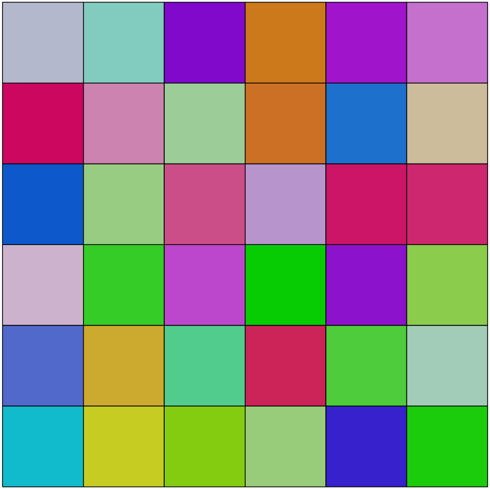
Graph-based segmentation

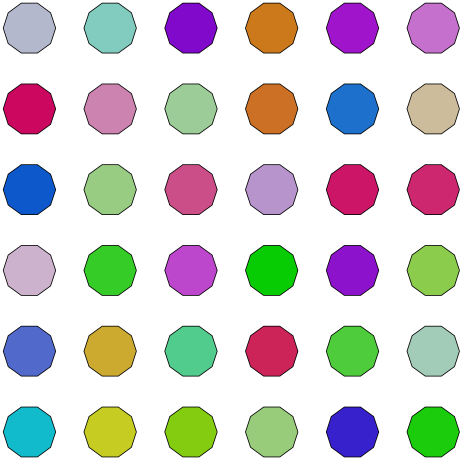
Ondrej Drbohlav

November 2012

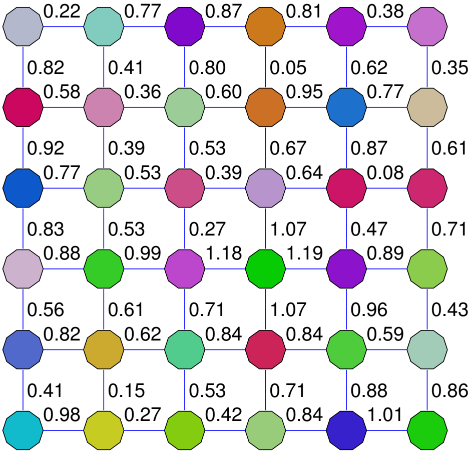
Image



Image, convert to graph



Image, convert to graph



pixels = vertices

edges capture dissimilarity of pixels

How to segment?

- Recall that segmentation is the partitioning of an image to connected components
- One of the simplest but very efficient algorithms based on greedily forming the components, bottom-up
- Felzenszwalb and Huttenlocher: *Efficient Graph-Based Image Segmentation*. International Journal of Computer Vision, Volume 59, Number 2, September 2004.

Base principles(1)

Partition to connected components such that within-component differences are lower than between-component differences.

Within-component difference in component C:

$$Int(C) = \max_{e \in MST(C)} w(e)$$

Difference between components C_1 and C_2:

$$Dif(C_1, C_2) = \min_{v_1 \in C_1, v_2 \in C_2, (v_1, v_2) \in E} w(v_1, v_2)$$

Base principles(2)

Algorithm:

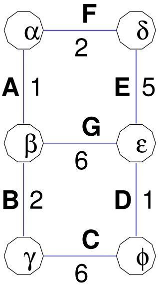
- init: components = vertices
- sort edges
- repeat: take an edge e and fuse the components C_1 and C_2 it connects if:
  - $C_1 \neq C_2$
  - $w(e) < min(Int(C_1) + \tau(C_1), Int(C_2) + \tau(C_2))$

τ is a threshold function. The larger it is, the more it encourages the two components to merge.

Toy example: segmentation process

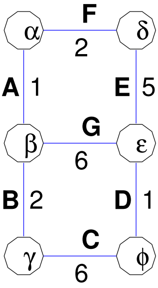
pixels $\alpha - \phi$, edges A-G

- initizalization: components = $\{\alpha, \beta, \gamma, \delta, \epsilon, \phi\}$



Toy example: segmentation process

pixels $\alpha - \phi$, edges A-G

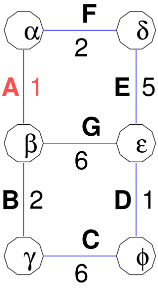


- initizalization: components $=$ $\{\alpha, \beta, \gamma, \delta, \epsilon, \phi\}$
- sort edges by weight; and let $\tau(\cdot) = k = 3$

	A	D	F	B	E	C	G
weight	1	1	2	2	5	6	6

Toy example: segmentation process

pixels $\alpha - \phi$, edges A-G



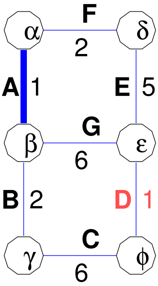
- initizalization: components $= \{\alpha, \beta, \gamma, \delta, \epsilon, \phi\}$
- sort edges by weight; and let $\tau(\cdot) = k = 3$

	A	D	F	B	E	C	G
weight	1	1	2	2	5	6	6

- A: merge α and β? $1 < 0 + k \Rightarrow$ yes.

Toy example: segmentation process

pixels $\alpha - \phi$, edges A-G



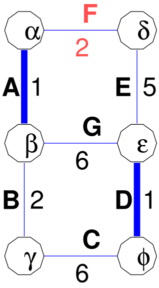
- initizalization: components = $\{\alpha, \beta, \gamma, \delta, \epsilon, \phi\}$
- sort edges by weight; and let $\tau(\cdot) = k = 3$

	A	D	F	B	E	C	G
weight	1	1	2	2	5	6	6

- A: merge α and β? $1 < 0 + k \Rightarrow$ yes.
- D: merge ϵ and ϕ? $1 < 0 + k \Rightarrow$ yes.

Toy example: segmentation process

pixels $\alpha - \phi$, edges A-G



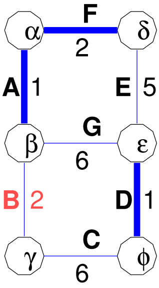
- initizalization: components = $\{\alpha, \beta, \gamma, \delta, \epsilon, \phi\}$
- sort edges by weight; and let $\tau(\cdot) = k = 3$

	A	D	F	B	E	C	G
weight	1	1	2	2	5	6	6

- A: merge α and β? $1 < 0 + k \Rightarrow$ yes.
- D: merge ϵ and ϕ? $1 < 0 + k \Rightarrow$ yes.
- F: merge $\alpha\beta$ and δ? $1 < 0 + k \Rightarrow$ yes.

Toy example: segmentation process

pixels $\alpha - \phi$, edges A-G



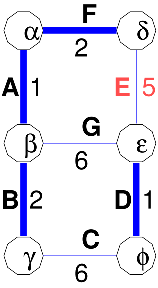
- initizalization: components = $\{\alpha, \beta, \gamma, \delta, \epsilon, \phi\}$
- sort edges by weight; and let $\tau(\cdot) = k = 3$

	A	D	F	B	E	C	G
weight	1	1	2	2	5	6	6

- A: merge α and β? $1 < 0 + k \Rightarrow$ yes.
- D: merge ϵ and ϕ? $1 < 0 + k \Rightarrow$ yes.
- F: merge $\alpha\beta$ and δ? $1 < 0 + k \Rightarrow$ yes.
- B: merge $\alpha\beta\delta$ and γ? $2 < 0 + k \Rightarrow$ yes.

Toy example: segmentation process

pixels $\alpha - \phi$, edges A-G



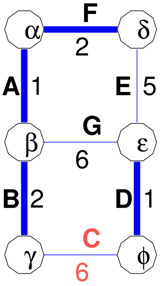
- initizalization: components = $\{\alpha, \beta, \gamma, \delta, \epsilon, \phi\}$
- sort edges by weight; and let $\tau(\cdot) = k = 3$

	A	D	F	B	E	C	G
weight	1	1	2	2	5	6	6

- A: merge α and β? $1 < 0 + k \Rightarrow$ yes.
- D: merge ϵ and ϕ? $1 < 0 + k \Rightarrow$ yes.
- F: merge $\alpha\beta$ and δ? $1 < 0 + k \Rightarrow$ yes.
- B: merge $\alpha\beta\delta$ and γ? $2 < 0 + k \Rightarrow$ yes.
- E: merge $\alpha\beta\gamma\delta$ and $\epsilon\phi$? $5 \not< 1 + k \Rightarrow$ no.

Toy example: segmentation process

pixels $\alpha - \phi$, edges A-G



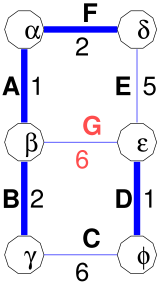
- initizalization: components = $\{\alpha, \beta, \gamma, \delta, \epsilon, \phi\}$
- sort edges by weight; and let $\tau(\cdot) = k = 3$

	A	D	F	B	E	C	G
weight	1	1	2	2	5	6	6

- A: merge α and β? $1 < 0 + k \Rightarrow$ yes.
- D: merge ϵ and ϕ? $1 < 0 + k \Rightarrow$ yes.
- F: merge $\alpha\beta$ and δ? $1 < 0 + k \Rightarrow$ yes.
- B: merge $\alpha\beta\delta$ and γ? $2 < 0 + k \Rightarrow$ yes.
- E: merge $\alpha\beta\gamma\delta$ and $\epsilon\phi$? $5 \not< 1 + k \Rightarrow$ no.
- C: merge $\alpha\beta\gamma\delta$ and $\epsilon\phi$? $6 \not< 0 + k \Rightarrow$ no.

Toy example: segmentation process

pixels $\alpha - \phi$, edges A-G

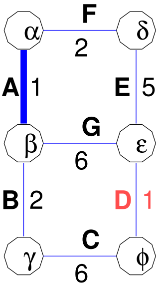


- initizalization: components = $\{\alpha, \beta, \gamma, \delta, \epsilon, \phi\}$
- sort edges by weight; and let $\tau(\cdot) = k = 3$

	A	D	F	B	E	C	G
weight	1	1	2	2	5	6	6

- A: merge α and β? $1 < 0 + k \Rightarrow$ yes.
- D: merge ϵ and ϕ? $1 < 0 + k \Rightarrow$ yes.
- F: merge $\alpha\beta$ and δ? $1 < 0 + k \Rightarrow$ yes.
- B: merge $\alpha\beta\delta$ and γ? $2 < 0 + k \Rightarrow$ yes.
- E: merge $\alpha\beta\gamma\delta$ and $\epsilon\phi$? $5 \not< 1 + k \Rightarrow$ no.
- C: merge $\alpha\beta\gamma\delta$ and $\epsilon\phi$? $6 \not< 0 + k \Rightarrow$ no.
- G: ditto $\Rightarrow$ no.

Toy example: final segmentation



- Final segmentation: two components, $\alpha\beta\gamma\delta$ and $\epsilon\phi$.

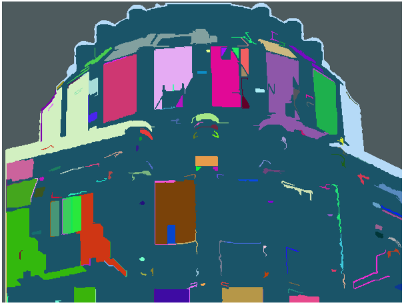
Real example

$\sigma = 0.7,\ k = 100$

Real example

$\sigma = 0.7,\ k = 500$

Real example



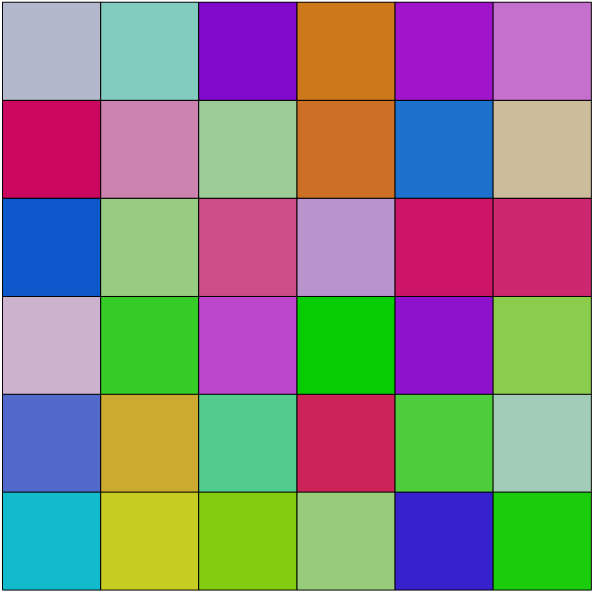
$\sigma = 0.7,\ k = 2000$

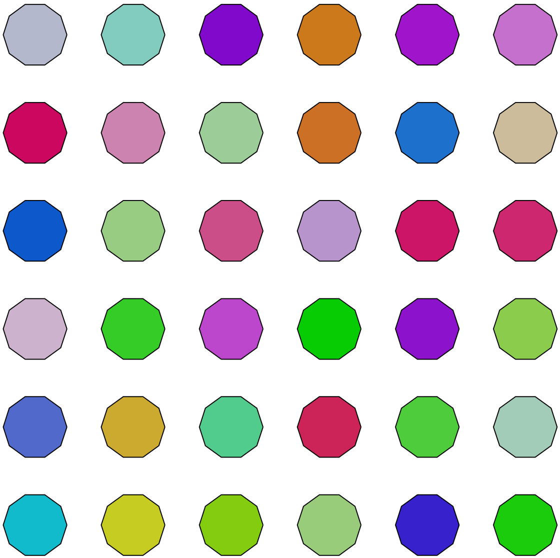
Final notes (for this algorithm)

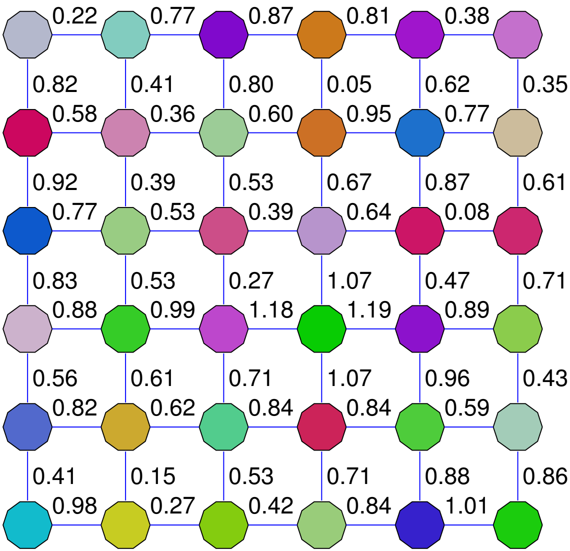
- Threshold $\tau(C) = k$ was used here for simplicity of explanation. In the real implementation, $\tau(C) = k/|C|$, $|C|$ being the cardinality of component C.
- The complexity of the algorithm is very low. The edge sorting can be done in $O(N \log N)$ time, and when edges are integers then in $O(N)$ time.
- During the algorithms, components need to be merged. This is done using an efficient union-find algorithm whose complexity is close to linear.

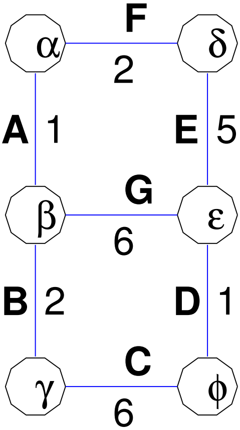
Graph Cuts for Image Segmentation

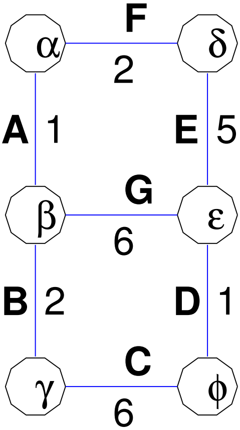
- Basic principle, demonstration on a toy example: blackboard explanation.
- plus, see file **mrf.pdf**, pages 34-50

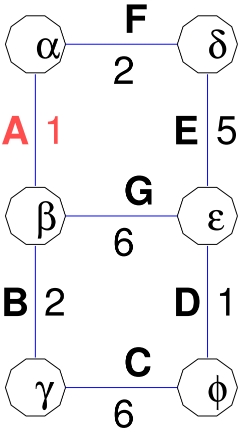


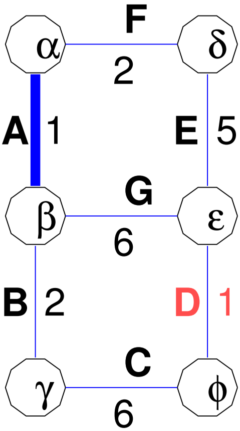


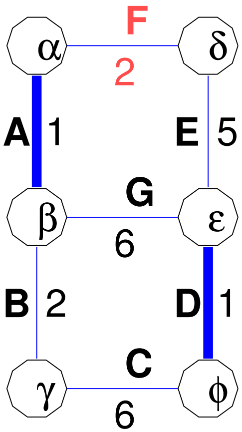

0.22
0.77
0.87
0.81
0.38
0.82
0.41
0.80
0.05
0.62
0.35
0.58
0.36
0.60
0.95
0.77
0.92
0.39
0.53
0.67
0.87
0.61
0.77
0.53
0.39
0.64
0.08
0.83
0.53
0.27
1.07
0.47
0.71
0.88
0.99
1.18
1.19
0.89
0.56
0.61
0.71
1.07
0.96
0.43
0.82
0.62
0.84
0.84
0.59
0.41
0.15
0.53
0.71
0.88
0.86
0.98
0.27
0.42
0.84
1.01

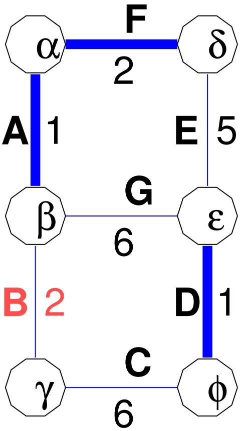

F
α
δ
2
A
1
E
5
G
β
ε
6
B
2
D
1
C
γ
ϕ
6

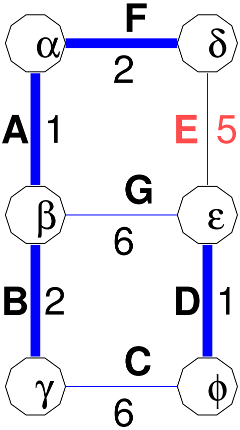

F
α
2
δ
A
1
E
5
G
β
6
ε
B
2
D
1
C
γ
6
ϕ

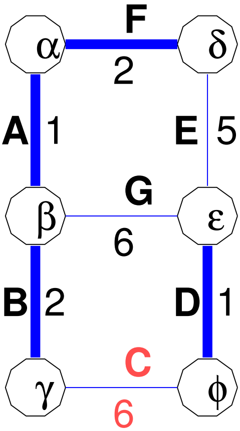

F
α
δ
2
A
1
E
5
G
β
ε
6
B
2
D
1
C
γ
ϕ
6

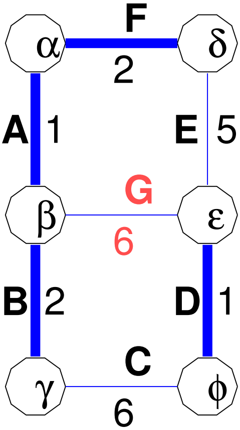

F
α
δ
2
A
1
E
5
G
β
ε
6
B
2
D
1
C
γ
φ
6

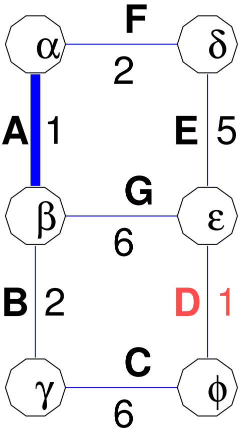

F
α
δ
2
A
1
E
5
G
β
ε
6
B
2
D
1
C
γ
ϕ
6


F
α
δ
2
A
1
E
5
G
β
ε
6
B
2
D
1
C
γ
ϕ
6


F
α
δ
2
A
1
E
5
G
β
ε
6
B
2
D
1
C
γ
ϕ
6


F
α
δ
2
A
1
E
5
G
β
ε
6
B
2
D
1
C
γ
φ
6


F
2
α
δ
A
1
E
5
G
6
β
ε
B
2
D
1
C
6
γ
ϕ


F
α
δ
2
A
1
E
5
G
β
ε
6
B
2
D
1
C
γ
ϕ
6