Power Watersheds

A unifying graph-based optimization framework

Camille Couprie[1], Leo Grady[2], **Laurent Najman**[1], Hugues Talbot[1]

[1]LIGM, UPE-MLV [2]Siemens Corporate Research

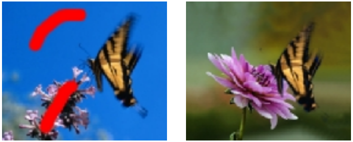
Workshop honouring Professor Jean Serra
Indian Statistical Institute, October 25-26, 2010

Outline

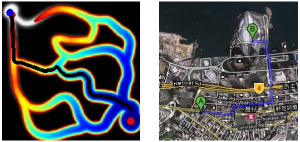
- A new graph-based optimization framework
- Application to image segmentation
- Deblurring with anisotropic-diffusion

What does all those algorithms have in common ?

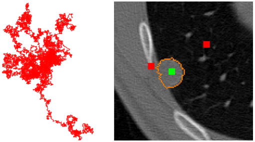
Graph cuts



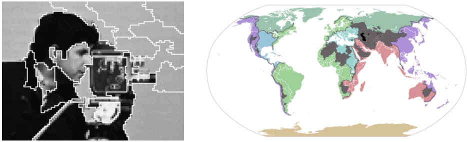
Shortest paths



Random walker

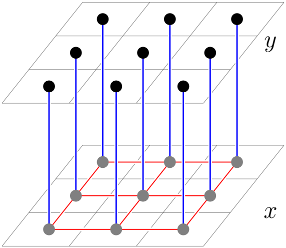


Watersheds



Power Watersheds : An energy minimization framework

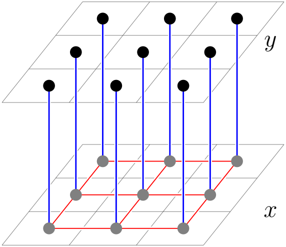
$$\min_x \underbrace{\sum_{e_{ij}\in E} {w_{ij}}^p |x_i - x_j|^q}_{\text{Smoothness term}} + \underbrace{\sum_{v_i\in V} {w_i}^p |x_i - y_i|^q}_{\text{Data term}}$$



Algorithms optimizing this energy :

Power Watersheds : An energy minimization framework

$$\min_x \underbrace{\sum_{e_{ij}\in E} {w_{ij}}^p |x_i - x_j|^q}_{\text{Smoothness term}} + \underbrace{\sum_{v_i \in V} {w_i}^p |x_i - y_i|^q}_{\text{Data term}}$$

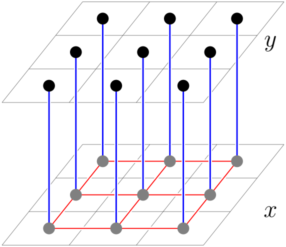


Algorithms optimizing this energy :

p finite, $q = 1$: Graph cuts [Boykov-Joly 2001 (only for 2 labels y)]

Power Watersheds : An energy minimization framework

$$\min_x \underbrace{\sum_{e_{ij}\in E} {w_{ij}}^p |x_i - x_j|^q}_{\text{Smoothness term}} + \underbrace{\sum_{v_i \in V} {w_i}^p |x_i - y_i|^q}_{\text{Data term}}$$

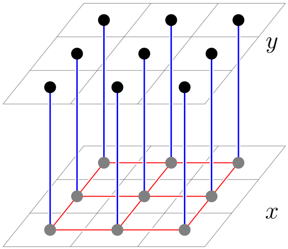


Algorithms optimizing this energy :

p finite, $q = 2$: Random walker [Grady 2006]

Power Watersheds : An energy minimization framework

$$\min_x \underbrace{\sum_{e_{ij}\in E} w_{ij}{}^p |x_i - x_j|^q}_{\text{Smoothness term}} + \underbrace{\sum_{v_i \in V} w_i{}^p |x_i - y_i|^q}_{\text{Data term}}$$

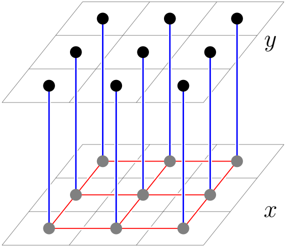


Algorithms optimizing this energy :

$p = q \to \infty$: Shortest paths [Sinop *et al* 2007]

Power Watersheds : An energy minimization framework

$$\min_x \underbrace{\sum_{e_{ij}\in E} {w_{ij}}^p |x_i - x_j|^q}_{\text{Smoothness term}} + \underbrace{\sum_{v_i\in V} {w_i}^p |x_i - y_i|^q}_{\text{Data term}}$$

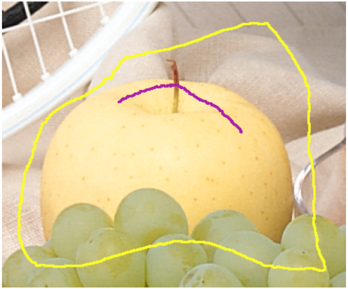


Algorithms optimizing this energy :

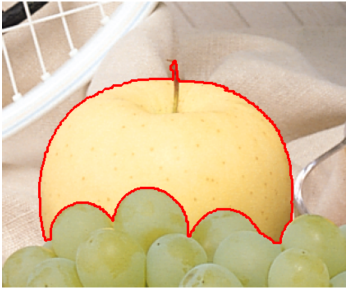
$p \to \infty$, q finite : Power watershed [Couprie *et al* 2009]

Power watershed for image segmentation

Input seeds



Segmentation



Power watershed for image segmentation

$$\min_x \underbrace{\sum_{e_{ij}\in E} {w_{ij}}^p |x_i - x_j|^q}_{\text{Smoothness term}} + \underbrace{\sum_{v_i \in V} {w_i}^p |x_i - y_i|^q}_{\text{Data term}}$$

Power watershed for image segmentation

$$\min_x \underbrace{\sum_{e_{ij}\in E} w_{ij}{}^p |x_i - x_j|^q}_{\text{Smoothness term}} + \underbrace{\sum_{v_i\in V} w_i{}^p |x_i - y_i|^q}_{\text{Data term}}$$

- Vertices = pixels, edges between neighboring pixels

Power watershed for image segmentation

$$\min_x \underbrace{\sum_{e_{ij}\in E} {w_{ij}}^p |x_i - x_j|^q}_{\text{Smoothness term}} + \underbrace{\sum_{v_i \in V} {w_i}^p |x_i - y_i|^q}_{\text{Data term}}$$

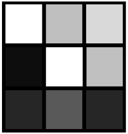
- Vertices = pixels, edges between neighboring pixels
- Pairwise weights w_{ij} inversely (nonlinear) function of image gradient

Power watershed for image segmentation

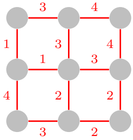
$$\min_x \underbrace{\sum_{e_{ij}\in E} {w_{ij}}^p |x_i - x_j|^q}_{\text{Smoothness term}} + \underbrace{\sum_{v_i\in V} {w_i}^p |x_i - y_i|^q}_{\text{Data term}}$$

- Vertices = pixels, edges between neighboring pixels
- Pairwise weights w_{ij} inversely (nonlinear) function of image gradient

Image



Graph



Power watershed for image segmentation

$$\min_x \underbrace{\sum_{e_{ij}\in E} {w_{ij}}^p |x_i - x_j|^q}_{\text{Smoothness term}} + \underbrace{\sum_{v_i \in V} {w_i}^p |x_i - y_i|^q}_{\text{Data term}}$$

Power watershed for image segmentation

$$\min_x \underbrace{\sum_{e_{ij}\in E} w_{ij}{}^p |x_i - x_j|^q}_{\text{Smoothness term}} + \underbrace{\sum_{v_i \in V} w_i{}^p |x_i - y_i|^q}_{\text{Data term}}$$

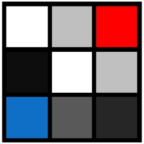
seeds enforced by y : $\quad y_i = \begin{cases} 1 & \text{if } v_i \in F, \\ 0 & \text{if } v_i \in B. \end{cases}$

Power watershed for image segmentation

$$\min_x \underbrace{\sum_{e_{ij} \in E} {w_{ij}}^p |x_i - x_j|^q}_{\text{Smoothness term}} + \underbrace{\sum_{v_i \in V} {w_i}^p |x_i - y_i|^q}_{\text{Data term}}$$

seeds enforced by y : $\quad y_i = \begin{cases} 1 & \text{if } v_i \in F, \\ 0 & \text{if } v_i \in B. \end{cases}$

Seeds



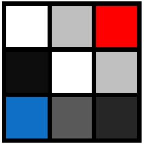
Graph

Power watershed for image segmentation

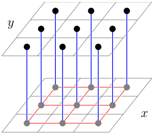
$$\min_{x} \underbrace{\sum_{e_{ij}\in E} {w_{ij}}^{p}|x_i - x_j|^q}_{\text{Smoothness term}} + \underbrace{\sum_{v_i\in V} {w_i}^{p}|x_i - y_i|^q}_{\text{Data term}}$$

seeds enforced by y : $\quad y_i = \begin{cases} 1 & \text{if } v_i \in F, \\ 0 & \text{if } v_i \in B. \end{cases}$

Seeds



Graph

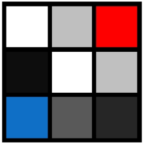


Power watershed for image segmentation

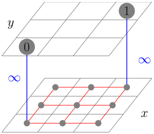
$$\min_x \underbrace{\sum_{e_{ij}\in E} {w_{ij}}^p |x_i - x_j|^q}_{\text{Smoothness term}} + \underbrace{\sum_{v_i\in V} {w_i}^p |x_i - y_i|^q}_{\text{Data term}}$$

$$\text{seeds enforced by } y: \quad y_i = \begin{cases} 1 & \text{if } v_i \in F, \\ 0 & \text{if } v_i \in B. \end{cases}$$

Seeds



Graph

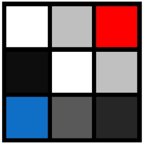


Power watershed for image segmentation

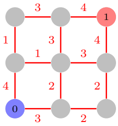
$$\min_x \underbrace{\sum_{e_{ij}\in E} {\color{red} w_{ij}}^p |x_i - x_j|^q}_{\color{red}\text{Smoothness term}} + \underbrace{\sum_{v_i\in V} {\color{blue} w_i}^p |x_i - y_i|^q}_{\color{blue}\text{Data term}}$$

$$\text{seeds enforced by } y: \quad y_i = \begin{cases} 1 & \text{if } v_i \in F, \\ 0 & \text{if } v_i \in B. \end{cases}$$

Seeds



Graph



Power watershed for image segmentation

- Simplification for algorithms comparison : only seeds used in the data fidelity term

$$\min_x \sum_{e_{ij}\in E} {w_{ij}}^p |x_i - x_j|^q$$
$$\text{s.t. } x(F) = 1,\ x(B) = 0$$

- Result : segmentation s defined $\forall i$ by $s_i = \begin{cases} 1 \text{ if } x_i \geq \frac{1}{2}, \\ 0 \text{ if } x_i < \frac{1}{2}. \end{cases}$

Algorithms deriving from values of p et q

Recall the energy function : $\min_x \sum_{e_{ij} \in E} w_{ij}^p |x_i - x_j|^q$

q \ p	0	finite	∞
1	Reduction to seeds	Graph cuts	Max Spanning Forest (watershed) [Allène et al. 07]
2	ℓ_2-norm Voronoi	Random walker	Power watershed [Couprie et al. 09]
∞	ℓ_1-norm Voronoi	ℓ_1-norm Voronoi	Shortest Path [Sinop et al. 07]

Algorithms deriving from values of p et q

Recall the energy function : $\min_x \sum_{e_{ij} \in E} w_{ij}^p |x_i - x_j|^q$

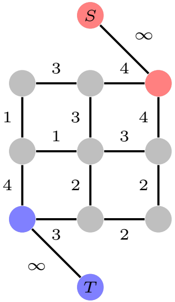
q \ p	0	finite	∞
1	Reduction to seeds	Graph cuts	Max Spanning Forest (watershed) [Allène et al. 07]
2	ℓ_2-norm Voronoi	Random walker	Power watershed [Couprie et al. 09]
∞	ℓ_1-norm Voronoi	ℓ_1-norm Voronoi	Shortest Path [Sinop et al. 07]

Graph Cuts

- Problem : compute x

$$x = \arg\min \sum_{e_{ij} \in E} {w_{ij}}^{p=1} |x_i - x_j|^{q=1}$$

- Min cut / Max flow duality
- Max Flow algorithm

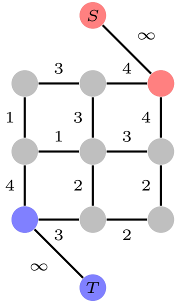


Graph Cuts

- Problem : compute x

$$x = \arg\min \sum_{e_{ij} \in E} w_{ij} |x_i - x_j|$$

- Min cut / Max flow duality
- Max Flow algorithm

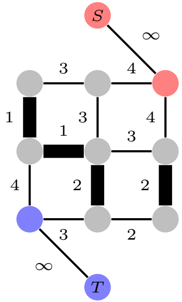


Graph Cuts

- Problem : compute x

$$x = \arg\min \sum_{e_{ij} \in E} w_{ij} |x_i - x_j|$$

- Min cut / Max flow duality
- Max Flow algorithm

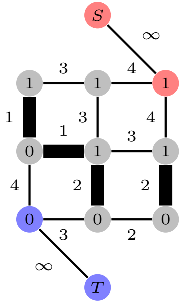


Graph Cuts

- Problem : compute x

$$x = \arg\min \sum_{e_{ij} \in E} w_{ij} |x_i - x_j|$$

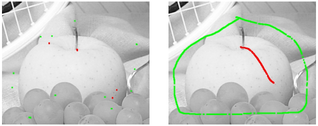
- Min cut / Max flow duality
- Max Flow algorithm



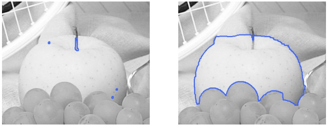
Graph Cuts : example

- favors small boundaries
- robust to seed placement

Input seeds



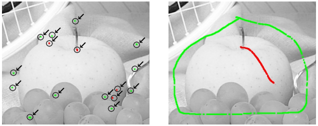
Corresponding segmentations



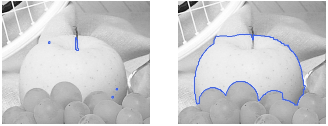
Graph Cuts : example

- favors small boundaries
- robust to seed placement

Input seeds



Corresponding segmentations



Algorithms deriving from values of p et q

Recall the energy function : $\min_x \sum_{e_{ij} \in E} w_{ij}^p |x_i - x_j|^q$

q \ p	0	finite	∞
1	Reduction to seeds	Graph cuts	Max Spanning Forest (watershed) [Allène et al. 07]
2	ℓ_2-norm Voronoi	Random walker	Power watershed [Couprie et al. 09]
∞	ℓ_1-norm Voronoi	ℓ_1-norm Voronoi	Shortest Path [Sinop et al. 07]

Algorithms deriving from values of p et q

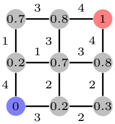
Recall the energy function : $\min_x \sum_{e_{ij} \in E} w_{ij}^p |x_i - x_j|^q$

q \ p	0	finite	∞
1	Reduction to seeds	Graph cuts	Max Spanning Forest (watershed) [Allène et al. 07]
2	ℓ_2-norm Voronoi	Random walker	Power watershed [Couprie et al. 09]
∞	ℓ_1-norm Voronoi	ℓ_1-norm Voronoi	Shortest Path [Sinop et al. 07]

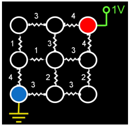
Random Walker

- Discrete version of the Dirichlet problem

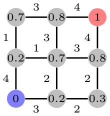
$$x = \arg\min \sum_{e_{ij}\in E} {w_{ij}}^{p=1}(x_i - x_j)^{q=2} \quad \leftarrow \quad u = \arg\min \int_\Omega |\nabla u|^2 d\Omega$$



- Potentials analogy



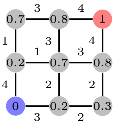
- Random walker analogy



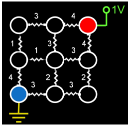
Random Walker

- Discrete version of the Dirichlet problem

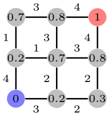
$$x = \arg\min \sum_{e_{ij} \in E} w_{ij}(x_i - x_j)^2 \quad \leftarrow \quad u = \arg\min \int_{\Omega} |\nabla u|^2 d\Omega$$



- Potentials analogy



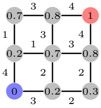
- Random walker analogy



Random Walker

- Discrete version of the Dirichlet problem

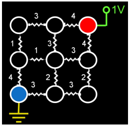
$$x = \arg\min \sum_{e_{ij} \in E} w_{ij}(x_i - x_j)^2 \quad \leftarrow \quad u = \arg\min \int_\Omega |\nabla u|^2 d\Omega$$



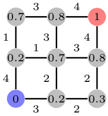
Stricly convex problem

$\Rightarrow$ unique optimal solution x^*

- Potentials analogy

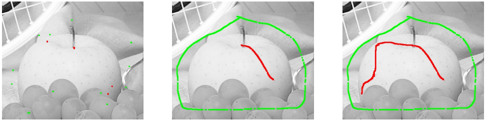


- Random walker analogy



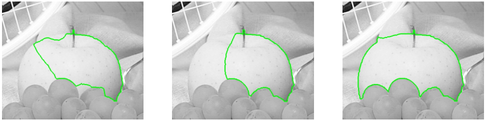
Random Walker : example

Input seeds



Corresponding probability/potential x

Corresponding segmentations (threshold of x)



Algorithms deriving from values of p et q

Recall the energy function : $\min_x \sum_{e_{ij} \in E} w_{ij}^p |x_i - x_j|^q$

q \ p	0	finite	∞
1	Reduction to seeds	Graph cuts	Max Spanning Forest (watershed) [Allène et al. 07]
2	ℓ_2-norm Voronoi	Random walker	Power watershed [Couprie et al. 09]
∞	ℓ_1-norm Voronoi	ℓ_1-norm Voronoi	Shortest Path [Sinop et al. 07]

Algorithms deriving from values of p et q

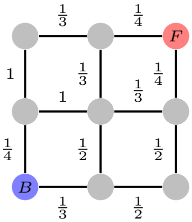
Recall the energy function : $\min_x \sum_{e_{ij} \in E} w_{ij}^p |x_i - x_j|^q$

q \ p	0	finite	∞
1	Reduction to seeds	Graph cuts	Max Spanning Forest (watershed) [Allène et al. 07]
2	ℓ_2-norm Voronoi	Random walker	Power watershed [Couprie et al. 09]
∞	ℓ_1-norm Voronoi	ℓ_1-norm Voronoi	Shortest Path [Sinop et al. 07]

Shortest path forest

- take the inverse of the weights
- the shortest path starting from each node to reach a seed node is computed
- Dijsktra algorithm
- [Sinop et al. 07] : optimizes

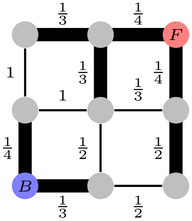
$$\min_x \sum_{e_{ij} \in E} {w_{ij}}^{p=q\to\infty} (x_i - x_j)^{q\to\infty}$$



Shortest paths

- take the inverse of the weights
- the shortest path starting from each node to reach a seed node is computed
- Dijsktra algorithm
- [Sinop et al. 07] : optimizes

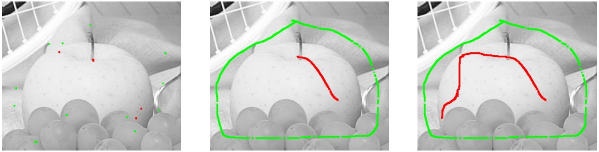
$$\min_{x} \sum_{e_{ij} \in E} {w_{ij}}^{p=q\to\infty} (x_i - x_j)^{q\to\infty}$$



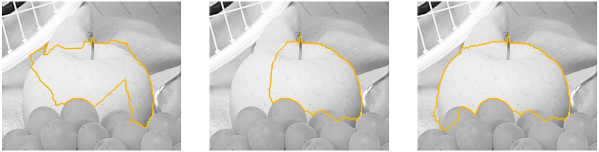
Shortest path : example

- Very sensitive to seeds placement

Input seeds



Corresponding segmentations



Algorithms deriving from values of p et q

Recall the energy function : $\min_x \sum_{e_{ij} \in E} w_{ij}^p |x_i - x_j|^q$

q \ p	0	finite	∞
1	Reduction to seeds	Graph cuts	Max Spanning Forest (watershed) [Allène et al. 07]
2	ℓ_2-norm Voronoi	Random walker	Power watershed [Couprie et al. 09]
∞	ℓ_1-norm Voronoi	ℓ_1-norm Voronoi	Shortest Path [Sinop et al. 07]

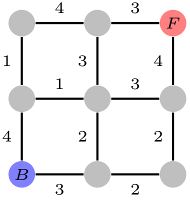
Algorithms deriving from values of p et q

Recall the energy function : $\min_x \sum_{e_{ij} \in E} w_{ij}^p |x_i - x_j|^q$

q \ p	0	finite	∞
1	Reduction to seeds	Graph cuts	Max Spanning Forest (watershed) [Allène et al. 07]
2	ℓ_2-norm Voronoi	Random walker	Power watershed [Couprie et al. 09]
∞	ℓ_1-norm Voronoi	ℓ_1-norm Voronoi	Shortest Path [Sinop et al. 07]

Watershed by Maximum Spanning Forest (MSF)

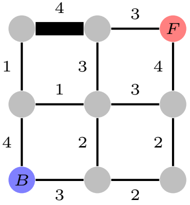
- maximize the sum of weights over the edges of a forest spanning the graph
- different labeled nodes have to belong to different trees
- Kruskal, Prim algorithms



Kruskal algorithm

Watershed by Maximum Spanning Forest (MSF)

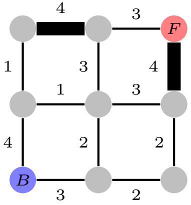
- maximize the sum of weights over the edges of a forest spanning the graph
- different labeled nodes have to belong to different trees
- Kruskal, Prim algorithms



Kruskal algorithm

Watershed by Maximum Spanning Forest (MSF)

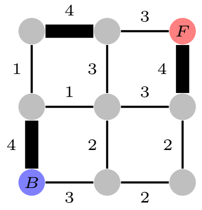
- maximize the sum of weights over the edges of a forest spanning the graph
- different labeled nodes have to belong to different trees
- Kruskal, Prim algorithms



Kruskal algorithm

Watershed by Maximum Spanning Forest (MSF)

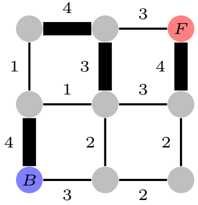
- maximize the sum of weights over the edges of a forest spanning the graph
- different labeled nodes have to belong to different trees
- Kruskal, Prim algorithms



Kruskal algorithm

Watershed by Maximum Spanning Forest (MSF)

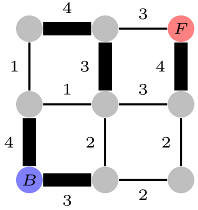
- maximize the sum of weights over the edges of a forest spanning the graph
- different labeled nodes have to belong to different trees
- Kruskal, Prim algorithms



Kruskal algorithm

Watershed by Maximum Spanning Forest (MSF)

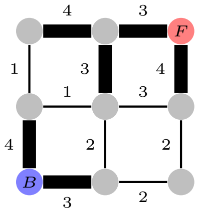
- maximize the sum of weights over the edges of a forest spanning the graph
- different labeled nodes have to belong to different trees
- Kruskal, Prim algorithms



Kruskal algorithm

Watershed by Maximum Spanning Forest (MSF)

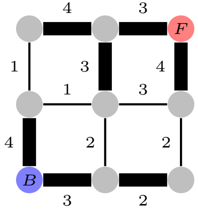
- maximize the sum of weights over the edges of a forest spanning the graph
- different labeled nodes have to belong to different trees
- Kruskal, Prim algorithms



Kruskal algorithm

Watershed by Maximum Spanning Forest (MSF)

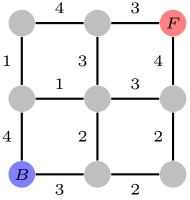
- maximize the sum of weights over the edges of a forest spanning the graph
- different labeled nodes have to belong to different trees
- Kruskal, Prim algorithms



Kruskal algorithm

Watershed by Maximum Spanning Forest (MSF)

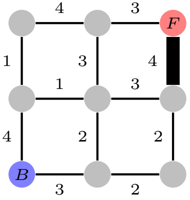
- maximize the sum of weights over the edges of a forest spanning the graph
- different labeled nodes have to belong to different trees
- Kruskal, Prim algorithms



Prim algorithm

Watershed by Maximum Spanning Forest (MSF)

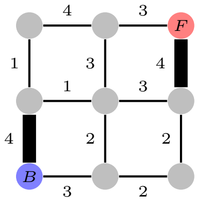
- maximize the sum of weights over the edges of a forest spanning the graph
- different labeled nodes have to belong to different trees
- Kruskal, Prim algorithms



Prim algorithm

Watershed by Maximum Spanning Forest (MSF)

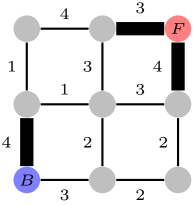
- maximize the sum of weights over the edges of a forest spanning the graph
- different labeled nodes have to belong to different trees
- Kruskal, Prim algorithms



Prim algorithm

Watershed by Maximum Spanning Forest (MSF)

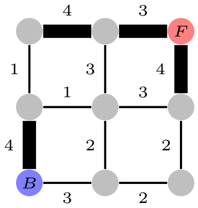
- maximize the sum of weights over the edges of a forest spanning the graph
- different labeled nodes have to belong to different trees
- Kruskal, Prim algorithms



Prim algorithm

Watershed by Maximum Spanning Forest (MSF)

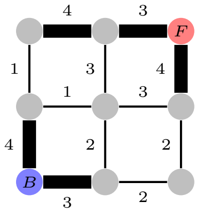
- maximize the sum of weights over the edges of a forest spanning the graph
- different labeled nodes have to belong to different trees
- Kruskal, Prim algorithms



Prim algorithm

Watershed by Maximum Spanning Forest (MSF)

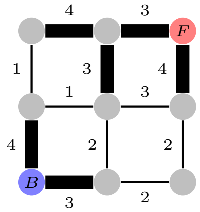
- maximize the sum of weights over the edges of a forest spanning the graph
- different labeled nodes have to belong to different trees
- Kruskal, Prim algorithms



Prim algorithm

Watershed by Maximum Spanning Forest (MSF)

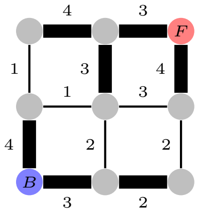
- maximize the sum of weights over the edges of a forest spanning the graph
- different labeled nodes have to belong to different trees
- Kruskal, Prim algorithms



Prim algorithm

Watershed by Maximum Spanning Forest (MSF)

- maximize the sum of weights over the edges of a forest spanning the graph
- different labeled nodes have to belong to different trees
- Kruskal, Prim algorithms

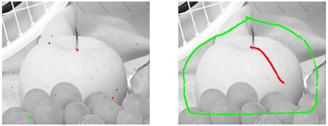


Prim algorithm

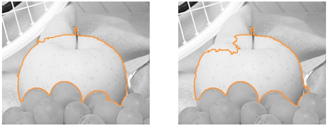
Maximum Spanning Forest (MSF) : example

- robust to small seeds : no bias toward small objects
- leaking effect

Input seeds

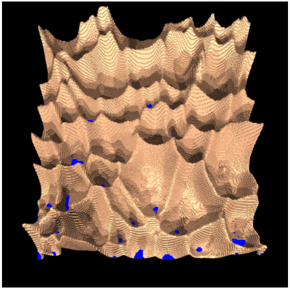


Corresponding segmentations



Watershed and Maximum Spanning Forest equivalence

- Watershed cut : edges where a drop of water could flow toward different catchment bassins [Cousty *et al.* 07].

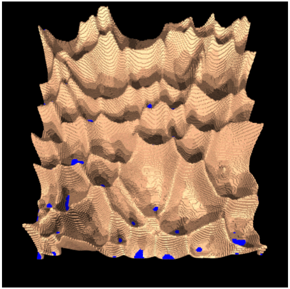


Theorem

If seeds are the minima of the weight function,
Equivalence between cuts by flooding and watershed cuts [Cousty et al 07]

Watershed and Maximum Spanning Forest equivalence

- Watershed cut : edges where a drop of water could flow toward different catchment bassins [Cousty *et al.* 07].

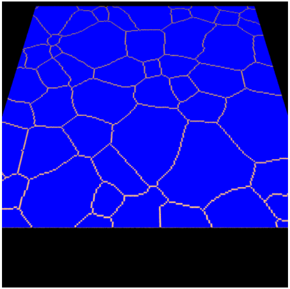


Theorem

If seeds are the minima of the weight function, any MSF cut on the weight function is a watershed cut (and conversely) [Cousty et al 07]

Watershed and Maximum Spanning Forest equivalence

- Watershed cut : edges where a drop of water could flow toward different catchment bassins [Cousty *et al.* 07].

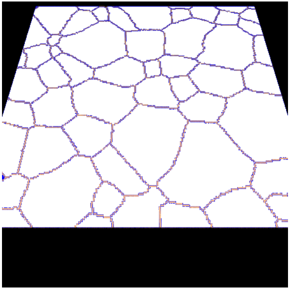


Theorem

If seeds are the minima of the weight function,
any MSF cut on the weight function is a watershed cut (and conversely) [Cousty et al 07]

Watershed and Maximum Spanning Forest equivalence

- Watershed cut : edges where a drop of water could flow toward different catchment bassins [Cousty *et al.* 07].

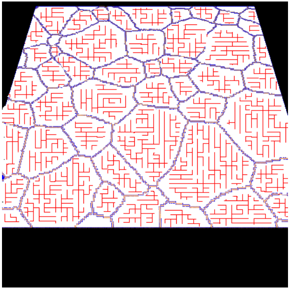


Theorem

If seeds are the minima of the weight function,
any MSF cut on the weight function is a watershed cut (and conversely) [Cousty et al 07]

Watershed and Maximum Spanning Forest equivalence

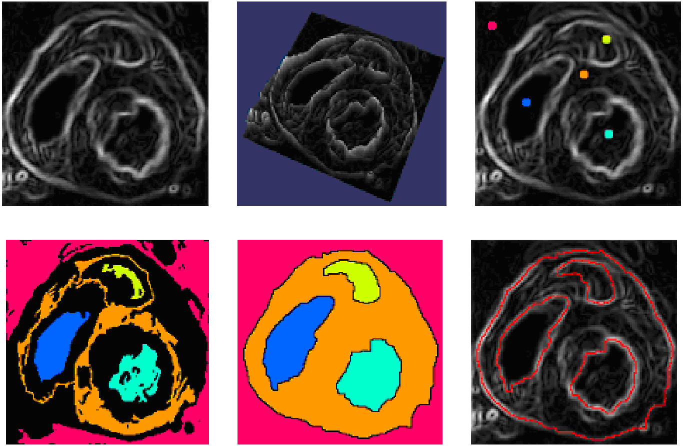
- Watershed cut : edges where a drop of water could flow toward different catchment bassins [Cousty *et al.* 07].



Theorem

If seeds are the minima of the weight function,
any MSF cut on the weight function is a watershed cut (and conversely) [Cousty et al 07]

Example of segmentation by flooding/Prim algorithm



Algorithms deriving from values of p et q

Recall the energy function : $\min_x \sum_{e_{ij} \in E} w_{ij}^p |x_i - x_j|^q$

q \ p	0	finite	∞
1	Reduction to seeds	Graph cuts	Max Spanning Forest (watershed) [Allène et al. 07]
2	ℓ_2-norm Voronoi	Random walker	Power watershed [Couprie et al. 09]
∞	ℓ_1-norm Voronoi	ℓ_1-norm Voronoi	Shortest Path [Sinop et al. 07]

Algorithms deriving from values of p et q

Recall the energy function : $\min_x \sum_{e_{ij} \in E} w_{ij}^p |x_i - x_j|^q$

q \ p	0	finite	∞
1	Reduction to seeds	Graph cuts	Max Spanning Forest (watershed) [Allène et al. 07]
2	ℓ_2-norm Voronoi	Random walker	Power watershed [Couprie et al. 09]
∞	ℓ_1-norm Voronoi	ℓ_1-norm Voronoi	Shortest Path [Sinop et al. 07]

Algorithms deriving from values of p et q

Recall the energy function : $\min_x \sum_{e_{ij} \in E} w_{ij}^p |x_i - x_j|^q$

q \ p	0	finite	∞
1	Reduction to seeds	Graph cuts	Max Spanning Forest (watershed) [Allène et al. 07]
2	ℓ_2-norm Voronoi	Random walker	Power watershed [Couprie et al. 09]
∞	ℓ_1-norm Voronoi	ℓ_1-norm Voronoi	Shortest Path [Sinop et al. 07]

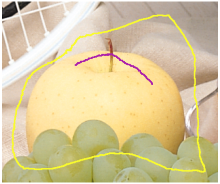
Algorithms deriving from values of p et q

Recall the energy function : $\min_x \sum_{e_{ij} \in E} w_{ij}^p |x_i - x_j|^q$

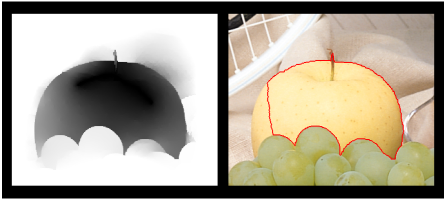
q \ p	0	finite	∞
1	Reduction to seeds	Graph cuts	Max Spanning Forest (watershed) [Allène et al. 07]
2	ℓ_2-norm Voronoi	Random walker	Power watershed [Couprie et al. 09]
∞	ℓ_1-norm Voronoi	ℓ_1-norm Voronoi	Shortest Path [Sinop et al. 07]

Convergence of RW when $p \to \infty$ toward PW

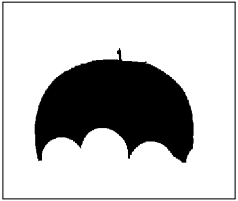
Input seeds



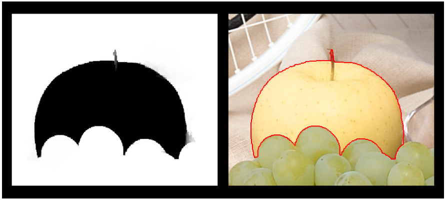
Random Walker $p = 1 \ldots 30$



PowerWatershed $q = 2$



Random Walker $p = 30$



Algorithm for the case $p \to \infty$, variable q

- Compute x minimizing

$$lim_{p\to\infty} \sum_{e_{ij}\in E} {w_{ij}}^p |x_i - x_j|^q$$

  subject to boundary conditions.

- We construct an MSF outside of plateaus, and optimize

$$\sum_{e_{ij}\in \text{plateau}} |x_i - x_j|^q$$

  on the plateaus.

- We call this algorithm "Power watershed"

Properties

Theorem

The cut obtained by the power watershed algorithm is a MSF cut.

Theorem

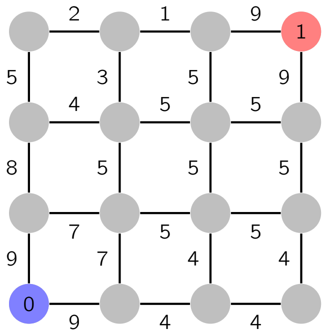
When $q > 1$, the solution x^* to the minimization of

$$\min_x lim_{p\to\infty} \sum_{e_{ij}\in E} {w_{ij}}^p |x_i - x_j|^q$$

is unique. Thus, when $q > 1$, the solution x obtained by the power watershed algorithm is unique.

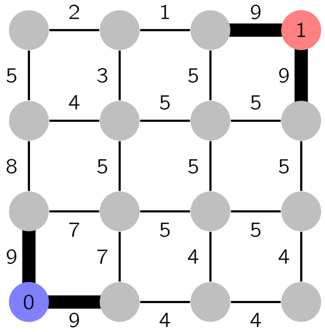
Power watershed algorithm

1. Choose an edge with maximal weight $e_{\max}$. Let S the set of edges connected to $e_{\max}$ with the same weight as $e_{\max}$.
2. If S does not contain vertices that have different labels, merge the nodes of S into one node, otherwise minimize $E_{1,q}$ on S.
3. Repeat steps 1 and 2 until all vertices are labeled.



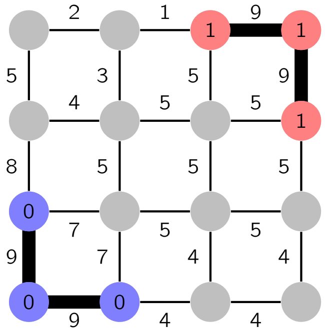
Power watershed algorithm

1. Choose an edge with maximal weight e_{max}. Let S the set of edges connected to e_{max} with the same weight as e_{max}.
2. If S does not contain vertices that have different labels, merge the nodes of S into one node, otherwise minimize $E_{1,q}$ on S.
3. Repeat steps 1 and 2 until all vertices are labeled.



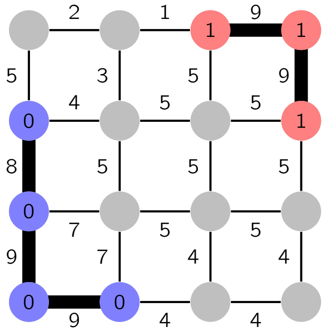
Power watershed algorithm

1. Choose an edge with maximal weight $e_{\max}$. Let S the set of edges connected to $e_{\max}$ with the same weight as $e_{\max}$.
2. If S does not contain vertices that have different labels, merge the nodes of S into one node, otherwise minimize $E_{1,q}$ on S.
3. Repeat steps 1 and 2 until all vertices are labeled.



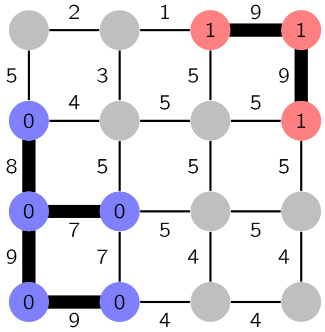
Power watershed algorithm

1. Choose an edge with maximal weight $e_{\max}$. Let S the set of edges connected to $e_{\max}$ with the same weight as $e_{\max}$.
2. If S does not contain vertices that have different labels, merge the nodes of S into one node, otherwise minimize $E_{1,q}$ on S.
3. Repeat steps 1 and 2 until all vertices are labeled.



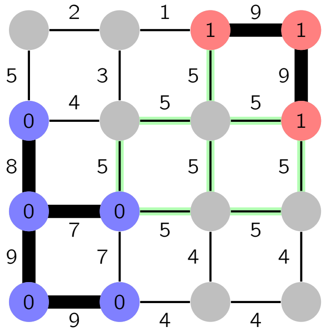
Power watershed algorithm

1. Choose an edge with maximal weight $e_{\max}$. Let S the set of edges connected to $e_{\max}$ with the same weight as $e_{\max}$.
2. If S does not contain vertices that have different labels, merge the nodes of S into one node, otherwise minimize $E_{1,q}$ on S.
3. Repeat steps 1 and 2 until all vertices are labeled.



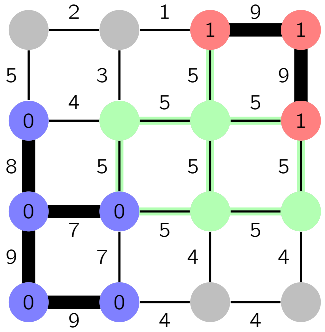
Power watershed algorithm

1. Choose an edge with maximal weight $e_{\max}$. Let S the set of edges connected to $e_{\max}$ with the same weight as $e_{\max}$.
2. If S does not contain vertices that have different labels, merge the nodes of S into one node, otherwise minimize $E_{1,q}$ on S.
3. Repeat steps 1 and 2 until all vertices are labeled.



Power watershed algorithm

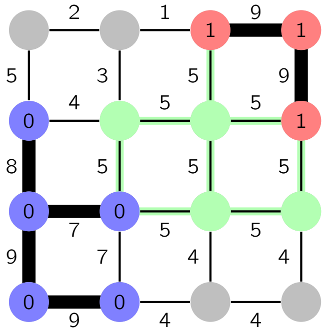
1. Choose an edge with maximal weight e_{max}. Let S the set of edges connected to e_{max} with the same weight as e_{max}.
2. If S does not contain vertices that have different labels, merge the nodes of S into one node, otherwise minimize $E_{1,q}$ on S.
3. Repeat steps 1 and 2 until all vertices are labeled.



$$\min_x \lim_{p\to\infty} \sum_{e_{ij}\in E} w_{ij}^p |x_i - x_j|^q$$

Power watershed algorithm

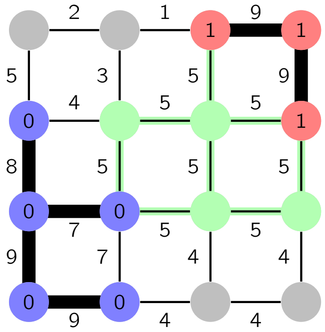
1. Choose an edge with maximal weight $e_{\max}$. Let S the set of edges connected to $e_{\max}$ with the same weight as $e_{\max}$.
2. If S does not contain vertices that have different labels, merge the nodes of S into one node, otherwise minimize $E_{1,q}$ on S.
3. Repeat steps 1 and 2 until all vertices are labeled.



$$\min_x \lim_{p\to\infty} \sum_{e_{ij}\in E} w_{ij}^p |x_i - x_j|^q$$

Power watershed algorithm

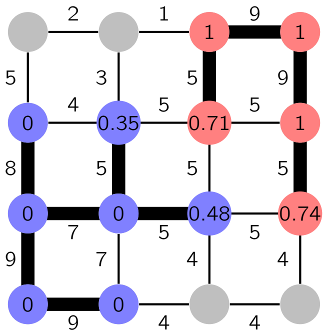
1. Choose an edge with maximal weight $e_{\max}$. Let S the set of edges connected to $e_{\max}$ with the same weight as $e_{\max}$.
2. If S does not contain vertices that have different labels, merge the nodes of S into one node, otherwise minimize $E_{1,q}$ on S.
3. Repeat steps 1 and 2 until all vertices are labeled.



$$\min_x \sum_{e_{ij} \in \text{plateau}} |x_i - x_j|^q$$

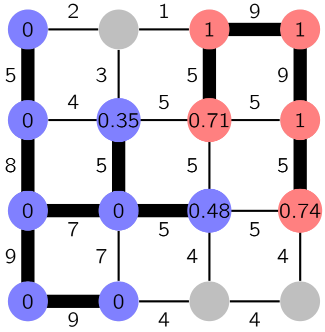
Power watershed algorithm

1. Choose an edge with maximal weight $e_{\max}$. Let S the set of edges connected to $e_{\max}$ with the same weight as $e_{\max}$.
2. If S does not contain vertices that have different labels, merge the nodes of S into one node, otherwise minimize $E_{1,q}$ on S.
3. Repeat steps 1 and 2 until all vertices are labeled.



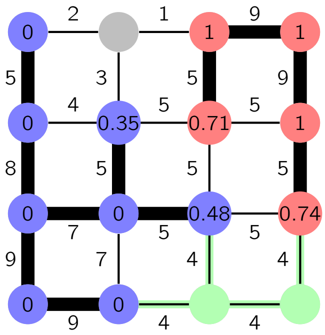
Power watershed algorithm

1. Choose an edge with maximal weight $e_{\max}$. Let S the set of edges connected to $e_{\max}$ with the same weight as $e_{\max}$.
2. If S does not contain vertices that have different labels, merge the nodes of S into one node, otherwise minimize $E_{1,q}$ on S.
3. Repeat steps 1 and 2 until all vertices are labeled.



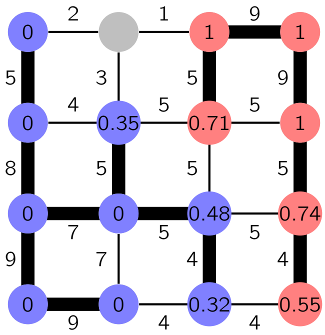
Power watershed algorithm

1. Choose an edge with maximal weight $e_{\max}$. Let S the set of edges connected to $e_{\max}$ with the same weight as $e_{\max}$.
2. If S does not contain vertices that have different labels, merge the nodes of S into one node, otherwise minimize $E_{1,q}$ on S.
3. Repeat steps 1 and 2 until all vertices are labeled.



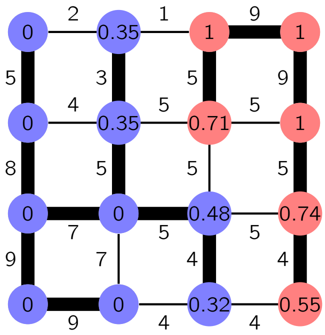
Power watershed algorithm

1. Choose an edge with maximal weight e_{max}. Let S the set of edges connected to e_{max} with the same weight as e_{max}.
2. If S does not contain vertices that have different labels, merge the nodes of S into one node, otherwise minimize $E_{1,q}$ on S.
3. Repeat steps 1 and 2 until all vertices are labeled.



Power watershed algorithm

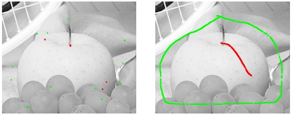
1. Choose an edge with maximal weight $e_{\max}$. Let S the set of edges connected to $e_{\max}$ with the same weight as $e_{\max}$.
2. If S does not contain vertices that have different labels, merge the nodes of S into one node, otherwise minimize $E_{1,q}$ on S.
3. Repeat steps 1 and 2 until all vertices are labeled.



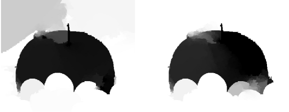
Power watershed (q=2) : example

- robust in case of small seeds
- less leaking than with standard Maximum Spanning Forest

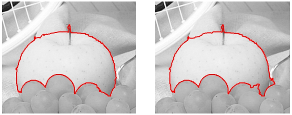
Input seeds



Corresponding probability x

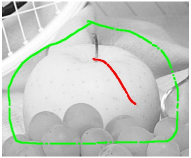


Corresponding segmentations (threshold of x)



Power watershed (q=2) : example

Input seeds

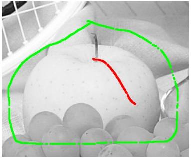


Prim (MSF, watershed by flooding)

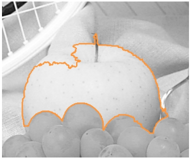
Power watershed ($q = 2$)

Power watershed (q=2) : example

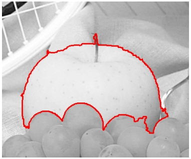
Input seeds



Prim (MSF, watershed by flooding)

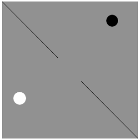


Power watershed ($q = 2$)



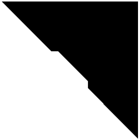
Algorithms behavior on plateaus

Seeded image

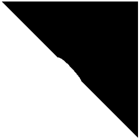


Graph Cuts

Shortest Paths, Watershed

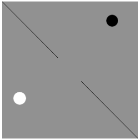


Random Walker, PW $q = 2$



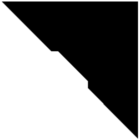
Algorithms behavior on plateaus

Seeded image

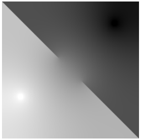


Graph Cuts

Shortest Paths, Watershed

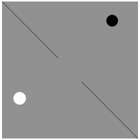


Random Walker, PW $q = 2$



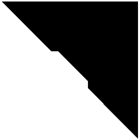
Algorithms behavior on plateaus

Seeded image

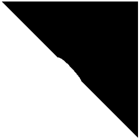


Graph Cuts

Shortest Paths, Watershed



Random Walker, PW $q = 2$



Algorithms comparison

- Evaluation on GrabCut database
- Ground truths
- 2 sets of seeds to study robustness to seeds centering :
  1. seeds well centered around boundaries
  2. seeds less centered around boundaries

Quantitative Results

Mean errors between ground truths and the algorithms results on GrabCut database with the seeds centered around boundaries.

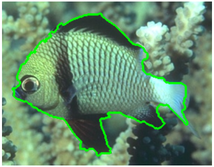
	BE	RI	GCE	Vol	**Average rank**
Shortest paths	2.821	0.972	0.233	0.204	**1**
Random walker	2.957	0.971	0.0234	0.0204	**2.5**
MSF (Prim)	2.859	0.971	0.0244	0.209	**3**
Power wshed ($q = 2$)	2.873	0.971	0.0245	0.210	**3.25**
Graph cuts	3.122	0.970	0.0249	0.212	**5**

Examples

Input seeds

Graph Cuts

Random Walker



Shortest Paths

Max Spanning Forests

Power Watersheds $q = 2$

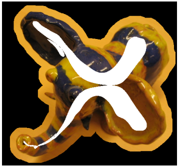
Quantitative Results

Mean errors between ground truths and the algorithms results on GrabCut database with the seeds less centered around boundaries.

	BE	RI	GCE	VoI	**Average rank**
Graph cuts	4.691	0.953	0.0380	0.284	**1**
Power wshed ($q = 2$)	4.928	0.951	0.0407	0.297	**2.5**
Random walker	5.124	0.950	0.0398	0.294	**2.75**
MSF (Prim)	5.111	0.950	0.0408	0.298	**3.5**
Shortest paths	5.330	0.947	0.0426	0.308	**5**

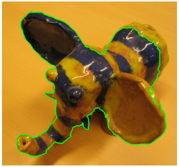
Examples

Input seeds



Graph Cuts

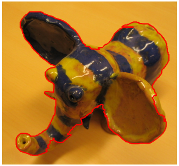
Random Walker



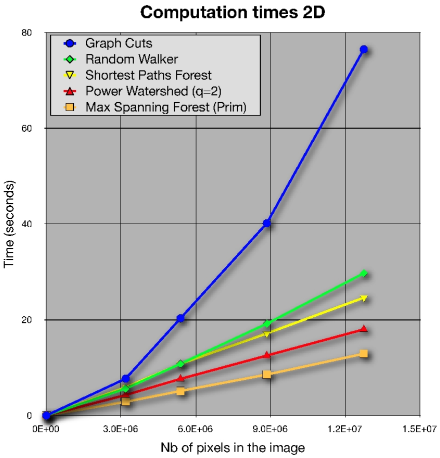
Shortest Paths

Max Spanning Forests

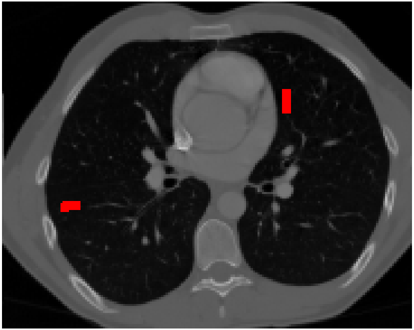
Power Watersheds $q = 2$



Computation time 2D

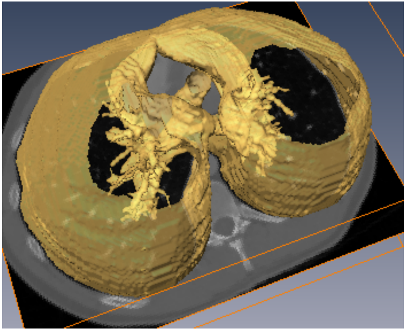


3D example



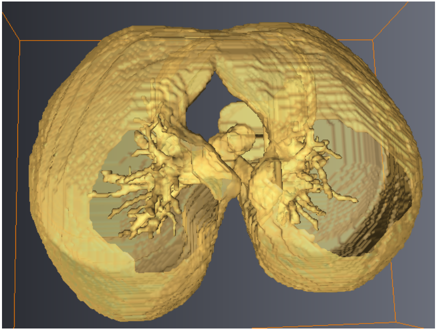
Foreground seeds

3D example



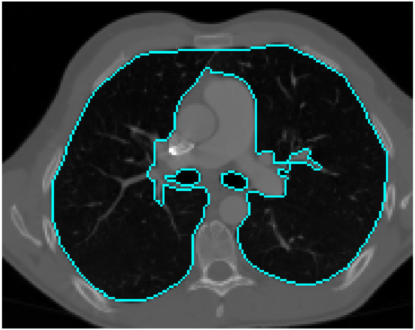
Powerwatershed result

3D example



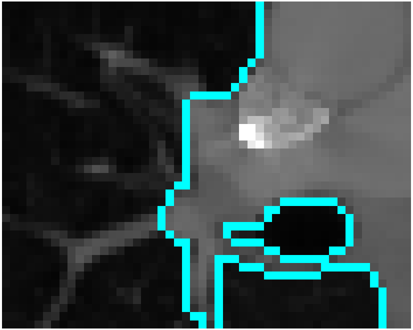
Powerwatershed result

3D example



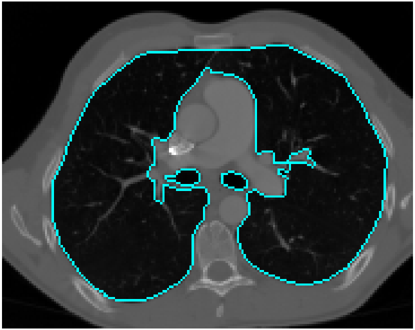
Graph-cut result

3D example



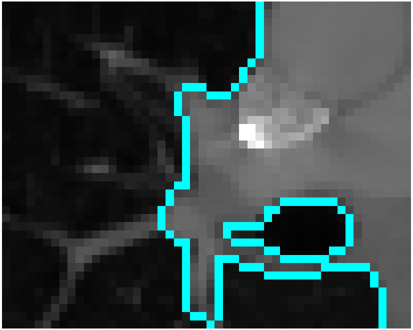
Graph-cut result (detail)

3D example



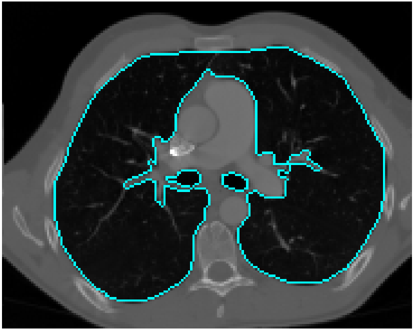
Random-walker result

3D example



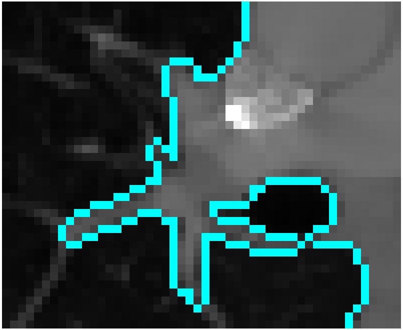
Random-walker result (detail)

3D example



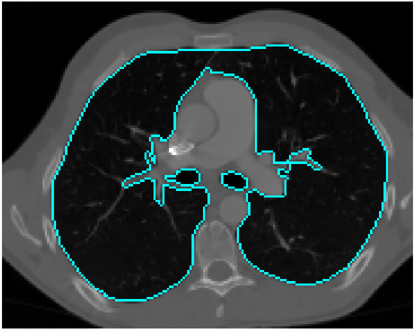
Shortest-path result

3D example



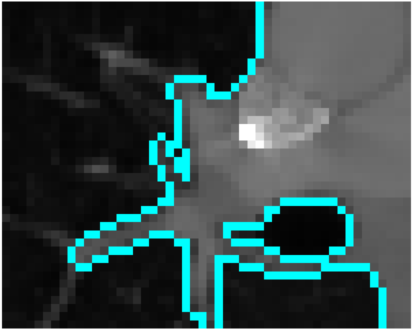
Shortest-path result (detail)

3D example



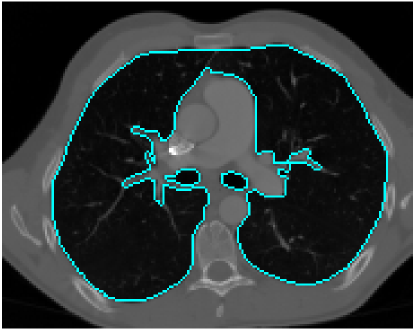
MSF-Watershed result

3D example



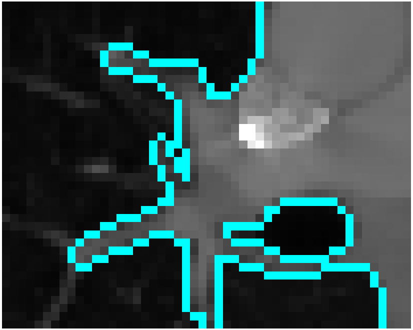
MSF-Watershed result (detail)

3D example



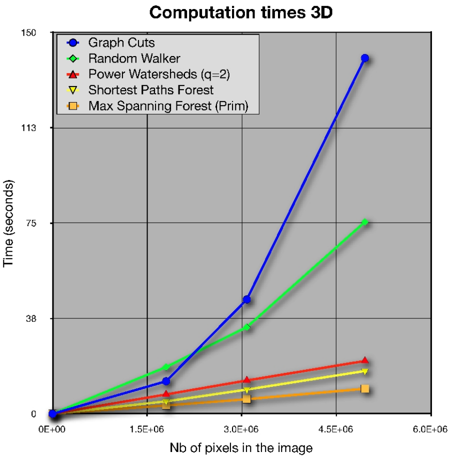
Powerwatershed result

3D example



Powerwatershed result (detail)

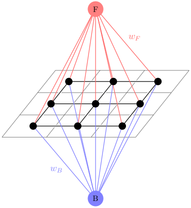
Computation time 3D



Unseeded segmentation

- Possibility to add unary terms to the energy function

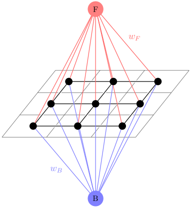
$$\min_x \sum_{e_{ij} \in E} w_{ij}^p |x_i - x_j|^q$$



Unseeded segmentation

- Possibility to add unary terms to the energy function

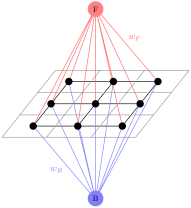
$$\min_x \sum_{e_{ij} \in E} w_{ij}^p |x_i - x_j|^q + \sum_{v_i} {w_{F_i}}^p |x_i - 1|^q + \sum_{v_i} {w_{Bi}}^p |x_i|^q$$



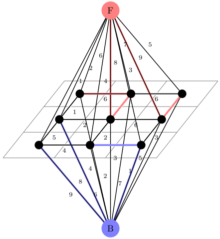
Unseeded segmentation

- Possibility to add unary terms to the energy function

$$\min_{x} \sum_{e_{ij}\in E} w_{ij}^{p}|x_i - x_j|^q + \sum_{v_i} {w_{F_i}}^{p}|x_i - 1|^q + \sum_{v_i} {w_{Bi}}^{p}|x_i|^q$$

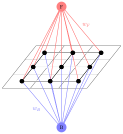


Maximum Spanning forest in the resulting graph



Unseeded segmentation

$$\min_x \sum_{e_{ij}\in E} w_{ij}^p |x_i - x_j|^q + \sum_{v_i} {w_{F_i}}^p |x_i - 1|^q + \sum_{v_i} {w_{Bi}}^p |x_i|^q$$



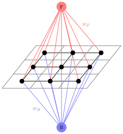
Image

Graph Cuts

Watershed

Unseeded segmentation

$$\min_x \sum_{e_{ij}\in E} w_{ij}^p |x_i - x_j|^q + \sum_{v_i} {w_{F_i}}^p |x_i - 1|^q + \sum_{v_i} {w_{B_i}}^p |x_i|^q$$



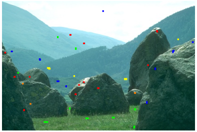
This is the first time that we show how to incorporate data unary terms into watershed computation.

Optimal multilabels segmentation

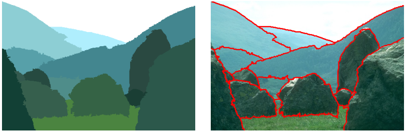
- More than 2-labels segmentation : NP-hard for Graph cuts
- Exact $n \geq 2$ labels segmentation for the other algorithms :
- n solutions $x^1, x^2, \ldots x^n$ computed
- x^k computed by enforcing $\begin{cases} x^k(n^k) = 1 \\ x^k(n^q) = 0 \text{ for all } q \neq k. \end{cases}$
- Each node i is affected to the label for which x_i^k is maximum :

$$s_i = \arg\max_k x_i^k$$

Input seeds



Segmentation by PowerWatershed ($q = 2$)



Segmentation : which algorithm to use ?

- Graph Cuts :
  - robust to seeds placement for 2D image segmentation with 2 labels only
  - too slow for 3D segmentation
- Shortest Paths : fast but requires well centered seeds around boundaries
- Random Walker :
  - efficient with uncentered seeds around boundaries
  - defined behavior on plateaus
- Watershed :
  - better segmentations than Shortest paths with uncentered seeds around boundaries
  - fast → 3D segmentation

Segmentation : which algorithm to use ?

- Power watershed $q = 2$:
  - Watershed properties (fast, multiseeds)
  - Random walker properties on plateaus and interacting plateaus
  - Unique solution
  - Less sensitive to leaking than standard watershed

What else can be done ?

- This efficient watershed algorithm can be used with data unary terms

Question

Can we apply watershed to other vision (optimization) problems ?

Anisotropic diffusion [Perona-Malik 1990]

- Optimization procedure blurring objects while preserving contours

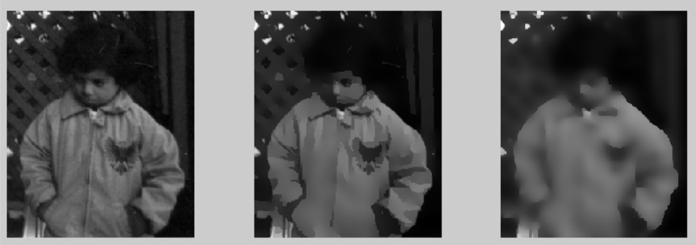


Image 100 iterations 200 iterations

- Goal of this work : perform anisotropic diffusion using an ℓ_0 norm to avoid the blurring effect

Anisotropic diffusion

- f : original image
- x : denoised image
- Perona-Malik algorithm

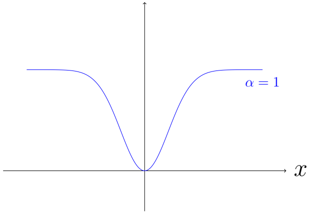
$$\frac{dx_i}{dt} = \sum_{e_{ij}\in E} e^{-\alpha(x_i - x_j)^2}(x_i - x_j)$$

- Black *et al.* energy

$$E(x) = \sum_{e_{ij}\in E} \sigma(x_i - x_j)$$

- Robust error function σ

$$\sigma(x) = 1 - \mathrm{e}^{-\alpha x^2}$$



- Perona-Malik algorithm is a gradient descent minimization of the Black *et al.* energy.

Anisotropic diffusion

- f : original image
- x : denoised image
- Perona-Malik algorithm

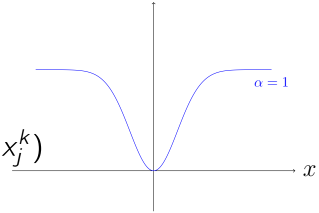
$$x_i^{k+1} = x_i^k + dt \sum_{e_{ij} \in E} e^{-\alpha (x_i^k - x_j^k)^2} (x_i^k - x_j^k)$$

- Black *et al.* energy

$$E(x) = \sum_{e_{ij} \in E} \sigma(x_i - x_j)$$

- Robust error function σ

$$\sigma(x) = 1 - \mathrm{e}^{-\alpha x^2}$$



- Perona-Malik algorithm is a gradient descent minimization of the Black *et al.* energy.

Anisotropic diffusion and ℓ_0 norm

$$\min_x \underbrace{\sum_{e_{ij}\in E} \sigma(x_i - x_j)}_{\text{smoothness term}} + \underbrace{\lambda \sum_{v_i \in V} (x_i - f_i)^2}_{\text{data fidelity term}}$$

Anisotropic diffusion and ℓ_0 norm

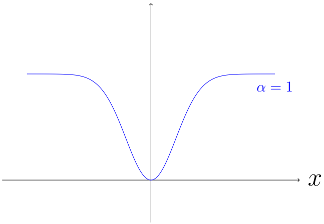
$$\min_x \underbrace{\sum_{e_{ij}\in E} \sigma(x_i - x_j)}_{\text{smoothness term}} + \lambda \underbrace{\sum_{v_i \in V} \sigma(x_i - f_i)}_{\text{data fidelity term}}$$

Anisotropic diffusion and ℓ_0 norm

$$\min_x \underbrace{\sum_{e_{ij}\in E} \sigma(x_i - x_j)}\; + \underbrace{\lambda \sum_{v_i \in V} \sigma(x_i - f_i)}_{\text{data fidelity term}}$$

$\alpha \to \infty$: approximation of ℓ_0 norm

$$\sigma(x) = 1 - \mathrm{e}^{-\alpha x^2}$$

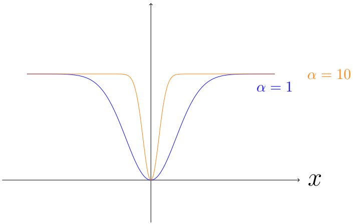


Anisotropic diffusion and ℓ_0 norm

$$\min_x \underbrace{\sum_{e_{ij} \in E} \sigma(x_i - x_j)}_{} + \underbrace{\lambda \sum_{v_i \in V} \sigma(x_i - f_i)}_{\text{data fidelity term}}$$

$\alpha \to \infty$: approximation of ℓ_0 norm

$\sigma(x) = 1 - e^{-\alpha x^2}$

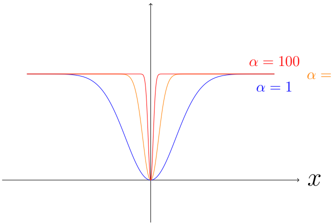


Anisotropic diffusion and ℓ_0 norm

$$\min_x \underbrace{\sum_{e_{ij}\in E} \sigma(x_i - x_j)} + \underbrace{\lambda \sum_{v_i \in V} \sigma(x_i - f_i)}_{\text{data fidelity term}}$$

$\alpha \to \infty$: approximation of ℓ_0 norm

$\sigma(x) = 1 - \mathrm{e}^{-\alpha x^2}$

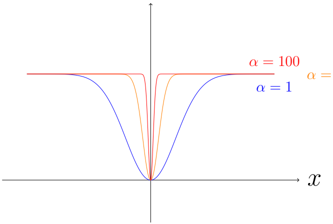


Anisotropic diffusion and ℓ_0 norm

$$\min_{x} \underbrace{\sum_{e_{ij} \in E} \sigma(x_i - x_j)} + \lambda \underbrace{\sum_{v_i \in V} \sigma(x_i - f_i)}_{\text{data fidelity term}}$$

$\alpha \to \infty$: approximation of ℓ_0 norm

$$\sigma(x) = 1 - \mathrm{e}^{-\alpha x^2}$$



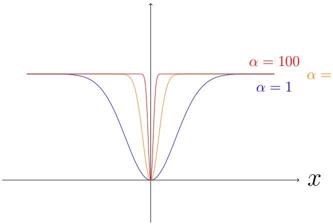
- high gradient $x_i - x_j \Rightarrow \sigma = 1$

Anisotropic diffusion and ℓ_0 norm

$$\min_x \underbrace{\sum_{e_{ij}\in E} \sigma(x_i - x_j)} + \lambda \underbrace{\sum_{v_i\in V} \sigma(x_i - f_i)}_{\text{data fidelity term}}$$

$\alpha \to \infty$: approximation of ℓ_0 norm

$$\sigma(x) = 1 - \mathrm{e}^{-\alpha x^2}$$



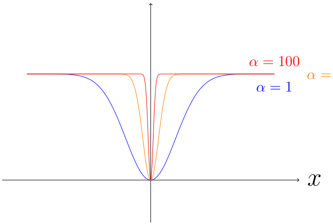
- high gradient $x_i - x_j \Rightarrow \sigma = 1$
- no gradient $\Rightarrow \sigma = 0$

Anisotropic diffusion and ℓ_0 norm

$$\min_x \underbrace{\sum_{e_{ij}\in E} \sigma(x_i - x_j)} + \underbrace{\lambda \sum_{v_i \in V} \sigma(x_i - f_i)}_{\text{data fidelity term}}$$

$\alpha \to \infty$: approximation of ℓ_0 norm

$$\sigma(x) = 1 - \mathrm{e}^{-\alpha x^2}$$



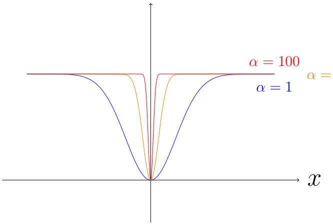
- high gradient $x_i - x_j \Rightarrow \sigma = 1$
- no gradient $\Rightarrow \sigma = 0$
- Finite α, low gradient $\Rightarrow$ $0 < \sigma < 1$ Piecewise smooth result

Anisotropic diffusion and ℓ_0 norm

$$\min_x \underbrace{\sum_{e_{ij}\in E} \sigma(x_i - x_j)}_{} + \underbrace{\lambda \sum_{v_i \in V} \sigma(x_i - f_i)}_{\text{data fidelity term}}$$

$\alpha \to \infty$: approximation of ℓ_0 norm

$\sigma(x) = 1 - e^{-\alpha x^2}$



- high gradient $x_i - x_j \Rightarrow \sigma = 1$
- no gradient $\Rightarrow \sigma = 0$
- Finite α, low gradient $\Rightarrow$ $0 < \sigma < 1$ Piecewise smooth result
- $\alpha \to \infty$, low gradient $\Rightarrow \sigma = 1$ Piecewise constant result

Anisotropic diffusion using power watershed

$$\min_{x} \underbrace{\sum_{e_{ij}\in E} \sigma(x_i - x_j)}_{\text{smoothness term}} + \lambda \underbrace{\sum_{v_i\in V} \sigma(x_i - f_i)}_{\text{data fidelity term}}$$

Anisotropic diffusion using power watershed

$$\min_x \underbrace{\sum_{e_{ij} \in E} \sigma(x_i - x_j)}_{\text{smoothness term}} + \underbrace{\lambda \sum_{v_i \in V} \sigma(x_i - f_i)}_{\text{data fidelity term}}$$

- Nonconvex energy

Anisotropic diffusion using power watershed

$$\min_x \underbrace{\sum_{e_{ij}\in E} \sigma(x_i - x_j)}_{\text{smoothness term}} + \underbrace{\lambda \sum_{v_i \in V} \sigma(x_i - f_i)}_{\text{data fidelity term}}$$

- Nonconvex energy
- Set the gradient of this energy to zero

Anisotropic diffusion using power watershed

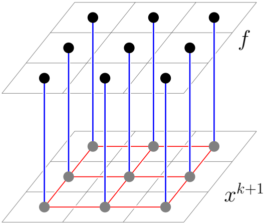
$$\min_x \underbrace{\sum_{e_{ij}\in E} \sigma(x_i - x_j)}_{\text{smoothness term}} + \lambda \underbrace{\sum_{v_i \in V} \sigma(x_i - f_i)}_{\text{data fidelity term}}$$

- Nonconvex energy
- Set the gradient of this energy to zero
- Fixed point iteration scheme with energy at step k :

Anisotropic diffusion using power watershed

$$\min_x \underbrace{\sum_{e_{ij}\in E} \sigma(x_i - x_j)}_{\text{smoothness term}} + \lambda \underbrace{\sum_{v_i \in V} \sigma(x_i - f_i)}_{\text{data fidelity term}}$$

- Nonconvex energy
- Set the gradient of this energy to zero
- Fixed point iteration scheme with energy at step k :

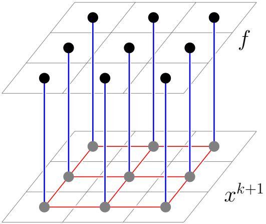


$$E_{k+1} = \sum_{e_{ij}\in E} e^{-\alpha(x_i^k - x_j^k)^2}(x_i^{k+1} - x_j^{k+1})^2 + \lambda \sum_{v_i \in V} e^{-\alpha(x_i^k - f_i)^2}(x_i^{k+1} - f_i)^2$$

Anisotropic diffusion using power watershed

$$\min_x \underbrace{\sum_{e_{ij}\in E} \sigma(x_i - x_j)}_{\text{smoothness term}} + \lambda \underbrace{\sum_{v_i \in V} \sigma(x_i - f_i)}_{\text{data fidelity term}}$$

- Nonconvex energy
- Set the gradient of this energy to zero
- Fixed point iteration scheme with energy at step k :

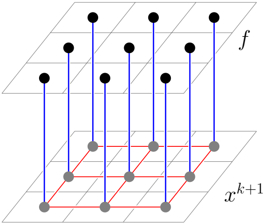


$$E_{k+1} = \sum_{e_{ij}\in E} e^{-\alpha(x_i^k - x_j^k)^2}(x_i^{k+1} - x_j^{k+1})^2 + \lambda \sum_{v_i\in V} e^{-\alpha(x_i^k - f_i)^2}(x_i^{k+1} - f_i)^2$$

Anisotropic diffusion using power watershed

$$\min_x \underbrace{\sum_{e_{ij}\in E} \sigma(x_i - x_j)}_{\text{smoothness term}} + \lambda \underbrace{\sum_{v_i \in V} \sigma(x_i - f_i)}_{\text{data fidelity term}}$$

- Nonconvex energy
- Set the gradient of this energy to zero
- Fixed point iteration scheme with energy at step k :



$$E_{k+1} = \sum_{e_{ij}\in E} \left(e^{-(x_i^k - x_j^k)^2}\right)^{\alpha} (x_i^{k+1} - x_j^{k+1})^2 + \lambda \sum_{v_i\in V} \left(e^{-(x_i^k - f_i)^2}\right)^{\alpha} (x_i^{k+1} - f_i)^2$$

Graph construction and algorithm

algoruled

Data: An image f, an initial solution x^0, $\lambda \in \mathbb{R}_+^*$

Result: A **filtered image** x^k

Set $k = 0$. Build the graph on the right

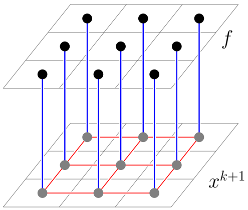
repeat

Generate the pairwise weights $\exp -(x_j^k - x_i^k)^2$, and unary weights $\exp -(x^k - f)^2$.

Use PW with $y = f$ to obtain x^{k+1}.

$k = k + 1$;

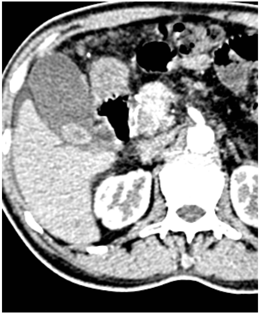
until $||x^{k+1} - x^k||_2 < \epsilon$;



Results

Leads to piecewise constant results

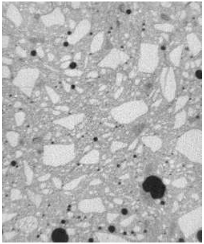
Original image



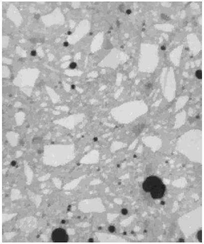
PW result

Results

Original image
(size 250×300)

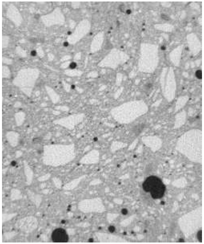


PW result
6 iterations, 1.78 sec.

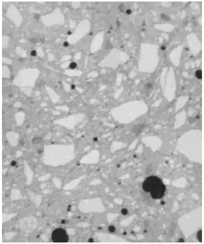


Results

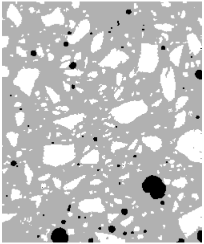
Original image
(size 250×300)



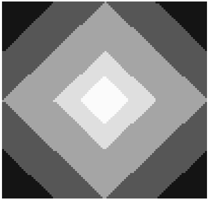
PW result
6 iterations, 1.78 sec.



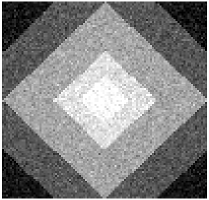
Segmentation
by thresholds



Comparison with Perona-Malik results

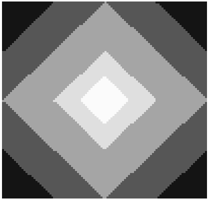


Original image

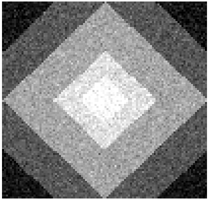


Noisy image, PSNR = 24.24dB

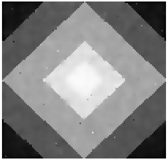
Comparison with Perona-Malik results



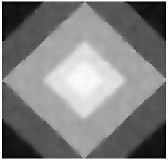
Original image



Noisy image, PSNR = 24.24dB

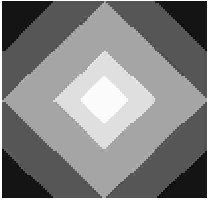


Perona-Malik
PSNR = 34.03dB

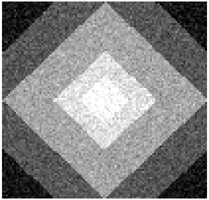


Perona-Malik
PSNR = 30.46dB

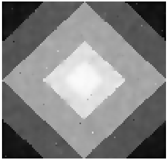
Comparison with Perona-Malik results



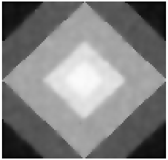
Original image



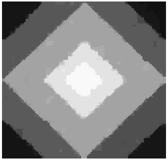
Noisy image, PSNR = 24.24dB



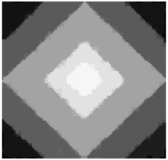
Perona-Malik
PSNR = 34.03dB



Perona-Malik
PSNR = 30.46dB



Power watershed
$x^0 = GF(f)$
PSNR = 31.40dB



Power watershed
$x^0 = MF(f)$
PSNR = 31.54dB

Conclusion and future work

- New framework unifying Graph Cuts, Random Walker, Shortest paths and Watershed.

Conclusion and future work

- New framework unifying Graph Cuts, Random Walker, Shortest paths and Watershed.
- The $p \to \infty, \ q = 2$ algorithm shows segmentation improvement while retaining watershed speed.

Conclusion and future work

- New framework unifying Graph Cuts, Random Walker, Shortest paths and Watershed.
- The $p \to \infty, \ q = 2$ algorithm shows segmentation improvement while retaining watershed speed.
- Unary terms formulation makes power watershed useful beyond segmentation, for example anisotropic diffusion.

Conclusion and future work

- New framework unifying Graph Cuts, Random Walker, Shortest paths and Watershed.
- The $p \to \infty,\ q = 2$ algorithm shows segmentation improvement while retaining watershed speed.
- Unary terms formulation makes power watershed useful beyond segmentation, for example anisotropic diffusion.
- Efficient robust error minimization with ℓ_0 norm

Conclusion and future work

- New framework unifying Graph Cuts, Random Walker, Shortest paths and Watershed.
- The $p \to \infty, \ q = 2$ algorithm shows segmentation improvement while retaining watershed speed.
- Unary terms formulation makes power watershed useful beyond segmentation, for example anisotropic diffusion.
- Efficient robust error minimization with ℓ_0 norm

Future work

- Caracterize the different energies that can be minimized in this framework
- Apply the power watershed algorithm to other computer vision problems

Questions

Reference books

- *Leo Grady and Jonathan R. Polimeni, "Discrete Calculus : Applied Analysis on Graphs for Computational Science", Springer, 2010.*
- *Laurent Najman and Hugues Talbot, "Mathematical morphology : from theory to applications", ISTE-Wiley, 2010.*

Source code for segmentation available from:

http ://sourceforge.net/projects/powerwatershed/

References

Bibliography

- Couprie, C., Grady, L., Najman, L. and Talbot, H. :
  Power Watersheds : A unifying graph-based optimization framework
  In *PAMI 2010*
- Couprie, C., Grady, L., Najman, L. and Talbot, H. :
  Power watersheds : A new image segmentation framework extending graph cuts, random walker and optimal spanning forest.
  In *Proc. of ICCV 2009*
- Couprie, C., Grady, L., Najman, L. and Talbot, H. :
  Anisotropic Diffusion Using Power Watersheds.
  In *Proc. of ICIP 2010*

References

Bibliography

- A. K. Sinop, L. Grady
  A Seeded Image Segmentation Framework Unifying Graph Cuts and Random Walker Which Yields a New Algorithm
  In *ICCV 2007*
- C. Allène, J-Y. Audibert, M. Couprie, R. Keriven
  Some links between min cuts, optimal spanning forests and watersheds
  In *Image and Vision Computing 2010*
- J. Cousty, G. Bertrand, L. Najman, M. Couprie.
  Watershed cuts : minimum spanning forests, and the drop of water principle.
  In *PAMI, 2009*

Properties

Definition

Let s be the segmentation defined by a thresholding of the labels

$$x = \arg\min \sum_{e_{ij} \in E} w_{ij}^p |x_i - x_j|^q.$$

The set of edges e_{ij} *that verify* $s_i \neq s_j$ *constitute a q-cut for* w^p.

Properties

Definition

Let s be the segmentation defined by a thresholding of the labels

$$x = \arg\min \sum_{e_{ij} \in E} w_{ij}^p |x_i - x_j|^q.$$

The set of edges e_{ij} that verify $s_i \neq s_j$ constitute a q-cut for w^p.

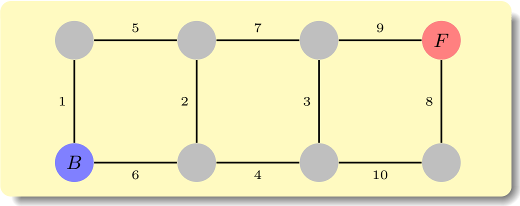
Theorem

If seeds correspond to maxima of the weight function, then any q-cut ($q \geq 1$) when $p \to \infty$ is an MSF cut.

Theorem and proof illustration

Theorem

If seeds correspond to maxima of the weight function, then any q-cut ($q \geq 1$) when $p \to \infty$ is an MSF cut.

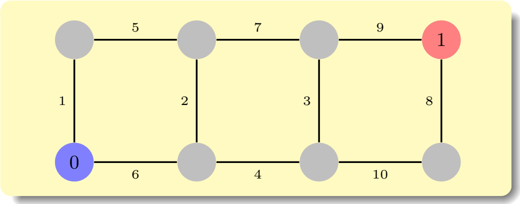


Recall the energy function : $\min_x \sum_{e_{ij} \in E} w_{ij}^p |x_i - x_j|^q$

Theorem and proof illustration

Theorem

If seeds correspond to maxima of the weight function, then any q-cut ($q \geq 1$) when $p \to \infty$ is an MSF cut.

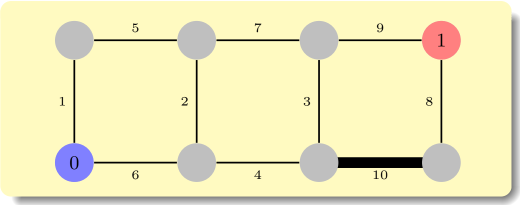


Recall the energy function : $\min_x \sum_{e_{ij} \in E} w_{ij}^p |x_i - x_j|^q$

Theorem and proof illustration

Theorem

If seeds correspond to maxima of the weight function, then any q-cut ($q \geq 1$) when $p \to \infty$ is an MSF cut.

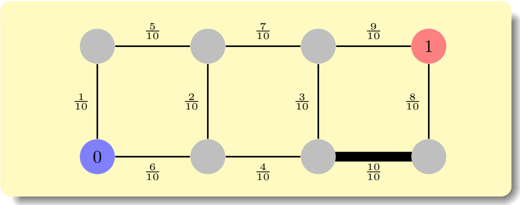


Recall the energy function : $\min_x \sum_{e_{ij} \in E} w_{ij}^p |x_i - x_j|^q$

Theorem and proof illustration

Theorem

If seeds correspond to maxima of the weight function, then any q-cut ($q \geq 1$) when $p \to \infty$ is an MSF cut.

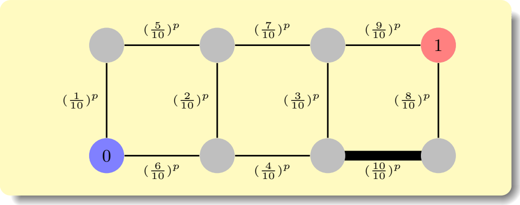


Recall the energy function : $\min_x \sum_{e_{ij} \in E} w_{ij}^p |x_i - x_j|^q$

Theorem and proof illustration

Theorem

If seeds correspond to maxima of the weight function, then any q-cut ($q \geq 1$) when $p \to \infty$ is an MSF cut.

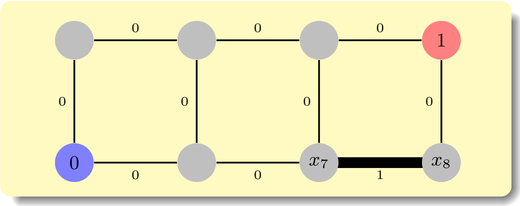


Recall the energy function : $\min_x \sum_{e_{ij} \in E} w_{ij}^p |x_i - x_j|^q$

Theorem and proof illustration

Theorem

If seeds correspond to maxima of the weight function, then any q-cut ($q \geq 1$) when $p \to \infty$ is an MSF cut.

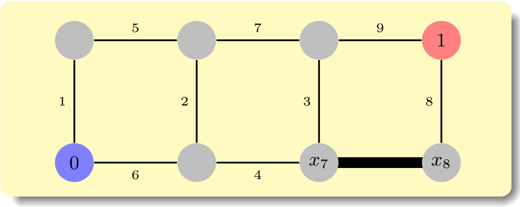


Recall the energy function : $\min_x \sum_{e_{ij} \in E} w_{ij}^p |x_i - x_j|^q$

Theorem and proof illustration

Theorem

If seeds correspond to maxima of the weight function, then any q-cut ($q \geq 1$) when $p \to \infty$ is an MSF cut.



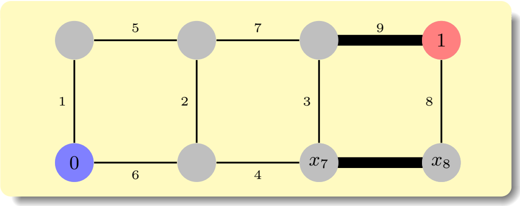
$x_7 = x_8$

Recall the energy function : $\min_x \sum_{e_{ij} \in E} w_{ij}^p |x_i - x_j|^q$

Theorem and proof illustration

Theorem

If seeds correspond to maxima of the weight function, then any q-cut ($q \geq 1$) when $p \to \infty$ is an MSF cut.



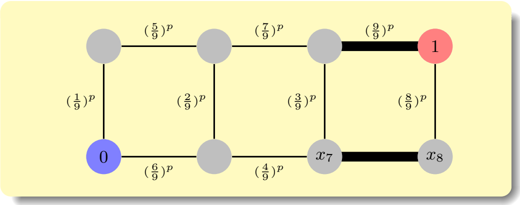
$x_7 = x_8$

Recall the energy function : $\min_x \sum_{e_{ij} \in E} w_{ij}^p |x_i - x_j|^q$

Theorem and proof illustration

Theorem

If seeds correspond to maxima of the weight function, then any q-cut ($q \geq 1$) when $p \rightarrow \infty$ is an MSF cut.



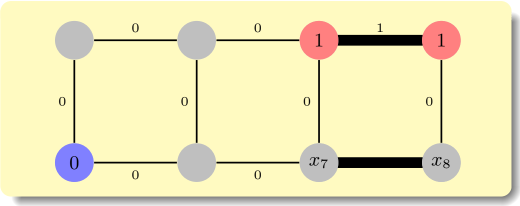
$x_7 = x_8$

Recall the energy function : $\min_x \sum_{e_{ij} \in E} w_{ij}^p |x_i - x_j|^q$

Theorem and proof illustration

Theorem

If seeds correspond to maxima of the weight function, then any q-cut ($q \geq 1$) when $p \to \infty$ is an MSF cut.



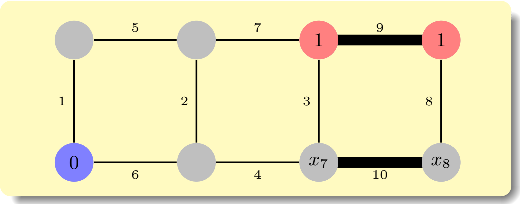
$x_7 = x_8$

Recall the energy function : $\min_x \sum_{e_{ij} \in E} w_{ij}^p |x_i - x_j|^q$

Theorem and proof illustration

Theorem

If seeds correspond to maxima of the weight function, then any q-cut ($q \geq 1$) when $p \to \infty$ is an MSF cut.



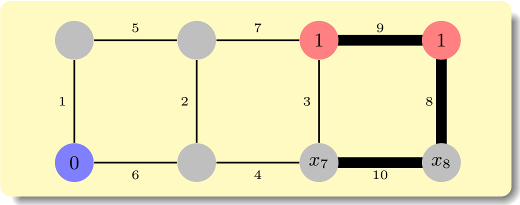
$x_7 = x_8$

Recall the energy function : $\min_x \sum_{e_{ij} \in E} w_{ij}^p |x_i - x_j|^q$

Theorem and proof illustration

Theorem

If seeds correspond to maxima of the weight function, then any q-cut ($q \geq 1$) when $p \to \infty$ is an MSF cut.



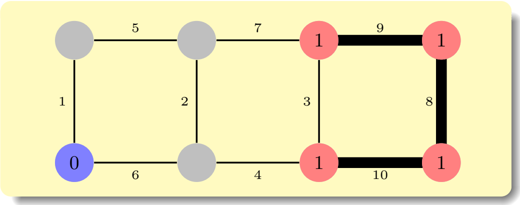
$x_7 = x_8$

Recall the energy function : $\min_x \sum_{e_{ij} \in E} w_{ij}^p |x_i - x_j|^q$

Theorem and proof illustration

Theorem

If seeds correspond to maxima of the weight function, then any q-cut ($q \geq 1$) when $p \to \infty$ is an MSF cut.

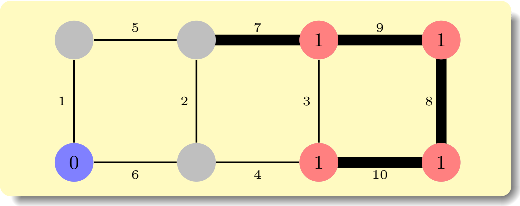


Recall the energy function : $\min_x \sum_{e_{ij} \in E} w_{ij}^p |x_i - x_j|^q$

Theorem and proof illustration

Theorem

If seeds correspond to maxima of the weight function, then any q-cut ($q \geq 1$) when $p \to \infty$ is an MSF cut.

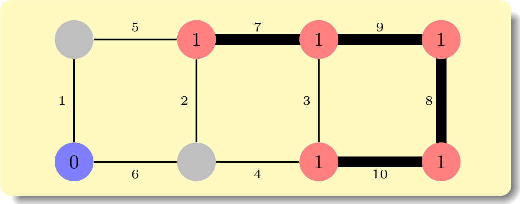


Recall the energy function : $\min_x \sum_{e_{ij} \in E} w_{ij}^p |x_i - x_j|^q$

Theorem and proof illustration

Theorem

If seeds correspond to maxima of the weight function, then any q-cut ($q \geq 1$) when $p \to \infty$ is an MSF cut.

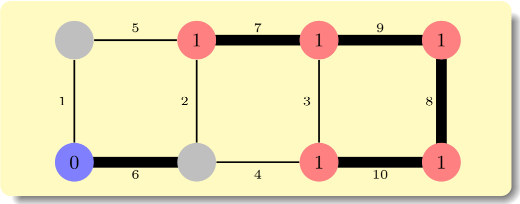


Recall the energy function : $\min_x \sum_{e_{ij} \in E} w_{ij}^p |x_i - x_j|^q$

Theorem and proof illustration

Theorem

If seeds correspond to maxima of the weight function, then any q-cut ($q \geq 1$) when $p \to \infty$ is an MSF cut.

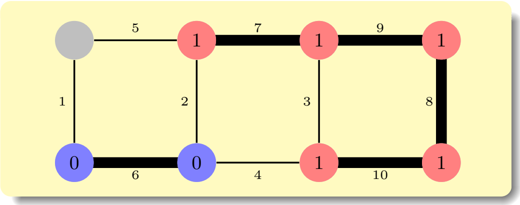


Recall the energy function : $\min_x \sum_{e_{ij} \in E} w_{ij}^p |x_i - x_j|^q$

Theorem and proof illustration

Theorem

If seeds correspond to maxima of the weight function, then any q-cut ($q \geq 1$) when $p \to \infty$ is an MSF cut.

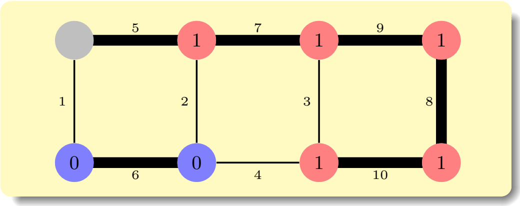


Recall the energy function : $\min_x \sum_{e_{ij} \in E} w_{ij}^p |x_i - x_j|^q$

Theorem and proof illustration

Theorem

If seeds correspond to maxima of the weight function, then any q-cut ($q \geq 1$) when $p \to \infty$ is an MSF cut.

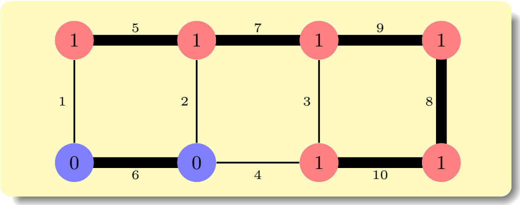


Recall the energy function : $\min_x \sum_{e_{ij} \in E} w_{ij}^p |x_i - x_j|^q$

Theorem and proof illustration

Theorem

If seeds correspond to maxima of the weight function, then any q-cut ($q \geq 1$) when $p \to \infty$ is an MSF cut.



Recall the energy function : $\min_x \sum_{e_{ij} \in E} w_{ij}^p |x_i - x_j|^q$

Example where RW with $p \to \infty$ is not a MSF

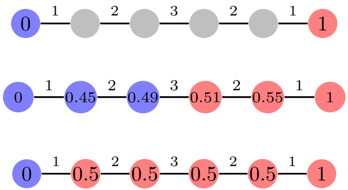


Figure: Example of graph where the q-cut computed by the minimization of $E_{p,q}$ is not a MaxSF cut. (a) weighted seeded graph, (b) Random walker result (q=2) when the weights are at the power p=5. The q-cut is in the center of the graph. (c) power watershed result (q=2) corresponding to the limit of the Random walker result (q=2) when the power of the weights converges toward infinity.