Contour cut: identifying salient contours in images by solving a Hermitian eigenvalue problem

Ryan Kennedy Jean Gallier Jianbo Shi

Problem

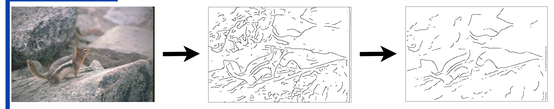


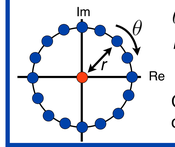
Image → Edgels → Contours

We want to *locate* and *rank* the most salient image contours. Edge linking fails in cluttered regions where multiple local paths exist.

Circular Embedding

Contours are made of edgels, and are represented by:

1) *Indicator* of whether each edgel is on the contour
2) *Ordering* of edgels on the contours



θ = *Ordering*

r = *Indicator*

Edgel j assigned: $\mathbf{x_j} = \mathbf{r_j} e^{i\theta_j}$

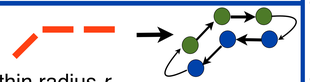
Ideal contour: $r_j \in \{0, 1\}$

$\theta_j = O(j)\delta$

$\delta = \frac{2\pi}{|C|}$

Contour x represented as vector of complex numbers, one per edgel.

Directed Graph Setup



Vertices: 2 per edgel

Edges: connect all edgels within radius r

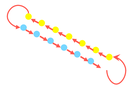
Edge weights: based on collinearity of edgels

Image contour

Open contour

Closed contour

Graph cycle



What makes a good contour?

A good contour has:

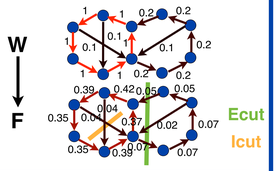
1) **high** weights locally on the contour
2) **low** weights otherwise
   a) **low** weights between distant nodes on the contour
   b) **low** weights between nodes on and off the contour

- It is ***persistent***: a random walk will return at regular intervals

External cut: sum of weights between nodes **on** and **off** the contour

Internal cut: sum of weights between **distant** nodes **on** the contour

	Internal cut	External cut
Clique	↑	↓
Contour	↓	↓



Directed graph cuts

- Graph cuts are not symmetric for non-symmetric W
- Instead of W, we use a *graph circulation matrix* F such that $\sum_i F_{ij} = \sum_k F_{jk}$
- In particular, we use $F = \Pi P$
  - $P = D^{-1} W$ is the random-walk matrix
  - $\Pi = diag(\pi)$, where π is the stationary distribution of P
- Graph cuts on F are symmetric
- If W is symmetric, cuts on W and F are the same

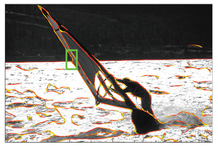
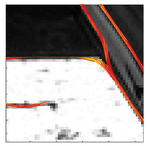
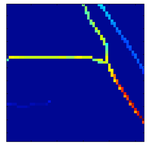


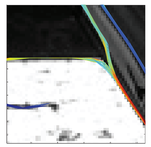
Image with graph edges colored by weight of W



Zoom in of junction



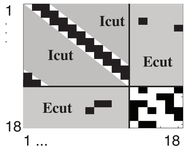
Stationary distribution



Graph weights colored by $F = \Pi P$

Cost function

$$Ccut(C, O) = \frac{Icut(C, O) + Ecut(C, O)}{Vol(C)}$$



Similar to *Normalized Cut*,

$$Ncut(A) = \frac{Ecut(A, \bar{A})}{Vol(A)} + \frac{Ecut(\bar{A}, A)}{Vol(\bar{A})}$$

but includes ***internal cut*** and is ***one-sided***.

$$Icut(\mathbf{x}) = \sum_{i,j} F_{ij} \mathbf{r}_i \mathbf{r}_j [1 - \cos(\theta_j - \theta_i - \delta)] \qquad Ecut(\mathbf{x}) = \sum_{i,j} F_{ij} \mathbf{r}_i (1 - \mathbf{r}_j)$$

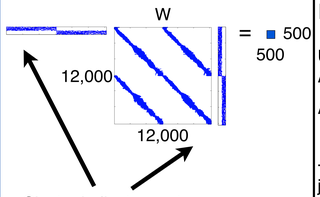
$$Ccut(\mathbf{x}) = \frac{\mathbf{x}^*(\Pi - H(\delta))\mathbf{x}}{\mathbf{x}^* \Pi \mathbf{x}} \qquad H(\delta) = \frac{F \exp(-i\delta) + F^T \exp(i\delta)}{2}$$

Relaxed optimization problem: $\max_{\mathbf{x} \in \mathbb{C}^{|C|}, \delta \in [0, 2\pi]} \frac{\mathbf{x}^* H(\delta) \mathbf{x}}{\mathbf{x}^* \Pi \mathbf{x}}$

We want to find all local maxima, each of which is a contour.

Approximations

Compress the graph using Ncut



Cluster indicator vectors

Use left/right eigenvectors of *P*

$$\Pi^{-1} H = \frac{Pe^{-i\delta} + \Pi^{-1} P^T \Pi e^{i\delta}}{2}$$

Use only **left** or only **right** side

Approximation 1: x where $Px = \lambda x$.

Approximation 2: x where $P^T y = \lambda y$

$x = \Pi^{-1} y$

- No need to search over δ: changing δ just scales the eigenvalues.
- This was (Zhu et al., 2007)'s solution.

Previous work

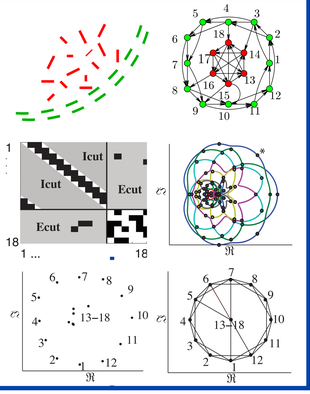
(Zhu et al., 2007) relaxed the problem further. Their solution was directly the eigenvectors of P. This is only a solution for us when P is ***normal***: $PP^T = P^T P$.

Solution

- For a fixed δ, solutions are eigenvectors of $\Pi^{-1} H(\delta)$.
- Search over all δ, find eigenvalues that are local maxima w.r.t. δ.

Algorithm 1 Contour cut

Calculate matrices P and Π from W.
for $\delta = \delta_{\min}$ to $\delta_{\max}$ **do**
  $H(\delta) \leftarrow (\Pi P \exp(-i\delta) + P^T \Pi \exp(i\delta))/2$
  Solve $\Pi^{-1} H(\delta) x = \lambda x$ for top k eigenvectors.
end for
Find local maxima of each λ_i over all δ. The associated x's are critical points with respect to both x and δ.



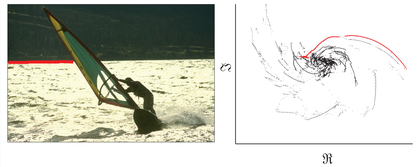
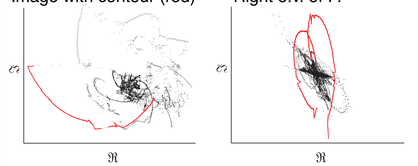


Image with contour (red)

Right e.v. of P.



Scaled left e.v. of P.

Our solution

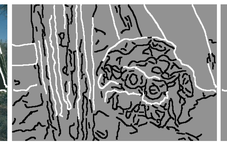
Results

Graph size	Algorithm	Top 10 contours	Top 20 contours
n=500	H(F)	**0.8312**	**0.7088**
	P	0.6047	0.5003
	P^T	0.7033	0.5098
	H(P)	0.8246	0.6896
n=1000	H(F)	**0.8292**	**0.7403**
	P	0.4569	0.4277
	P^T	0.6470	0.5460
	H(P)	0.7887	0.6743

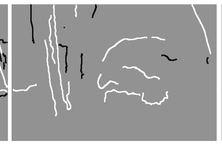
Graph size	Algorithm	Top 10 contours	Top 20 contours
Full graph	H(F)	**0.8411**	**0.7452**
	P	0.6833	0.4970
	P^T	0.7391	0.6037

Average value of the cost function, $\frac{x^* H(\delta) x}{x^* \Pi x}$, with respect to the full graph.

Image and top 20 contours found by our algorithm



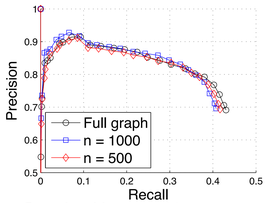
Pb edges and top 20 contours found by our algorithm



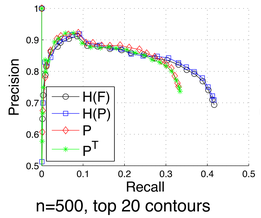
XOR versus eigenvectors of P

XOR versus scaled eigenvectors of P^T

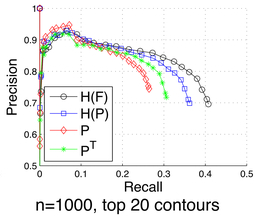
White: only our algorithm found
Black: only approximate algorithm found



Our algorithm, top 20 contours. Graph compression has little effect on the results.

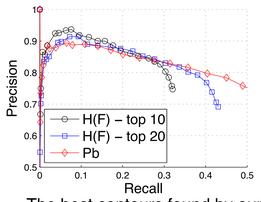


n=500, top 20 contours



n=1000, top 20 contours

Comparison of algorithms for different graph sizes. Our algorithm outperforms others, especially for n=1000.



The best contours found by our algorithm are more salient than the average input of Pb.