Energy, Plane-based Stereo and Tracking

Raquel Urtasun

TTI Chicago

March 5, 2013
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 \( \mathcal{P} \).
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 $l \neq \alpha, \beta$. 

\[\text{Raquel Urtasun (TTI-C)}\] 
\[\text{Computer Vision}\] 
\[\text{March 5, 2013 2 / 66}\]
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.
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.

- 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 \( P = \{ P_l | l \in \mathcal{L} \} \), 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 \( \mathcal{P} \).

- 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 \( \mathcal{P}_\alpha \subset \mathcal{P}'_\alpha \) 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 \( P = \{ \mathcal{P}_l | l \in \mathcal{L} \} \), 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 \( \alpha \).
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 $\{\alpha, \beta\} \subset \mathcal{L}$
   3.1. Find $\hat{f} = \arg\min E(f')$ 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 $f$

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 $\alpha$-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 $f$
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 $\mathcal{P}$) 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 $\alpha, \beta$. 
Graph Construction

- The set of vertices includes the two terminals $\alpha$ and $\beta$, 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 $\alpha$ and $\beta$, called $t$-links.
- Each set of pixels $p, q \in \mathcal{P}_{\alpha\beta}$ which are neighbors is connected by an edge $e_{p,q}$

<table>
<thead>
<tr>
<th>edge</th>
<th>weight</th>
<th>for</th>
</tr>
</thead>
<tbody>
<tr>
<td>$t^\alpha_p$</td>
<td>$D_p(\alpha) + \sum_{q \in \mathcal{N}<em>p, q \notin \mathcal{P}</em>{\alpha\beta}} V(\alpha, f_q)$</td>
<td>$p \in \mathcal{P}_{\alpha\beta}$</td>
</tr>
<tr>
<td>$t^\beta_p$</td>
<td>$D_p(\beta) + \sum_{q \in \mathcal{N}<em>p, q \notin \mathcal{P}</em>{\alpha\beta}} V(\beta, f_q)$</td>
<td>$p \in \mathcal{P}_{\alpha\beta}$</td>
</tr>
<tr>
<td>$e_{{p,q}}$</td>
<td>$V(\alpha, \beta)$</td>
<td>${p,q} \in \mathcal{N}$</td>
</tr>
</tbody>
</table>
Computing the Cut

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

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

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

For any cut, then

(a) If $t_p^\alpha, t_q^\alpha \in \mathcal{C}$ then $e_{\{p,q\}} \not\in \mathcal{C}$.
(b) If $t_p^\beta, t_q^\beta \in \mathcal{C}$ then $e_{\{p,q\}} \not\in \mathcal{C}$.
(c) If $t_p^\beta, t_q^\alpha \in \mathcal{C}$ then $e_{\{p,q\}} \in \mathcal{C}$.
(d) If $t_p^\alpha, t_q^\beta \in \mathcal{C}$ then $e_{\{p,q\}} \in \mathcal{C}$. 

\[ \]
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 $G_{\alpha} = (\mathcal{V}_{\alpha}, \mathcal{E}_{\alpha})$. 
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 $\alpha$. 
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.
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.
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 $a_{p,q}$. 
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.
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.
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 $\{e_{p,a}, e_{a,q}, t_{\bar{\alpha}}\}$. 
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 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 edges is then

\[
\mathcal{E}_\alpha = \left\{ \bigcup_{p \in \mathcal{P}} \{t_p^\alpha, t_p^\bar{\alpha}\}, \bigcup_{(p,q) \in \mathcal{N}} \mathcal{E}_{\{p,q\}} \right\}
\]
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

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

<table>
<thead>
<tr>
<th>edge</th>
<th>weight</th>
<th>for</th>
</tr>
</thead>
<tbody>
<tr>
<td>$e_{p,q}$</td>
<td>$i^\alpha_p$</td>
<td>$\infty$ $p \in \mathcal{P}_\alpha$</td>
</tr>
<tr>
<td>$e_{p,q}$</td>
<td>$D_p(f_p)$</td>
<td>$p \notin \mathcal{P}_\alpha$</td>
</tr>
<tr>
<td>$e_{p,q}$</td>
<td>$D_p(\alpha)$</td>
<td>$p \in \mathcal{P}$</td>
</tr>
<tr>
<td>$e_{p,q}$</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>$e_{p,q}$</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>
There is a one-to-one correspondences between a cut and a labeling.

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

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) \( t_p^\alpha, t_q^\alpha \in C \) then \( C \cap E_{\{p,q\}} = \emptyset \).

(b) \( t_p^\alpha, t_q^\alpha \in C \) then \( C \cap E_{\{p,q\}} = t_a^\alpha \).

(c) \( t_p^\alpha, t_q^\alpha \in C \) then \( C \cap E_{\{p,q\}} = e_{\{p,a\}} \).

(d) \( t_p^\alpha, t_q^\alpha \in C \) then \( C \cap E_{\{p,q\}} = e_{\{a,q\}} \).
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/)
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!
Novel Challenges

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

**Middlebury, Errors: 2.7%**

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

Middlebury
- Laboratory
- Lambertian

KITTI
- 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
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>
<th>Compare</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>
<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>
<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>
<td></td>
</tr>
<tr>
<td>4</td>
<td>SNCC</td>
<td>6.27 %</td>
<td>7.23 %</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>
<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>
<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>
<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>
<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>
<td></td>
</tr>
<tr>
<td>9</td>
<td>MS-D31</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>
<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>
<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>
<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>
<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>
<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>
<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>
<td></td>
</tr>
</tbody>
</table>

Raquel Urtasun (TTI-C)  | Computer Vision  | March 5, 2013 22 / 66
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

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

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 aggregates 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
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

- This are continuous variables. Is this a problem?
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

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 is a continuous variable. 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

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

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
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

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

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

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

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

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

 Agreement with result of input disparity map

Truncated quadratic function $\phi_{i}^{TP}(\mathbf{p}, \mathbf{y}_i, K) = \min \left( |D(\mathbf{p}) - \hat{d}_i(\mathbf{p}, \mathbf{y}_i)|, K \right)^2$

On boundary

“Occlusion” – Foreground segment owns boundary

Computed by any matching method (Modified semi-global matching)
Energy of PCBP-Stereo

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

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

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

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

(2) Boundary labels match Slanted planes

- "Occlusion" $\hat{d}_{front}(p) > \hat{d}_{back}(p)$
- "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**

1. \[ \text{Front} \quad \text{Back} \quad \text{Occlusion} \]
2. \[ \text{Hinge} \quad \text{Coplanar} \]
Stereo Evaluation

[K. Yamaguchi, T. Hazan, D. McAllester and R. Urtasun, ECCV12]

Easy Scenarios:

- Natural scenes, lots of texture, no objects
- A couple of errors at thin structures (poles)

Errors: < 0.5%
Easy Scenarios:

- Shadows help the disambiguation process
- Errors at thin structures and far away textureless regions
Hard Scenarios:

- Textureless or saturated areas
- Ambiguous reflections

Errors: 22.1%  
Errors: 17.4%
Stereo Evaluation

[K. Yamaguchi, T. Hazan, D. McAllester and R. Urtasun, ECCV12]

Hard Scenarios:

- Depth discontinuities / complicated geometries

Errors: 11.2%

Errors: 10.5%
A different view on tracking
**Goal**: Given a set of detections in video, link the detections into tracks

Discover which detections are of the same object, and how many objects there are.
Problem: Given a set of detections in video, link the detections into tracks

Discover which detections are of the same object, and how many objects there are

This can be solved optimally as a network flow problem, with non-overlapping constraints in trajectories
Problem: Given a set of detections in video, link the detections into tracks

Discover which detections are of the same object, and how many objects there are

This can be solved optimally as a network flow problem, with non-overlapping constraints in trajectories

The optimal data association is found by a min-cost flow algorithm in the network
Problem: Given a set of detections in video, link the detections into tracks

Discover which detections are of the same object, and how many objects there are

This can be solved optimally as a network flow problem, with non-overlapping constraints in trajectories

The optimal data association is found by a min-cost flow algorithm in the network
Notation and Problem Definition

- Let $\mathcal{X} = \{x_i\}$ be a set of object observations.
- Each $x_i$ is a detection response $x_i = (x_i, s_i, a_i, t_i)$, where $x_i$ is the position, $s_i$ is the scale, $a_i$ is the appearance, and $t_i$ is the time step (frame index).

A single trajectory hypothesis is defined as an ordered list of object observations, $T_k = \{x_{k1}, \ldots, x_{kl}\}$, with $x_{ki} \in \mathcal{X}$.

An association hypothesis $T$ is defined as a set of single trajectory hypotheses, $T = \{T_k\}$.

The association is given by:

$$T^* = \arg\max_T P(T|\mathcal{X}) = \arg\max_T P(\mathcal{X}|T) P(T)$$

We have assumed that the likelihood probabilities are conditionally independent given $T$. 

Raquel Urtasun (TTI-C)
Notation and Problem Definition

- Let $\mathcal{X} = \{x_i\}$ be a set of object observations
- Each $x_i$ is detection response $x_i = (x_i, s_i, a_i, t_i)$, where $x_i$ is the position, $s_i$ is the scale, $a_i$ is the appearance and $t_i$ is the time step (frame index)
- A single trajectory hypothesis is defined as an ordered list of object observations, $T_k = \{x_{k_1}, \cdots, x_{k_{l_k}}\}$, with $x_{k_i} \in \mathcal{X}$
Let $\mathcal{X} = \{x_i\}$ be a set of object observations.

Each $x_i$ is a detection response $x_i = (x_i, s_i, a_i, t_i)$, where $x_i$ is the position, $s_i$ is the scale, $a_i$ is the appearance, and $t_i$ is the time step (frame index).

A single trajectory hypothesis is defined as an ordered list of object observations, $T_k = \{x_{k1}, \ldots, x_{kl}\}$, with $x_{ki} \in \mathcal{X}$.

An association hypothesis $\mathcal{T}$ is defined as a set of single trajectory hypotheses, $\mathcal{T} = \{T_k\}$.
Notation and Problem Definition

- Let \( \mathcal{X} = \{x_i\} \) be a set of object observations.
- Each \( x_i \) is a detection response \( x_i = (x_i, s_i, a_i, t_i) \), where \( x_i \) is the position, \( s_i \) is the scale, \( a_i \) is the appearance, and \( t_i \) is the time step (frame index).
- A single trajectory hypothesis is defined as an ordered list of object observations, \( T_k = \{x_{k_1}, \cdots, x_{k_{k_l}}\} \), with \( x_{k_i} \in \mathcal{X} \).
- An association hypothesis \( T \) is defined as a set of single trajectory hypotheses, \( T = \{T_k\} \).
- The association is given by:
  \[
  T^* = \arg \max_T P(T|\mathcal{X}) \\
  = \arg \max_T P(\mathcal{X}|T)P(T) \\
  = \arg \max_T \prod_i P(x_i|T)P(T)
  \]
Notation and Problem Definition

- Let $\mathcal{X} = \{x_i\}$ be a set of object observations.
- Each $x_i$ is a detection response $x_i = (x_i, s_i, a_i, t_i)$, where $x_i$ is the position, $s_i$ is the scale, $a_i$ is the appearance, and $t_i$ is the time step (frame index).
- A single trajectory hypothesis is defined as an ordered list of object observations, $T_k = \{x_{k_1}, \ldots, x_{k_{lk}}\}$, with $x_{k_i} \in \mathcal{X}$.
- An association hypothesis $T$ is defined as a set of single trajectory hypotheses, $\mathcal{T} = \{T_k\}$.
- The association is given by
  \[
  T^* = \arg \max_T P(T|\mathcal{X}) \approx \arg \max_T P(\mathcal{X}|T)P(T) = \arg \max_T \prod_i P(x_i|T)P(T)
  \]
- We have assumed that the likelihood probs. are conditionally independent given $T$. 

Raquel Urtasun (TTI-C)  
Computer Vision  
March 5, 2013 38 / 66
Notation and Problem Definition

- Let \( \mathcal{X} = \{x_i\} \) be a set of object observations.
- Each \( x_i \) is detection response \( x_i = (x_i, s_i, a_i, t_i) \), where \( x_i \) is the position, \( s_i \) is the scale, \( a_i \) is the appearance and \( t_i \) is the time step (frame index).
- A single trajectory hypothesis is defined as an ordered list of object observations, \( T_k = \{x_{k1}, \cdots, x_{k_l}\} \), with \( x_{ki} \in \mathcal{X} \).
- An association hypothesis \( T \) is defined as a set of single trajectory hypotheses, \( \mathcal{T} = \{T_k\} \).
- The association is given by
  \[
  \mathcal{T}^* = \arg \max_{\mathcal{T}} P(\mathcal{T}|\mathcal{X}) \\
  = \arg \max_{\mathcal{T}} P(\mathcal{X}|\mathcal{T})P(\mathcal{T}) \\
  = \arg \max_{\mathcal{T}} \prod_i P(x_i|\mathcal{T})P(\mathcal{T})
  \]
- We have assumed that the likelihood prob. are conditionally independent given \( \mathcal{T} \).
We want to solve the following optimization

\[ T^* = \arg \max_T \prod_i P(x_i|T)P(T) \]

The space \( T \) is very large, so difficult to optimize.

There is one more constraint: one object can only belong to one trajectory.

\[ T_k \cap T_l = \emptyset, \quad \forall k \neq l \]

If we assume that the motion of each object is independent

\[ T^* = \arg \max_T \prod_i P(x_i|T)P(T) \]

s.t.

\[ T_k \cap T_l = \emptyset, \quad \forall k \neq l \]

When is this assumption not good?
Optimization problem

- We want to solve the following optimization

\[ \mathcal{T}^* = \arg \max_{\mathcal{T}} \prod_i P(x_i|\mathcal{T}) P(\mathcal{T}) \]

- The space \( \mathcal{T} \) is very large, so difficult to optimize

- There is one more constraint: one object can only belong to one trajectory.

\[ \mathcal{T}_k \cap \mathcal{T}_l = \emptyset, \quad \forall k \neq l \]
Optimization problem

- We want to solve the following optimization

\[
\mathcal{T}^* = \arg \max \mathcal{T} \prod_i P(x_i | \mathcal{T}) P(\mathcal{T})
\]

- The space \( \mathcal{T} \) is very large, so difficult to optimize

- There is one more constraint: one object can only belong to one trajectory.

\[
\mathcal{T}_k \cap \mathcal{T}_l = \emptyset, \quad \forall k \neq l
\]

- If we assume that the motion of each object is independent

\[
\mathcal{T}^* = \arg \max \mathcal{T} \prod_i P(x_i | \mathcal{T}) \prod_{\mathcal{T}_k \in \mathcal{T}} P(\mathcal{T}_k)
\]

\[
s.t. \quad \mathcal{T}_k \cap \mathcal{T}_l = \emptyset, \quad \forall k \neq l
\]
Optimization problem

- We want to solve the following optimization

\[ T^* = \arg \max_T \prod_i P(x_i|T) P(T) \]

- The space \( T \) is very large, so difficult to optimize
- There is one more constraint: one object can only belong to one trajectory.

\[ T_k \cap T_l = \emptyset, \quad \forall k \neq l \]

- If we assume that the motion of each object is independent

\[ T^* = \arg \max_T \prod_i P(x_i|T) \prod_{T_k \in T} P(T_k) \]

\[ \text{s.t.} \quad T_k \cap T_l = \emptyset, \quad \forall k \neq l \]

- When is this assumption not good?
Optimization problem

- We want to solve the following optimization

\[
\mathcal{T}^* = \arg \max_{\mathcal{T}} \prod_i P(x_i | \mathcal{T}) P(\mathcal{T})
\]

- The space \( \mathcal{T} \) is very large, so difficult to optimize

- There is one more constraint: one object can only belong to one trajectory.

\[
\mathcal{T}_k \cap \mathcal{T}_l = \emptyset, \quad \forall k \neq l
\]

- If we assume that the motion of each object is independent

\[
\mathcal{T}^* = \arg \max_{\mathcal{T}} \prod_i P(x_i | \mathcal{T}) \prod_{\mathcal{T}_k \in \mathcal{T}} P(\mathcal{T}_k)
\]

\[
s.t. \quad \mathcal{T}_k \cap \mathcal{T}_l = \emptyset, \quad \forall k \neq l
\]

- When is this assumption not good?
Problem Formulation

\[ T^* = \arg \max_T \prod_i P(x_i|T) \prod_{T_k \in T} P(T_k) \]

s.t. \[ T_k \cap T_l = \emptyset, \forall k \neq l \]

- \( P(x_i|T) \) is the **likelihood** of observation \( x_i \). We can use a Bernoulli distribution for example to represent being an inlier or outlier

\[
P(x_i|T) = \begin{cases} 
1 - \beta_i & \text{if } \exists T_k \in T, x_i \in T_k \\
\beta_i & \text{otherwise.}
\end{cases}
\]
Problem Formulation

\[ T^* = \arg \max_T \prod_i P(x_i|T) \prod_{T_k \in T} P(T_k) \]
\[ \text{s.t. } T_k \cap T_l = \emptyset, \forall k \neq l \]

- \( P(x_i|T) \) is the **likelihood** of observation \( x_i \). We can use a Bernoulli distribution for example to represent being an inlier or outlier

\[
P(x_i|T) = \begin{cases} 
1 - \beta_i & \text{if } \exists T_k \in T, x_i \in T_k \\
\beta_i & \text{otherwise.}
\end{cases}
\]

- \( P(T_k) \) can be modeled as a Markov chain, with initialization probability \( P_{\text{ent}} \), termination probability \( P_{\text{exit}} \), and transition probability \( P_{\text{link}}(x_{k_i+1}|x_{k_i}) \)

\[
P(T_k) = P(\{x_{k_0}, \cdots, x_{k_{l_k}}\}) = P_{\text{ent}}(x_{k_0}) p_{\text{link}}(x_{k_1}|x_{k_0}) \cdots p_{\text{link}}(x_{k_{l_k}}|x_{k_{l_k}}) p_{\text{exit}}(x_{k_{l_k}})
\]
\[ T^* = \arg \max_T \prod_i P(x_i|T) \prod_{T_k \in T} P(T_k) \]

s.t. \( T_k \cap T_l = \emptyset, \forall k \neq l \)

- \( P(x_i|T) \) is the **likelihood** of observation \( x_i \). We can use a Bernoulli distribution for example to represent being an inlier or outlier

\[
P(x_i|T) = \begin{cases} 
1 - \beta_i & \text{ if } \exists T_k \in T, x_i \in T_k \\
\beta_i & \text{ otherwise.}
\end{cases}
\]

- \( P(T_k) \) can be modeled as a Markov chain, with initialization probability \( P_{ent} \), termination probability \( P_{exit} \), and transition probability \( P_{link}(x_{k_{i+1}}|x_{k_i}) \)

\[
P(T_k) = P(\{x_{k_0}, \cdots, x_{k_l}\})
= P_{ent}(x_{k_0})p_{link}(x_{k_1}|x_{k_0}) \cdots p_{link}(x_{k_l}|x_{k_{l-1}})p_{exit}(x_{k_l})
\]

- \( P(x_i|T) \) allows for selecting observations, rather than assume all the inputs to be true detections, without additional processing to remove false trajectories after association.
Problem Formulation

\[ T^* = \arg \max_T \prod_i P(x_i|T) \prod_{T_k \in T} P(T_k) \]

s.t. \( T_k \cap T_l = \emptyset, \ \forall k \neq l \)

- \( P(x_i|T) \) is the **likelihood** of observation \( x_i \). We can use a Bernoulli distribution for example to represent being an inlier or outlier

\[
P(x_i|T) = \begin{cases} 
1 - \beta_i & \text{if } \exists T_k \in T, x_i \in T_k \\
\beta_i & \text{otherwise.}
\end{cases}
\]

- \( P(T_k) \) can be modeled as a Markov chain, with initialization probability \( P_{\text{ent}} \), termination probability \( P_{\text{exit}} \), and transition probability \( P_{\text{link}}(x_{k_{i+1}}|x_{k_i}) \)

\[
P(T_k) = P(\{x_{k_0}, \ldots, x_{k_{l_k}}\}) = P_{\text{ent}}(x_{k_0}) p_{\text{link}}(x_{k_1}|x_{k_0}) \cdots p_{\text{link}}(x_{k_{l_k}}|x_{k_{l_k}}) p_{\text{exit}}(x_{k_{l_k}})
\]

- \( P(x_i|T) \) allows for selecting observations, rather than assume all the inputs to be true detections, without additional processing to remove false trajectories after association.
Useful definitions

- To couple the non-overlap constraints with the objective function we define 0-1 indicator variables

\[
\begin{align*}
  f_{en,i} & = \begin{cases} 
  1 & \text{if } \exists T_k \in \mathcal{T}, T_k \text{ starts from } x_i \\
  0 & \text{otherwise.} 
  \end{cases} \\
  f_{ex,i} & = \begin{cases} 
  1 & \text{if } \exists T_k \in \mathcal{T}, T_k \text{ ends at } x_i \\
  0 & \text{otherwise.} 
  \end{cases} \\
  f_{i,j} & = \begin{cases} 
  1 & \text{if } \exists T_k \in \mathcal{T}, x_j \text{ is after } x_i \text{ in } T_k \\
  0 & \text{otherwise.} 
  \end{cases} \\
  f_i & = \begin{cases} 
  1 & \text{if } \exists T_k \in \mathcal{T}, x_i \in T_k \\
  0 & \text{otherwise.} 
  \end{cases}
\end{align*}
\]

- \(\mathcal{T}\) is non-overlap if and only if

\[
f_{en,i} + \sum_j f_{j,i} = f_i = f_{ex,i} + \sum_j f_{i,j} \quad \forall i
\]
Useful definitions

- To couple the non-overlap constraints with the objective function we define 0-1 indicator variables

\[
\begin{align*}
\text{f}_{en,i} &= \begin{cases} 
1 & \text{if } \exists T_k \in T, T_k \text{ starts from } x_i \\
0 & \text{otherwise.}
\end{cases} \\
\text{f}_{ex,i} &= \begin{cases} 
1 & \text{if } \exists T_k \in T, T_k \text{ ends at } x_i \\
0 & \text{otherwise.}
\end{cases} \\
\text{f}_{i,j} &= \begin{cases} 
1 & \text{if } \exists T_k \in T, x_j \text{ is after } x_i \text{ in } T_k \\
0 & \text{otherwise.}
\end{cases} \\
\text{f}_i &= \begin{cases} 
1 & \text{if } \exists T_k \in T, x_i \in T_k \\
0 & \text{otherwise.}
\end{cases}
\end{align*}
\]

- $T$ is non-overlap if and only if

\[
f_{en,i} + \sum_j f_{j,i} = f_i = f_{ex,i} + \sum_j f_{i,j} \quad \forall i
\]
Min-cost flow problem

- We have the optimization problem

\[
\min_T - \sum_{T_k \in T} \log P(T_k) - \sum_i \log p(x_i|T)
\]

- This can be obtained as

\[
\min_T \sum_{T_k \in T} \left( C_{en,k_0} f_{en,k_0} + \sum_j C_{k_j,k_{j+1}} f_{k_j,k_{j+1}} + C_{ex,k_{l_k}} f_{ex,k_{l_k}} \right) + \\
+ \sum_i \left( -\log(1 - \beta_i) f_i - \log \beta_i (1 - f_i) \right)
\]

s.t. \( f_{en,i} + \sum_j f_{j,i} = f_i = f_{ex,i} + \sum_j f_{i,j} \quad \forall i \)
Min-cost flow problem

- We have the optimization problem

\[
\min_T - \sum_{T_k \in T} \log P(T_k) - \sum_i \log p(x_i|T)
\]

- This can be obtained as

\[
\min_T \sum_{T_k \in T} \left( C_{en,k_0} f_{en,k_0} + \sum_j C_{k_j,k_{j+1}} f_{k_j,k_{j+1}} + C_{ex,k_l} f_{ex,k_l} \right) + \\
+ \sum_i \left( -\log(1 - \beta_i)f_i - \log\beta_i(1 - f_i) \right)
\]

\[
s.t. \quad f_{en,i} + \sum_j f_{j,i} = f_i = f_{ex,i} + \sum_j f_{i,j} \quad \forall i
\]

- Which can be reformulated as

\[
\min_T \sum_i C_{en,i} f_{en,i} + \sum_{i,j} C_{i,j} f_{i,j} + \sum_i C_{ex,i} f_{ex,i} + \sum_i C_{i} f_{i}
\]

\[
s.t. \quad f_{en,i} + \sum_j f_{j,i} = f_i = f_{ex,i} + \sum_j f_{i,j} \quad \forall i
\]
Min-cost flow problem

- We have the optimization problem

\[
\min_{\mathcal{T}} - \sum_{T_k \in \mathcal{T}} \log P(T_k) - \sum_{i} \log p(x_i | \mathcal{T})
\]

- This can be obtained as

\[
\min_{\mathcal{T}} \sum_{T_k \in \mathcal{T}} \left( C_{en,k_0} f_{en,k_0} + \sum_{j} C_{k_j,k_{j+1}} f_{k_j,k_{j+1}} + C_{ex,k_{l_k}} f_{ex,k_{l_k}} \right) + \sum_{i} ( - \log(1 - \beta_i) f_i - \log \beta_i (1 - f_i) )
\]

\[
s.t. \quad f_{en,i} + \sum_{j} f_{j,i} = f_i = f_{ex,i} + \sum_{j} f_{i,j} \quad \forall i
\]

- Which can be reformulated as

\[
\min_{\mathcal{T}} \sum_{i} C_{en,i} f_{en,i} + \sum_{i,j} C_{i,j} f_{i,j} + \sum_{i} C_{ex,i} f_{ex,i} + \sum_{i} C_{i} f_{i}
\]

\[
s.t. \quad f_{en,i} + \sum_{j} f_{j,i} = f_i = f_{ex,i} + \sum_{j} f_{i,j} \quad \forall i
\]

- What are the relationships between the costs and the probabilities we had before?
We have the optimization problem

$$\min_{\mathcal{T}} - \sum_{T_k \in \mathcal{T}} \log P(T_k) - \sum_i \log p(x_i | \mathcal{T})$$

This can be obtained as

$$\min_{\mathcal{T}} \sum_{T_k \in \mathcal{T}} \left( C_{en,k_0} f_{en,k_0} + \sum_j C_{k_j,k_{j+1}} f_{k_j,k_{j+1}} + C_{ex,k_l} f_{ex,k_l} \right) +$$

$$+ \sum_i (-\log(1 - \beta_i)f_i - \log \beta_i(1 - \beta_i))$$

subject to

$$f_{en,i} + \sum_j f_{j,i} = f_i = f_{ex,i} + \sum_j f_{i,j} \quad \forall i$$

Which can be reformulated as

$$\min_{\mathcal{T}} \sum_i C_{en,i} f_{en,i} + \sum_{i,j} C_{i,j} f_{i,j} + \sum_i C_{ex,i} f_{ex,i} + \sum_i C_{i} f_{i}$$

subject to

$$f_{en,i} + \sum_j f_{j,i} = f_i = f_{ex,i} + \sum_j f_{i,j} \quad \forall i$$

What are the relationships between the costs and the probabilities we had before?
This can be mapped into a cost-flow network $G(\mathcal{X})$ with source $s$ and sink $t$

$$\min_{\mathcal{T}} \sum_i C_{en,i}f_{en,i} + \sum_{i,j} C_{i,j}f_{i,j} + \sum_i C_{ex,i}f_{ex,i} + \sum_i C_{i}f_{i}$$

subject to

$$f_{en,i} + \sum_j f_{j,i} = f_{i} = f_{ex,i} + \sum_j f_{i,j} \quad \forall i$$

For every observation $x_i \in \mathcal{X}$ create two nodes $u_i, v_i$, and an arc with cost $c(u_i, v_j) = C_i$ and flow $f_i$. 
This can be mapped into a cost-flow network $G(\mathcal{X})$ with source $s$ and sink $t$

$$\min_T \sum_i C_{en,i}f_{en,i} + \sum_{i,j} C_{i,j}f_{i,j} + \sum_{i} C_{ex,i}f_{ex,i} + \sum_{i} C_i f_i$$

s.t. $f_{en,i} + \sum_j f_{j,i} = f_i = f_{ex,i} + \sum_j f_{i,j} \quad \forall i$

For every observation $x_i \in \mathcal{X}$ create two nodes $u_i, v_i$, and an arc with cost $c(u_i, v_j) = C_{i}$ and flow $f_i$.

Add arcs $c(s, u_i) = C_{en,i}$ and flow $f_{en,i}$, as well as $c(t, u_i) = C_{ex,i}$ and flow $f_{ex,i}$.
Mapping to Min cost-flow network

- This can be mapped into a cost-flow network $G(\mathcal{X})$ with source $s$ and sink $t$
  \[
  \min_T \sum_i C_{en,i} f_{en,i} + \sum_{i,j} C_{i,j} f_{i,j} + \sum_i C_{ex,i} f_{ex,i} + \sum_i C_i f_i
  \]
  \[
  \text{s.t. } f_{en,i} + \sum_j f_{j,i} = f_i = f_{ex,i} + \sum_j f_{i,j} \quad \forall i
  \]

- For every observation $x_i \in \mathcal{X}$ create two nodes $u_i, v_i$, and an arc with cost $c(u_i, v_j) = C_i$ and flow $f_i$.

- Add arcs $c(s, u_i) = C_{en,i}$ and flow $f_{en,i}$, as well as $c(t, u_i) = C_{ex,i}$ and flow $f_{ex,i}$.

- For every transition $p_{link}(x_j|x_i) \neq 0$, create an arc with cost $c(v_i, u_j) = C_{i,j}$ and flow $f_{i,j}$. 
Mapping to Min cost-flow network

- This can be mapped into a cost-flow network $G(\mathcal{X})$ with source $s$ and sink $t$

$$\min_T\quad \sum_i C_{en,i}f_{en,i} + \sum_{i,j} C_{i,j}f_{i,j} + \sum_i C_{ex,i}f_{ex,i} + \sum_i C_i f_i$$

$$\text{s.t.} \quad f_{en,i} + \sum_j f_{j,i} = f_i = f_{ex,i} + \sum_j f_{i,j} \quad \forall i$$

- For every observation $x_i \in \mathcal{X}$ create two nodes $u_i, v_i$, and an arc with cost $c(u_i, v_j) = C_i$ and flow $f_i$.

- Add arcs $c(s, u_i) = C_{en,i}$ and flow $f_{en,i}$, as well as $c(t, u_i) = C_{ex,i}$ and flow $f_{ex,i}$

- For every transition $p_{\text{link}}(x_j|x_i) \neq 0$, create an arc with cost $c(v_i, u_j) = C_{i,j}$ and flow $f_{i,j}$.

- The constraint is equivalent to the flow conservation constraint
This can be mapped into a cost-flow network $G(\mathcal{X})$ with source $s$ and sink $t$

$$\min_T \sum_i C_{en,i} f_{en,i} + \sum_{i,j} C_{i,j} f_{i,j} + \sum_i C_{ex,i} f_{ex,i} + \sum_i C_i f_i$$

s.t. $f_{en,i} + \sum_j f_{j,i} = f_i = f_{ex,i} + \sum_j f_{i,j}$ $\forall i$

For every observation $x_i \in \mathcal{X}$ create two nodes $u_i, v_i$, and an arc with cost $c(u_i, v_j) = C_i$ and flow $f_i$.

Add arcs $c(s, u_i) = C_{en,i}$ and flow $f_{en,i}$, as well as $c(t, u_i) = C_{ex,i}$ and flow $f_{ex,i}$

For every transition $p_{\text{link}}(x_j | x_i) \neq 0$, create an arc with cost $c(v_i, u_j) = C_{i,j}$ and flow $f_{i,j}$.

The constraint is equivalent to the flow conservation constraint

The objective is the cost of the flow in $G$. 

Mapping to Min cost-flow network

This can be mapped into a cost-flow network $G(\mathcal{X})$ with source $s$ and sink $t$

$$\min_T \sum_i C_{en,i}f_{en,i} + \sum_{i,j} C_{i,j}f_{i,j} + \sum_i C_{ex,i}f_{ex,i} + \sum_i C_i f_i$$

s.t. \hspace{1cm} f_{en,i} + \sum_j f_{j,i} = f_i = f_{ex,i} + \sum_j f_{i,j} \quad \forall i

For every observation $x_i \in \mathcal{X}$ create two nodes $u_i, v_i$, and an arc with cost $c(u_i, v_j) = C_i$ and flow $f_i$.

Add arcs $c(s, u_i) = C_{en,i}$ and flow $f_{en,i}$, as well as $c(t, u_i) = C_{ex,i}$ and flow $f_{ex,i}$

For every transition $p_{\text{link}}(x_j|x_i) \neq 0$, create an arc with cost $c(v_i, u_j) = C_{i,j}$ and flow $f_{i,j}$.

The constraint is equivalent to the flow conservation constraint

The objective is the cost of the flow in $G$.

Finding optimal association hypothesis $\mathcal{T}^*$, is equivalent to sending the flow from source to sink that minimizes the cost.
This can be mapped into a cost-flow network $G(X)$ with source $s$ and sink $t$

$$\min_T \sum_i C_{en,i} f_{en,i} + \sum_{i,j} C_{i,j} f_{i,j} + \sum_i C_{ex,i} f_{ex,i} + \sum_i C_i f_i$$

s.t. $f_{en,i} + \sum_j f_{j,i} = f_i = f_{ex,i} + \sum_j f_{i,j} \quad \forall i$

For every observation $x_i \in X$ create two nodes $u_i, v_i$, and an arc with cost $c(u_i, v_j) = C_i$ and flow $f_i$.

Add arcs $c(s, u_i) = C_{en,i}$ and flow $f_{en,i}$, as well as $c(t, u_i) = C_{ex,i}$ and flow $f_{ex,i}$

For every transition $p_{\text{link}}(x_j|x_i) \neq 0$, create an arc with cost $c(v_i, u_j) = C_{i,j}$ and flow $f_{i,j}$.

The constraint is equivalent to the flow conservation constraint

The objective is the cost of the flow in $G$.

Finding optimal association hypothesis $T^\ast$, is equivalent to sending the flow from source to sink that minimizes the cost.
This can be mapped into a cost-flow network $G(X)$ with source $s$ and sink $t$

$$
\min_{T} \sum_{i} C_{\text{en},i} f_{\text{en},i} + \sum_{i,j} C_{i,j} f_{i,j} + \sum_{i} C_{\text{ex},i} f_{\text{ex},i} + \sum_{i} C_{i} f_{i}
$$

s.t. $f_{\text{en},i} + \sum_{j} f_{j,i} = f_{i} = f_{\text{ex},i} + \sum_{j} f_{i,j}$ \quad \forall i

Diagram:

- Observation edges $(u_i, v_i)$
- Transition edges $(v_i, u_i)$
- Enter/exit edges $(s, u_i)$ & $(v_i, t)$
How is to optimize the objective

- For a given $f(G)$, the minimal cost can be solved for in polynomial time by a min-cost flow algorithm

- Construct the graph $G(V, E, C, f)$ from observation set $\mathcal{X}$
- Start with empty flow
- WHILE ($f(G)$) can be augmented
  - Augment $f(G)$ by one.
  - Find the min cost flow by the algorithm of [12].
  - IF (current min cost < global optimal cost) 
    Store current min-cost assignment as global optimum.
- Return the global optimal flow as the best association hypothesis

- The minimal cost is a convex function w.r.t $f(G)$
How is to optimize the objective

- For a given \( f(G) \), the minimal cost can be solved for in polynomial time by a min-cost flow algorithm
  - Construct the graph \( G(V, E, C, f) \) from observation set \( \mathcal{X} \)
  - Start with empty flow
  - WHILE ( \( f(G') \) can be augmented )
    - Augment \( f(G') \) by one.
    - Find the min cost flow by the algorithm of [12].
    - IF ( current min cost < global optimal cost )
      Store current min-cost assignment as global optimum.
  - Return the global optimal flow as the best association hypothesis

- The minimal cost is a convex function w.r.t \( f(G) \)
  - Hence the enumeration over all possible \( f(G) \) can be replaced by a Fibonacci search, which finds the global minimal cost by at most \( O(\log n) \)
How is to optimize the objective

- For a given $f(G)$, the minimal cost can be solved for in polynomial time by a min-cost flow algorithm

  - Construct the graph $G(V, E, C, f)$ from observation set $\mathcal{X}$
  - Start with empty flow
  - WHILE ( $f(G)$ can be augmented )
    - Augment $f(G)$ by one.
    - Find the min cost flow by the algorithm of [12].
    - IF ( current min cost < global optimal cost )
      Store current min-cost assignment as global optimum.
  - Return the global optimal flow as the best association hypothesis

- The minimal cost is a convex function w.r.t $f(G)$
- Hence the enumeration over all possible $f(G)$ can be replaced by a Fibonacci search, which finds the global minimal cost by at most $O(\log n)$
What are the problems with this approach?
Grouping
When do we use grouping?

- In the case of frontal/slanted plane methods, we assume that the image has been over-segmented into a set of superpixels.
- This can be applied to the general problem of matching to do it in a more robust way.
When do we use grouping?

- In the case of frontal/slanted plane methods, we assume that the image has been over-segmented into a set of superpixels.
- This can be applied to the general problem of matching to do it in a more robust way.
- What is the model assumption then?
When do we use grouping?

- In the case of frontal/slanted plane methods, we assume that the image has been over-segmented into a set of superpixels.
- This can be applied to the general problem of matching to do it in a more robust way.
- What is the model assumption then?
- How are those superpixels computed?
When do we use grouping?

- In the case of frontal/slanted plane methods, we assume that the image has been over-segmented into a set of superpixels.
- This can be applied to the general problem of matching to do it in a more robust way.
- What is the model assumption then?
- How are those superpixels computed?
- We will see a few different approaches.
When do we use grouping?

- In the case of frontal/slanted plane methods, we assume that the image has been over-segmented into a set of superpixels.
- This can be applied to the general problem of matching to do it in a more robust way.
- What is the model assumption then?
- How are those superpixels computed?
- We will see a few different approaches.
- At first sight, the problem is very similar to clustering.
When do we use grouping?

- In the case of frontal/slanted plane methods, we assume that the image has been over-segmented into a set of superpixels.
- This can be applied to the general problem of matching to do it in a more robust way.
- What is the model assumption then?
- How are those superpixels computed?
- We will see a few different approaches.
- At first sight, the problem is very similar to clustering.
- We can draw inspiration from clustering algorithms.
When do we use grouping?

- In the case of frontal/slanted plane methods, we assume that the image has been over-segmented into a set of superpixels.
- This can be applied to the general problem of matching to do it in a more robust way.
- What is the model assumption then?
- How are those superpixels computed?
- We will see a few different approaches.
- At first sight, the problem is very similar to clustering.
- We can draw inspiration from clustering algorithms.
Techniques we will see

- K-means style clustering, e.g., SLIC superpixels
- Normalized cuts
- Graph-based superpixels
- Watershed transform
- Mean-shift
Find three clusters in this data

Figure: From M. Tappan
Simple K-means

- Find three clusters in this data

Figure: From M. Tappen
Simple K-means

- Find three clusters in this data

**Figure:** From M. Tappen
Simple K-means

- Find three clusters in this data

Figure: From M. Tappan
K-means style algorithms

- We would like to encode
  - Super-pixels have regular shape
We would like to encode

- Super-pixels have regular shape
- Pixels in super-pixels have similar appearance
K-means style algorithms

- We would like to encode
  - Super-pixels have regular shape
  - Pixels in super-pixels have similar appearance
- Let $S = \{s_1, \cdots, s_m\}$ be the set of superpixel assignments

$$E(p) = E_{col}(p, c_{sp}) + \lambda_{pos} E_{pos}(p, \mu_{sp})$$

The problem becomes

$$\min_{S, \mu, c} \sum p E(p, s_p, \mu_{sp}, c_{sp})$$
K-means style algorithms

- We would like to encode
  - Super-pixels have regular shape
  - Pixels in super-pixels have similar appearance
- Let $S = \{s_1, \cdots, s_m\}$ be the set of superpixel assignments
- We define $\mu = \{\mu_1, \cdots, \mu_m\}$ as the mean location of each superpixel, and $c = \{c_1, \cdots, c_m\}$ as the mean appearance descriptor.
K-means style algorithms

- We would like to encode
  - Super-pixels have regular shape
  - Pixels in super-pixels have similar appearance

- Let \( S = \{s_1, \cdots, s_m\} \) be the set of superpixel assignments

- We define \( \mu = \{\mu_1, \cdots, \mu_m\} \) as the mean location of each superpixel, and \( c = \{c_1, \cdots, c_m\} \) as the mean appearance descriptor.

- We can define the total energy of a pixel as
  \[
  E(p) = E_{\text{col}}(p, c_{s_p}) + \lambda_{\text{pos}} E_{\text{pos}}(p, \mu_{s_p})
  \]
K-means style algorithms

- We would like to encode
  - Super-pixels have regular shape
  - Pixels in super-pixels have similar appearance
- Let \( S = \{s_1, \cdots, s_m\} \) be the set of superpixel assignments
- We define \( \mu = \{\mu_1, \cdots, \mu_m\} \) as the mean location of each superpixel, and \( c = \{c_1, \cdots, c_m\} \) as the mean appearance descriptor.
- We can define the total energy of a pixel as
  \[
  E(p) = E_{\text{col}}(p, c_{s_p}) + \lambda_{\text{pos}} E_{\text{pos}}(p, \mu_{s_p})
  \]
- The problem becomes
  \[
  \min_{S, \mu, c} \sum_p E(p, s_p, \mu_{s_p}, c_{s_p}).
  \]
K-means style algorithms

- We would like to encode
  - Super-pixels have regular shape
  - Pixels in super-pixels have similar appearance
- Let $S = \{s_1, \cdots, s_m\}$ be the set of superpixel assignments
- We define $\mu = \{\mu_1, \cdots, \mu_m\}$ as the mean location of each superpixel, and $c = \{c_1, \cdots, c_m\}$ as the mean appearance descriptor.
- We can define the total energy of a pixel as
  \[
  E(p) = E_{\text{col}}(p, c_{s_p}) + \lambda_{\text{pos}} E_{\text{pos}}(p, \mu_{s_p})
  \]
- The problem becomes
  \[
  \min_{S, \mu, c} \sum_p E(p, s_p, \mu_{s_p}, c_{s_p}).
  \]
K-means style algorithms

- We can define the total energy of a pixel as
  \[ E(p) = E_{\text{col}}(p, c_{sp}) + \lambda_{\text{pos}} E_{\text{pos}}(p, \mu_{sp}) \]

- The problem becomes
  \[ \min_{S, \mu, c} \sum_p E(p, s_p, \mu_{s_p}, c_{s_p}) . \]

- Simple iterative algorithm:
  - Solve for the assignments \( S \)
  - Solve in parallel for the positions \( \mu \) and appearances \( c \)
We can define the total energy of a pixel as

\[ E(p) = E_{\text{col}}(p, c_p) + \lambda_{\text{pos}} E_{\text{pos}}(p, \mu_{s_p}) \]

The problem becomes

\[ \min_{S, \mu, c} \sum_p E(p, s_p, \mu_{s_p}, c_{s_p}). \]

Simple iterative algorithm:
- Solve for the assignments \( S \)
- Solve in parallel for the positions \( \mu \) and appearances \( c \)

Is this easy to do?
K-means style algorithms

- We can define the total energy of a pixel as

\[ E(p) = E_{\text{col}}(p, c_{sp}) + \lambda_{\text{pos}} E_{\text{pos}}(p, \mu_{sp}) \]

- The problem becomes

\[ \min_{S, \mu, c} \sum_p E(p, s_p, \mu_{sp}, c_{sp}). \]

- Simple iterative algorithm:
  - Solve for the assignments \( S \)
  - Solve in parallel for the positions \( \mu \) and appearances \( c \)

- Is this easy to do?
[R. Achanta and A. Shaji and K. Smith and A. Lucchi and P. Fua and S. Susstrunk, PAMI12]
Let $S = \{s_1, \cdots, s_m\}$ be the set of superpixel assignments.

Let $\Theta = \{\theta_1, \cdots, \theta_m\}$ be the set of plane parameters.
Joint Segmentation and Depth Estimation

- Let $S = \{s_1, \ldots, s_m\}$ be the set of superpixel assignments.
- Let $\Theta = \{\theta_1, \ldots, \theta_m\}$ be the set of plane parameters.
- We define $\mu = \{\mu_1, \ldots, \mu_m\}$ as the mean location of each superpixel, and $c = \{c_1, \ldots, c_m\}$ as the mean appearance descriptor.
Joint Segmentation and Depth Estimation

- Let $S = \{s_1, \cdots, s_m\}$ be the set of superpixel assignments.
- Let $\Theta = \{\theta_1, \cdots, \theta_m\}$ be the set of plane parameters.
- We define $\mu = \{\mu_1, \cdots, \mu_m\}$ as the mean location of each superpixel, and $c = \{c_1, \cdots, c_m\}$ as the mean appearance descriptor.
- We can define the total energy of a pixel as

$$E(p) = E_{\text{col}}^l_r(p, c_{sp}, \theta_{sp}) + \lambda_{\text{pos}} E_{\text{pos}}(p, \mu_{sp}) + \lambda_{\text{disp}} E_{\text{disp}}^l_r(p, \theta_{sp}),$$
Let $S = \{s_1, \cdots, s_m\}$ be the set of superpixel assignments.

Let $\Theta = \{\theta_1, \cdots, \theta_m\}$ be the set of plane parameters.

We define $\mu = \{\mu_1, \cdots, \mu_m\}$ as the mean location of each superpixel, and $c = \{c_1, \cdots, c_m\}$ as the mean appearance descriptor.

We can define the total energy of a pixel as

$$E(p) = E_{col}^l(p, c_{sp}, \theta_{sp}) + \lambda_{pos}E_{pos}(p, \mu_{sp}) + \lambda_{disp}E_{disp}^l(p, \theta_{sp}),$$
Joint Segmentation and Depth Estimation

- Let $S = \{s_1, \cdots, s_m\}$ be the set of superpixel assignments.
- Let $\Theta = \{\theta_1, \cdots, \theta_m\}$ be the set of plane parameters.
- We define $\mu = \{\mu_1, \cdots, \mu_m\}$ as the mean location of each superpixel, and $c = \{c_1, \cdots, c_m\}$ as the mean appearance descriptor.
- We can define the total energy of a pixel as

$$E(p) = E_{\text{col}}^l(p, c_{sp}, \theta_{sp}) + \lambda_{\text{pos}}E_{\text{pos}}(p, \mu_{sp}) + \lambda_{\text{disp}}E_{\text{disp}}^l(p, \theta_{sp}),$$

- We can use:

$$E_{\text{pos}}(p, \mu_{sp}) = \|p - \mu_{sp}\|^2/g \quad E_{\text{col}}(p, c_{sp}) = (I_t(p) - c_{sp})^2$$

and

$$E_{\text{disp}}(p, \theta_{sp}) = \begin{cases} (d(p, \theta_{sp}) - \hat{d}(p))^2 & \text{if } p \in F \\ \lambda & \text{otherwise} \end{cases}$$
Let $S = \{s_1, \cdots, s_m\}$ be the set of superpixel assignments.
Let $\Theta = \{\theta_1, \cdots, \theta_m\}$ be the set of plane parameters.
We define $\mu = \{\mu_1, \cdots, \mu_m\}$ as the mean location of each superpixel, and $c = \{c_1, \cdots, c_m\}$ as the mean appearance descriptor.
We can define the total energy of a pixel as

$$E(p) = E_{col}^l(p, c_{sp}, \theta_{sp}) + \lambda_{pos} E_{pos}(p, \mu_{sp}) + \lambda_{disp} E_{disp}^l(p, \theta_{sp}),$$

We can use:

$$E_{pos}(p, \mu_{sp}) = \|p - \mu_{sp}\|_2^2 / g \quad E_{col}(p, c_{sp}) = (l_t(p) - c_{sp})^2$$

and

$$E_{disp}(p, \theta_{sp}) = \begin{cases} (d(p, \theta_{sp}) - \hat{d}(p))^2 & \text{if } p \in F \\ \lambda & \text{otherwise} \end{cases}$$
We can define the total energy of a pixel as

\[ E(p) = E_{col}^{l,r}(p, c_s, \theta_s) + \lambda_{pos} E_{pos}(p, \mu_s) + \lambda_{disp} E_{disp}^{l,r}(p, \theta_s), \]

The problem of joint unsupervised segmentation and flow estimation becomes

\[ \min_{\Theta, S, \mu, c} \sum_p E(p, s_p, \theta_{s_p}, \mu_{s_p}, c_{s_p}). \]
Joint Segmentation and Depth Estimation

- We can define the total energy of a pixel as
  \[ E(p) = E_{col}^{l,r}(p, c_{sp}, \theta_{sp}) + \lambda_{pos}E_{pos}(p, \mu_{sp}) + \lambda_{disp}E_{disp}^{l,r}(p, \theta_{sp}), \]

- The problem of joint unsupervised segmentation and flow estimation becomes
  \[
  \min_{\Theta, S, \mu, c} \sum_{p} E(p, s_{p}, \theta_{sp}, \mu_{sp}, c_{sp}).
  \]

- Simple iterative algorithm
  - Solve for the assignments \( S \)
  - Solve in parallel for the planes \( \Theta \), positions \( \mu \) and appearances \( c \)
We can define the total energy of a pixel as

\[ E(p) = E_{\text{col}}(p, c_{sp}, \theta_{sp}) + \lambda_{\text{pos}} E_{\text{pos}}(p, \mu_{sp}) + \lambda_{\text{disp}} E_{\text{disp}}^{l,r}(p, \theta_{sp}), \]

The problem of joint unsupervised segmentation and flow estimation becomes

\[ \min_{\Theta, S, \mu, c} \sum_p E(p, s_{sp}, \theta_{sp}, \mu_{sp}, c_{sp}). \]

Simple iterative algorithm

- Solve for the assignments \( S \)
- Solve in parallel for the planes \( \Theta \), positions \( \mu \) and appearances \( c \)

How do we do this?
Joint Segmentation and Depth Estimation

- We can define the total energy of a pixel as

\[ E(p) = E_{\text{col}}^{l,r}(p, c_s, \theta_s) + \lambda_{\text{pos}} E_{\text{pos}}(p, \mu_s) + \lambda_{\text{disp}} E_{\text{disp}}^{l,r}(p, \theta_s), \]

- The problem of joint unsupervised segmentation and flow estimation becomes

\[ \min_{\Theta, S, \mu, \mathbf{c}} \sum_p E(p, s_p, \theta_s, \mu_s, c_s). \]

- Simple iterative algorithm
  - Solve for the assignments \( S \)
  - Solve in parallel for the planes \( \Theta \), positions \( \mu \) and appearances \( \mathbf{c} \)

- How do we do this?
Techniques we will see

- K-means style clustering, e.g., SLIC superpixels
- Normalized cuts
- Graph-based superpixels
- Watershed transform
- Mean-shift
Segmentation as a mincut problem

- Examines the affinities (similarities) between nearby pixels and tries to separate groups that are connected with weak affinities.

- The cut separate the nodes into two groups
The cut between two groups $A$ and $B$ is defined as the sum of all the weights being cut:

$$cut(A, B) = \sum_{i \in A, j \in B} w_{i,j}$$

Problem: Results in small cuts that isolates single pixels.

We need to normalize somehow.
Normalized Cuts

- Better measure is the normalized cuts

\[ N_{\text{cut}}(A, B) = \frac{\text{cut}(A, B)}{\text{assoc}(A, V)} + \frac{\text{cut}(A, B)}{\text{assoc}(B, V)} \]

with \( \text{assoc}(A, A) = \sum_{i \in A, j \in A} w_{ij} \) is the association term within a cluster and \( \text{Assoc}(A, V) = \text{assoc}(A, A) + \text{cut}(A, B) \) is the sum of all the weights associated with nodes in A.

We want minimize the disassociation between the groups and maximize the association within the groups.
Normalize Cuts

- Computing the optimal normalized cut is NP-Complete.
- Instead, relax by computing a real value assignment
Normalize Cuts

- Computing the optimal normalized cut is NP-Complete.
- Instead, relax by computing a real value assignment
  - Let $x$ be an indicator vector, with $x_i = 1$ if $x_i \in A$, and $x_i = -1$ otherwise.
  - Let $d = W1$ be the row sums of the symmetric matrix $W$, and $D = \text{diag}(d)$ be the corresponding diagonal matrix.
Normalize Cuts

- Computing the optimal normalized cut is NP-Complete.
- Instead, relax by computing a real value assignment
- Let $\mathbf{x}$ be an indicator vector, with $x_i = 1$ if $x_i \in A$, and $x_i = -1$ otherwise.
  Let $d = W\mathbf{1}$ be the row sums of the symmetric matrix $W$, and $D = \text{diag}(d)$ be the corresponding diagonal matrix.
- Shi and Malik, compute the cut by solving
  \[
  \min_{y} \frac{y^T(D - W)y}{y^TDy}
  \]
  with $y = ((1 + x) - b(1 - x))/2$ is a vector with all 1's and -b's such that $y \cdot d = 0$, by relaxing $y$ to be real value.
Normalize Cuts

- Computing the optimal normalized cut is NP-Complete.
- Instead, relax by computing a real value assignment.

Let \( x \) be an indicator vector, with \( x_i = 1 \) if \( x_i \in A \), and \( x_i = -1 \) otherwise.

Let \( d = W1 \) be the row sums of the symmetric matrix \( W \), and \( D = \text{diag}(d) \) be the corresponding diagonal matrix.

Shi and Malik, compute the cut by solving

\[
\min_y \frac{y^T(D - W)y}{y^TDy}
\]

with \( y = \frac{((1 + x) - b(1 - x))}{2} \) is a vector with all 1’s and -b’s such that \( y \cdot d = 0 \), by relaxing \( y \) to be real value.

- \( D - W \) is the Laplacian
Normalize Cuts

- Computing the optimal normalized cut is NP-Complete.
- Instead, relax by computing a real value assignment
- Let \( x \) be an indicator vector, with \( x_i = 1 \) if \( x_i \in A \), and \( x_i = -1 \) otherwise.
- Let \( d = W1 \) be the row sums of the symmetric matrix \( W \), and \( D = \text{diag}(d) \) be the corresponding diagonal matrix.
- Shi and Malik, compute the cut by solving

\[
\min_y \frac{y^T(D - W)y}{y^T Dy}
\]

with \( y = ((1 + x) - b(1 - x))/2 \) is a vector with all 1’s and -b’s such that \( y \cdot d = 0 \), by relaxing \( y \) to be real value.

- \( D - W \) is the Laplacian
Solving for the cut

- Minimizing this **Rayleigh quotient** is equivalent to solving the generalized eigenvalue system

\[(D - W)y = \lambda Dy\]

- This is a normal eigenvalue problem

\[(I - N)z = \lambda z\]

with \(N = D^{1/2}WD^{1/2}\) is the normalized affinity matrix, and \(z = D^{1/2}y\).
Solving for the cut

Minimizing this **Rayleigh quotient** is equivalent to solving the generalized eigenvalue system

\[(D - W)y = \lambda Dy\]

This is a normal eigenvalue problem

\[(I - N)z = \lambda z\]

with \(N = D^{1/2}WD^{1/2}\) is the normalized affinity matrix, and \(z = D^{1/2}y\).

This is an example of a spectral method for segmentation, solution is the second smallest eigenvector/eigenvalue.
Solving for the cut

- Minimizing this **Rayleigh quotient** is equivalent to solving the generalized eigenvalue system

\[(D - W)y = \lambda Dy\]

- This is a normal eigenvalue problem

\[(I - N)z = \lambda z\]

with \(N = D^{1/2}WD^{1/2}\) is the normalized affinity matrix, and \(z = D^{1/2}y\).

- This is an example of a spectral method for segmentation, solution is the second smallest eigenvector/eigenvalue

- This process can be applied in a hierarchical manner to have more clusters
Solving for the cut

- Minimizing this **Rayleigh quotient** is equivalent to solving the generalized eigenvalue system

\[(D - W)y = \lambda Dy\]

- This is a normal eigenvalue problem

\[(I - N)z = \lambda z\]

with \(N = D^{1/2}WD^{1/2}\) is the normalized affinity matrix, and \(z = D^{1/2}y\).

- This is an example of a spectral method for segmentation, solution is the second smallest eigenvector/eigenvalue

- This process can be applied in a hierarchical manner to have more clusters

- Shi and Malik employ the following affinity

\[w_{i,j} = \exp \left( - \frac{||F_i - F_j||_2^2}{\sigma_f^2} - \frac{||p_i - p_j||_2^2}{\sigma_s^2} \right)\]

for pixels within a radius \(||p_i - p_j||_2 < r\), and \(F\) is a feature vector with color, intensities, histograms, gradients, etc.
Solving for the cut

- Minimizing this **Rayleigh quotient** is equivalent to solving the generalized eigenvalue system

\[(D - W)y = \lambda Dy\]

- This is a normal eigenvalue problem

\[(I - N)z = \lambda z\]

with \(N = D^{1/2}WD^{1/2}\) is the normalized affinity matrix, and \(z = D^{1/2}y\).

- This is an example of a spectral method for segmentation, solution is the second smallest eigenvector/eigenvalue

- This process can be applied in a hierarchical manner to have more clusters

- Shi and Malik employ the following affinity

\[w_{i,j} = \exp \left( - \frac{||F_i - F_j||_2^2}{\sigma_f^2} - \frac{||p_i - p_j||_2^2}{\sigma_s^2} \right)\]

for pixels within a radius \(||p_i - p_j||_2 < r\), and \(F\) is a feature vector with color, intensities, histograms, gradients, etc.
1. Given an image or image sequence, set up a weighted graph $G = (V, E)$ and set the weight on the edge connecting two nodes to be a measure of the similarity between the two nodes.

2. Solve $(D - W)x = \lambda Dx$ for eigenvectors with the smallest eigenvalues.

3. Use the eigenvector with the second smallest eigenvalue to bipartition the graph.

4. Decide if the current partition should be subdivided and recursively repartition the segmented parts if necessary.
Examples

Figure: Shi and Malik N-Cuts