Energy Minimization

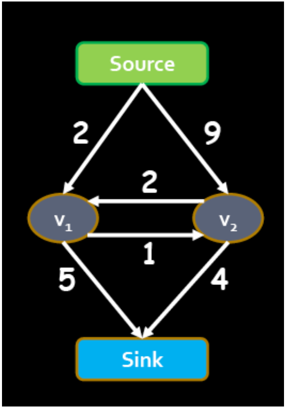
Raquel Urtasun

TTI Chicago

Feb 28, 2013

The ST-mincut problem

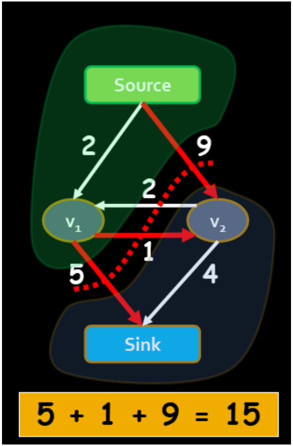
- Suppose we have a graph $G = \{V, E, C\}$, with vertices V, Edges E and costs C.



[Source: P. Kohli]

The ST-mincut problem

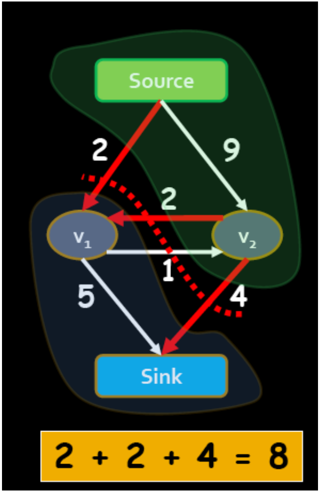
- An st-cut (S,T) divides the nodes between source and sink.
- The cost of a st-cut is the sum of cost of all edges going from S to T



[Source: P. Kohli]

The ST-mincut problem

- The st-mincut is the st-cut with the minimum cost

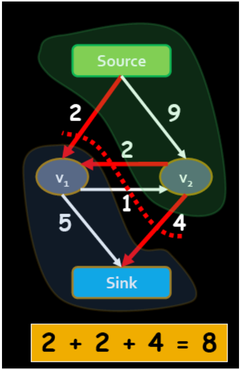


[Source: P. Kohli]

Back to our energy minimization

Construct a graph such that

1. Any st-cut corresponds to an assignment of x
2. The cost of the cut is equal to the energy of x : E(x)



[Source: P. Kohli]

St-mincut and Energy Minimization

$$E(x) = \sum_i \theta_i(x_i) + \sum_{i,j} \theta_{ij}(x_i, x_j)$$

For all ij $\theta_{ij}(0,1) + \theta_{ij}(1,0) \geq \theta_{ij}(0,0) + \theta_{ij}(1,1)$

Equivalent (transformable)

$$E(x) = \sum_i c_i x_i + \sum_{i,j} c_{ij} x_i(1 - x_j)$$

$c_{ij} \geq 0$

[Source: P. Kohli]

How are they equivalent?

$A = \theta_{ij}(0,0)$ $B = \theta_{ij}(0,1)$ $C = \theta_{ij}(1,0)$ $D = \theta_{ij}(1,1)$

$$\boxed{\begin{array}{cc|cc} & & \multicolumn{2}{c}{x_j} \\ & & 0 & 1 \\ \hline x_i & 0 & A & B \\ & 1 & C & D \end{array}} = A + \begin{array}{c|cc} & 0 & 1 \\ \hline 0 & 0 & 0 \\ 1 & C\text{-}A & C\text{-}A \end{array} + \begin{array}{c|cc} & 0 & 1 \\ \hline 0 & 0 & D\text{-}C \\ 1 & 0 & D\text{-}C \end{array} + \begin{array}{c|cc} & 0 & 1 \\ \hline 0 & 0 & B+C\text{-}A\text{-}D \\ 1 & 0 & 0 \end{array}$$

if x_1=1 add C-A

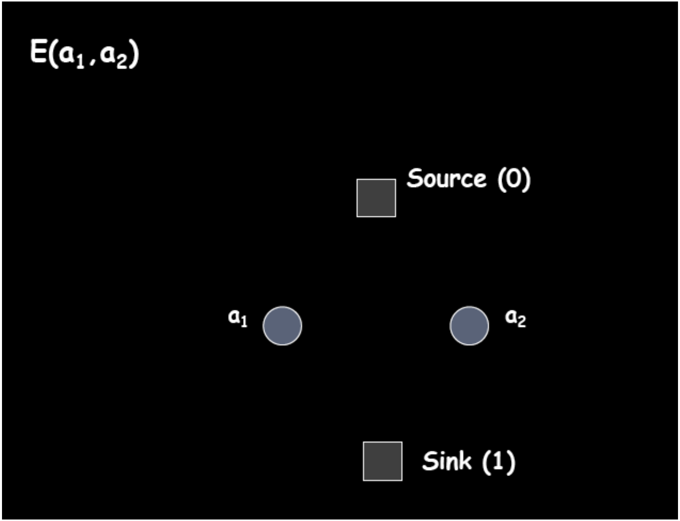
if x_2 = 1 add D-C

$$\boxed{\theta_{ij}(x_i,x_j)} = \theta_{ij}(0,0)$$
$$+ (\theta_{ij}(1,0)-\theta_{ij}(0,0))\, x_i + (\theta_{ij}(1,0)-\theta_{ij}(0,0))\, x_j$$
$$+ (\theta_{ij}(1,0) + \theta_{ij}(0,1) - \theta_{ij}(0,0) - \theta_{ij}(1,1))\,(1-x_i)\, x_j$$

$B+C-A-D \geq 0$ is true from the submodularity of θ_{ij}

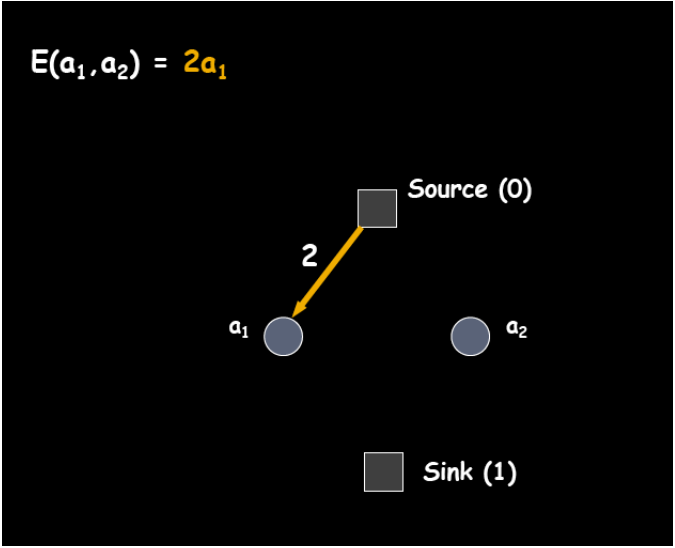
[Source: P. Kohli]

Graph Construction



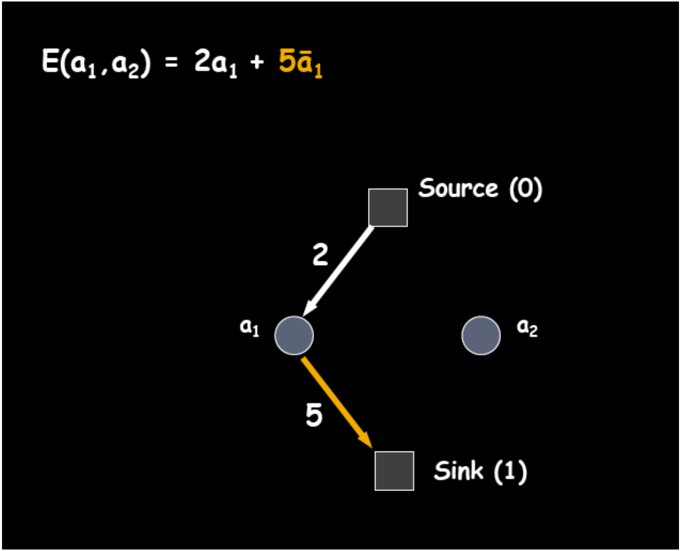
[Source: P. Kohli]

Graph Construction



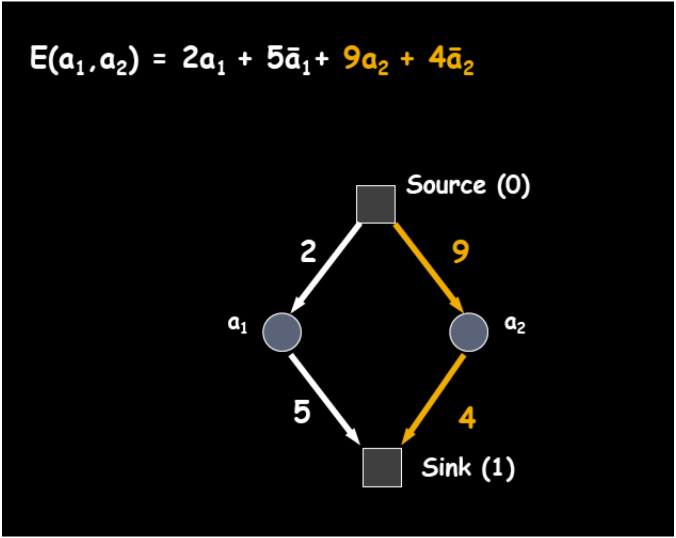
[Source: P. Kohli]

Graph Construction



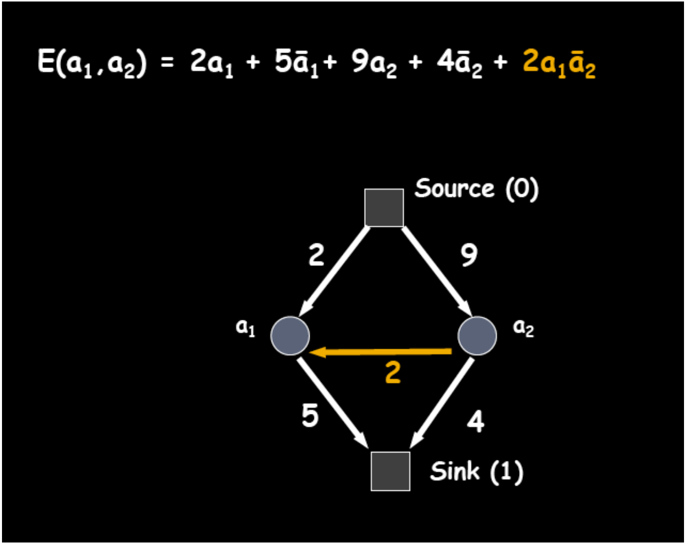
[Source: P. Kohli]

Graph Construction



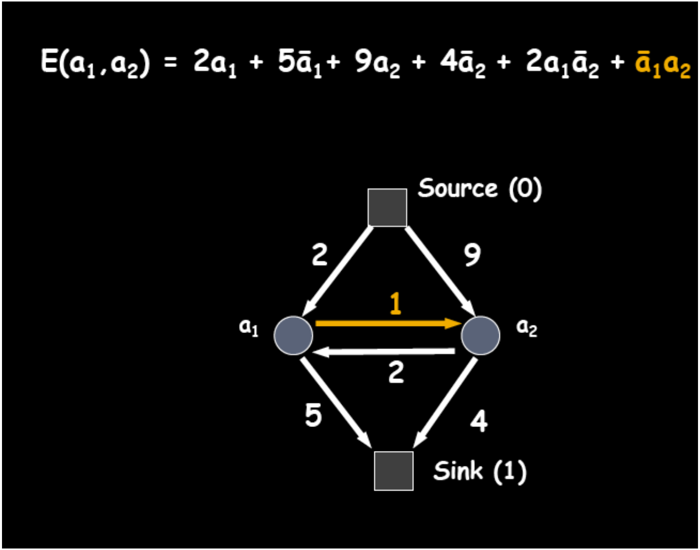
[Source: P. Kohli]

Graph Construction



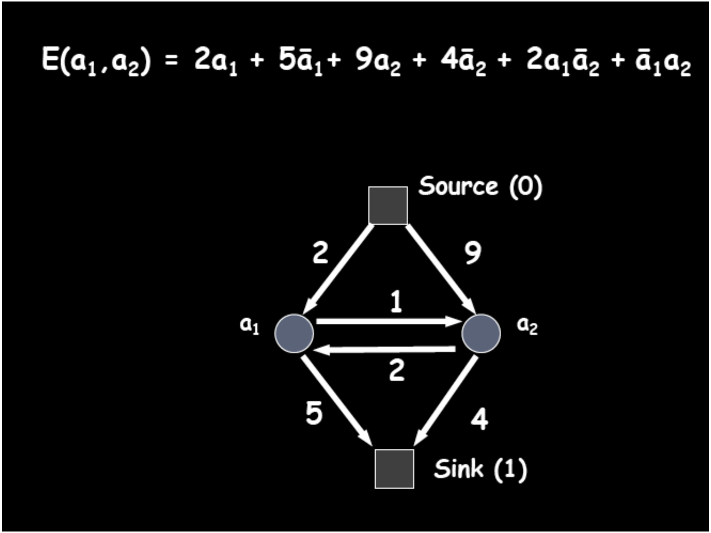
[Source: P. Kohli]

Graph Construction



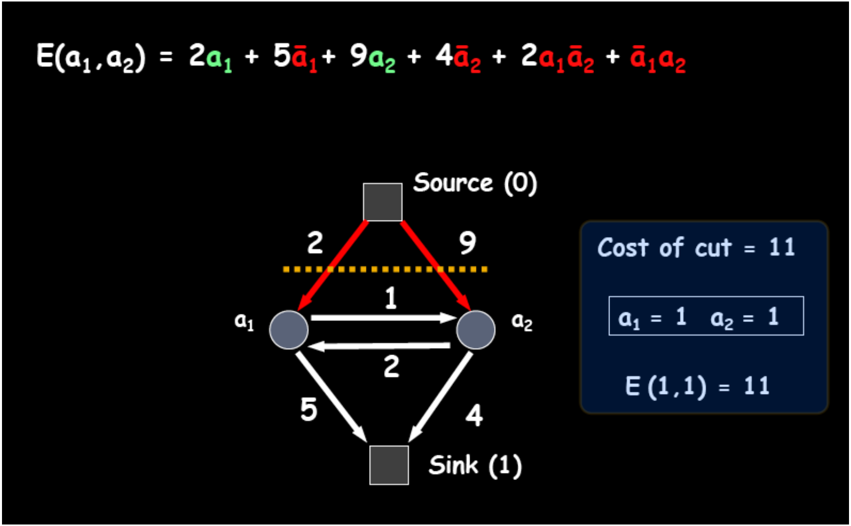
[Source: P. Kohli]

Graph Construction



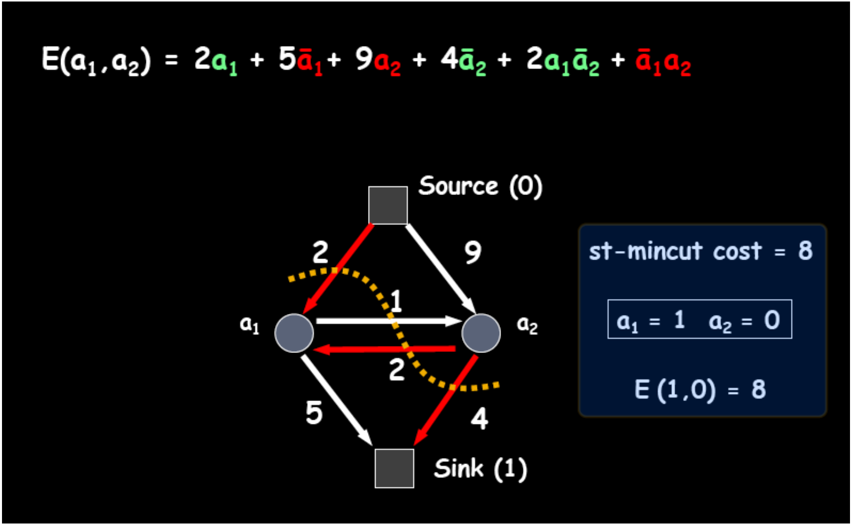
[Source: P. Kohli]

Graph Construction



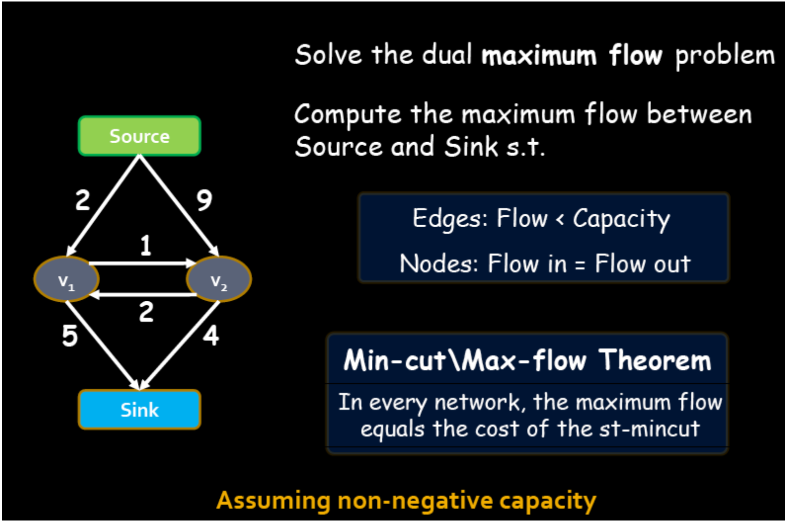
[Source: P. Kohli]

Graph Construction



[Source: P. Kohli]

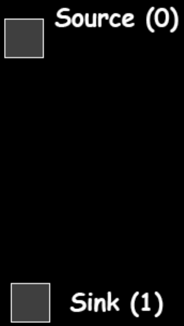
How to compute the St-mincut?



[Source: P. Kohli]

How does the code look like

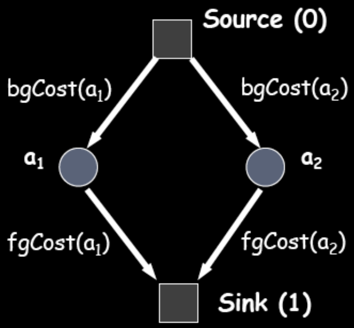
```
Graph *g;

For all pixels p

        /* Add a node to the graph */
        nodeID(p) = g->add_node();

        /* Set cost of terminal edges */
        set_weights(nodeID(p), fgCost(p), bgCost(p));

end

for all adjacent pixels p,q
        add_weights(nodeID(p), nodeID(q),  cost(p,q));
end

g->compute_maxflow();

label_p = g->is_connected_to_source(nodeID(p));
// is the label of pixel p (0 or 1)
```



[Source: P. Kohli]

How does the code look like

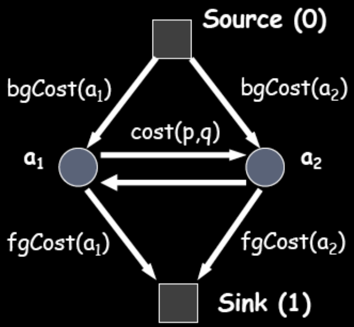
```
Graph *g;
For all pixels p

    /* Add a node to the graph */
    nodeID(p) = g->add_node();

    /* Set cost of terminal edges */
    set_weights(nodeID(p), fgCost(p), bgCost(p));

end

for all adjacent pixels p,q
    add_weights(nodeID(p), nodeID(q), cost(p,q));
end

g->compute_maxflow();

label_p = g->is_connected_to_source(nodeID(p));
// is the label of pixel p (0 or 1)
```



[Source: P. Kohli]

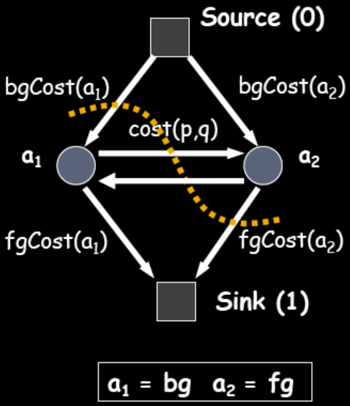
How does the code look like

```
Graph *g;

For all pixels p

        /* Add a node to the graph */
        nodeID(p) = g->add_node();

        /* Set cost of terminal edges */
        set_weights(nodeID(p), fgCost(p), bgCost(p));

end

for all adjacent pixels p,q
        add_weights(nodeID(p), nodeID(q),  cost(p,q));
end

g->compute_maxflow();

label_p = g->is_connected_to_source(nodeID(p));
// is the label of pixel p (0 or 1)
```



[Source: P. Kohli]

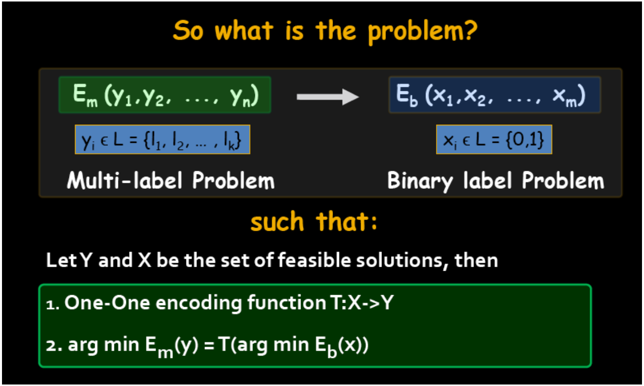
```
Graph *g;

For all pixels p

        /* Add a node to the graph */
        nodeID(p) = g->add_node();

        /* Set cost of terminal edges */
        set_weights(nodeID(p), fgCost(p), bgCost(p));

end

for all adjacent pixels p,q
        add_weights(nodeID(p), nodeID(q),  cost(p,q));
end

g->compute_maxflow();

label_p = g->is_connected_to_source(nodeID(p));
// is the label of pixel p (0 or 1)
```



[Source: P. Kohli]

Graph cuts for multi-label problems

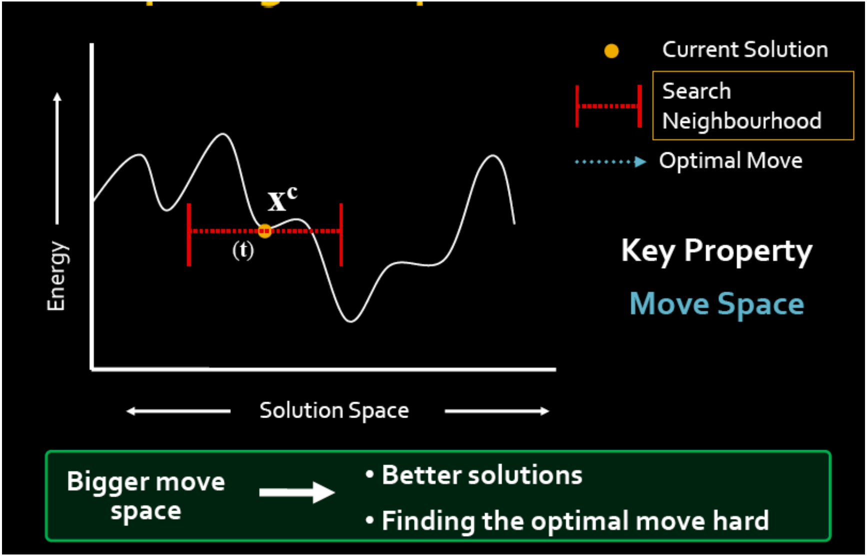
- Exact Transformation to QPBF [Roy and Cox 98] [Ishikawa 03] [Schlesinger et al. 06] [Ramalingam et al. 08]



- Very high computational cost

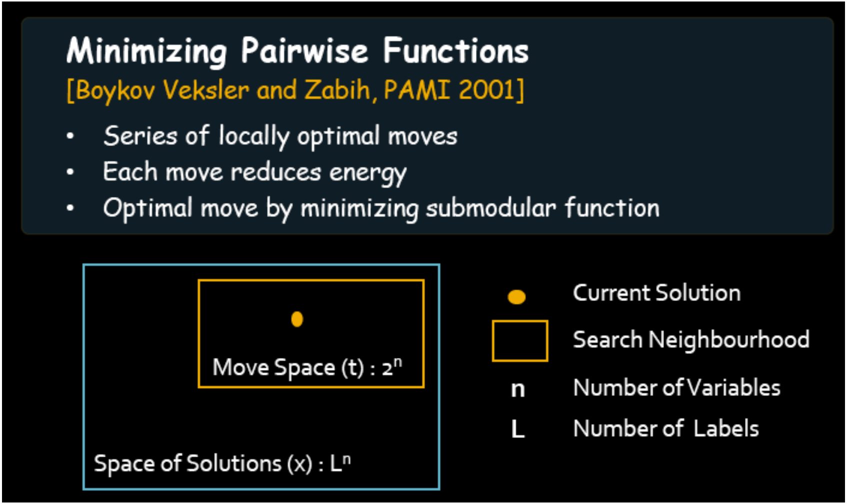
[Source: P. Kohli]

Computing the Optimal Move



[Source: P. Kohli]

Move Making Algorithms



[Source: P. Kohli]

Energy Minimization

- Consider pairwise MRFs

$$E(f) = \sum_{\{p,q\} \in \mathcal{N}} V_{p,q}(f_p, f_q) + \sum_p D_p(f_p)$$

with $\mathcal{N}$ defining the interactions between nodes, e.g., pixels

- D_p non-negative, but arbitrary.

Energy Minimization

- Consider pairwise MRFs

$$E(f) = \sum_{\{p,q\} \in \mathcal{N}} V_{p,q}(f_p, f_q) + \sum_p D_p(f_p)$$

with $\mathcal{N}$ defining the interactions between nodes, e.g., pixels

- D_p non-negative, but arbitrary.
- This is the graph-cuts notation.
- Important to notice it's the same thing.

Energy Minimization

- Consider pairwise MRFs

$$E(f) = \sum_{\{p,q\} \in \mathcal{N}} V_{p,q}(f_p, f_q) + \sum_p D_p(f_p)$$

  with $\mathcal{N}$ defining the interactions between nodes, e.g., pixels
- D_p non-negative, but arbitrary.
- This is the graph-cuts notation.
- Important to notice it's the same thing.

Metric vs Semimetric

Two general classes of pairwise interactions

- **Metric** if it satisfies for any set of labels α, β, γ

$$\begin{aligned} V(\alpha,\beta) = 0 &\leftrightarrow \alpha = \beta \\ V(\alpha,\beta) &= V(\beta,\alpha) \geq 0 \\ V(\alpha,\beta) &\leq V(\alpha,\gamma) + V(\gamma,\beta) \end{aligned}$$

- **Semi-metric** if it satisfies for any set of labels α, β, γ

$$\begin{aligned} V(\alpha,\beta) = 0 &\leftrightarrow \alpha = \beta \\ V(\alpha,\beta) &= V(\beta,\alpha) \geq 0 \end{aligned}$$

Metric vs Semimetric

Two general classes of pairwise interactions

- **Metric** if it satisfies for any set of labels α, β, γ

$$\begin{aligned} V(\alpha,\beta) = 0 &\leftrightarrow \alpha = \beta \\ V(\alpha,\beta) &= V(\beta,\alpha) \geq 0 \\ V(\alpha,\beta) &\leq V(\alpha,\gamma) + V(\gamma,\beta) \end{aligned}$$

- **Semi-metric** if it satisfies for any set of labels α, β, γ

$$\begin{aligned} V(\alpha,\beta) = 0 &\leftrightarrow \alpha = \beta \\ V(\alpha,\beta) &= V(\beta,\alpha) \geq 0 \end{aligned}$$

Examples for 1D label set

- Truncated quadratic is a semi-metric

$$V(\alpha, \beta) = \min(K, |\alpha - \beta|^2)$$

with K a constant.

- Truncated absolute distance is a metric

$$V(\alpha, \beta) = \min(K, |\alpha - \beta|)$$

with K a constant.

Examples for 1D label set

- Truncated quadratic is a semi-metric

$$V(\alpha, \beta) = \min(K, |\alpha - \beta|^2)$$

  with K a constant.

- Truncated absolute distance is a metric

$$V(\alpha, \beta) = \min(K, |\alpha - \beta|)$$

  with K a constant.

- For multi-dimensional, replace $|\cdot|$ by any norm.

Examples for 1D label set

- Truncated quadratic is a semi-metric

$$V(\alpha, \beta) = \min(K, |\alpha - \beta|^2)$$

  with K a constant.

- Truncated absolute distance is a metric

$$V(\alpha, \beta) = \min(K, |\alpha - \beta|)$$

  with K a constant.

- For multi-dimensional, replace $|\cdot|$ by any norm.

- Potts model is a metric

$$V(\alpha, \beta) = K \cdot T(\alpha \neq \beta)$$

  with $T(\cdot) = 1$ if the argument is true and 0 otherwise.

Examples for 1D label set

- Truncated quadratic is a semi-metric

$$V(\alpha, \beta) = \min(K, |\alpha - \beta|^2)$$

  with K a constant.

- Truncated absolute distance is a metric

$$V(\alpha, \beta) = \min(K, |\alpha - \beta|)$$

  with K a constant.

- For multi-dimensional, replace $|\cdot|$ by any norm.
- Potts model is a metric

$$V(\alpha, \beta) = K \cdot T(\alpha \neq \beta)$$

  with $T(\cdot) = 1$ if the argument is true and 0 otherwise.

Binary Moves

- $\alpha - \beta$ **moves** works for semi-metrics
- α **expansion** works for V being a metric

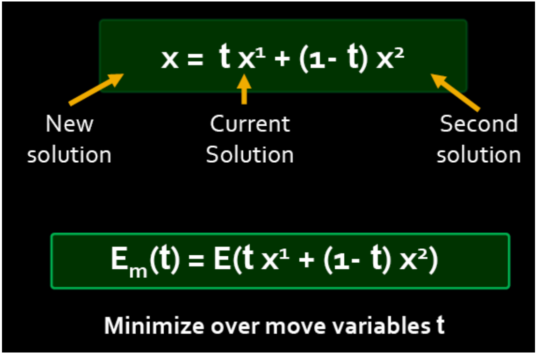
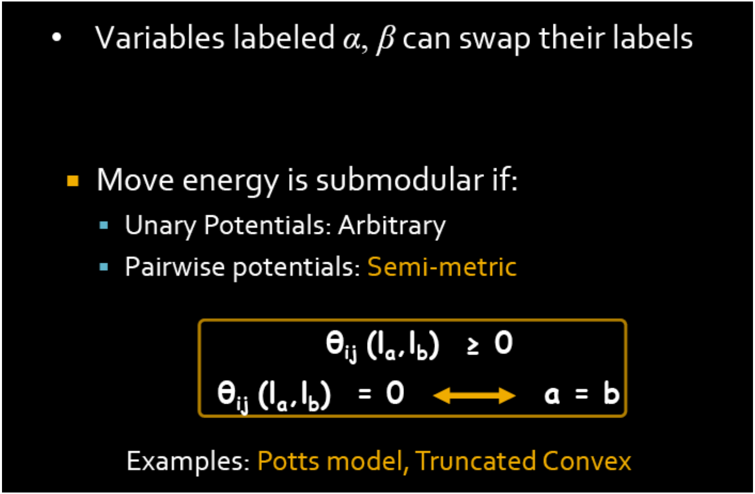


Figure: Figure from P. Kohli tutorial on graph-cuts

- For certain x^1 and x^2, the move energy is sub-modular QPBF

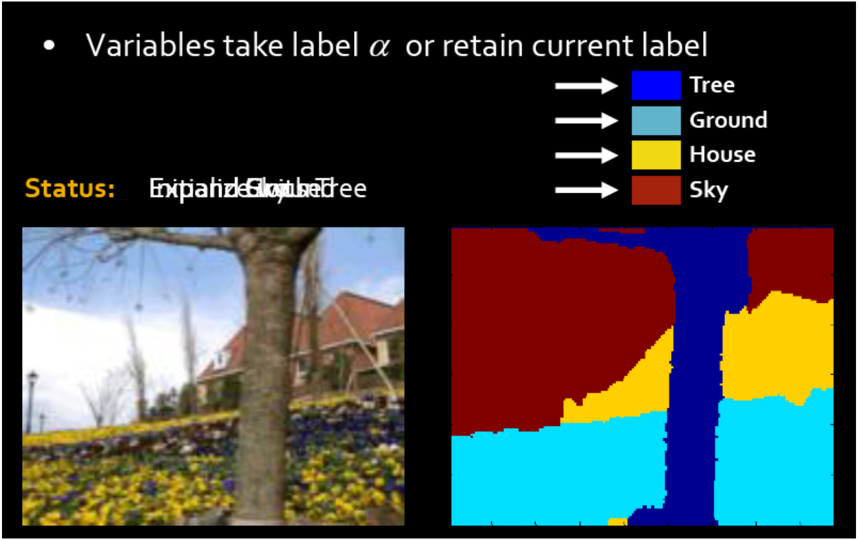
[Source: P. Kohli]

Swap Move



[Source: P. Kohli]

Expansion Move



[Source: P. Kohli]

Expansion Move

- Variables take label α or retain current label

- Move energy is submodular if:
  - Unary Potentials: Arbitrary
  - Pairwise potentials: Metric

Semi metric
+
Triangle Inequality

$$\boxed{\theta_{ij}(l_a, l_b) + \theta_{ij}(l_b, l_c) \geq \theta_{ij}(l_a, l_c)}$$

Examples: Potts model, Truncated linear

Cannot solve truncated quadratic

[Source: P. Kohli]

More formally

- Any labeling can be uniquely represented by a partition of image pixels $\mathbf{P} = \{\mathcal{P}_l | l \in \mathcal{L}\}$, where $\mathcal{P}_l = \{p \in \mathcal{P} | f_p = l\}$ is a subset of pixels assigned label l.
- There is a one to one correspondence between labelings f and partitions $\mathcal{P}$.

More formally

- Any labeling can be uniquely represented by a partition of image pixels $\mathbf{P} = \{\mathcal{P}_l | l \in \mathcal{L}\}$, where $\mathcal{P}_l = \{p \in \mathcal{P} | f_p = l\}$ is a subset of pixels assigned label l.
- There is a one to one correspondence between labelings f and partitions $\mathcal{P}$.
- Given a pair of labels α, β, a move from a partition $\mathcal{P}$ (labeling f) to a new partition $\mathcal{P}$' (labeling f') is called an $\alpha - \beta$ **swap** if $\mathcal{P}_l = \mathcal{P}'$ for any label $l \neq \alpha, \beta$.

More formally

- Any labeling can be uniquely represented by a partition of image pixels $\mathbf{P} = \{\mathcal{P}_l | l \in \mathcal{L}\}$, where $\mathcal{P}_l = \{p \in \mathcal{P} | f_p = l\}$ is a subset of pixels assigned label l.
- There is a one to one correspondence between labelings f and partitions $\mathcal{P}$.
- Given a pair of labels α, β, a move from a partition $\mathcal{P}$ (labeling f) to a new partition $\mathcal{P}$' (labeling f') is called an $\alpha - \beta$ **swap** if $\mathcal{P}_l = \mathcal{P}'$ for any label $l \neq \alpha, \beta$.
- The only difference between $\mathcal{P}$ and $\mathcal{P}'$ is that some pixels that were labeled in $\mathcal{P}$ are now labeled in $\mathcal{P}'$, and vice-versa.

More formally

- Any labeling can be uniquely represented by a partition of image pixels $\mathbf{P} = \{\mathcal{P}_l | l \in \mathcal{L}\}$, where $\mathcal{P}_l = \{p \in \mathcal{P} | f_p = l\}$ is a subset of pixels assigned label l.
- There is a one to one correspondence between labelings f and partitions $\mathcal{P}$.
- Given a pair of labels α, β, a move from a partition $\mathcal{P}$ (labeling f) to a new partition $\mathcal{P}$' (labeling f') is called an $\alpha - \beta$ **swap** if $\mathcal{P}_l = \mathcal{P}'$ for any label $l \neq \alpha, \beta$.
- The only difference between $\mathcal{P}$ and $\mathcal{P}'$ is that some pixels that were labeled in $\mathcal{P}$ are now labeled in $\mathcal{P}'$, and vice-versa.
- Given a label l, a move from a partition $\mathcal{P}$ (labeling f) to a new partition $\mathcal{P}'$ (labeling f') is called an α-**expansion** if $\mathcal{P}_\alpha \subset \mathcal{P}'_\alpha$ and $\mathcal{P}'_l \subset \mathcal{P}_l$.

More formally

- Any labeling can be uniquely represented by a partition of image pixels $\mathbf{P} = \{\mathcal{P}_l | l \in \mathcal{L}\}$, where $\mathcal{P}_l = \{p \in \mathcal{P} | f_p = l\}$ is a subset of pixels assigned label l.
- There is a one to one correspondence between labelings f and partitions $\mathcal{P}$.
- Given a pair of labels α, β, a move from a partition $\mathcal{P}$ (labeling f) to a new partition $\mathcal{P}$' (labeling f') is called an $\alpha - \beta$ **swap** if $\mathcal{P}_l = \mathcal{P}'$ for any label $l \neq \alpha, \beta$.
- The only difference between $\mathcal{P}$ and $\mathcal{P}'$ is that some pixels that were labeled in $\mathcal{P}$ are now labeled in $\mathcal{P}'$, and vice-versa.
- Given a label l, a move from a partition $\mathcal{P}$ (labeling f) to a new partition $\mathcal{P}'$ (labeling f') is called an α-**expansion** if $\mathcal{P}_\alpha \subset \mathcal{P}'_\alpha$ and $\mathcal{P}'_l \subset \mathcal{P}_l$.
- An α-**expansion** move allows any set of image pixels to change their labels to α.

More formally

- Any labeling can be uniquely represented by a partition of image pixels $\mathbf{P} = \{\mathcal{P}_l | l \in \mathcal{L}\}$, where $\mathcal{P}_l = \{p \in \mathcal{P} | f_p = l\}$ is a subset of pixels assigned label l.
- There is a one to one correspondence between labelings f and partitions $\mathcal{P}$.
- Given a pair of labels α, β, a move from a partition $\mathcal{P}$ (labeling f) to a new partition $\mathcal{P}$' (labeling f') is called an $\alpha - \beta$ **swap** if $\mathcal{P}_l = \mathcal{P}'$ for any label $l \neq \alpha, \beta$.
- The only difference between $\mathcal{P}$ and $\mathcal{P}'$ is that some pixels that were labeled in $\mathcal{P}$ are now labeled in $\mathcal{P}'$, and vice-versa.
- Given a label l, a move from a partition $\mathcal{P}$ (labeling f) to a new partition $\mathcal{P}'$ (labeling f') is called an α-**expansion** if $\mathcal{P}_\alpha \subset \mathcal{P}'_\alpha$ and $\mathcal{P}'_l \subset \mathcal{P}_l$.
- An α-**expansion** move allows any set of image pixels to change their labels to α.

Example

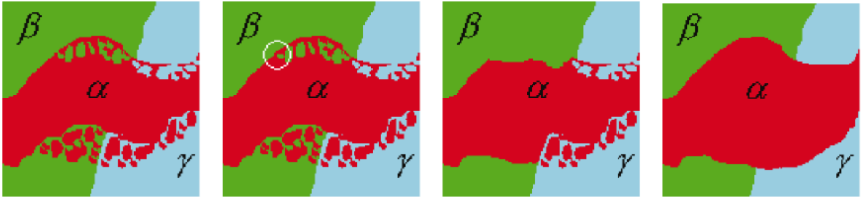


Figure: (a) Current partition (b) local move (c) $\alpha - \beta$-swap (d) α-expansion.

Algorithms

1. Start with an arbitrary labeling f
2. Set success := 0
3. For each pair of labels $\{\alpha, \beta\} \subset \mathcal{L}$
   3.1. Find $\hat{f} = \arg\min E(f')$ among f' within one α-β swap of f
   3.2. If $E(\hat{f}) < E(f)$, set f := $\hat{f}$ and success := 1
4. If success = 1 goto 2
5. Return f

---

1. Start with an arbitrary labeling f
2. Set success := 0
3. For each label $\alpha \in \mathcal{L}$
   3.1. Find $\hat{f} = \arg\min E(f')$ among f' within one α-expansion of f
   3.2. If $E(\hat{f}) < E(f)$, set f := $\hat{f}$ and success := 1
4. If success = 1 goto 2
5. Return f

Finding optimal Swap move

- Given an input labeling f (partition $\mathcal{P}$) and a pair of labels α, β we want to find a labeling $\hat{f}$ that minimizes E over all labelings within one $\alpha - \beta$-swap of f.
- This is going to be done by computing a labeling corresponding to a minimum cut on a graph $\mathcal{G}_{\alpha\beta} = (\mathcal{V}_{\alpha\beta}, \mathcal{E}_{\alpha\beta})$.

Finding optimal Swap move

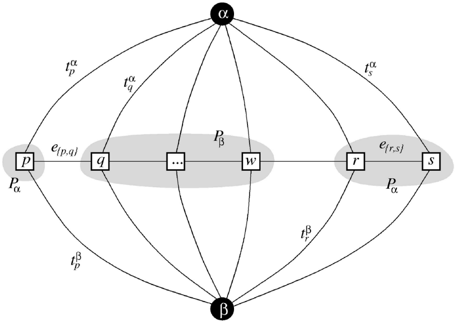
- Given an input labeling f (partition $\mathcal{P}$) and a pair of labels α, β we want to find a labeling $\hat{f}$ that minimizes E over all labelings within one $\alpha - \beta$-swap of f.
- This is going to be done by computing a labeling corresponding to a minimum cut on a graph $\mathcal{G}_{\alpha\beta} = (\mathcal{V}_{\alpha\beta}, \mathcal{E}_{\alpha\beta})$.
- The **structure** of this graph is **dynamically determined** by the current partition $\mathcal{P}$ and by the labels α, β.

Finding optimal Swap move

- Given an input labeling f (partition $\mathcal{P}$) and a pair of labels α, β we want to find a labeling $\hat{f}$ that minimizes E over all labelings within one $\alpha - \beta$-swap of f.
- This is going to be done by computing a labeling corresponding to a minimum cut on a graph $\mathcal{G}_{\alpha\beta} = (\mathcal{V}_{\alpha\beta}, \mathcal{E}_{\alpha\beta})$.
- The **structure** of this graph is **dynamically determined** by the current partition $\mathcal{P}$ and by the labels α, β.

Graph Construction

- The set of vertices includes the two terminals α and β, as well as image pixels p in the sets $\mathcal{P}_\alpha$ and $\mathcal{P}_\beta$ (i.e., $f_p \in \{\alpha, \beta\}$).
- Each pixel $p \in \mathcal{P}_{\alpha\beta}$ is connected to the terminals α and β, called t-links.
- Each set of pixels $p, q \in \mathcal{P}_{\alpha\beta}$ which are neighbors is connected by an edge $e_{p,q}$



edge	weight	for
t_p^α	$D_p(\alpha) + \sum_{q \in \mathcal{N}_p, q \notin \mathcal{P}_{\alpha\beta}} V(\alpha, f_q)$	$p \in \mathcal{P}_{\alpha\beta}$
t_p^β	$D_p(\beta) + \sum_{q \in \mathcal{N}_p, q \notin \mathcal{P}_{\alpha\beta}} V(\beta, f_q)$	$p \in \mathcal{P}_{\alpha\beta}$
$e_{\{p,q\}}$	$V(\alpha, \beta)$	$\{p,q\} \in \mathcal{N}$, $p,q \in \mathcal{P}_{\alpha\beta}$

Computing the Cut

- Any cut must have a single t-link not cut.
- This defines a labeling

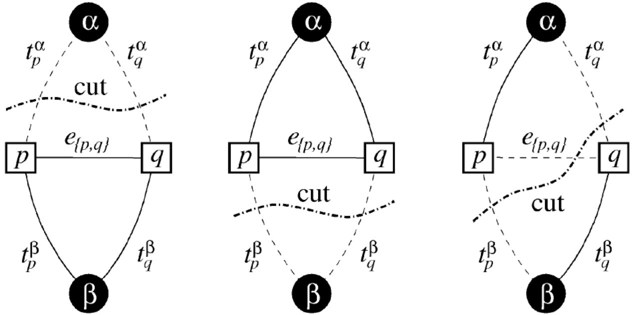
$$f_p^{\mathcal{C}} = \begin{cases} \alpha & \text{if } \ t_p^{\alpha} \in \mathcal{C} \ \text{ for } \ p \in \mathcal{P}_{\alpha\beta} \\ \beta & \text{if } \ t_p^{\beta} \in \mathcal{C} \ \text{ for } \ p \in \mathcal{P}_{\alpha\beta} \\ f_p & \text{for } \ p \in \mathcal{P}, \ p \notin \mathcal{P}_{\alpha\beta}. \end{cases}$$

- There is a one-to-one correspondences between a cut and a labeling.
- The energy of the cut is the energy of the labeling.
- See Boykov et al, "*fast approximate energy minimization via graph cuts*" PAMI 2001.

Properties

- For any cut, then

$$\begin{array}{llllll}
(a) & \text{If} & t_p^\alpha, t_q^\alpha \in \mathcal{C} & \text{then} & e_{\{p,q\}} \notin \mathcal{C}. \\
(b) & \text{If} & t_p^\beta, t_q^\beta \in \mathcal{C} & \text{then} & e_{\{p,q\}} \notin \mathcal{C}. \\
(c) & \text{If} & t_p^\beta, t_q^\alpha \in \mathcal{C} & \text{then} & e_{\{p,q\}} \in \mathcal{C}. \\
(d) & \text{If} & t_p^\alpha, t_q^\beta \in \mathcal{C} & \text{then} & e_{\{p,q\}} \in \mathcal{C}.
\end{array}$$



Finding the optimal α expansion

- Given an input labeling f (partition $\mathcal{P}$) and a label α we want to find a labeling $\hat{f}$ that minimizes E over all labelings within one α-expansion of f.
- This is going to be done by computing a labeling corresponding to a minimum cut on a graph $\mathcal{G}_\alpha = (\mathcal{V}_\alpha, \mathcal{E}_\alpha)$.

Finding the optimal α expansion

- Given an input labeling f (partition $\mathcal{P}$) and a label α we want to find a labeling $\hat{f}$ that minimizes E over all labelings within one α-expansion of f.
- This is going to be done by computing a labeling corresponding to a minimum cut on a graph $\mathcal{G}_\alpha = (\mathcal{V}_\alpha, \mathcal{E}_\alpha)$.
- The **structure** of this graph is **dynamically determined** by the current partition $\mathcal{P}$ and by the label α.

Finding the optimal α expansion

- Given an input labeling f (partition $\mathcal{P}$) and a label α we want to find a labeling $\hat{f}$ that minimizes E over all labelings within one α-expansion of f.
- This is going to be done by computing a labeling corresponding to a minimum cut on a graph $\mathcal{G}_\alpha = (\mathcal{V}_\alpha, \mathcal{E}_\alpha)$.
- The **structure** of this graph is **dynamically determined** by the current partition $\mathcal{P}$ and by the label α.
- Different graph than the $\alpha - \beta$ swap.

Finding the optimal α expansion

- Given an input labeling f (partition $\mathcal{P}$) and a label α we want to find a labeling $\hat{f}$ that minimizes E over all labelings within one α-expansion of f.
- This is going to be done by computing a labeling corresponding to a minimum cut on a graph $\mathcal{G}_\alpha = (\mathcal{V}_\alpha, \mathcal{E}_\alpha)$.
- The **structure** of this graph is **dynamically determined** by the current partition $\mathcal{P}$ and by the label α.
- Different graph than the $\alpha - \beta$ swap.

Graph Construction

- The set of vertices includes the two terminals α and $\bar{\alpha}$, as well as all image pixels $p \in \mathcal{P}$.
- Additionally, for each pair of neighboring pixels p, q such that $f_p \neq f_q$ we create an auxiliary node $a_{p,q}$.

Graph Construction

- The set of vertices includes the two terminals α and $\bar{\alpha}$, as well as all image pixels $p \in \mathcal{P}$.
- Additionally, for each pair of neighboring pixels p, q such that $f_p \neq f_q$ we create an auxiliary node $a_{p,q}$.
- Each pixel p is connected to the terminals α and $\bar{\alpha}$, called t-links.

Graph Construction

- The set of vertices includes the two terminals α and $\bar{\alpha}$, as well as all image pixels $p \in \mathcal{P}$.
- Additionally, for each pair of neighboring pixels p, q such that $f_p \neq f_q$ we create an auxiliary node $a_{p,q}$.
- Each pixel p is connected to the terminals α and $\bar{\alpha}$, called t-links.
- Each set of pixels p, q which are neighbors and $f_p = f_q$, we connect with and n-link.

Graph Construction

- The set of vertices includes the two terminals α and $\bar{\alpha}$, as well as all image pixels $p \in \mathcal{P}$.
- Additionally, for each pair of neighboring pixels p, q such that $f_p \neq f_q$ we create an auxiliary node $a_{p,q}$.
- Each pixel p is connected to the terminals α and $\bar{\alpha}$, called t-links.
- Each set of pixels p, q which are neighbors and $f_p = f_q$, we connect with and n-link.
- For each pair of neighboring pixels such that $f_p \neq f_q$, we create a triplet $\{e_{p,a}, e_{a,q}, t_a^{\bar{\alpha}}\}$.

Graph Construction

- The set of vertices includes the two terminals α and $\bar{\alpha}$, as well as all image pixels $p \in \mathcal{P}$.
- Additionally, for each pair of neighboring pixels p, q such that $f_p \neq f_q$ we create an auxiliary node $a_{p,q}$.
- Each pixel p is connected to the terminals α and $\bar{\alpha}$, called t-links.
- Each set of pixels p, q which are neighbors and $f_p = f_q$, we connect with and n-link.
- For each pair of neighboring pixels such that $f_p \neq f_q$, we create a triplet $\{e_{p,a}, e_{a,q}, t_a^{\bar{\alpha}}\}$.
- The set of edges is then

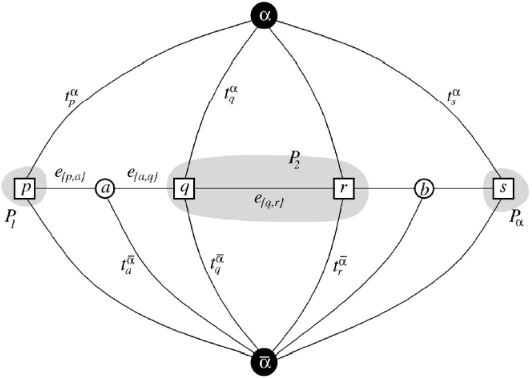
$$\mathcal{E}_\alpha = \left\{ \bigcup_{p \in \mathcal{P}} \{t_p^\alpha, t_p^{\bar{\alpha}}\}, \bigcup_{\substack{\{p,q\} \in \mathcal{N} \\ f_p \neq f_q}} \mathcal{E}_{\{p,q\}} \ , \bigcup_{\substack{\{p,q\} \in \mathcal{N} \\ f_p = f_q}} e_{\{p,q\}} \right\}$$

Graph Construction

- The set of vertices includes the two terminals α and $\bar{\alpha}$, as well as all image pixels $p \in \mathcal{P}$.
- Additionally, for each pair of neighboring pixels p, q such that $f_p \neq f_q$ we create an auxiliary node $a_{p,q}$.
- Each pixel p is connected to the terminals α and $\bar{\alpha}$, called t-links.
- Each set of pixels p, q which are neighbors and $f_p = f_q$, we connect with and n-link.
- For each pair of neighboring pixels such that $f_p \neq f_q$, we create a triplet $\{e_{p,a}, e_{a,q}, t_a^{\bar{\alpha}}\}$.
- The set of edges is then

$$\mathcal{E}_\alpha = \left\{ \bigcup_{p \in \mathcal{P}} \{t_p^\alpha, t_p^{\bar{\alpha}}\}, \bigcup_{\substack{\{p,q\} \in \mathcal{N} \\ f_p \neq f_q}} \mathcal{E}_{\{p,q\}} \;, \bigcup_{\substack{\{p,q\} \in \mathcal{N} \\ f_p = f_q}} e_{\{p,q\}} \right\}$$

Graph Construction



edge	weight	for
$t_p^{\bar\alpha}$	∞	$p \in \mathcal{P}_\alpha$
$t_p^{\bar\alpha}$	$D_p(f_p)$	$p \notin \mathcal{P}_\alpha$
t_p^{α}	$D_p(\alpha)$	$p \in \mathcal{P}$
$e_{\{p,a\}}$	$V(f_p,\alpha)$	
$e_{\{a,q\}}$	$V(\alpha,f_q)$	$\{p,q\} \in \mathcal{N},\ f_p \neq f_q$
$t_a^{\bar\alpha}$	$V(f_p,f_q)$	
$e_{\{p,q\}}$	$V(f_p,\alpha)$	$\{p,q\} \in \mathcal{N},\ f_p = f_q$

Properties

- There is a one-to-one correspondences between a cut and a labeling.

$$f_p^{\mathcal{C}} = \begin{cases} \alpha & \text{if} \quad t_p^{\alpha} \in \mathcal{C} \\ f_p & \text{if} \quad t_p^{\bar{\alpha}} \in \mathcal{C} \end{cases} \qquad \forall p \in \mathcal{P}.$$

- The energy of the cut is the energy of the labeling.
- See Boykov et al, "*fast approximate energy minimization via graph cuts*" PAMI 2001.

Property 5.2. *If* $\{p,q\} \in \mathcal{N}$ *and* $f_p \neq f_q$, *then a minimum cut* $\mathcal{C}$ *on* $\mathcal{G}_\alpha$ *satisfies:*

$$\begin{array}{lllll} (a) & \textit{If} & t_p^{\alpha}, t_q^{\alpha} \in \mathcal{C} & \textit{then} & \mathcal{C} \cap \mathcal{E}_{\{p,q\}} = \emptyset. \\ (b) & \textit{If} & t_p^{\bar{\alpha}}, t_q^{\bar{\alpha}} \in \mathcal{C} & \textit{then} & \mathcal{C} \cap \mathcal{E}_{\{p,q\}} = t_a^{\bar{\alpha}}. \\ (c) & \textit{If} & t_p^{\bar{\alpha}}, t_q^{\alpha} \in \mathcal{C} & \textit{then} & \mathcal{C} \cap \mathcal{E}_{\{p,q\}} = e_{\{p,a\}}. \\ (d) & \textit{If} & t_p^{\alpha}, t_q^{\bar{\alpha}} \in \mathcal{C} & \textit{then} & \mathcal{C} \cap \mathcal{E}_{\{p,q\}} = e_{\{a,q\}}. \end{array}$$

Global Minimization Techniques

Ways to get an approximate solution typically

- Dynamic programming approximations
- Sampling
- Simulated annealing
- Graph-cuts: imposes restrictions on the type of pairwise cost functions
- Message passing: iterative algorithms that pass messages between nodes in the graph.

Now we can solve for the MAP (approximately) in general energies. We can solve for other problems than stereo

Let's look at data/bechmarks

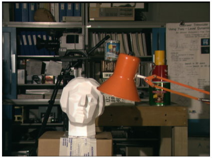
Benchmarks

Two benchmarks with very different characteristics

(Middlebury)

(KITTI)

Middlebury Stereo Evaluation – Version 2

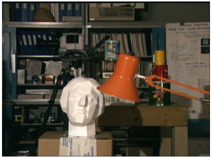




- Laboratory
- Lambertian

Middlebury Dataset

Middlebury Stereo Evaluation – Version 2



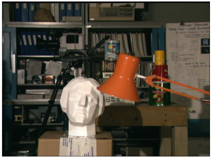
- Laboratory
- Lambertian
- Rich in texture

Middlebury Dataset

Middlebury Stereo Evaluation – Version 2

- Laboratory
- Lambertian
- Rich in texture
- Medium-size label set

Middlebury Stereo Evaluation – Version 2





- Laboratory
- Lambertian
- Rich in texture
- Medium-size label set
- Largely fronto-parallel

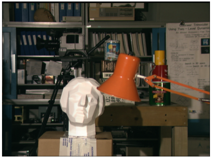
Middlebury Dataset

Middlebury Stereo Evaluation – Version 2

- Laboratory
- Lambertian
- Rich in texture
- Medium-size label set
- Largely fronto-parallel

Benchmarks for Stereo and metrics

Middlebury Stereo Evaluation – Version 2





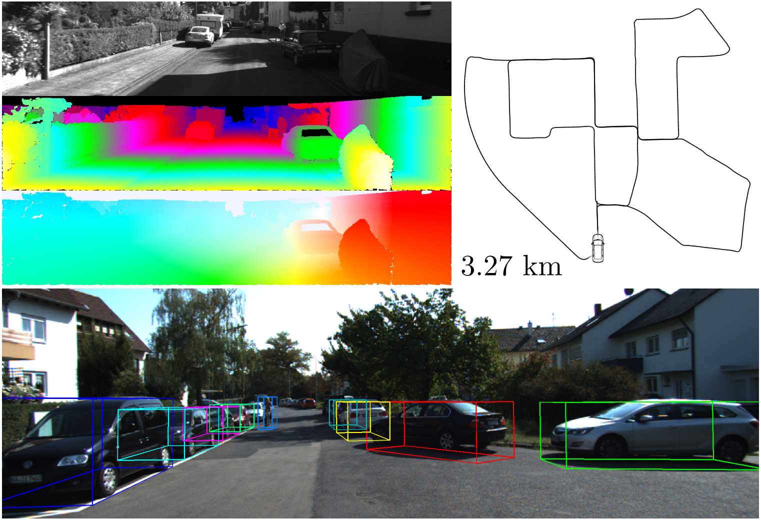
Error Threshold = 1 Algorithm	Avg.	Tsukuba ground truth			Venus ground truth			Teddy ground truth			Cones ground truth		
CoopRegion [41]	8.8	0.87 4	**1.16** 1	4.61 3	0.11 4	0.21 3	1.54 7	5.16 16	8.31 11	13.0 13	2.79 17	7.18 4	8.01 23
AdaptingBP [17]	9.0	1.11 19	1.37 7	5.79 19	0.10 3	0.21 4	1.44 5	4.22 8	7.06 6	11.8 9	2.48 7	7.92 11	7.32 10
ADCensus [94]	7.3	1.07 15	1.48 13	5.73 17	0.09 2	0.25 7	1.15 3	4.10 6	6.22 3	10.9 6	2.42 5	7.25 5	6.95 6
SurfaceStereo [79]	18.2	1.28 32	1.65 21	6.78 37	0.19 18	0.28 10	2.61 32	3.12 2	**5.10** 1	**8.65** 1	2.89 21	7.95 13	8.26 30
GC+SegmBorder [57]	27.1	1.47 45	1.82 32	7.86 58	0.19 19	0.31 12	2.44 26	4.25 9	5.55 2	10.9 7	4.99 77	**5.78** 1	8.66 37
WarpMat [55]	20.8	1.16 20	1.35 6	6.04 24	0.18 17	0.24 6	2.44 26	5.02 13	9.30 17	13.0 15	3.49 39	8.47 22	9.01 44
RDP [102]	12.5	0.97 10	1.39 9	5.00 9	0.21 23	0.38 19	1.89 13	4.84 10	9.94 19	12.6 11	2.53 8	7.69 8	7.38 11
RVbased [116]	11.6	0.95 9	1.42 11	4.98 8	0.11 6	0.29 11	**1.07** 1	5.98 21	11.6 31	15.4 27	2.35 3	7.61 6	6.81 5
OutlierConf [42]	12.9	0.88 5	1.43 12	4.74 5	0.18 16	0.26 9	2.40 22	5.01 12	9.12 16	12.8 12	2.78 16	8.57 23	6.99 7

- Best methods $< 3\%$ errors (for all non-occluded regions)
- `http://vision.middlebury.edu/stereo/data/`

Benchmarks: KITTI Data Collection

- **Two stereo rigs** (1392×512 px, 54 cm base, 90° opening)
- **Velodyne** laser scanner, **GPS+IMU** localization
- **6 hours** at 10 frames per second!

The KITTI Vision Benchmark Suite



Novel Challenges

Fast guided cost-volume filtering (Rhemann et al., CVPR 2011)

Middlebury, Errors: 2.7%

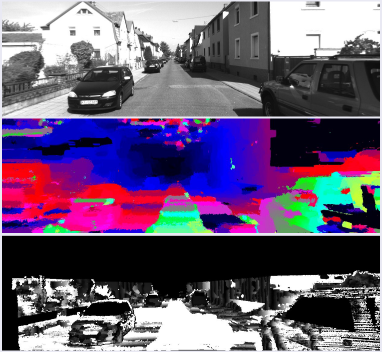
- Error threshold: 1 px (Middlebury) / 3 px (KITTI)

Novel Challenges

Fast guided cost-volume filtering (Rhemann et al., CVPR 2011)

Middlebury, Errors: 2.7%

KITTI, Errors: 46.3%



- Error threshold: 1 px (Middlebury) / 3 px (KITTI)

Novel Challenges

So what is the difference?

Middlebury

- Laboratory
- Lambertian

KITTI

- Moving vehicle
- Specularities

So what is the difference?

Middlebury

- Laboratory
- Lambertian
- Rich in texture

KITTI

- Moving vehicle
- Specularities
- Sensor saturation

Novel Challenges

So what is the difference?

Middlebury

- Laboratory
- Lambertian
- Rich in texture
- Medium-size label set

KITTI

- Moving vehicle
- Specularities
- Sensor saturation
- Large label set

Novel Challenges

So what is the difference?

Middlebury

KITTI

- Laboratory
- Lambertian
- Rich in texture
- Medium-size label set
- Largely fronto-parallel

- Moving vehicle
- Specularities
- Sensor saturation
- Large label set
- Strong slants

Novel Challenges

So what is the difference?

Middlebury

KITTI

- Laboratory
- Lambertian
- Rich in texture
- Medium-size label set
- Largely fronto-parallel

- Moving vehicle
- Specularities
- Sensor saturation
- Large label set
- Strong slants

Stereo Evaluation

Rank	Method	Setting	Out-Noc	Out-All	Avg-Noc	Avg-All	Density	Runtime	Environment	Compare
1	PCBP		4.13 %	5.45 %	0.9 px	1.2 px	100.00 %	5 min	4 cores @ 2.5 Ghz (Matlab + C/C++)	☐
Koichiro Yamaguchi, Tamir Hazan, David McAllester and Raquel Urtasun. Continuous Markov Random Fields for Robust Stereo Estimation. ECCV 2012.										
2	iSGM		5.16 %	7.19 %	1.2 px	2.1 px	94.70 %	8 s	2 cores @ 2.5 Ghz (C/C++)	☐
Simon Hermann and Reinhard Klette. Iterative Semi-Global Matching for Robust Driver Assistance Systems. ACCV 2012.										
3	SGM		5.83 %	7.08 %	1.2 px	1.3 px	85.80 %	3.7 s	1 core @ 3.0 Ghz (C/C++)	☐
Heiko Hirschmueller. Stereo Processing by Semi-Global Matching and Mutual Information. IEEE Transactions on Pattern Analysis and Machine Intelligence 2008.										
4	SNCC		6.27 %	7.33 %	1.4 px	1.5 px	100.00 %	0.27 s	1 core @ 3.0 Ghz (C/C++)	☐
N. Einecke and J. Eggert. A Two-Stage Correlation Method for Stereoscopic Depth Estimation. DICTA 2010.										
5	ITGV		6.31 %	7.40 %	1.3 px	1.5 px	100.00 %	7 s	1 core @ 3.0 Ghz (Matlab + C/C++)	☐
Rene Ranftl, Stefan Gehrig, Thomas Pock and Horst Bischof. Pushing the Limits of Stereo Using Variational Stereo Estimation. IEEE Intelligent Vehicles Symposium 2012.										
6	BSSM		7.50 %	8.89 %	1.4 px	1.6 px	94.87 %	20.7 s	1 core @ 3.5 Ghz (C/C++)	☐
Anonymous submission										
7	OCV-SGBM		7.64 %	9.13 %	1.8 px	2.0 px	86.50 %	1.1 s	1 core @ 2.5 Ghz (C/C++)	☐
Heiko Hirschmueller. Stereo processing by semiglobal matching and mutual information. PAMI 2008.										
8	ELAS		8.24 %	9.95 %	1.4 px	1.6 px	94.55 %	0.3 s	1 core @ 2.5 Ghz (C/C++)	☐
Andreas Geiger, Martin Roser and Raquel Urtasun. Efficient Large-Scale Stereo Matching. ACCV 2010.										
9	MS-DSI		10.68 %	12.11 %	1.9 px	2.2 px	100.00 %	10.3 s	>8 cores @ 2.5 Ghz (C/C++)	☐
Anonymous submission										
10	SDM		10.98 %	12.19 %	2.0 px	2.3 px	63.58 %	1 min	1 core @ 2.5 Ghz (C/C++)	☐
Jana Kostkova. Stratified dense matching for stereopsis in complex scenes. BMVC 2003.										
11	GCSF		12.06 %	13.26 %	1.9 px	2.1 px	60.77 %	2.4 s	1 core @ 2.5 Ghz (C/C++)	☐
Jan Cech, Jordi Sanchez-Riera and Radu P. Horaud. Scene Flow Estimation by Growing Correspondence Seeds. CVPR 2011.										
12	GCS		13.37 %	14.54 %	2.1 px	2.3 px	51.06 %	2.2 s	1 core @ 2.5 Ghz (C/C++)	☐
Jan Cech and Radim Sara. Efficient Sampling of Disparity Space for Fast And Accurate Matching. BenCOS 2007.										
13	CostFilter		19.96 %	21.05 %	5.0 px	5.4 px	100.00 %	4 min	1 core @ 2.5 Ghz (Matlab)	☐
Christoph Rhemann, Asmaa Hosni, Michael Bleyer, Carsten Rother and Margrit Gelautz. Fast Cost-Volume Filtering for Visual Correspondence and Beyond. CVPR 2011.										
14	OCV-BM		25.39 %	26.72 %	7.6 px	7.9 px	55.84 %	0.1 s	1 core @ 2.5 Ghz (C/C++)	☐
G. Bradski. The OpenCV Library. Dr. Dobb's Journal of Software Tools 2000.										
15	GC+occ		33.50 %	34.74 %	8.6 px	9.2 px	87.57 %	6 min	1 core @ 2.5 Ghz (C/C++)	☐
Vladimir Kolmogorov and Ramin Zabih. Computing Visual Correspondence with Occlusions using Graph Cuts. ICCV 2001.										

MRFs for stereo

Global methods: define a Markov random field over

- Pixel-level
- Fronto-parallel planes
- Slanted planes

Plane MRFs

- First segment an image into small regions, i.e., superpixels
- Assume that the 3D world is compose of small frontal/slanted planes

Plane MRFs

- First segment an image into small regions, i.e., superpixels
- Assume that the 3D world is compose of small frontal/slanted planes
- Good representation if the superpixels are small and respect boundaries

$$E(\mathbf{x}_1, \cdots, \mathbf{x}_n) = \sum_i C(\mathbf{x}_i) + \sum_i \sum_{j \in \mathcal{N}_i} C(\mathbf{x}_i, \mathbf{x}_j)$$

with $\mathbf{x}_i \in \Re$ for the fronto-parallel planes, and $\mathbf{x}_i \in \Re^3$ for the slanted planes

Plane MRFs

- First segment an image into small regions, i.e., superpixels
- Assume that the 3D world is compose of small frontal/slanted planes
- Good representation if the superpixels are small and respect boundaries

$$E(\mathbf{x}_1, \cdots, \mathbf{x}_n) = \sum_i C(\mathbf{x}_i) + \sum_i \sum_{j \in \mathcal{N}_j} C(\mathbf{x}_i, \mathbf{x}_j)$$

  with $\mathbf{x}_i \in \Re$ for the fronto-parallel planes, and $\mathbf{x}_i \in \Re^3$ for the slanted planes
- This are continuous variables. Is this a problem?

Plane MRFs

- First segment an image into small regions, i.e., superpixels
- Assume that the 3D world is compose of small frontal/slanted planes
- Good representation if the superpixels are small and respect boundaries

$$E(\mathbf{x}_1, \cdots, \mathbf{x}_n) = \sum_i C(\mathbf{x}_i) + \sum_i \sum_{j \in \mathcal{N}_j} C(\mathbf{x}_i, \mathbf{x}_j)$$

  with $\mathbf{x}_i \in \Re$ for the fronto-parallel planes, and $\mathbf{x}_i \in \Re^3$ for the slanted planes
- This are continuous variables. Is this a problem?
- What can I do to solve this? Discretize the problem

Plane MRFs

- First segment an image into small regions, i.e., superpixels
- Assume that the 3D world is compose of small frontal/slanted planes
- Good representation if the superpixels are small and respect boundaries

$$E(\mathbf{x}_1, \cdots, \mathbf{x}_n) = \sum_i C(\mathbf{x}_i) + \sum_i \sum_{j \in \mathcal{N}_j} C(\mathbf{x}_i, \mathbf{x}_j)$$

  with $\mathbf{x}_i \in \Re$ for the fronto-parallel planes, and $\mathbf{x}_i \in \Re^3$ for the slanted planes
- This are continuous variables. Is this a problem?
- What can I do to solve this? Discretize the problem
- The unitary are usually agreegation of cost over the local matching on the pixels in that superpixel

Plane MRFs

- First segment an image into small regions, i.e., superpixels
- Assume that the 3D world is compose of small frontal/slanted planes
- Good representation if the superpixels are small and respect boundaries

$$E(\mathbf{x}_1, \cdots, \mathbf{x}_n) = \sum_i C(\mathbf{x}_i) + \sum_i \sum_{j \in \mathcal{N}_j} C(\mathbf{x}_i, \mathbf{x}_j)$$

  with $\mathbf{x}_i \in \Re$ for the fronto-parallel planes, and $\mathbf{x}_i \in \Re^3$ for the slanted planes
- This are continuous variables. Is this a problem?
- What can I do to solve this? Discretize the problem
- The unitary are usually agreegation of cost over the local matching on the pixels in that superpixel
- Pairwise is typically smoothness

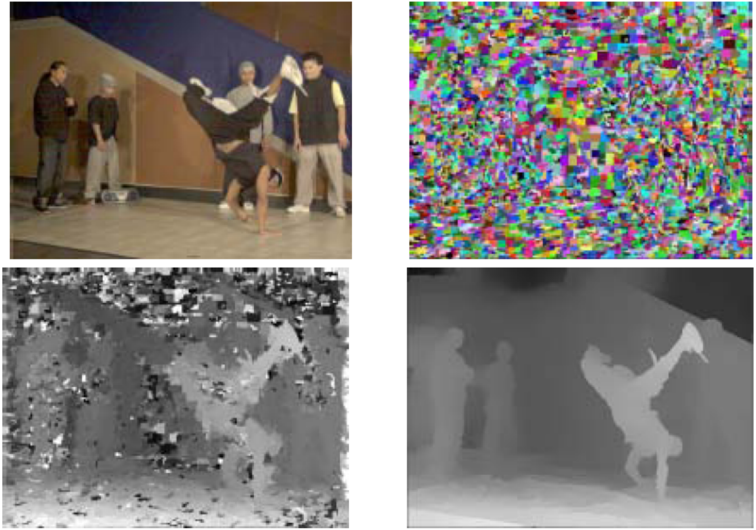
Plane MRFs

- First segment an image into small regions, i.e., superpixels
- Assume that the 3D world is compose of small frontal/slanted planes
- Good representation if the superpixels are small and respect boundaries

$$E(\mathbf{x}_1, \cdots, \mathbf{x}_n) = \sum_i C(\mathbf{x}_i) + \sum_i \sum_{j \in \mathcal{N}_j} C(\mathbf{x}_i, \mathbf{x}_j)$$

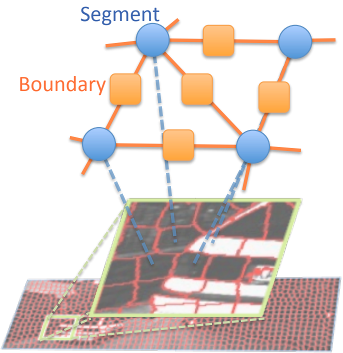
  with $\mathbf{x}_i \in \Re$ for the fronto-parallel planes, and $\mathbf{x}_i \in \Re^3$ for the slanted planes
- This are continuous variables. Is this a problem?
- What can I do to solve this? Discretize the problem
- The unitary are usually agreegation of cost over the local matching on the pixels in that superpixel
- Pairwise is typically smoothness

Slanted-plane MRFs



A more sophisticated occlusion model

- MRF on continuous variables (slanted planes) and discrete var. (boundary)
- Combines depth ordering (segmentation) and stereo



Superpixels (UCM [Arbelaez, et al. 2011] and SLIC [Achanta, et al. 2010])

Segment variable $\mathbf{y}_i = (\alpha_i, \beta_i, \gamma_i)$

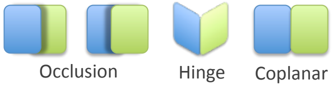
Slanted 3D plane of segment

Continuous variable

Boundary variable o_{ij}

Relationship between segments

4 states



Discrete variable

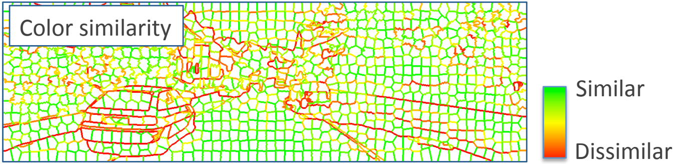
- Takes as input disparities computed by any local algorithm

Energy of PCBP-Stereo

- **y** the set of slanted 3D planes, **o** the set of discrete boundary variables

$$E(\mathbf{y},\mathbf{o}) = E_{color}(\mathbf{o}) + E_{match}(\mathbf{y},\mathbf{o}) + E_{compatibility}(\mathbf{y},\mathbf{o}) + E_{junction}(\mathbf{o})$$

Similar color → Likely to be coplanar

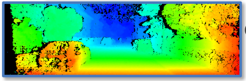


Energy of PCBP-Stereo

- **y** the set of slanted 3D planes, **o** the set of discrete boundary variables

$$E(\mathbf{y}, \mathbf{o}) = E_{color}(\mathbf{o}) + E_{match}(\mathbf{y}, \mathbf{o}) + E_{compatibility}(\mathbf{y}, \mathbf{o}) + E_{junction}(\mathbf{o})$$

Agreement with result of input disparity map



Computed by any matching method
(Modified semi-global matching)

Truncated quadratic function $\phi_i^{\mathrm{TP}}(\mathbf{p}, \mathbf{y}_i, K) = \min\left(|\underbrace{\mathcal{D}(\mathbf{p})}_{\text{Disparity map}} - \underbrace{\hat{d}_i(\mathbf{p}, \mathbf{y}_i)}_{\text{Slanted plane}}|, K\right)^2$

On boundary

"Occlusion" – Foreground segment owns boundary

Energy of PCBP-Stereo

- $\mathbf{y}$ the set of slanted 3D planes, $\mathbf{o}$ the set of discrete boundary variables

$$E(\mathbf{y},\mathbf{o}) = E_{color}(\mathbf{o}) + E_{match}(\mathbf{y},\mathbf{o}) + E_{compatibility}(\mathbf{y},\mathbf{o}) + E_{junction}(\mathbf{o})$$

(1) Preference of boundary label (Coplanar > Hinge > Occlusion)

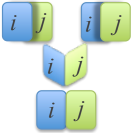
Impose penalty $\lambda_{\text{occ}} > \lambda_{\text{hinge}} > 0$

(2) Boundary labels $\xleftrightarrow{\text{match}}$ Slanted planes

"Occlusion" $\longleftrightarrow$ $\hat{d}_{\text{front}}(\mathbf{p}) > \hat{d}_{\text{back}}(\mathbf{p})$

"Hinge" $\longleftrightarrow$ $\hat{d}_i(\mathbf{p}) = \hat{d}_j(\mathbf{p})$ on boundary

"Coplanar" $\longleftrightarrow$ $\hat{d}_i(\mathbf{p}) = \hat{d}_j(\mathbf{p})$ in both segments



Energy of PCBP-Stereo

- $\mathbf{y}$ the set of slanted 3D planes, $\mathbf{o}$ the set of discrete boundary variables

$$E(\mathbf{y},\mathbf{o}) = E_{color}(\mathbf{o}) + E_{match}(\mathbf{y},\mathbf{o}) + E_{compatibility}(\mathbf{y},\mathbf{o}) + E_{junction}(\mathbf{o})$$

Occlusion boundary reasoning [Malik 1987]

Penalize impossible junctions

Impossible cases