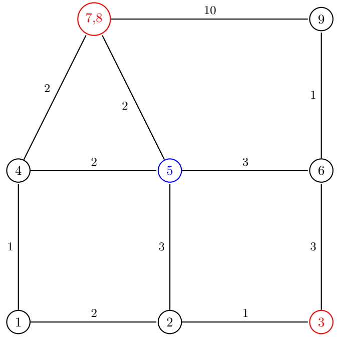
[10] Camille Couprie, Leo Grady, Laurent Najman, and Hugues Talbot. Power watershed: A unifying graph-based optimization framework. IEEE Trans. Pattern Anal. Mach. Intell., 33(7):1384–1399, 2011.

# Power Watershed-based Random Walker: A Nested Random Walker

**Left**: Image with two labels

**Right**: First Pass of PW

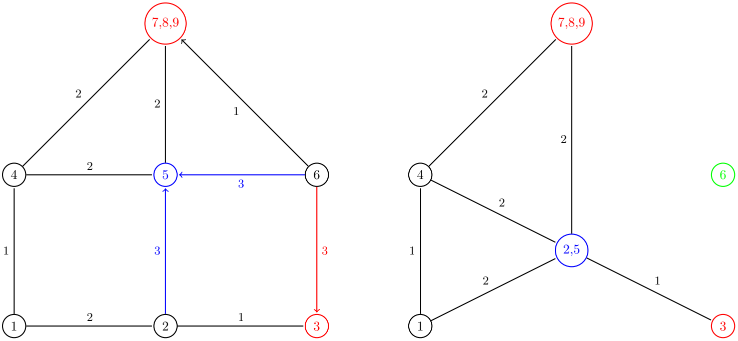


# Power Watershed-based Random Walker: A Nested Random Walker

**Left**: Second Pass of PW

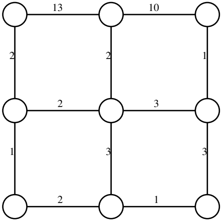
**Right**: Label of 6 obtained by solving a RW on a small subgraph

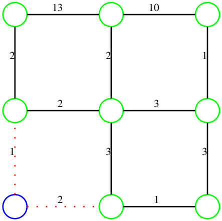


# Isoperimetric Segmentation

$$IsoCost(A) = \frac{W(A, \bar{A})}{min\{|A|, |\bar{A}|\}}, \tag{17}$$

Avoids small cuts!



# Isoperimetric Partitioning: NP-Hard Problem

$$IsoCost(A) = \frac{\mathbf{x}^T L \mathbf{x}}{min\{\mathbf{x}^T \mathbf{1}, (\mathbf{1} - \mathbf{x})^T \mathbf{1}\}}, \quad (18)$$

where

$L$: unnormalized graph Laplacian,

$$x_i = \begin{cases} 1 & \text{if} \quad i \in A \\ 0 & i \in \bar{A} \end{cases} \quad (19)$$

# Continuous Relaxation Solution to Isoperimetric Partitioning

Set $x_r = 0$ and

$$Minimize\ \frac{\mathbf{x}_{-r}^{T} L_{-r} \mathbf{x}_{-r}}{min\{\mathbf{x}_{-r}^{T}\mathbf{1}, (\mathbf{1} - \mathbf{x}_{-r})^{T}\mathbf{1}\}}, subject\ to\ each\ x_i \in [0, 1] \tag{20}$$

Lagrange Multipliers $\Rightarrow$ Enough to solve

$$L_{-r}\mathbf{x}_{-r} = \mathbf{1} \tag{21}$$

# Fast Isoperimetric Partitioning

Maximum spanning tree solver [11]

$$L_{-r}^{MaxST}\mathbf{x}_{-r} = \mathbf{1} \tag{22}$$

Power watershed solution [12]

$$L_{-r}^{UMaxST}\mathbf{x}_{-r} = \mathbf{1} \tag{23}$$

[11] Leo Grady. Fast, quality, segmentation of large volumes - isoperimetric distance trees. In Computer Vision - ECCV 2006, 9th European Conference on Computer Vision, Graz, Austria, May 7-13, 2006, Proceedings, Part III, pages 449–462, 2006.

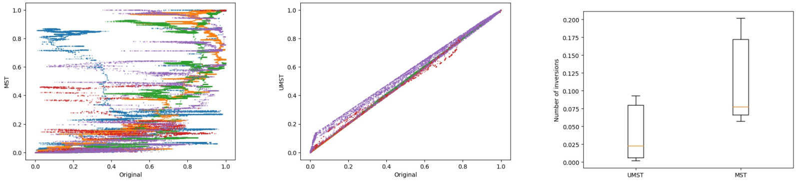
[12] Sravan Danda, Aditya Challa, BS Daya Sagar, and Laurent Najman. Revisiting the isoperimetric graph partitioning problem. IEEE Access, 7:50636–50649, 2019.

# Comparison of MaxST and UMaxST Solutions [14]

**Left**: Matrix Solver on a MaxST vs Original Image graph.
Monotonous ⇒ consistent solutions [13]

**Middle**: UMaxST vs Original Image Graph

**Right**: Boxplot of inversions



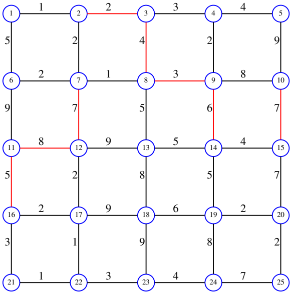

[13] Each color is the solution map comparison of an image.

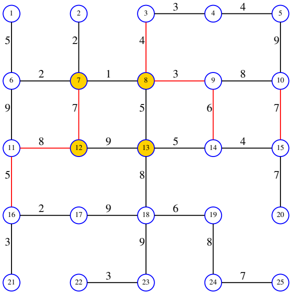
[14] Sravan Danda, Aditya Challa, BS Daya Sagar, and Laurent Najman. Revisiting the isoperimetric graph partitioning problem. IEEE Access, 7:50636–50649, 2019.

# Mutex Watershed: Graph-Cut for Simultaneous Similarity and Dissimilarity [15]

**Left**: An image graph capturing similarities and dissimilarities with varied levels of confidence

**Right**: An ambiguous partitioning




[15] Steffen Wolf et al. The mutex watershed and its objective: Efficient, parameter-free graph partitioning. IEEE PAMI, 2020.

# Mutex Watershed: Greedy Approximation to Multi-Cut [16]

Subject to the consistency constraint, minimizing cut edges is same as minimizing negative sum of leftover edges:

$$\text{Minimize } Q(\mathbf{a}) = -\sum_{e \in E} a_e w_e$$
$$\text{subject to } \mathbf{a} \in \{0,1\}^{|E|}, \mathcal{C}_1(A) = \emptyset \text{ with } A = \{e \in E | a_e = 1\} \quad (24)$$

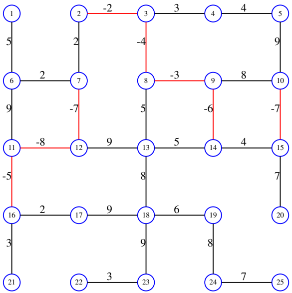
NP-hard!

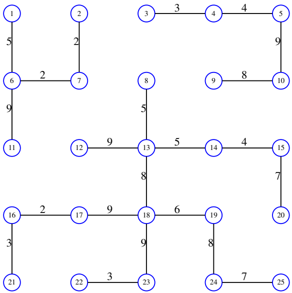
[16] Steffen Wolf et al. The mutex watershed and its objective: Efficient, parameter-free graph partitioning. IEEE PAMI, 2020.

# Mutex Watershed: Greedy Approximation to Multi-Cut [17]

**Left**: Adding edges greedily w.r.t. the confidence subject to cycle constraint

**Right**: Final partitioning removing the dissimilar edges




[17] Steffen Wolf et al. The mutex watershed and its objective: Efficient, parameter-free graph partitioning. IEEE PAMI, 2020.

# Mutex Watershed: Power Watershed Approximation to Multi-Cut

$$\textit{Minimize } Q^{(p)}(\mathbf{a}) = -\sum_{e\in E} a_e w_e^p$$
$$\textit{subject to } \mathbf{a} \in \{0,1\}^{|E|}, \mathcal{C}_1(A) = \emptyset \text{ with } A = \{e \in E | a_e = 1\} \tag{25}$$

# Mutex Watershed: Power Watershed Approximation to Multi-Cut

$E_r = \{e \in E | w_e = w_r\}$ for each $1 \leq r \leq l$

$w_l > \cdots > w_1$ are distinct weights

# Mutex Watershed: Power Watershed Approximation to Multi-Cut

Level 1: No longer NP-Hard

$$\begin{aligned} & \textit{Minimize} \quad -\sum_{e \in E_l} a_e \\ & \textit{subject to } \mathbf{a} \in \{0,1\}^{|E_l|}, \mathcal{C}_1(A) = \emptyset \text{ with } A = \{e \in E_l | a_e = 1\} \end{aligned} \tag{26}$$

$A_l$: Solution space

# Mutex Watershed: Power Watershed Approximation to Multi-Cut

Level 2:

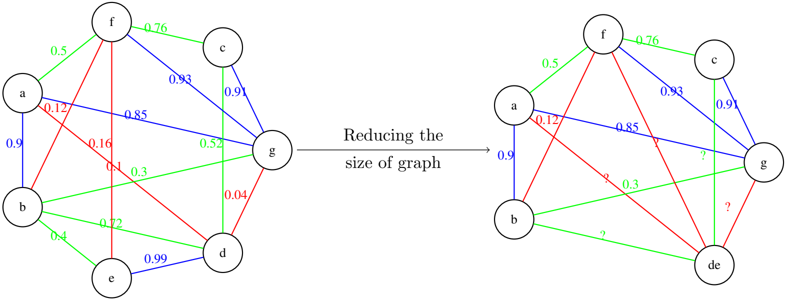
$$\textit{Minimize} \ -\sum_{e \in E_{l-1}} a_e$$
$$\textit{subject to } \mathbf{a} \in \{0,1\}^{|E_{l-1}|}, \mathcal{C}_1(A) = \emptyset \text{ with } A = A_l \cup \{e \in E_{l-1} | a_e = 1\} \tag{27}$$

$A_{l-1}$: Solution space

Continue until edges at all $l$ levels are exhausted

# A Glimpse of Other Applications: Fast Ratio Cut

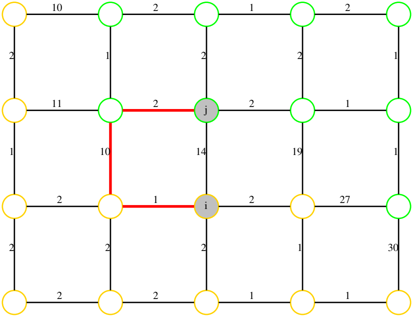
## Fast Spectral Clustering [18]




[18] Aditya Challa, Sravan Danda, B S Daya Sagar, and Laurent Najman. Power spectral clustering. Journal of Mathematical Imaging and Vision, 62(9):1195-1213, 2020.

# A Glimpse of Other Applications: Fast Approximation to Shortest Path Filters

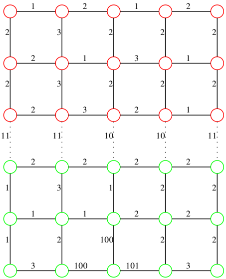
Shortest path pairwise weights




$\Theta_i(j) = 3$

# A Glimpse of Other Applications: Fast Approximation to Shortest Path Filters

Tree Filter [19] as a PW approximation to Shortest Path Weighted Average Filters [20]



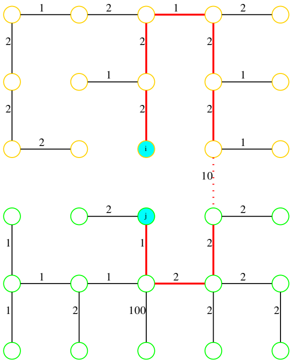

Gradient image

[19] Bao et al. Tree filtering: Efficient structure-preserving smoothing with a minimum spanning tree. IEEE TIP, 23(2):555–569, 2014.

[20] Sravan Danda, Aditya Challa, B S Daya Sagar, and Laurent Najman. Some theoretical links between shortest path filters and minimum spanning tree filters. Journal of Mathematical Imaging and Vision, 61(6):745-762, 2019

# A Glimpse of Other Applications: Fast Approximation to Shortest Path Filters

Pairwise weights on UMinST act as PW approximation




$t_i(j) = 9$

# Summary and Perspectives

1. Scalability of image segmentation algorithms based on cost optimization on graphs
2. Pairwise cost $\Rightarrow$ Enough to work on UMaxST/UMinST
3. Can be used at test phase for end-to-end learned counterparts of these algorithms [21]

[21] end-to-end random walker for example: Lorenzo Cerrone, Alexander Zeilmann, and Fred A Hamprecht. End-to-end learned random walker for seeded image segmentation. In Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, pages 12559–12568, 2019.