Graph cuts and computer vision
an introduction

Guillaume Charpiat

Pulsar Project, INRIA

INRIA Sophia-Antipolis

15/05/09

Map

- **Introduction :** energie minimization
- **Max flow** and min cut (graph theory)
- **Images as graphs :** an efficient minimization tool
- **Extensions**
- **Discussion**

Introduction : energies

Algorithms and energies

Usual method in computer science :
state the problem $\Longrightarrow$ write an algorithm $\Longrightarrow$ suitable ?

Good case : proof available

$\hookrightarrow$ prove that the algorithm solves the problem
$\hookrightarrow$ ex: to sort a list of words in alphabetical order

Bad case : no proof

$\hookrightarrow$ problem is not precise

$\hookrightarrow$ or it is not clear how the algorithm compares to other ones.

$\Longrightarrow$ there is something wrong !

Introduction : energies

Algorithms and energies

Usual method in computer science :
state the problem $\Longrightarrow$ write an algorithm $\Longrightarrow$ suitable ?

Good case : proof available

$\hookrightarrow$ prove that the algorithm solves the problem
$\hookrightarrow$ ex: to sort a list of words in alphabetical order

Bad case : no proof

$\hookrightarrow$ problem is not precise

$\hookrightarrow$ or it is not clear how the algorithm compares to other ones.

$\Longrightarrow$ there is something wrong !

Introduction : energies

Algorithms and energies

Usual method in computer science :
state the problem $\Longrightarrow$ write an algorithm $\Longrightarrow$ suitable ?

Good case : proof available

$\hookrightarrow$ prove that the algorithm solves the problem
$\hookrightarrow$ ex: to sort a list of words in alphabetical order

Bad case : no proof

$\hookrightarrow$ problem is not precise

$\hookrightarrow$ or it is not clear how the algorithm compares to other ones.

$\Longrightarrow$ there is something wrong !

Introduction : energies

Algorithms and energies

Usual method in computer science :

state the problem $\Longrightarrow$ write an algorithm $\Longrightarrow$ suitable ?

Good case : proof available

$\hookrightarrow$ prove that the algorithm solves the problem

$\hookrightarrow$ ex: to sort a list of words in alphabetical order

Bad case : no proof

$\hookrightarrow$ problem is not precise

$\hookrightarrow$ or it is not clear how the algorithm compares to other ones.

$\Longrightarrow$ there is something wrong !

Introduction : energies

Algorithms and energies

Usual method in computer science :

state the problem $\Longrightarrow$ write an algorithm $\Longrightarrow$ suitable ?

Good case : proof available

$\hookrightarrow$ prove that the algorithm solves the problem

$\hookrightarrow$ ex: to sort a list of words in alphabetical order

Bad case : no proof

$\hookrightarrow$ problem is not precise $\Longrightarrow$ need for more modelisation

$\hookrightarrow$ or it is not clear how the algorithm compares to other ones.

$\Longrightarrow$ need for an objective criterion for quantitative comparison

Energies

Quantitative criterion C related to the current problem:

$\hookrightarrow$ comparison of possible answers : $C(A_1) > C(A_2)$?

$\hookrightarrow$ state the problem mathematically : search for the optimal answer A_0 s.t.:

$$A_0 \in \arg\sup_{A \in \mathcal{X}} C(A)$$

Usually expressed as an energy $E(A)$ to be minimized:

$\hookrightarrow$ search for the optimum $A_0 \in \arg\inf_{A \in \mathcal{X}} E(A)$

$\hookrightarrow$ $\mathcal{X}$: search space (including constraints)

Energies

Quantitative criterion C related to the current problem:

$\hookrightarrow$ comparison of possible answers : $C(A_1) > C(A_2)$?

$\hookrightarrow$ state the problem mathematically : search for the optimal answer A_0 s.t.:

$$A_0 \in \arg\sup_{A \in \mathcal{X}} C(A)$$

Usually expressed as an energy $E(A)$ to be minimized:

$\hookrightarrow$ search for the optimum $A_0 \in \arg\inf_{A \in \mathcal{X}} E(A)$

$\hookrightarrow$ $\mathcal{X}$: search space (including constraints)

How to minimize energies

- Best case : explicit formula for the solution
  - ↪ ex: search for the center of a cloud of n points P_i
  - ↪ set a mathematical definition : center = mean coordinates
  - ↪ $$\overrightarrow{M} = \frac{1}{n} \sum_i \overrightarrow{P_i}$$
  - ↪ why average ?
- Special cases : ad hoc minimization method suited
  - ↪ Energy and constraints write a particular way
  - ↪ Graph-cuts, loopy belief propagation, kernel methods, dynamic time warping, linear programming, minimum cycles, etc.
- General case : ?
  - ↪ discrete variables : exhaustive search... or stochastic methods (Gibbs...)
  - ↪ continuous variables : gradient descents (possibly stochastic)
    $\Longrightarrow$ local optima, result depends on initialization if non-convex problem

How to minimize energies

- Best case : explicit formula for the solution
  - $\hookrightarrow$ ex: search for the center of a cloud of n points P_i
  - $\hookrightarrow$ set a mathematical definition : center = closest point to all
  - $\hookrightarrow$ $$E(M) = \sum_{1 \leqslant i \leqslant n} \|\overrightarrow{MP_i}\|_2^2$$
  - $\hookrightarrow$ $$\Longrightarrow \overrightarrow{M} = \frac{1}{n} \sum_i \overrightarrow{P_i}$$
  - $\hookrightarrow$ Warning : solution changes with energy design (choice of norm, power, weights, outliers...)
- Special cases : ad hoc minimization method suited
  - $\hookrightarrow$ Energy and constraints write a particular way
  - $\hookrightarrow$ Graph-cuts, loopy belief propagation, kernel methods, dynamic time warping, linear programming, minimum cycles, etc.
- General case : ?
  - $\hookrightarrow$ discrete variables : exhaustive search... or stochastic methods (Gibbs...)
  - $\hookrightarrow$ continuous variables : gradient descents (possibly stochastic) $\Longrightarrow$ local optima, result depends on initialization if non-convex problem

How to minimize energies

- Best case : explicit formula for the solution
  - ↪ ex: search for the center of a cloud of n points P_i
  - ↪ set a mathematical definition : center = best fitting Gaussian
  - ↪ $p(P_i) = \frac{1}{(2\pi)^{N/2}|\mathbf{S}|^{1/2}} e^{-\frac{1}{2}\overrightarrow{MP_i}\,\mathbf{S}\,\overrightarrow{MP_i}}$ parameters: M, $\mathbf{S}$
  - ↪ maximize likelihood : $L(M, \mathbf{S}) = \prod_i p(P_i)$
  - ↪ $E = -lnL = \frac{-n}{2}\left(\ln|\mathbf{S}| + M\mathbf{S}M - \frac{2}{n}M\mathbf{S}\sum_i P_i\right) + c^{st} \implies \overrightarrow{M} = \frac{1}{n}\sum_i \overrightarrow{P_i}$
- Special cases : ad hoc minimization method suited
  - ↪ Energy and constraints write a particular way
  - ↪ Graph-cuts, loopy belief propagation, kernel methods, dynamic time warping, linear programming, minimum cycles, etc.
- General case : ?
  - ↪ discrete variables : exhaustive search... or stochastic methods (Gibbs...)
  - ↪ continuous variables : gradient descents (possibly stochastic)
    $\implies$ local optima, result depends on initialization if non-convex problem

How to minimize energies

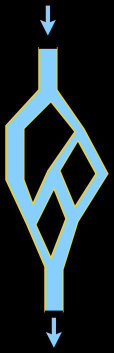
- Best case : explicit formula for the solution
  - $\hookrightarrow$ ex: search for the center of a cloud of n points P_i
  - $\hookrightarrow$ set a mathematical definition : center = best fitting Gaussian
  - $\hookrightarrow$ $p(P_i) = \frac{1}{(2\pi)^{N/2}|\mathbf{S}|^{1/2}} e^{-\frac{1}{2}\overrightarrow{MP_i}\,\mathbf{S}\,\overrightarrow{MP_i}}$ parameters: M, $\mathbf{S}$
  - $\hookrightarrow$ maximize likelihood : $L(M, \mathbf{S}) = \prod_i p(P_i)$
  - $\hookrightarrow$ $E = -lnL = \frac{-n}{2}\left(\ln|\mathbf{S}| + M\mathbf{S}M - \frac{2}{n}M\mathbf{S}\sum_i P_i\right) + c^{st} \implies \overrightarrow{M} = \frac{1}{n}\sum_i \overrightarrow{P_i}$
- Special cases : ad hoc minimization method suited
  - $\hookrightarrow$ Energy and constraints write a particular way
  - $\hookrightarrow$ Graph-cuts, loopy belief propagation, kernel methods, dynamic time warping, linear programming, minimum cycles, etc.
- General case : ?
  - $\hookrightarrow$ discrete variables : exhaustive search... or stochastic methods (Gibbs...)
  - $\hookrightarrow$ continuous variables : gradient descents (possibly stochastic)
    $\implies$ local optima, result depends on initialization if non-convex problem

How to minimize energies

- Best case : explicit formula for the solution
  - $\hookrightarrow$ ex: search for the center of a cloud of n points P_i
  - $\hookrightarrow$ set a mathematical definition : center = best fitting Gaussian
  - $\hookrightarrow$ $p(P_i) = \frac{1}{(2\pi)^{N/2}|\mathbf{S}|^{1/2}} e^{-\frac{1}{2}\overrightarrow{MP_i}\,\mathbf{S}\,\overrightarrow{MP_i}}$ parameters: M, $\mathbf{S}$
  - $\hookrightarrow$ maximize likelihood : $L(M, \mathbf{S}) = \prod_i p(P_i)$
  - $\hookrightarrow$ $E = -lnL = \frac{-n}{2}\left(\ln|\mathbf{S}| + M\mathbf{S}M - \frac{2}{n}M\mathbf{S}\sum_i P_i\right) + c^{st} \implies \overrightarrow{M} = \frac{1}{n}\sum_i \overrightarrow{P_i}$
- Special cases : ad hoc minimization method suited
  - $\hookrightarrow$ Energy and constraints write a particular way
  - $\hookrightarrow$ Graph-cuts, loopy belief propagation, kernel methods, dynamic time warping, linear programming, minimum cycles, etc.
- General case : ?
  - $\hookrightarrow$ discrete variables : exhaustive search... or stochastic methods (Gibbs...)
  - $\hookrightarrow$ continuous variables : gradient descents (possibly stochastic)
    $\implies$ local optima, result depends on initialization if non-convex problem

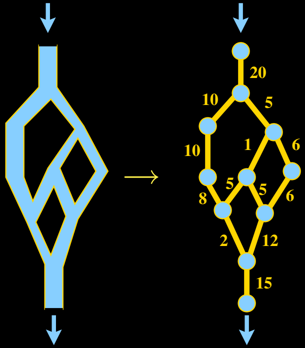
Max flow problem

Water pipes



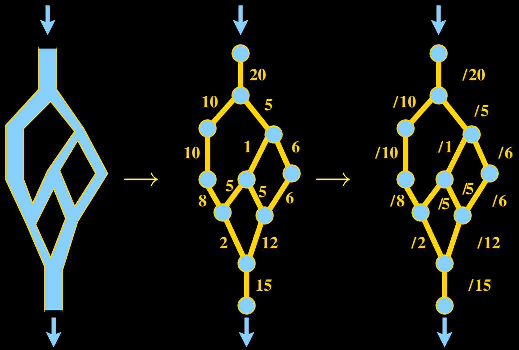
Max flow problem

Water pipes



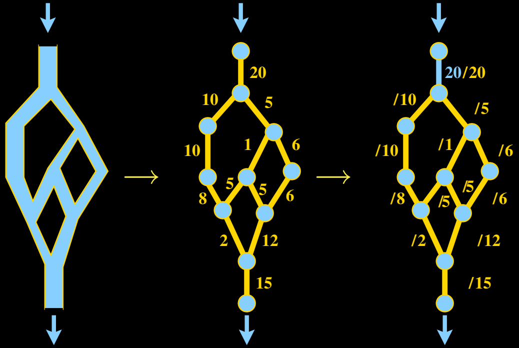
Max flow problem

Water pipes



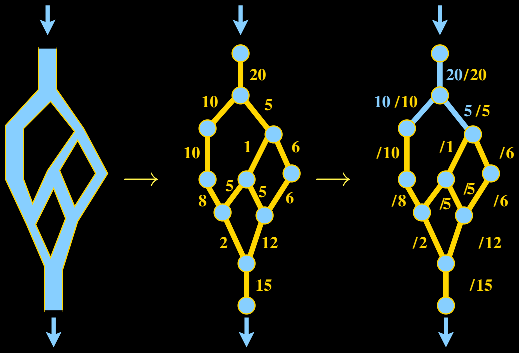
Max flow problem

Water pipes



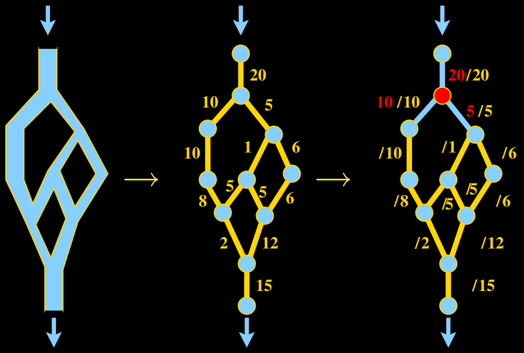
Max flow problem

Water pipes



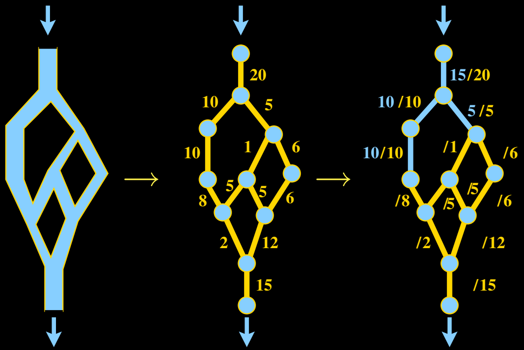
Max flow problem

Water pipes



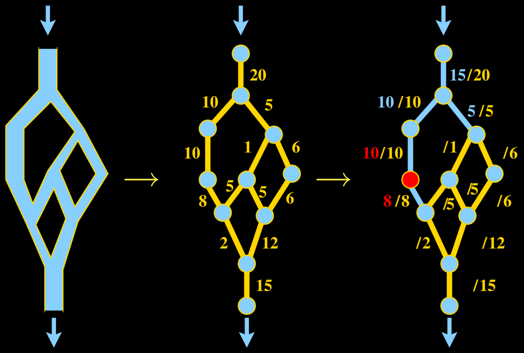
Max flow problem

Water pipes



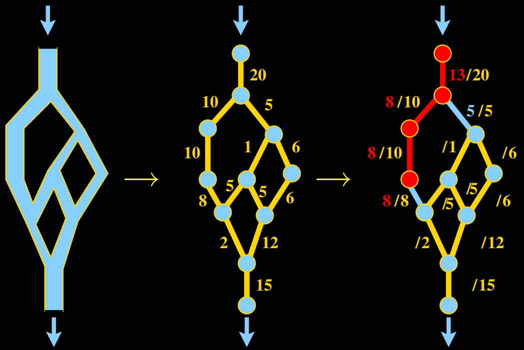
Max flow problem

Water pipes



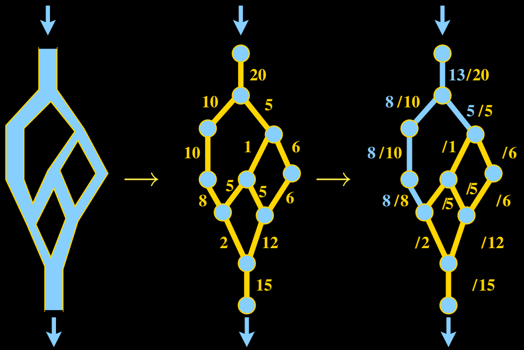
Max flow problem

Water pipes



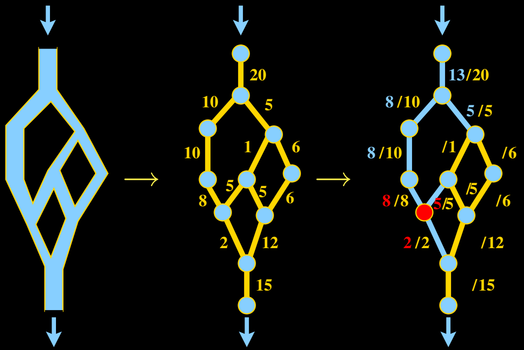
Max flow problem

Water pipes



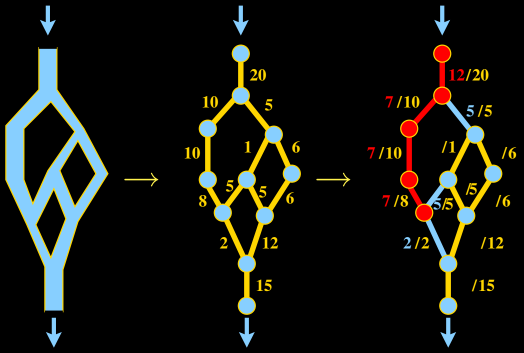
Max flow problem

Water pipes



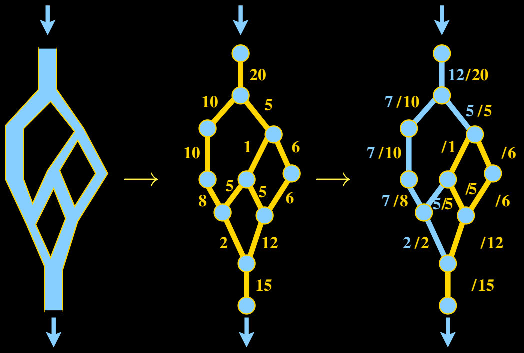
Max flow problem

Water pipes



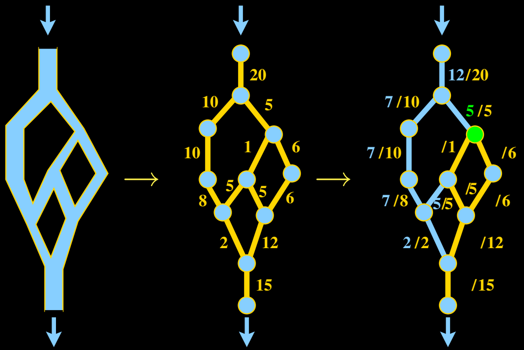
Max flow problem

Water pipes



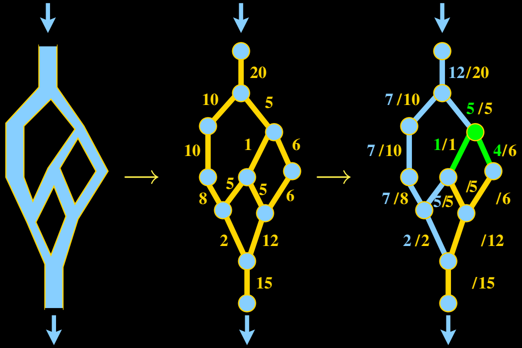
Max flow problem

Water pipes



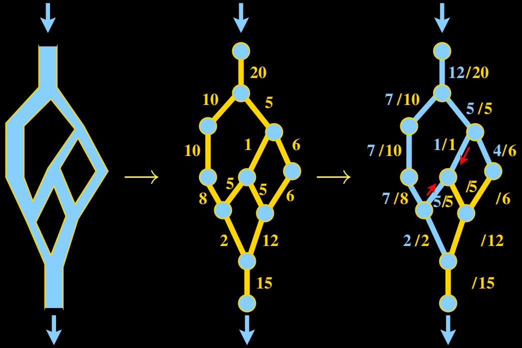
Max flow problem

Water pipes



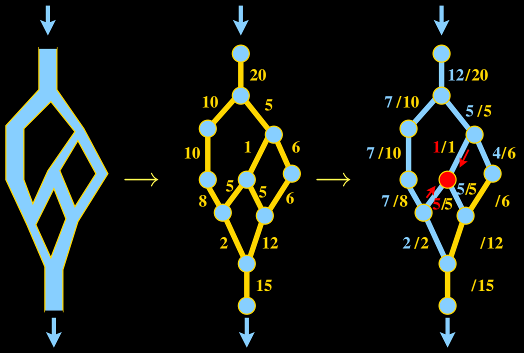
Max flow problem

Water pipes



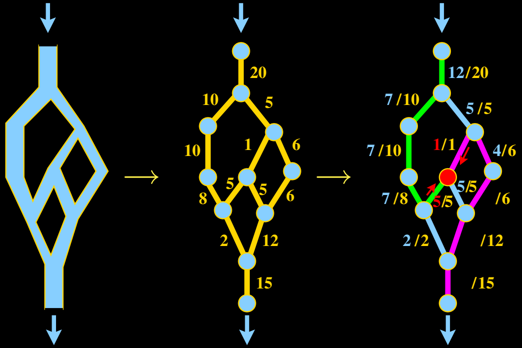
Max flow problem

Water pipes



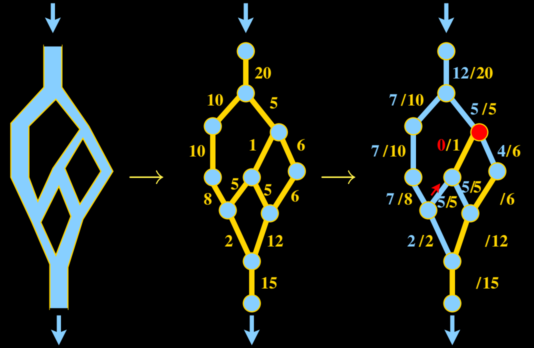
Max flow problem

Water pipes



Max flow problem

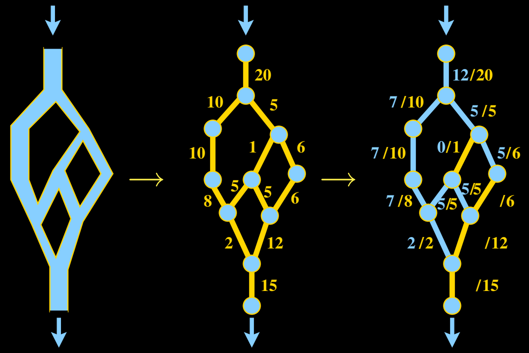
Water pipes



push - relabel

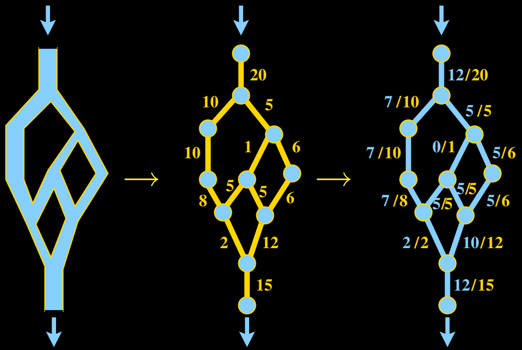
Max flow problem

Water pipes



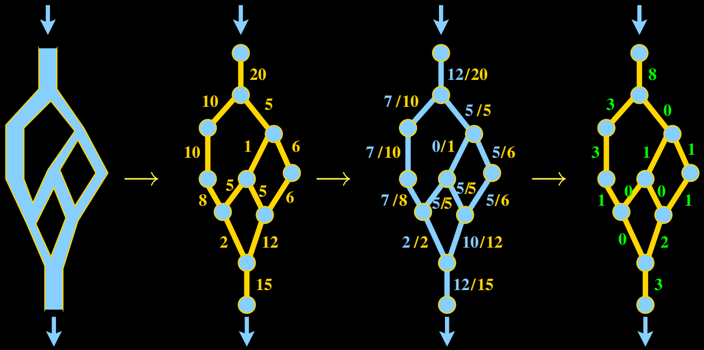
Max flow problem

Water pipes



Max flow problem

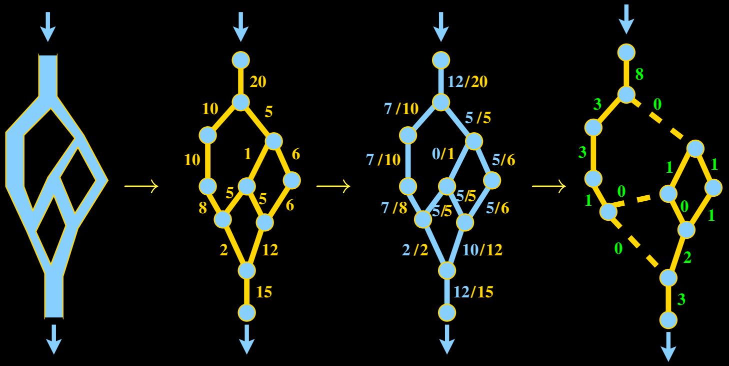
Water pipes



Theorem : maximum flow $\Longleftrightarrow$ no augmenting path in the residual graph

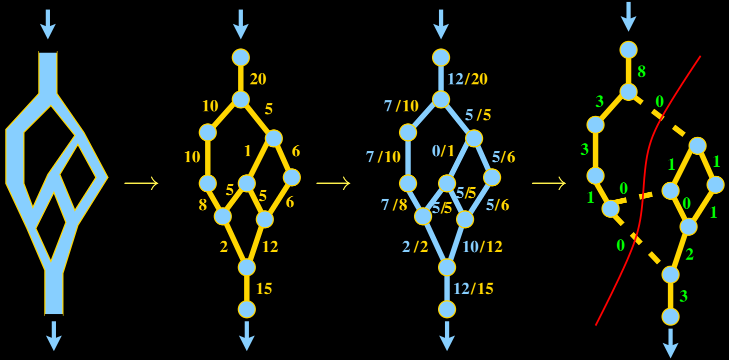
Max flow problem

Water pipes



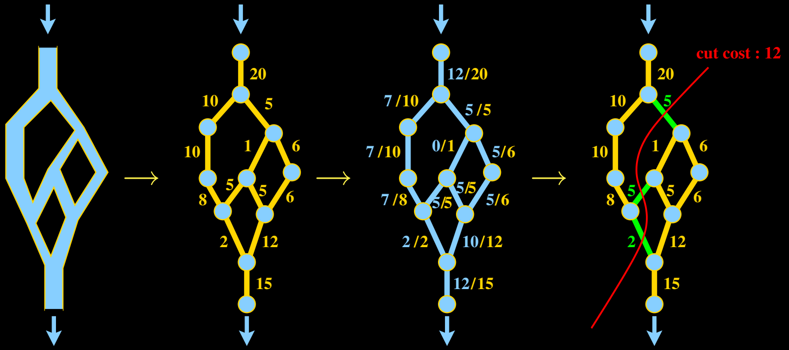
Max flow problem

Water pipes



Max flow problem

Water pipes



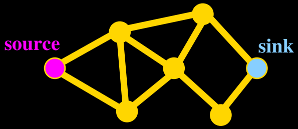
Theorem : maximum flow $\iff$ minimal cut

From graphs to images

Energy minimized on graphs

Graph :

- nodes N_i (including *source* and *sink*)
- weights w_{ij} between nodes N_i and N_j



A cut :

- a partition of the nodes
- a binary function L which associates to each node N_i a label $L(i)$: *source* or *sink*
- cost of a cut : $\sum_{ij} \delta_{L(i) \neq L(j)} w_{ij}$; min cut found e.g. by *Push relabel*

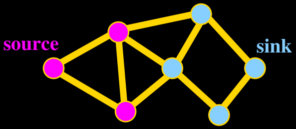
$\Longrightarrow$ **graph cuts give the optimal solution to any binary problem written this way !**

From graphs to images

Energy minimized on graphs

Graph :

- nodes N_i (including *source* and *sink*)
- weights w_{ij} between nodes N_i and N_j



A cut :

- a partition of the nodes
- a binary function L which associates to each node N_i a label $L(i)$: *source* or *sink*
- cost of a cut : $\sum_{ij} \delta_{L(i) \neq L(j)} w_{ij}$; min cut found e.g. by *Push relabel*

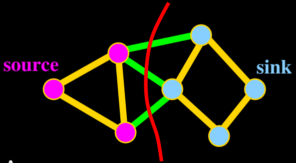
$\Longrightarrow$ **graph cuts give the optimal solution to any binary problem written this way !**

From graphs to images

Energy minimized on graphs

Graph :

- nodes N_i (including *source* and *sink*)
- weights w_{ij} between nodes N_i and N_j



A cut :

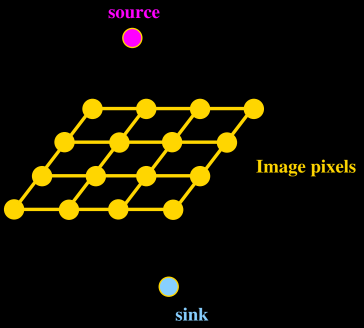
- a partition of the nodes
- a binary function L which associates to each node N_i a label $L(i)$: *source* or *sink*
- cost of a cut : $\sum_{ij} \delta_{L(i) \neq L(j)} w_{ij}$; min cut found e.g. by *Push relabel*

$\Longrightarrow$ **graph cuts give the optimal solution to any binary problem written this way !**

Images as graphs

Build a graph :

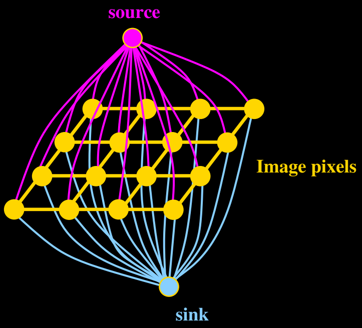
- one node for each pixel N_i
- edges between adjacent pixels
- two more nodes : the source A and the sink B
- edges from A and B to all pixels



Images as graphs

Build a graph :

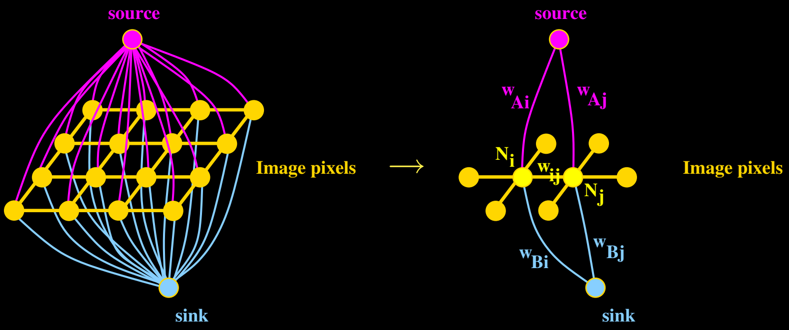
- one node for each pixel N_i
- edges between adjacent pixels
- two more nodes : the source A and the sink B
- edges from A and B to all pixels



Images as graphs

Build a graph :

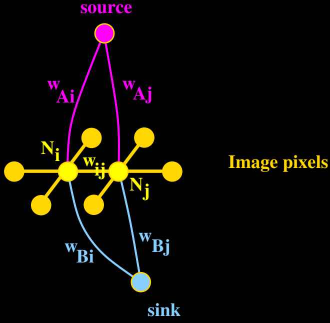
- one node for each pixel N_i
- edges between adjacent pixels
- two more nodes : the source A and the sink B
- edges from A and B to all pixels



Choosing costs to design energies

Cut cost : sum over edges cut

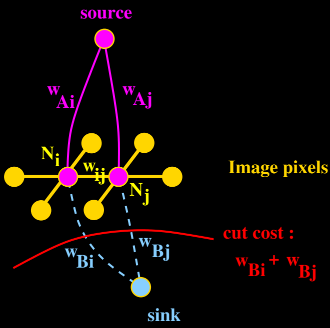
- vertical edges : for each pixel, either Ai or Bi is cut :
  if $L(i) = A$: w_{Bi}, if $L(i) = B$: w_{Ai}
  $\Longrightarrow \sum_i w_{\neg L(i),\, i}$
- horizontal edges : sum over edges between nodes of different labels
  $\Longrightarrow \sum_{ij} \delta_{L(i) \neq L(j)} w_{ij}$
- Total : $\sum_i w_{\neg L(i),\, i} + \sum_{ij} \delta_{L(i) \neq L(j)} w_{ij}$



Choosing costs to design energies

Cut cost : sum over edges cut

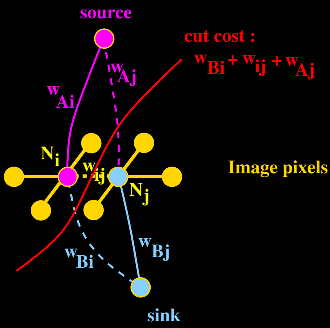
- vertical edges : for each pixel, either Ai or Bi is cut :
  if $L(i) = A$: w_{Bi}, if $L(i) = B$: w_{Ai}
  $\Longrightarrow \quad \sum_i w_{\neg L(i),\, i}$
- horizontal edges : sum over edges between nodes of different labels
  $\Longrightarrow \quad \sum_{ij} \delta_{L(i) \neq L(j)} w_{ij}$
- Total : $\displaystyle\sum_i w_{\neg L(i),\, i} + \sum_{ij} \delta_{L(i) \neq L(j)} w_{ij}$



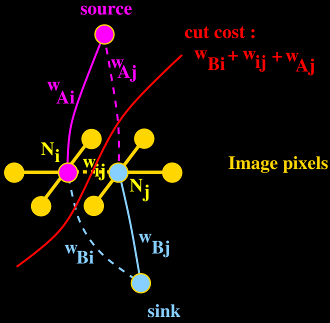
Choosing costs to design energies

Cut cost : sum over edges cut

- vertical edges : for each pixel, either Ai or Bi is cut :
  if $L(i) = A$: w_{Bi}, if $L(i) = B$: w_{Ai}
  $\Longrightarrow \sum_i w_{\neg L(i),\, i}$
- horizontal edges : sum over edges between nodes of different labels
  $\Longrightarrow \sum_{ij} \delta_{L(i) \neq L(j)} w_{ij}$
- Total : $\sum_i w_{\neg L(i),\, i} + \sum_{ij} \delta_{L(i) \neq L(j)} w_{ij}$



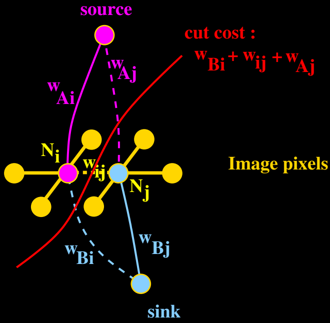
- Total : $\sum_i w_{\neg L(i),\, i} + \sum_{ij} \delta_{L(i) \neq L(j)} w_{ij}$



- Total : $\sum_i w_{\neg L(i),\, i} + \sum_{ij} \delta_{L(i) \neq L(j)} w_{ij}$

Choose the weights that suit your problem

- vertical edges : individual label preferences for each pixel
  ↪ rename $V_i(\ L(i)\ ) = w_{\neg L(i),\, i}$
  ↪ only constraint : $V_i(L(i))$ should be $\geqslant 0$
- horizontal edges : pairwise interaction between neighboring pixels, 0 if same labels
  ↪ rename $D_{ij}(\ L(i),\ L(j)\ ) = \delta_{L(i) \neq L(j)} w_{ij}$
  ↪ constraints : $D_{ij}(A,A) = D_{ij}(B,B) = 0$ and $D_{ij}(A,B) = D_{ij}(B,A) \geqslant 0$
- Total : $E(L) = \sum_i V_i(L(i)) + \sum_{ij} D_{ij}(L(i), L(j))$
  + constants : $+\sum_i K_i \quad + \sum_{ij} K_{ij}$

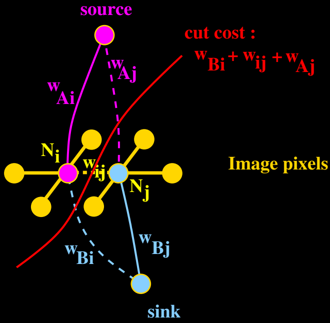


- real constraints :
  ↪ no constraint on potentials V_i
  ↪ no constraint on neighborhood choices
  ↪ for each interaction ij : $D_{ij}(A,A) = D_{ij}(B,B) \leqslant D_{ij}(A,B) = D_{ij}(B,A)$ i.e.locally, labels are preferred to be homogeneous

- Total : $\sum_i w_{\neg L(i),\, i} + \sum_{ij} \delta_{L(i) \neq L(j)} w_{ij}$

Choose the weights that suit your problem

- vertical edges : individual label preferences for each pixel
  - ↪ rename $V_i(\ L(i)\ ) = w_{\neg L(i),\, i}$
  - ↪ only constraint : $V_i(L(i))$ should be $\geqslant 0$
- horizontal edges : pairwise interaction between neighboring pixels, 0 if same labels
  - ↪ rename $D_{ij}(\ L(i),\ L(j)\ ) = \delta_{L(i) \neq L(j)} w_{ij}$
  - ↪ constraints : $D_{ij}(A, A) = D_{ij}(B, B) = 0$ and $D_{ij}(A, B) = D_{ij}(B, A) \geqslant 0$



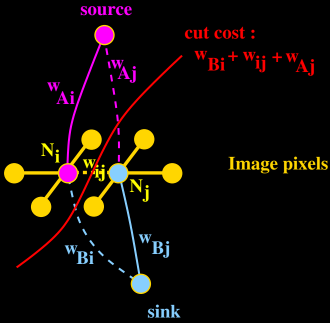
- Total : $E(L) = \sum_i V_i(L(i)) + \sum_{ij} D_{ij}(L(i), L(j))$

  + constants : $+ \sum_i K_i \quad + \sum_{ij} K_{ij}$

- real constraints :
  - ↪ no constraint on potentials V_i
  - ↪ no constraint on neighborhood choices
  - ↪ for each interaction ij : $D_{ij}(A, A) = D_{ij}(B, B) \leqslant D_{ij}(A, B) = D_{ij}(B, A)$ i.e.locally, labels are preferred to be homogeneous

- Total : $\sum_i w_{\neg L(i),\, i} + \sum_{ij} \delta_{L(i) \neq L(j)} w_{ij}$

Choose the weights that suit your problem



- vertical edges : individual label preferences for each pixel
  ↪ rename $V_i(\ L(i)\ ) = w_{\neg L(i),\, i}$
  ↪ only constraint : $V_i(L(i))$ should be $\geqslant 0$
- horizontal edges : pairwise interaction between neighboring pixels, 0 if same labels
  ↪ rename $D_{ij}(\ L(i),\ L(j)\ ) = \delta_{L(i) \neq L(j)} w_{ij}$
  ↪ constraints : $D_{ij}(A, A) = D_{ij}(B, B) = 0$ and $D_{ij}(A, B) = D_{ij}(B, A) \geqslant 0$
- Total : $E(L) = \sum_i V_i(L(i)) + \sum_{ij} D_{ij}(L(i), L(j))$

  + constants : $+\sum_i K_i \quad + \sum_{ij} K_{ij}$

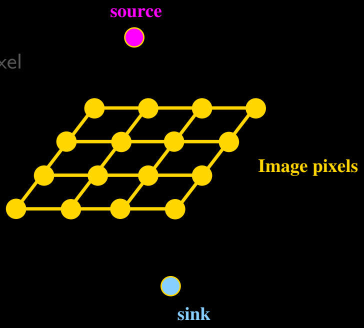
- real constraints :
  ↪ no constraint on potentials V_i
  ↪ no constraint on neighborhood choices
  ↪ for each interaction ij : $D_{ij}(A, A) = D_{ij}(B, B) \leqslant D_{ij}(A, B) = D_{ij}(B, A)$ i.e.locally, labels are preferred to be homogeneous

- Total : $\sum_i w_{\neg L(i),\, i} + \sum_{ij} \delta_{L(i) \neq L(j)} w_{ij}$

Choose the weights that suit your problem

- vertical edges : individual label preferences for each pixel
  $\hookrightarrow$ rename $V_i(\ L(i)\ ) = w_{\neg L(i),\, i}$
  $\hookrightarrow$ only constraint : $V_i(L(i))$ should be $\geqslant 0$
- horizontal edges : pairwise interaction between neighboring pixels, 0 if same labels
  $\hookrightarrow$ rename $D_{ij}(\ L(i),\ L(j)\ ) = \delta_{L(i) \neq L(j)} w_{ij}$
  $\hookrightarrow$ constraints : $D_{ij}(A, A) = D_{ij}(B, B) = 0$ and $D_{ij}(A, B) = D_{ij}(B, A) \geqslant 0$
- Total : $E(L) = \sum_i V_i(L(i)) + \sum_{ij} D_{ij}(L(i), L(j))$

  $+$ constants : $+\sum_i K_i$ $+\sum_{ij} K_{ij}$



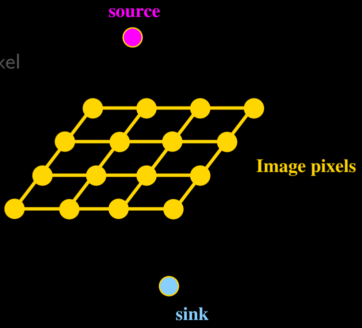
- real constraints :
  $\hookrightarrow$ no constraint on potentials V_i
  $\hookrightarrow$ no constraint on neighborhood choices
  $\hookrightarrow$ for each interaction ij :
  $D_{ij}(A, A) = D_{ij}(B, B) \leqslant D_{ij}(A, B) = D_{ij}(B, A)$
  i.e. locally, labels are preferred to be homogeneous

- Total : $\sum_i w_{\neg L(i),\, i} + \sum_{ij} \delta_{L(i) \neq L(j)} w_{ij}$

Choose the weights that suit your problem

- vertical edges : individual label preferences for each pixel
  ↪ rename $V_i(\ L(i)\ ) = w_{\neg L(i),\, i}$
  ↪ only constraint : $V_i(L(i))$ should be $\geqslant 0$
- horizontal edges : pairwise interaction between neighboring pixels, 0 if same labels
  ↪ rename $D_{ij}(\ L(i),\ L(j)\ ) = \delta_{L(i) \neq L(j)} w_{ij}$
  ↪ constraints : $D_{ij}(A, A) = D_{ij}(B, B) = 0$ and $D_{ij}(A, B) = D_{ij}(B, A) \geqslant 0$
- Total : $E(L) = \sum_i V_i(L(i)) + \sum_{ij} D_{ij}(L(i), L(j))$

  + constants : $+\sum_i K_i \quad +\sum_{ij} K_{ij}$



- real constraints :
  ↪ no constraint on potentials V_i
  ↪ no constraint on neighborhood choices
  ↪ for each interaction ij : $D_{ij}(A, A) = D_{ij}(B, B) \leqslant D_{ij}(A, B) = D_{ij}(B, A)$ i.e. locally, labels are preferred to be homogeneous

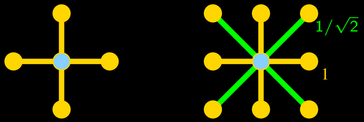
$\Longrightarrow$ graph cuts give the global optimum to any binary problem written this way !

Neighborhoods

Any neighborhood can be chosen
but
the choice will influence the shape of the cut

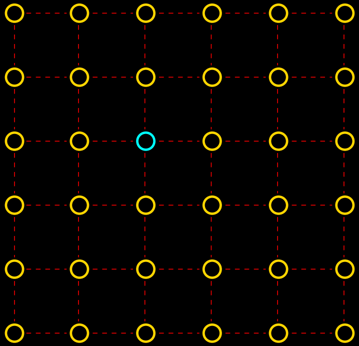
$\hookrightarrow$ 4-neighborhood $\Longrightarrow$ vertical and horizontal segments

$\hookrightarrow$ 8-neighborhood $\Longrightarrow$ $\approx$ ok in practice



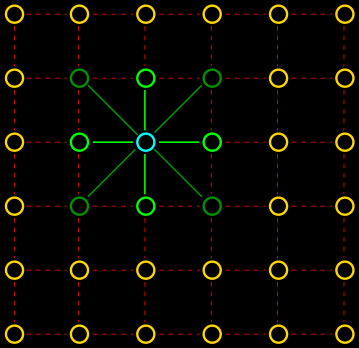
Example : binary segmentation

- For each pixel i of image A associate a node
- define 8-neighborhood
- two possible labels : black (B) and white (W)
- set potentials: $V_i(B) = A(i), \quad V_i(W) = 256 - A(i)$
- set spatial coherency: $D_{ij}(L_i, L_j) = K\, \delta_{L_i \neq L_j} \left(\frac{1}{\varepsilon + |A(i) - A(j)|} \right)$



Example : binary segmentation

- For each pixel i of image A associate a node
- define 8-neighborhood
- two possible labels : black (B) and white (W)
- set potentials: $V_i(B) = A(i), \quad V_i(W) = 256 - A(i)$
- set spatial coherency: $D_{ij}(L_i, L_j) = K\, \delta_{L_i \neq L_j} \left(\frac{1}{\varepsilon + |A(i) - A(j)|} \right)$



Example : binary segmentation

original image

$+$ noise

threshold

graph cut

Extension to multi-label problems

Previously : binary problems : $L(i) \in \{0, 1\}$
Now : multi-label problems : $L(i) \in \mathcal{X}$ (discrete set, indep. of i)

In some cases : possible to build a graph to get the global optimum
Most often : use (α, β)-swap or α-expansions

α-expansions :

$\hookrightarrow$ $D_{ij}(\cdot, \cdot)$: required to be distance on labels

$\hookrightarrow$ iteratively : choose one particular label α, and consider the binary problem : for each pixel i, is it better to keep current label $L(i)$, or to move to label α ?

$\hookrightarrow$ each step solved by graph-cuts

$\hookrightarrow$ repeat until no evolution

$\hookrightarrow$ convergence and good local optimum guaranteed

Energies that can be minimized

- Minimizing $E(L) = \sum_i V_i(\, L(i)\,) + \sum_{ij} D_{ij}(\, L(i),\, L(j)\,)$
  is NP-hard in the general case
- The sub-modularity condition $D_{ij}(A, A) = D_{ij}(B, B) < D_{ij}(A, B) = D_{ij}(B, A)$ makes it minimizable by graph-cuts
- If labels are ordered : $D_{ij}(\, L(i),\, L(j)\,) = g_{ij}(\, L(i) - L(j)\,)$ with g_{ij} convex $\Longrightarrow$ global optimum
- (α, β) swap : if D_{ij} is a semi-metric on labels
- α expansion : if D_{ij} is a metric : good local minimum guaranteed theoretically

Probabilistic / Markovian rewriting

$$
\begin{aligned}
p(L) &\propto \exp(-E(L)) \\
&\propto \exp\Big(-\sum_i V_i(\,L(i)\,)\Big)\ \exp\Big(-\sum_{ij} D_{ij}(\,L(i),\,L(j)\,)\Big) \\
&\propto \prod_i e^{-V_i(\,L(i)\,)} \prod_{i\sim j} e^{-D_{ij}(\,L(i),\,L(j)\,)} \\
&\propto \prod_i p_i(\,L(i)\,) \prod_{i\sim j} q_{ij}(\,L(i),\,L(j)\,)
\end{aligned}
$$

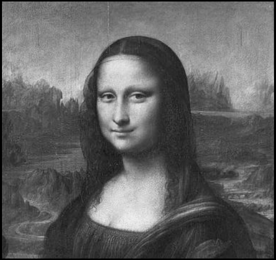
$$
p(\,L(i)\,|\,L(k)\,\forall k \neq i) \;\propto\; p_i(\,L(i)\,) \prod_{j\sim i} q_{ij}(\,L(i),\,L(j)\,) \;\propto\; p(\,L(i)\,|\,L(j)\,\forall j \in \mathcal{N}_i)
$$

$$
p(\,L(i)\,|\,L(j)\,) \;\propto\; p_i(\,L(i)\,)\ q_{ij}(\,L(i),\,L(j)\,)
$$

An example in the probabilistic setting : colorization

- Learning how to color greyscale images
- training set : one or several color images
- main idea : copy colors from patches with similar greyscale texture

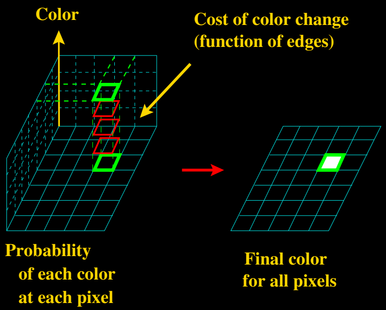
$+$



$= ?$

An other example

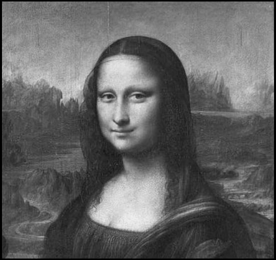
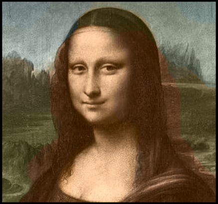
- pixel $\mapsto$ greyscale patch $\mapsto$ texture features (SURF)
- SVR : features $\mapsto$ proba(each color)
- SVR : features $\mapsto$ norm of the gradient of the color
- graph cut : proba(each color) $\times$ cost of color change $\mapsto$ color



Minimize energy: $\prod_i \psi(color\ c_i, patch\ p_i) \times \prod_{i \sim j} \Psi(c_i, c_j | p_i, p_j)$

An other example

Examples of results

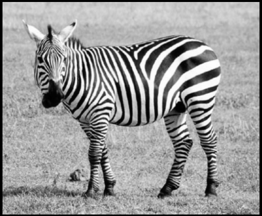

+

=


Examples of results

An other example

Examples of results

+



=

An other example

Examples of results

+

=

Variations

- directed vs. undirected graph
- higher-order interaction, with m variables : $V_{i,j,k\ldots}(\,L(i),\,L(j),\,L(k)\ldots)$
- dynamic cut : knowing a solution to a close problem $\Longrightarrow$ iterative
- active graph cut : knowing a solution on a part of the graph $\Longrightarrow$ multi-scale
- multiple sources or sinks within the image

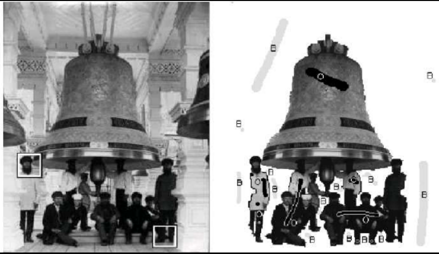
Complexity

- Theoretical complexity, worse case : about $(\#\text{pixels})^3 \times (\#\text{labels})^2$ (depends on the algorithm)
- In practice : worse case never reached, much faster
- GPU implementation possible $\Longrightarrow$ incredibly fast

Some applications

Any problem that can be written as a multi-labelling problem with simple local interaction terms (Markov fields)

- image denoising
- image segmentation, knowing color histograms of objects
- segmentation knowing seeds (points inside and outside the object)



Some applications

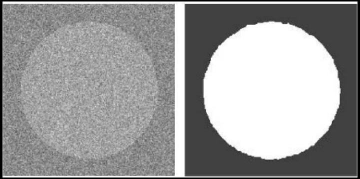
Any problem that can be written as a multi-labelling problem with simple local interaction terms (Markov fields)

- image denoising
- image segmentation, knowing color histograms of objects
- segmentation knowing seeds (points inside and outside the object)
- active contours : iterative segmentation within a narrow band
- multi-scale approach for segmentation
- iterative segmentation with parameter estimations (e.g. color histograms)

Some applications

Any problem that can be written as a multi-labelling problem with simple local interaction terms (Markov fields)

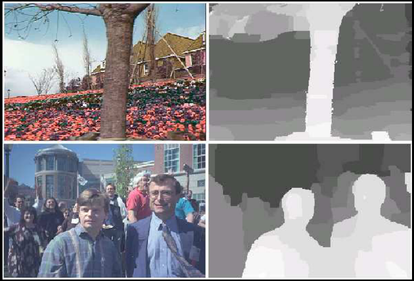
- image denoising
- image segmentation, knowing color histograms of objects
- segmentation knowing seeds (points inside and outside the object)
- active contours : iterative segmentation within a narrow band
- multi-scale approach for segmentation
- iterative segmentation with parameter estimations (e.g. color histograms)
- EM algorithms : iterative clustering / parameter-estimation



Some applications

Any problem that can be written as a multi-labelling problem with simple local interaction terms (Markov fields)

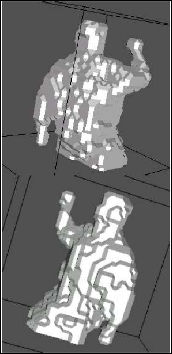
- ...
- iterative segmentation with parameter estimations (e.g. color histograms)
- EM algorithms : iterative clustering / parameter-estimation
- stereovison



Some applications

Any problem that can be written as a multi-labelling problem with simple local interaction terms (Markov fields)

- ...
- stereovison
- 3D-reconstruction



Some applications

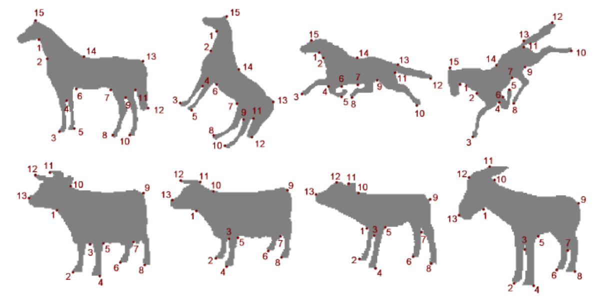
Any problem that can be written as a multi-labelling problem with simple local interaction terms (Markov fields)

- ...
- stereovison
- 3D-reconstruction
- video segmentation : based on motion
- texture synthesis

Some applications

Any problem that can be written as a multi-labelling problem with simple local interaction terms (Markov fields)

- ...
- texture synthesis
- shape matching (different kind of graph)
- segmentation with rigid shape prior (insanely huge graph)



Discussion

Pros:

↪ gives the global optimum of certain types of energies

↪ gives a very good local optimum of all Markov-like energies with discrete values

↪ practical way to bring spatial coherency

↪ it's fast

Cons :

↪ only those kinds of simple energies

↪ tends to make people do simplisitic modelings

Competitor :

↪ loopy belief propagation

References

Tutorials

- *Introduction aux GraphCuts en Vision par Ordinateur*, by Mickaël Péchaud (Odyssée Team)

Papers

- G. B. Dantzig and D. R. Fulkerson, *On the max-flow min-cut theorem of networks*, Annals of Mathematics Studies, 1956
- D.M. Grieg, B.T. Porteous and A.H. Seheult, *Exact maximum a posteriori estimation for binary images*, Journal of the Royal Statistical Society, 1989
- V. Kolmogorov and R. Zabih, *What energy functions can be minimized via graph cuts*, ICCV 2002
- Y. Boykov and V. Kolmogorov, *An experimental comparison of min-cut/max-flow algorithms for energy minimization in vision*, TPAMI 2004.