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 \(T\)

\[
\begin{align*}
5 + 1 + 9 &= 15
\end{align*}
\]

[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 \( i,j \)
\[ \theta_{ij}(0,1) + \theta_{ij}(1,0) \geq \theta_{ij}(0,0) + \theta_{ij}(1,1) \]

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?

\[ A = \theta_{ij}(0,0) \quad B = \theta_{ij}(0,1) \quad C = \theta_{ij}(1,0) \quad D = \theta_{ij}(1,1) \]

\[
\begin{array}{c|c}
0 & 1 \\
\hline
0 & \begin{array}{c}
A \\
B \\
C \\
D \\
\end{array} \\
\end{array}
\]

\[
\begin{array}{c|c}
0 & 1 \\
\hline
0 & \begin{array}{c}
\theta_{ij}(0,0) \\
\theta_{ij}(1,0)-\theta_{ij}(0,0) x_i + \theta_{ij}(1,0)-\theta_{ij}(0,0) x_j \\
\theta_{ij}(1,0)+\theta_{ij}(0,1)-\theta_{ij}(0,0)-\theta_{ij}(1,1) (1-x_i) x_j \\
\end{array} \\
\end{array}
\]

\[
\begin{array}{c|c}
0 & 1 \\
\hline
0 & \begin{array}{c}
\theta_{ij}(1,0)-\theta_{ij}(0,0) x_i + \theta_{ij}(1,0)-\theta_{ij}(0,0) x_j \\
\theta_{ij}(1,0)+\theta_{ij}(0,1)-\theta_{ij}(0,0)-\theta_{ij}(1,1) (1-x_i) x_j \\
\end{array} \\
\end{array}
\]

\[
\begin{array}{c|c}
0 & 1 \\
\hline
0 & \begin{array}{c}
B + C - A - D \geq 0 \quad \text{is true from the submodularity of } \theta_{ij} \\
\end{array} \\
\end{array}
\]

[Source: P. Kohli]
Graph Construction

$E(a_1, a_2)$

Source (0)  

$\begin{array}{c} a_1 \\ a_2 \end{array}$

Sink (1)

[Source: P. Kohli]
Graph Construction

\[ E(a_1, a_2) = 2a_1 \]

[Source: P. Kohli]
Graph Construction

\[ E(a_1, a_2) = 2a_1 + 5a_1 \]

Source (0)

\( a_1 \)

Sink (1)

\( a_2 \)

Source: P. Kohli
Graph Construction

\[ E(a_1, a_2) = 2a_1 + 5\bar{a}_1 + 9a_2 + 4\bar{a}_2 \]

[Source: P. Kohli]
Graph Construction

\[ E(a_1, a_2) = 2a_1 + 5\overline{a}_1 + 9a_2 + 4\overline{a}_2 + 2a_1\overline{a}_2 \]

[Source: P. Kohli]
Graph Construction

\[ E(a_1, a_2) = 2a_1 + 5\bar{a}_1 + 9a_2 + 4\bar{a}_2 + 2a_1\bar{a}_2 + \bar{a}_1a_2 \]
Graph Construction

\[ E(a_1, a_2) = 2a_1 + 5\bar{a}_1 + 9a_2 + 4\bar{a}_2 + 2a_1\bar{a}_2 + \bar{a}_1a_2 \]

[Source: P. Kohli]
Graph Construction

\[ E(a_1, a_2) = 2a_1 + 5\bar{a}_1 + 9a_2 + 4\bar{a}_2 + 2a_1\bar{a}_2 + \bar{a}_1a_2 \]

[Source: P. Kohli]
Graph Construction

\[ E(a_1, a_2) = 2a_1 + 5\bar{a}_1 + 9a_2 + 4\bar{a}_2 + 2a_1\bar{a}_2 + \bar{a}_1a_2 \]

[Source: P. Kohli]
How to compute the St-mincut?

Solve the dual maximum flow problem

Compute the maximum flow between Source and Sink s.t.

Edges: Flow < Capacity
Nodes: Flow in = Flow out

Min-cut/Max-flow Theorem
In every network, the maximum flow equals the cost of the st-mincut

Assuming non-negative capacity

[Source: P. Kohli]
How does the code look like

```c
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 *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

/* 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

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)
Graph cuts for multi-label problems

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

**So what is the problem?**

\[ E_m(y_1, y_2, \ldots, y_n) \rightarrow E_b(x_1, x_2, \ldots, x_m) \]

\[ y_i \in L = \{l_1, l_2, \ldots, l_k\} \]

\[ x_i \in L = \{0, 1\} \]

**Multi-label Problem** \hspace{1cm} **Binary label Problem**

**such that:**

Let \( Y \) and \( X \) be the set of feasible solutions, then

1. One-One encoding function \( T: X \rightarrow Y \)
2. \( \arg \min E_m(y) = T(\arg \min E_b(x)) \)

- Very high computational cost

[Source: P. Kohli]
Computing the Optimal Move

[Source: P. Kohli]
Minimizing Pairwise Functions
[Boykov Veksler and Zabih, PAMI 2001]

- Series of locally optimal moves
- Each move reduces energy
- Optimal move by minimizing submodular function

Space of Solutions \( (x) : L^n \)

Move Space \( (t) : 2^n \)

Current Solution
Search Neighbourhood
\( n \) Number of Variables
\( L \) Number of Labels

[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.
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 $\alpha, \beta, \gamma$
  
  \[
  V(\alpha, \beta) = 0 \iff \alpha = \beta \\
  V(\alpha, \beta) = V(\beta, \alpha) \geq 0 \\
  V(\alpha, \beta) \leq V(\alpha, \gamma) + V(\gamma, \beta)
  \]

- **Semi-metric** if it satisfies for any set of labels $\alpha, \beta, \gamma$
  
  \[
  V(\alpha, \beta) = 0 \iff \alpha = \beta \\
  V(\alpha, \beta) = V(\beta, \alpha) \geq 0
  \]
Metric vs Semimetric

Two general classes of pairwise interactions

- **Metric** if it satisfies for any set of labels $\alpha, \beta, \gamma$

  \[
  V(\alpha, \beta) = 0 \iff \alpha = \beta \\
  V(\alpha, \beta) = V(\beta, \alpha) \geq 0 \\
  V(\alpha, \beta) \leq V(\alpha, \gamma) + V(\gamma, \beta)
  \]

- **Semi-metric** if it satisfies for any set of labels $\alpha, \beta, \gamma$

  \[
  V(\alpha, \beta) = 0 \iff \alpha = \beta \\
  V(\alpha, \beta) = V(\beta, \alpha) \geq 0
  \]
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
- $\alpha$ expansion works for $V$ being a metric

\[ x = t x^1 + (1 - t) x^2 \]

Minimize over move variables $t$

\[ E_m(t) = E(t x^1 + (1 - t) x^2) \]

**Figure**: Figure from P. Kohli's tutorial on graph-cuts

For certain $x^1$ and $x^2$, the move energy is sub-modular QPBF
Variables labeled \( \alpha, \beta \) can swap their labels

Swap Sky, House

[Source: P. Kohli]
• Variables labeled $\alpha$, $\beta$ can swap their labels

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

\[
\theta_{ij}(l_a, l_b) \geq 0 \\
\theta_{ij}(l_a, l_b) = 0 \quad \text{if} \quad a = b
\]

Examples: Potts model, Truncated Convex

[Source: P. Kohli]
Expansion Move

- Variables take label $\alpha$ or retain current label

**Status:** Expand Sky to Tree

[Source: P. Kohli]
Variables take label $\alpha$ or retain current label

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

\[ \theta_{ij} (l_a, l_b) + \theta_{ij} (l_b, l_c) \geq \theta_{ij} (l_a, l_c) \]

Examples: Potts model, Truncated linear

Semi metric + Triangle Inequality

Cannot solve truncated quadratic

[Source: P. Kohli]
More formally

- Any labeling can be uniquely represented by a partition of image pixels \( P = \{ P_l | l \in L \} \), where \( P_l = \{ p \in P | f_p = l \} \) is a subset of pixels assigned label \( l \).

- There is a one to one correspondence between labelings \( f \) and partitions \( P \).
More formally

- Any labeling can be uniquely represented by a partition of image pixels \( \mathcal{P} = \{ \mathcal{P}_l | l \in \mathcal{L} \} \), 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 \( \alpha, \beta \), a move from a partition \( \mathcal{P} \) (labeling \( f \)) to a new partition \( \mathcal{P}' \) (labeling \( f' \)) is called an \( \alpha - \beta \) swap if \( \mathcal{P}_l = \mathcal{P}'_l \) for any label \( l \neq \alpha, \beta \).
More formally

- Any labeling can be uniquely represented by a partition of image pixels $P = \{\mathcal{P}_l | l \in \mathcal{L}\}$, 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 $\alpha, \beta$, a move from a partition $\mathcal{P}$ (labeling $f$) to a new partition $\mathcal{P}'$ (labeling $f'$) is called an $\alpha - \beta$ swap if $\mathcal{P}_l = \mathcal{P}'_l$ 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 \( \mathcal{P} = \{ \mathcal{P}_l | l \in \mathcal{L} \} \), 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 \( \alpha, \beta \), a move from a partition \( \mathcal{P} \) (labeling \( f \)) to a new partition \( \mathcal{P}' \) (labeling \( f' \)) is called an \( \alpha - \beta \) swap if \( \mathcal{P}_l = \mathcal{P}'_l \) 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 \( \alpha \)-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 $\mathcal{P} = \{\mathcal{P}_l | l \in \mathcal{L}\}$, 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 $\alpha, \beta$, a move from a partition $\mathcal{P}$ (labeling $f$) to a new partition $\mathcal{P}'$ (labeling $f'$) is called an $\alpha - \beta$ swap if $\mathcal{P}_l = \mathcal{P}'_l$ 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 $\alpha$-expansion if $\mathcal{P}_\alpha \subset \mathcal{P}'_\alpha$ and $\mathcal{P}'_l \subset \mathcal{P}_l$.

- An $\alpha$-expansion move allows any set of image pixels to change their labels to $\alpha$. 
More formally

- Any labeling can be uniquely represented by a partition of image pixels \( P = \{ P_l | l \in \mathcal{L} \} \), where \( P_l = \{ p \in \mathcal{P} | f_p = l \} \) 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 \( \alpha, \beta \), a move from a partition \( \mathcal{P} \) (labeling \( f \)) to a new partition \( \mathcal{P}' \) (labeling \( f' \)) is called an \( \alpha-\beta \) swap if \( P_l = P'_l \) 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 \( \alpha\)-expansion if \( P_\alpha \subset P'_\alpha \) and \( P'_l \subset P_l \).

- An \( \alpha\)-expansion move allows any set of image pixels to change their labels to \( \alpha \).
Figure: (a) Current partition (b) local move (c) $\alpha - \beta$-swap (d) $\alpha$-expansion.
1. Start with an arbitrary labeling $f$
2. Set success := 0
3. For each pair of labels $\{\alpha, \beta\} \in \mathcal{L}$
   3.1. Find $\hat{f} = \text{arg min } E(f')$ among $f'$ within one $\alpha$-$\beta$ 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 $f$
Given an input labeling $f$ (partition $\mathcal{P}$) and a pair of labels $\alpha, \beta$ 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 $G_{\alpha \beta} = (V_{\alpha \beta}, E_{\alpha \beta})$. 
Finding optimal Swap move

- Given an input labeling $f$ (partition $\mathcal{P}$) and a pair of labels $\alpha, \beta$ 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 $G_{\alpha\beta} = (V_{\alpha\beta}, E_{\alpha\beta})$.

- The structure of this graph is dynamically determined by the current partition $\mathcal{P}$ and by the labels $\alpha, \beta$.
Finding optimal Swap move

- Given an input labeling \( f \) (partition \( P \)) and a pair of labels \( \alpha, \beta \) 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 \( G_{\alpha\beta} = (V_{\alpha\beta}, E_{\alpha\beta}) \).

- The **structure** of this graph is **dynamically determined** by the current partition \( \mathcal{P} \) and by the labels \( \alpha, \beta \).
The set of vertices includes the two terminals $\alpha$ and $\beta$, as well as image pixels $p$ in the sets $P_\alpha$ and $P_\beta$ (i.e., $f_p \in \{\alpha, \beta\}$).

Each pixel $p \in P_{\alpha\beta}$ is connected to the terminals $\alpha$ and $\beta$, called $t$-links.

Each set of pixels $p, q \in P_{\alpha\beta}$ which are neighbors is connected by an edge $e_{p,q}$.

<table>
<thead>
<tr>
<th>edge</th>
<th>weight</th>
<th>for</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t^\alpha_p$</td>
<td>$D_p(\alpha) + \sum_{q \in N_p \cap P_{\alpha\beta}} V(\alpha, f_q)$</td>
<td>$p \in P_{\alpha\beta}$</td>
</tr>
<tr>
<td>$t^\beta_p$</td>
<td>$D_p(\beta) + \sum_{q \in N_p \cap P_{\alpha\beta}} V(\beta, f_q)$</td>
<td>$p \in P_{\alpha\beta}$</td>
</tr>
<tr>
<td>$e_{{p,q}}$</td>
<td>$V(\alpha, \beta)$</td>
<td>${p,q} \in N_{p,q \in P_{\alpha\beta}}$</td>
</tr>
</tbody>
</table>
Computing the Cut

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

$$f_p^c = \begin{cases} 
\alpha & \text{if } t^\alpha_p \in C \text{ for } p \in P_{\alpha\beta} \\
\beta & \text{if } t^\beta_p \in C \text{ for } p \in P_{\alpha\beta} \\
f_p & \text{for } p \in P, p \notin P_{\alpha\beta}.
\end{cases}$$

- There is a one-to-one correspondence between a cut and a labeling.
- The energy of the cut is the energy of the labeling.
Properties

- For any cut, then

  \begin{align*}
  (a) \quad & \text{If } t_p^\alpha, t_q^\alpha \in \mathcal{C} \quad \text{then} \quad e_{\{p,q\}} \notin \mathcal{C}. \\
  (b) \quad & \text{If } t_p^\beta, t_q^\beta \in \mathcal{C} \quad \text{then} \quad e_{\{p,q\}} \notin \mathcal{C}. \\
  (c) \quad & \text{If } t_p^\beta, t_q^\alpha \in \mathcal{C} \quad \text{then} \quad e_{\{p,q\}} \in \mathcal{C}. \\
  (d) \quad & \text{If } t_p^\alpha, t_q^\beta \in \mathcal{C} \quad \text{then} \quad e_{\{p,q\}} \in \mathcal{C}. 
  \end{align*}
Finding the optimal $\alpha$ expansion

- Given an input labeling $f$ (partition $\mathcal{P}$) and a label $\alpha$ we want to find a labeling $\hat{f}$ that minimizes $E$ over all labelings within one $\alpha$-expansion of $f$.

- This is going to be done by computing a labeling corresponding to a minimum cut on a graph $G_\alpha = (\mathcal{V}_\alpha, \mathcal{E}_\alpha)$. 

Raquel Urtasun (TTI-C)
Finding the optimal $\alpha$ expansion

- Given an input labeling $f$ (partition $\mathcal{P}$) and a label $\alpha$ we want to find a labeling $\hat{f}$ that minimizes $E$ over all labelings within one $\alpha$-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 $\alpha$. 
Finding the optimal $\alpha$ expansion

- Given an input labeling $f$ (partition $\mathcal{P}$) and a label $\alpha$ we want to find a labeling $\hat{f}$ that minimizes $E$ over all labelings within one $\alpha$-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 $\alpha$.

- Different graph than the $\alpha - \beta$ swap.
Finding the optimal $\alpha$ expansion

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

- The set of vertices includes the two terminals $\alpha$ 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 $\alpha$ 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 $\alpha$ and $\bar{\alpha}$, called $t$-links.
The set of vertices includes the two terminals $\alpha$ and $\bar{\alpha}$, as well as all image pixels $p \in P$.

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 $\alpha$ 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 $\alpha$ and $\bar{\alpha}$, as well as all image pixels $p \in P$.
- 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 $\alpha$ and $\bar{\alpha}$, called $t$-links.
- Each set of pixels $p, q$ which are neighbors and $f_p = f_q$, we connect with an $n$-link.
- For each pair of neighboring pixels such that $f_p \neq f_q$, we create a triplet $\{e_{p,a}, e_{a,q}, t_{\bar{\alpha}}\}$.
The set of vertices includes the two terminals $\alpha$ 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 $\alpha$ 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 $\{e_{p,a}, e_{a,q}, t_{\bar{\alpha}}\}$.

The set of edges is then

$$\mathcal{E}_\alpha = \left\{ \bigcup_{p \in \mathcal{P}} \{t^\alpha_p, t_{\bar{\alpha}}_p\}, \bigcup_{\{p,q\} \in \mathcal{N} \atop f_p \neq f_q} \mathcal{E}_{\{p,q\}}, \bigcup_{\{p,q\} \in \mathcal{N} \atop f_p = f_q} e_{\{p,q\}} \right\}$$
Graph Construction

- The set of vertices includes the two terminals $\alpha$ 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 $\alpha$ 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 $\{e_{p,a}, e_{a,q}, t_{\bar{\alpha}}\}$.

- The set of edges is then

$$\mathcal{E}_\alpha = \left\{ \bigcup_{p \in \mathcal{P}} \{t^\alpha_p, t^{\bar{\alpha}}_p\}, \bigcup_{\{p,q\} \in \mathcal{N}} \mathcal{E}_{\{p,q\}} \cup \bigcup_{\{p,q\} \in \mathcal{N}} \mathcal{E}_{\{p,q\}} \right\}$$
Graph Construction

<table>
<thead>
<tr>
<th>edge</th>
<th>weight</th>
<th>for</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t^\alpha_p$</td>
<td>$\infty$</td>
<td>$p \in \mathcal{P}_\alpha$</td>
</tr>
<tr>
<td>$t^\alpha_p$</td>
<td>$D_p(f_p)$</td>
<td>$p \notin \mathcal{P}_\alpha$</td>
</tr>
<tr>
<td>$t^\alpha_p$</td>
<td>$D_p(\alpha)$</td>
<td>$p \in \mathcal{P}$</td>
</tr>
<tr>
<td>$e_{{p,\alpha}}$</td>
<td>$V(f_p, \alpha)$</td>
<td></td>
</tr>
<tr>
<td>$e_{{p,q}}$</td>
<td>$V(\alpha, f_q)$</td>
<td>${p, q} \in \mathcal{N}$, $f_p \neq f_q$</td>
</tr>
<tr>
<td>$t^\alpha_a$</td>
<td>$V(f_p, f_q)$</td>
<td></td>
</tr>
<tr>
<td>$e_{{p,q}}$</td>
<td>$V(f_p, \alpha)$</td>
<td>${p, q} \in \mathcal{N}$, $f_p = f_q$</td>
</tr>
</tbody>
</table>
Properties

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

\[ f_p^C = \begin{cases} \alpha & \text{if } t_p^\alpha \in C \\ f_p & \text{if } t_p^\alpha \not\in C \end{cases} \quad \forall p \in \mathcal{P} \]

- The energy of the cut is the energy of the labeling.


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

- (a) If \( t_p^\alpha, t_q^\alpha \in C \) then \( C \cap \mathcal{E}_{\{p, q\}} = \emptyset \).

- (b) If \( t_p^\alpha, t_q^\alpha \in C \) then \( C \cap \mathcal{E}_{\{p, q\}} = t_p^\alpha \).

- (c) If \( t_p^\alpha, t_q^\alpha \in C \) then \( C \cap \mathcal{E}_{\{p, q\}} = c_{\{p, q\}} \).

- (d) If \( t_p^\alpha, t_q^\alpha \in C \) then \( C \cap \mathcal{E}_{\{p, q\}} = c_{\{a, q\}} \).
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/benchmarks
Two benchmarks with very different characteristics

(Middlebury) (KITTI)
Middlebury Dataset

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 Dataset

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

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

<table>
<thead>
<tr>
<th>Algorithm</th>
<th>Error Threshold = 1</th>
<th>Avg.</th>
<th>Tsukuba ground truth</th>
<th>Venus ground truth</th>
<th>Teddy ground truth</th>
<th>Cones ground truth</th>
</tr>
</thead>
<tbody>
<tr>
<td>CoopRegion [41]</td>
<td>8.8</td>
<td>0.87</td>
<td>1.16</td>
<td>4.61</td>
<td>0.11</td>
<td>5.16</td>
</tr>
<tr>
<td>AdaptingBP [17]</td>
<td>9.0</td>
<td>1.11</td>
<td>1.37</td>
<td>5.79</td>
<td>0.10</td>
<td>4.22</td>
</tr>
<tr>
<td>ADCensus [94]</td>
<td>7.3</td>
<td>1.07</td>
<td>1.48</td>
<td>5.73</td>
<td>0.19</td>
<td>4.10</td>
</tr>
<tr>
<td>SurfaceStereo [79]</td>
<td>18.2</td>
<td>1.28</td>
<td>1.65</td>
<td>6.78</td>
<td>0.19</td>
<td>3.12</td>
</tr>
<tr>
<td>GC+SegmBorder [57]</td>
<td>27.1</td>
<td>1.47</td>
<td>1.82</td>
<td>7.86</td>
<td>0.19</td>
<td>4.25</td>
</tr>
<tr>
<td>WarpMat [55]</td>
<td>20.8</td>
<td>1.16</td>
<td>1.35</td>
<td>6.04</td>
<td>0.18</td>
<td>5.02</td>
</tr>
<tr>
<td>RDP [102]</td>
<td>12.5</td>
<td>0.97</td>
<td>1.39</td>
<td>5.00</td>
<td>0.21</td>
<td>4.84</td>
</tr>
<tr>
<td>RVbased [116]</td>
<td>11.6</td>
<td>0.95</td>
<td>1.42</td>
<td>4.98</td>
<td>0.11</td>
<td>5.98</td>
</tr>
<tr>
<td>OutlierConf [42]</td>
<td>12.9</td>
<td>0.88</td>
<td>1.43</td>
<td>4.74</td>
<td>0.18</td>
<td>5.01</td>
</tr>
</tbody>
</table>
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
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%

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

KITTI, Errors: 46.3%
## Novel Challenges

**So what is the difference?**

<table>
<thead>
<tr>
<th>Middlebury</th>
<th>KITTI</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="middlebury1.png" alt="Image" /></td>
<td><img src="kitti.png" alt="Image" /></td>
</tr>
<tr>
<td><img src="middlebury2.png" alt="Image" /></td>
<td><img src="kitti2.png" alt="Image" /></td>
</tr>
</tbody>
</table>

- **Laboratory**
  - Lambertian
- **Moving vehicle**
  - Specularities
Novel Challenges

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

KITTI
- Moving vehicle
- Specularities
- Sensor saturation
- Large label set
- Strong slants
Novel Challenges

So what is the difference?

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

KITTI
- Moving vehicle
- Specularities
- Sensor saturation
- Large label set
- Strong slants
## Stereo Evaluation

<table>
<thead>
<tr>
<th>Rank</th>
<th>Method</th>
<th>Setting</th>
<th>Out-Noc</th>
<th>Out-All</th>
<th>Avg-Noc</th>
<th>Avg-All</th>
<th>Density</th>
<th>Runtime</th>
<th>Environment</th>
</tr>
</thead>
<tbody>
<tr>
<td>1</td>
<td>PCBP</td>
<td>4.13 %</td>
<td>5.45 %</td>
<td>0.9 px</td>
<td>1.2 px</td>
<td>100.00 %</td>
<td>5 min</td>
<td>4 cores @ 2.5 Ghz (Matlab + C/C++)</td>
<td></td>
</tr>
<tr>
<td>2</td>
<td>ISCM</td>
<td>5.16 %</td>
<td>7.19 %</td>
<td>1.2 px</td>
<td>2.1 px</td>
<td>94.70 %</td>
<td>8 s</td>
<td>2 cores @ 2.5 Ghz (C/C++)</td>
<td></td>
</tr>
<tr>
<td>3</td>
<td>SGM</td>
<td>5.83 %</td>
<td>7.08 %</td>
<td>1.2 px</td>
<td>1.3 px</td>
<td>85.80 %</td>
<td>3.7 s</td>
<td>1 core @ 3.0 Ghz (C/C++)</td>
<td></td>
</tr>
<tr>
<td>4</td>
<td>SNCC</td>
<td>6.27 %</td>
<td>7.33 %</td>
<td>1.4 px</td>
<td>1.5 px</td>
<td>100.00 %</td>
<td>0.27 s</td>
<td>1 core @ 3.0 Ghz (C/C++)</td>
<td></td>
</tr>
<tr>
<td>5</td>
<td>ITGV</td>
<td>6.31 %</td>
<td>7.40 %</td>
<td>1.3 px</td>
<td>1.5 px</td>
<td>100.00 %</td>
<td>7 s</td>
<td>1 core @ 3.0 Ghz (Matlab + C/C++)</td>
<td></td>
</tr>
<tr>
<td>6</td>
<td>BSSM</td>
<td>7.50 %</td>
<td>8.89 %</td>
<td>1.4 px</td>
<td>1.6 px</td>
<td>94.87 %</td>
<td>20.7 s</td>
<td>1 core @ 3.5 Ghz (C/C++)</td>
<td></td>
</tr>
<tr>
<td>7</td>
<td>OCV-SGBM</td>
<td>7.64 %</td>
<td>9.13 %</td>
<td>1.8 px</td>
<td>2.0 px</td>
<td>86.50 %</td>
<td>1.1 s</td>
<td>1 core @ 2.5 Ghz (C/C++)</td>
<td></td>
</tr>
<tr>
<td>8</td>
<td>ELAS</td>
<td>8.24 %</td>
<td>9.95 %</td>
<td>1.4 px</td>
<td>1.6 px</td>
<td>94.55 %</td>
<td>0.3 s</td>
<td>1 core @ 2.5 Ghz (C/C++)</td>
<td></td>
</tr>
<tr>
<td>9</td>
<td>MS-DSI</td>
<td>10.68 %</td>
<td>12.11 %</td>
<td>1.9 px</td>
<td>2.2 px</td>
<td>100.00 %</td>
<td>10.3 s</td>
<td>&gt;8 cores @ 2.5 Ghz (C/C++)</td>
<td></td>
</tr>
<tr>
<td>10</td>
<td>SDM</td>
<td>10.98 %</td>
<td>12.19 %</td>
<td>2.0 px</td>
<td>2.3 px</td>
<td>63.58 %</td>
<td>1 min</td>
<td>1 core @ 2.5 Ghz (C/C++)</td>
<td></td>
</tr>
<tr>
<td>11</td>
<td>GCSF</td>
<td>12.06 %</td>
<td>13.26 %</td>
<td>1.9 px</td>
<td>2.1 px</td>
<td>60.77 %</td>
<td>2.4 s</td>
<td>1 core @ 2.5 Ghz (C/C++)</td>
<td></td>
</tr>
<tr>
<td>12</td>
<td>GCS</td>
<td>13.37 %</td>
<td>14.54 %</td>
<td>2.1 px</td>
<td>2.3 px</td>
<td>51.06 %</td>
<td>2.2 s</td>
<td>1 core @ 2.5 Ghz (C/C++)</td>
<td></td>
</tr>
<tr>
<td>13</td>
<td>CostFilter</td>
<td>19.96 %</td>
<td>21.05 %</td>
<td>5.0 px</td>
<td>5.4 px</td>
<td>100.00 %</td>
<td>4 min</td>
<td>1 core @ 2.5 Ghz (Matlab)</td>
<td></td>
</tr>
<tr>
<td>14</td>
<td>OCVBM</td>
<td>25.39 %</td>
<td>26.72 %</td>
<td>7.6 px</td>
<td>7.9 px</td>
<td>55.84 %</td>
<td>0.1 s</td>
<td>1 core @ 2.5 Ghz (C/C++)</td>
<td></td>
</tr>
<tr>
<td>15</td>
<td>GC-occ</td>
<td>33.50 %</td>
<td>34.74 %</td>
<td>8.6 px</td>
<td>9.2 px</td>
<td>87.57 %</td>
<td>6 min</td>
<td>1 core @ 2.5 Ghz (C/C++)</td>
<td></td>
</tr>
</tbody>
</table>


Anonymous submission


Anonymous submission


Christoph Rhemann, Asmaa Houri, Michael Bleyer, Carsten Rother and Margrit Gelautz. **Fast Cost Volume Filtering for Visual Correspondence and Beyond**, CVPR 2011.


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 composed of small frontal/slanted planes
Plane MRFs

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

\[
E(x_1, \ldots, x_n) = \sum_i C(x_i) + \sum_i \sum_{j \in \mathcal{N}_j} C(x_i, x_j)
\]

with \(x_i \in \mathbb{R}\) for the fronto-parallel planes, and \(x_i \in \mathbb{R}^3\) for the slanted planes
Plane MRFs

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

\[ E(x_1, \ldots, x_n) = \sum_i C(x_i) + \sum_i \sum_{j \in N_j} C(x_i, x_j) \]

with \( x_i \in \mathbb{R} \) for the fronto-parallel planes, and \( x_i \in \mathbb{R}^3 \) 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 composed of small frontal/slanted planes
- Good representation if the superpixels are small and respect boundaries

\[
E(x_1, \cdots, x_n) = \sum_i C(x_i) + \sum_i \sum_{j \in N_j} C(x_i, x_j)
\]

with \( x_i \in \mathbb{R} \) for the fronto-parallel planes, and \( x_i \in \mathbb{R}^3 \) 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 composed of small frontal/slanted planes
- Good representation if the superpixels are small and respect boundaries

\[ E(x_1, \cdots, x_n) = \sum_i C(x_i) + \sum_i \sum_{j \in N_j} C(x_i, x_j) \]

with \( x_i \in \mathbb{R} \) for the fronto-parallel planes, and \( x_i \in \mathbb{R}^3 \) 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 aggregation 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 composed of small frontal/slanted planes
- Good representation if the superpixels are small and respect boundaries

\[ E(x_1, \ldots, x_n) = \sum_i C(x_i) + \sum_i \sum_{j \in N_j} C(x_i, x_j) \]

with \( x_i \in \mathbb{R} \) for the fronto-parallel planes, and \( x_i \in \mathbb{R}^3 \) 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 aggregation 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 composed of small frontal/slanted planes
- Good representation if the superpixels are small and respect boundaries

\[
E(x_1, \cdots, x_n) = \sum_i C(x_i) + \sum_i \sum_{j \in N_j} C(x_i, x_j)
\]

with \( x_i \in \mathbb{R} \) for the fronto-parallel planes, and \( x_i \in \mathbb{R}^3 \) 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 aggregation 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

Takes as input disparities computed by any local algorithm

**Segment variable** \( y_i = (\alpha_i, \beta_i, \gamma_i) \)

Slanted 3D plane of segment

**Boundary variable** \( o_{i,j} \)

Relationship between segments

4 states

- Occlusion
- Hinge
- Coplanar

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

Raquel Urtasun (TTI-C)
Energy of PCBP-Stereo

- \( y \) the set of slanted 3D planes, \( o \) the set of discrete boundary variables

\[
E(y, o) = E_{\text{color}}(o) + E_{\text{match}}(y, o) + E_{\text{compatibility}}(y, o) + E_{\text{junction}}(o)
\]
Energy of PCBP-Stereo

- \( y \) the set of slanted 3D planes, \( o \) the set of discrete boundary variables

\[
E(y, o) = E_{\text{color}}(o) + E_{\text{match}}(y, o) + E_{\text{compatibility}}(y, o) + E_{\text{junction}}(o)
\]

Agreement with result of input disparity map

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

Truncated quadratic function

\[
\phi_i^{TP}(p, y_i, K) = \min \left( |D(p) - \hat{d}_i(p, y_i)|, K \right)^2
\]

On boundary

“Oclusion” – Foreground segment owns boundary
Energy of PCBP-Stereo

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

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

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

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

(2) Boundary labels match Slanted planes

- "Occlusion" $\hat{d}_{\text{front}}(p) > \hat{d}_{\text{back}}(p)$
- "Hinge" $\hat{d}_i(p) = \hat{d}_j(p)$ on boundary
- "Coplanar" $\hat{d}_i(p) = \hat{d}_j(p)$ in both segments
Energy of PCBP-Stereo

- \( y \) the set of slanted 3D planes, \( o \) the set of discrete boundary variables

\[
E(y, o) = E_{\text{color}}(o) + E_{\text{match}}(y, o) + E_{\text{compatibility}}(y, o) + E_{\text{junction}}(o)
\]

Occlusion boundary reasoning [Malik 1987]
Penalize impossible junctions

Impossible cases

Front
Back
Occlusion
Hinge
Coplanar