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 $\mathbf{P} = \{\mathcal{P}_l | l \in \mathcal{L}\}$, where $\mathcal{P}_l = \{p \in \mathcal{P} | f_p = l\}$ is a subset of pixels assigned label l.
- There is a one to one correspondence between labelings f and partitions $\mathcal{P}$.

More formally

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

More formally

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

More formally

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

More formally

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

More formally

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

Example

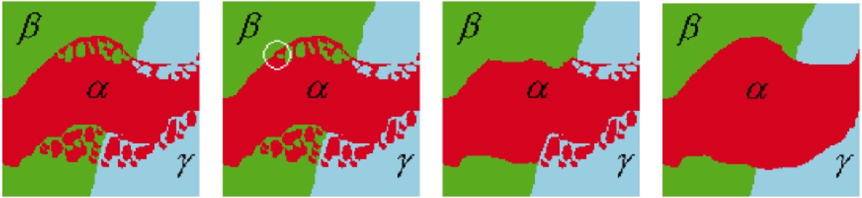


Figure: (a) Current partition (b) local move (c) $\alpha - \beta$-swap (d) α-expansion.

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

---

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

Finding optimal Swap move

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

Finding optimal Swap move

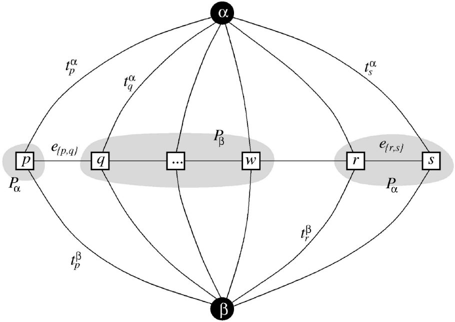
- Given an input labeling f (partition $\mathcal{P}$) and a pair of labels α, β we want to find a labeling $\hat{f}$ that minimizes E over all labelings within one $\alpha - \beta$-swap of f.
- This is going to be done by computing a labeling corresponding to a minimum cut on a graph $\mathcal{G}_{\alpha\beta}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

Finding optimal Swap move

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

Graph Construction

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



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

Computing the Cut

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

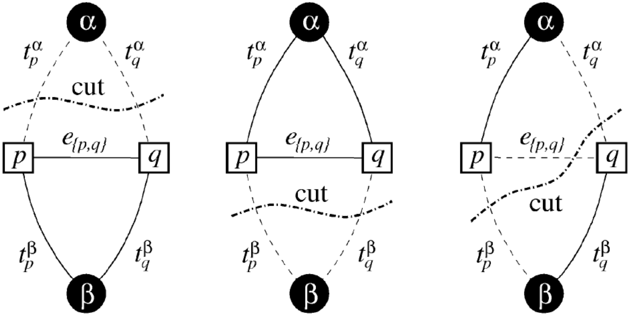
$$f_p^{\mathcal{C}} = \begin{cases} \alpha & \text{if } \; t_p^{\alpha} \in \mathcal{C} \;\; \text{for} \;\; p \in \mathcal{P}_{\alpha\beta} \\ \beta & \text{if } \; t_p^{\beta} \in \mathcal{C} \;\; \text{for} \;\; p \in \mathcal{P}_{\alpha\beta} \\ f_p & \text{for } \;\; p \in \mathcal{P}, \;\; p \notin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

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

Properties

- For any cut, then

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



Finding the optimal α expansion

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

Finding the optimal α expansion

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

Finding the optimal α expansion

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

Finding the optimal α expansion

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

Graph Construction

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

Graph Construction

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

Graph Construction

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

Graph Construction

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

Graph Construction

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

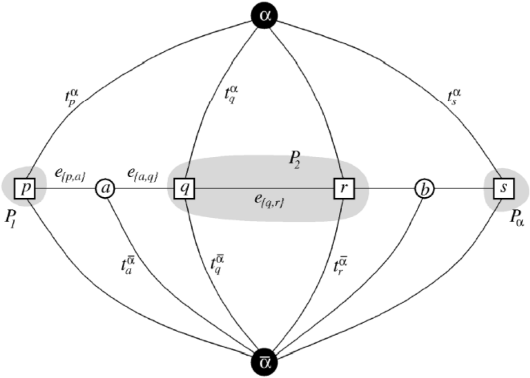
$$\mathcal{E}_\alpha = \left\{ \bigcup_{p \in \mathcal{P}} \{t_p^\alpha, t_p^{\bar{\alpha}}\}, \bigcup_{\substack{\{p,q\} \in \mathcal{N} \\ f_p \neq f_q}} \mathcal{E}_{\{p,q\}} \ , \bigcup_{\substack{\{p,q\} \in \mathcal{N} \\ f_p = f_q}} e_{\{p,q\}} \right\}$$

Graph Construction

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

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

Graph Construction



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

Properties

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

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

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

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

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

Global Minimization Techniques

Ways to get an approximate solution typically

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

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

Let's look at data/bechmarks

Benchmarks

Two benchmarks with very different characteristics

(Middlebury)

(KITTI)

Middlebury 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

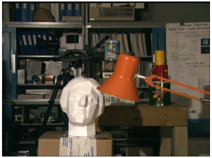
Middlebury Dataset

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



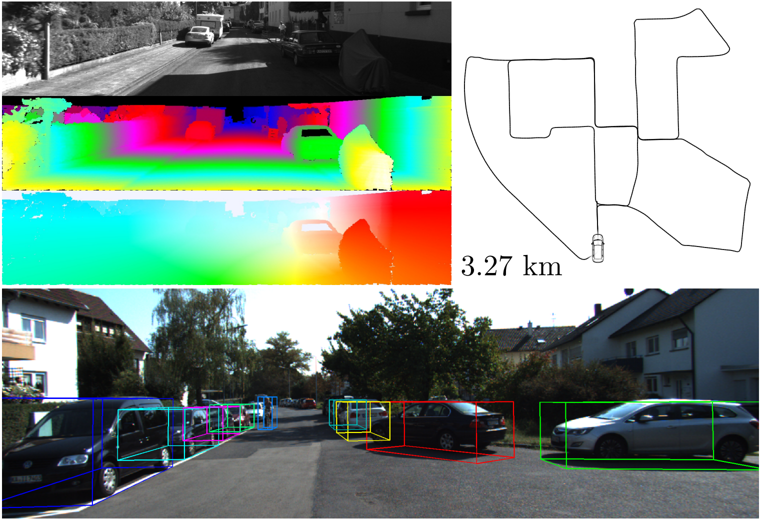
Error Threshold = 1 Algorithm	Avg.	Tsukuba ground truth			Venus ground truth			Teddy ground truth			Cones ground truth		
CoopRegion [41]	8.8	0.87 4	**1.16 1**	4.61 3	0.11 4	0.21 3	1.54 7	5.16 16	8.31 11	13.0 13	2.79 17	7.18 4	8.01 23
AdaptingBP [17]	9.0	1.11 19	1.37 7	5.79 19	0.10 3	0.21 4	1.44 5	4.22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

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

Benchmarks: KITTI Data Collection

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

The KITTI Vision Benchmark Suite



Novel Challenges

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

Middlebury, Errors: 2.7%

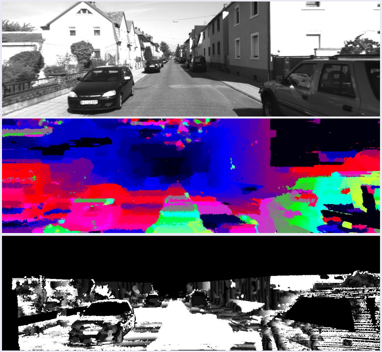
- Error threshold: 1 px (Middlebury) / 3 px (KITTI)

Novel Challenges

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

Middlebury, Errors: **2.7%**

KITTI, Errors: **46.3%**



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

So what is the difference?

Middlebury

KITTI

- Laboratory
- Lambertian

- Moving vehicle
- Specularities

Novel Challenges

So what is the difference?

Middlebury

KITTI

- Laboratory
- Lambertian
- Rich in texture

- Moving vehicle
- Specularities
- Sensor saturation

Novel Challenges

So what is the difference?

Middlebury

KITTI

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

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

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

KITTI

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

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

Stereo Evaluation

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

MRFs for stereo

Global methods: define a Markov random field over

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

Plane MRFs

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

Plane MRFs

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

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

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

Plane MRFs

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

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

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

- This are continuous variables. Is this a problem?

Plane MRFs

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

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

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

Plane MRFs

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

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

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

Plane MRFs

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

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

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

Plane MRFs

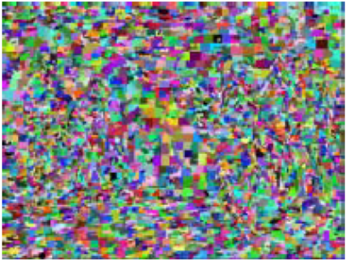
- First segment an image into small regions, i.e., superpixels
- Assume that the 3D world is compose of small frontal/slanted planes
- Good representation if the superpixels are small and respect boundaries

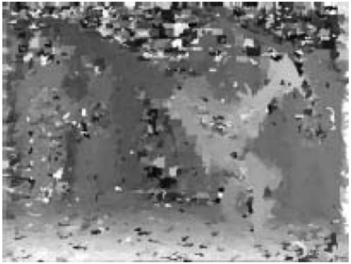
$$E(\mathbf{x}_1, \cdots, \mathbf{x}_n) = \sum_i C(\mathbf{x}_i) + \sum_i \sum_{j \in \mathcal{N}_j} C(\mathbf{x}_i, \mathbf{x}_j)$$

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

- This are continuous variables. Is this a problem?
- What can I do to solve this? Discretize the problem
- The unitary are usually agreegation of cost over the local matching on the pixels in that superpixel
- Pairwise is typically smoothness

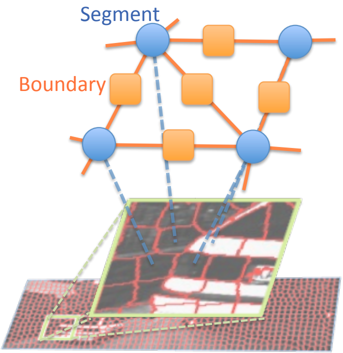
Slanted-plane MRFs





A more sophisticated occlusion model

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



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

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

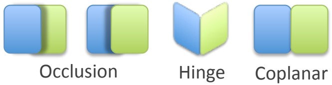
Slanted 3D plane of segment

Continuous variable

Boundary variable o_{ij}

Relationship between segments

4 states



Discrete variable

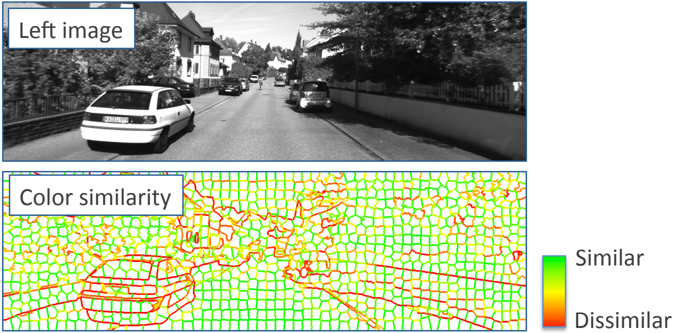
- Takes as input disparities computed by any local algorithm

Energy of PCBP-Stereo

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

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

Similar color → Likely to be coplanar

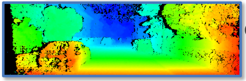


Energy of PCBP-Stereo

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

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

Agreement with result of input disparity map



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

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

Disparity map — Slanted plane

On boundary

"Occlusion" – Foreground segment owns boundary

Energy of PCBP-Stereo

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

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

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

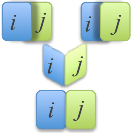
Impose penalty $\lambda_{\text{occ}} > \lambda_{\text{hinge}} > 0$

(2) Boundary labels ⟷ (match) Slanted planes

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

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

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



Energy of PCBP-Stereo

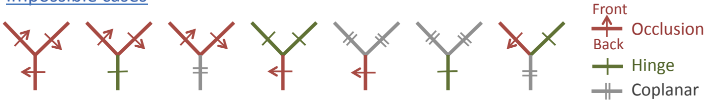
- **y** the set of slanted 3D planes, **o** the set of discrete boundary variables

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

Occlusion boundary reasoning [Malik 1987]

Penalize impossible junctions

Impossible cases

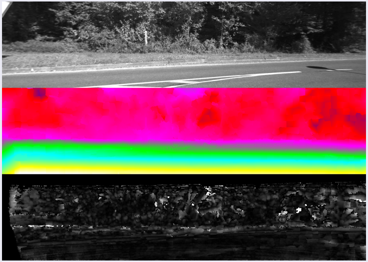


[K. Yamaguchi, T. Hazan, D. McAllester and R. Urtasun, ECCV12]

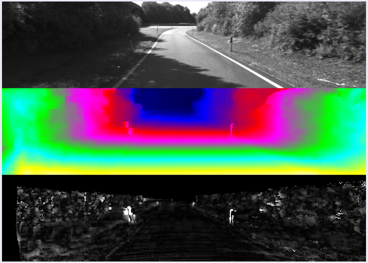
Easy Scenarios:

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

Errors: $< 0.5\%$



Errors: $< 0.5\%$



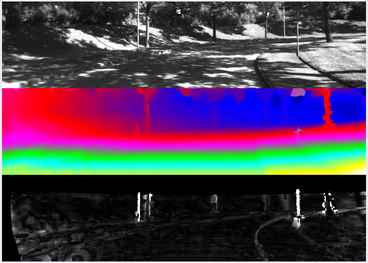
Stereo Evaluation

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

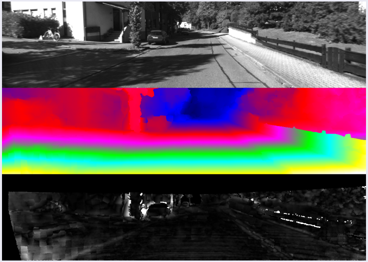
Easy Scenarios:

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

Errors: $< 0.5\%$



Errors: $< 0.5\%$



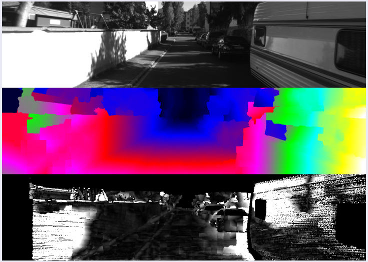
Stereo Evaluation

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

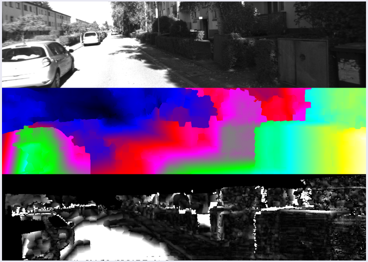
Hard Scenarios:

- Textureless or saturated areas
- Ambiguous reflections

Errors: 22.1%



Errors: 17.4%



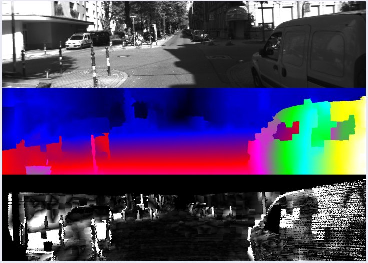
Stereo Evaluation

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

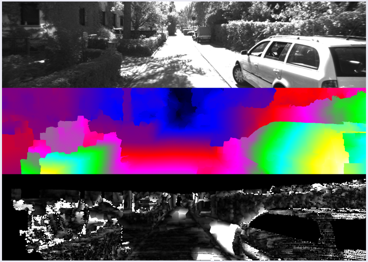
Hard Scenarios:

- Depth discontinuities / complicated geometries

Errors: 11.2%



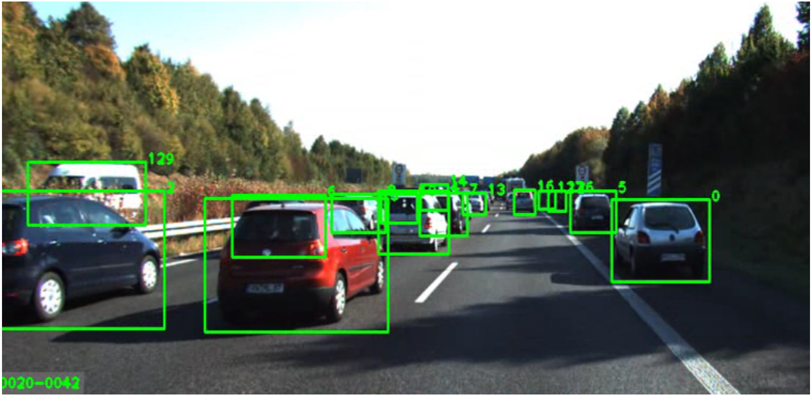
Errors: 10.5%



A different view on tracking

Tracking as a graph minimization

- **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



Tracking as a graph minimization

- 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-overlaping constraints in trajectories

Tracking as a graph minimization

- 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-overlaping constraints in trajectories
- The optimal data association is found by a min-cost flow algorithm in the network

Tracking as a graph minimization

- 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-overlaping constraints in trajectories
- The optimal data association is found by a min-cost flow algorithm in the network

Notation and Problem Definition

- Let $\mathcal{X} = \{\mathbf{x}_i\}$ be a set of object observations
- Each $\mathbf{x}_i$ is detection response $\mathbf{x}_i = (x_i, s_i, a_i, t_i)$, 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)

Notation and Problem Definition

- Let $\mathcal{X} = \{\mathbf{x}_i\}$ be a set of object observations
- Each $\mathbf{x}_i$ is detection response $\mathbf{x}_i = (x_i, s_i, a_i, t_i)$, 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, $T_k = \{\mathbf{x}_{k_1}, \cdots, \mathbf{x}_{k_{l_k}}\}$, with $\mathbf{x}_{k_i} \in \mathcal{X}$

Notation and Problem Definition

- Let $\mathcal{X} = \{\mathbf{x}_i\}$ be a set of object observations
- Each $\mathbf{x}_i$ is detection response $\mathbf{x}_i = (x_i, s_i, a_i, t_i)$, 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, $T_k = \{\mathbf{x}_{k_1}, \cdots, \mathbf{x}_{k_{l_k}}\}$, with $\mathbf{x}_{k_i} \in \mathcal{X}$
- An association hypothesis T is defined as a set of single trajectory hypotheses, $\mathcal{T} = \{T_k\}$

Notation and Problem Definition

- Let $\mathcal{X} = \{\mathbf{x}_i\}$ be a set of object observations
- Each $\mathbf{x}_i$ is detection response $\mathbf{x}_i = (x_i, s_i, a_i, t_i)$, 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, $T_k = \{\mathbf{x}_{k_1}, \cdots, \mathbf{x}_{k_{l_k}}\}$, with $\mathbf{x}_{k_i} \in \mathcal{X}$
- An association hypothesis T is defined as a set of single trajectory hypotheses, $\mathcal{T} = \{T_k\}$
- The association is given by

$$\begin{aligned} \mathcal{T}^* &= arg \max_{\mathcal{T}} P(\mathcal{T}|\mathcal{X}) \\ &= arg \max_{\mathcal{T}} P(\mathcal{X}|\mathcal{T})P(\mathcal{T}) \\ &= arg \max_{\mathcal{T}} \prod_i P(\mathbf{x}_i|\mathcal{T})P(\mathcal{T}) \end{aligned}$$

Notation and Problem Definition

- Let $\mathcal{X} = \{\mathbf{x}_i\}$ be a set of object observations
- Each $\mathbf{x}_i$ is detection response $\mathbf{x}_i = (x_i, s_i, a_i, t_i)$, 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, $T_k = \{\mathbf{x}_{k_1}, \cdots, \mathbf{x}_{k_{l_k}}\}$, with $\mathbf{x}_{k_i} \in \mathcal{X}$
- An association hypothesis T is defined as a set of single trajectory hypotheses, $\mathcal{T} = \{T_k\}$
- The association is given by

$$\begin{aligned} \mathcal{T}^* &= arg \max_{\mathcal{T}} P(\mathcal{T}|\mathcal{X}) \\ &= arg \max_{\mathcal{T}} P(\mathcal{X}|\mathcal{T})P(\mathcal{T}) \\ &= arg \max_{\mathcal{T}} \prod_i P(\mathbf{x}_i|\mathcal{T})P(\mathcal{T}) \end{aligned}$$

- We have assumed that the likelihood prob. are conditionally independent given $\mathcal{T}$.

Notation and Problem Definition

- Let $\mathcal{X} = \{\mathbf{x}_i\}$ be a set of object observations
- Each $\mathbf{x}_i$ is detection response $\mathbf{x}_i = (x_i, s_i, a_i, t_i)$, 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, $T_k = \{\mathbf{x}_{k_1}, \cdots, \mathbf{x}_{k_{l_k}}\}$, with $\mathbf{x}_{k_i} \in \mathcal{X}$
- An association hypothesis T is defined as a set of single trajectory hypotheses, $\mathcal{T} = \{T_k\}$
- The association is given by

$$\begin{aligned}\mathcal{T}^* &= arg \max_{\mathcal{T}} P(\mathcal{T}|\mathcal{X}) \\ &= arg \max_{\mathcal{T}} P(\mathcal{X}|\mathcal{T})P(\mathcal{T}) \\ &= arg \max_{\mathcal{T}} \prod_i P(\mathbf{x}_i|\mathcal{T})P(\mathcal{T})\end{aligned}$$

- We have assumed that the likelihood prob. are conditionally independent given $\mathcal{T}$.

Optimization problem

- We want to solve the following optimization

$$\mathcal{T}^* = arg \max_{\mathcal{T}} \prod_i P(\mathbf{x}_i|\mathcal{T})P(\mathcal{T})$$

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

Optimization problem

- We want to solve the following optimization

$$\mathcal{T}^* = arg \max_{\mathcal{T}} \prod_i P(\mathbf{x}_i|\mathcal{T})P(\mathcal{T})$$

- 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

$$\mathcal{T}^* = \arg\max_{\mathcal{T}} \prod_i P(\mathbf{x}_i|\mathcal{T})P(\mathcal{T})$$

- 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

$$\mathcal{T}^* = \arg\max_{\mathcal{T}} \prod_i P(\mathbf{x}_i|\mathcal{T}) \prod_{\mathcal{T}_k \in \mathcal{T}} P(\mathcal{T}_k)$$

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

Optimization problem

- We want to solve the following optimization

$$\mathcal{T}^* = arg \max_{\mathcal{T}} \prod_i P(\mathbf{x}_i|\mathcal{T})P(\mathcal{T})$$

- 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

$$\mathcal{T}^* = arg \max_{\mathcal{T}} \prod_i P(\mathbf{x}_i|\mathcal{T}) \prod_{\mathcal{T}_k \in \mathcal{T}} P(\mathcal{T}_k)$$
$$s.t. \quad \mathcal{T}_k \cap \mathcal{T}_l = \emptyset, \quad \forall k \neq l$$

- When is this assumption not good?

Optimization problem

- We want to solve the following optimization

$$\mathcal{T}^* = \arg\max_{\mathcal{T}} \prod_i P(\mathbf{x}_i|\mathcal{T})P(\mathcal{T})$$

- 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

$$\mathcal{T}^* = \arg\max_{\mathcal{T}} \prod_i P(\mathbf{x}_i|\mathcal{T}) \prod_{\mathcal{T}_k \in \mathcal{T}} P(\mathcal{T}_k)$$
$$s.t. \quad \mathcal{T}_k \cap \mathcal{T}_l = \emptyset, \quad \forall k \neq l$$

- When is this assumption not good?

Problem Formulation

$$\mathcal{T}^* = \arg\max_{\mathcal{T}} \prod_i P(\mathbf{x}_i|\mathcal{T}) \prod_{\mathcal{T}_k \in \mathcal{T}} P(\mathcal{T}_k)$$
$$s.t. \quad \mathcal{T}_k \cap \mathcal{T}_l = \emptyset, \quad \forall k \neq l$$

- $P(\mathbf{x}_i|\mathcal{T})$ is the **likelihood** of observation $\mathbf{x}_i$. We can use a Bernoulli distribution for example to represent being an inlier or outlier

$$P(\mathbf{x}_i|\mathcal{T}) = \begin{cases} 1 - \beta_i & \text{if } \exists \mathcal{T}_k \in \mathcal{T}, \mathbf{x}_i \in \mathbf{T}_k \\ \beta_i & \text{otherwise.} \end{cases}$$

Problem Formulation

$$\mathcal{T}^* = \arg\max_{\mathcal{T}} \prod_i P(\mathbf{x}_i|\mathcal{T}) \prod_{\mathcal{T}_k \in \mathcal{T}} P(\mathcal{T}_k)$$
$$s.t. \quad \mathcal{T}_k \cap \mathcal{T}_l = \emptyset, \quad \forall k \neq l$$

- $P(\mathbf{x}_i|\mathcal{T})$ is the **likelihood** of observation $\mathbf{x}_i$. We can use a Bernoulli distribution for example to represent being an inlier or outlier

$$P(\mathbf{x}_i|\mathcal{T}) = \begin{cases} 1 - \beta_i & \text{if } \exists \mathcal{T}_k \in \mathcal{T}, \mathbf{x}_i \in \mathbf{T}_k \\ \beta_i & \text{otherwise.} \end{cases}$$

- $P(\mathcal{T}_k)$ can be modeled as a Markov chain, with initialization probability P_{ent}, termination probability P_{exit}, and transition probability $P_{link}(\mathbf{x}_{k_{i+1}}|\mathbf{x}_{k_i})$

$$P(\mathcal{T}_k) = P(\{\mathbf{x}_{k_0}, \cdots, \mathbf{x}_{k_{l_k}}\})$$
$$= P_{ent}(\mathbf{x}_{k_0}) p_{link}(\mathbf{x}_{k_1}|\mathbf{x}_{k_0}) \cdots p_{link}(\mathbf{x}_{k_{l_k}}|\mathbf{x}_{k_{l_k}}) p_{exit}(\mathbf{x}_{k_{l_k}})$$

Problem Formulation

$$\mathcal{T}^* = \arg\max_{\mathcal{T}} \prod_i P(\mathbf{x}_i|\mathcal{T}) \prod_{\mathcal{T}_k \in \mathcal{T}} P(\mathcal{T}_k)$$
$$s.t. \quad \mathcal{T}_k \cap \mathcal{T}_l = \emptyset, \quad \forall k \neq l$$

- $P(\mathbf{x}_i|\mathcal{T})$ is the **likelihood** of observation $\mathbf{x}_i$. We can use a Bernoulli distribution for example to represent being an inlier or outlier

$$P(\mathbf{x}_i|\mathcal{T}) = \begin{cases} 1-\beta_i & \text{if } \exists \mathcal{T}_k \in \mathcal{T}, \mathbf{x}_i \in \mathbf{T}_k \\ \beta_i & \text{otherwise.} \end{cases}$$

- $P(\mathcal{T}_k)$ can be modeled as a Markov chain, with initialization probability P_{ent}, termination probability P_{exit}, and transition probability $P_{link}(\mathbf{x}_{k_{i+1}}|\mathbf{x}_{k_i})$

$$\begin{aligned} P(\mathcal{T}_k) &= P(\{\mathbf{x}_{k_0}, \cdots, \mathbf{x}_{k_{l_k}}\}) \\ &= P_{ent}(\mathbf{x}_{k_0}) p_{link}(\mathbf{x}_{k_1}|\mathbf{x}_{k_0}) \cdots p_{link}(\mathbf{x}_{k_{l_k}}|\mathbf{x}_{k_{l_k}}) p_{exit}(\mathbf{x}_{k_{l_k}}) \end{aligned}$$

- $P(\mathbf{x}_i|\mathcal{T})$ 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

$$\mathcal{T}^* = \arg\max_{\mathcal{T}} \prod_i P(\mathbf{x}_i|\mathcal{T}) \prod_{\mathcal{T}_k \in \mathcal{T}} P(\mathcal{T}_k)$$
$$s.t. \quad \mathcal{T}_k \cap \mathcal{T}_l = \emptyset, \quad \forall k \neq l$$

- $P(\mathbf{x}_i|\mathcal{T})$ is the **likelihood** of observation $\mathbf{x}_i$. We can use a Bernoulli distribution for example to represent being an inlier or outlier

$$P(\mathbf{x}_i|\mathcal{T}) = \begin{cases} 1 - \beta_i & \text{if } \exists \mathcal{T}_k \in \mathcal{T}, \mathbf{x}_i \in \mathbf{T}_k \\ \beta_i & \text{otherwise.} \end{cases}$$

- $P(\mathcal{T}_k)$ can be modeled as a Markov chain, with initialization probability P_{ent}, termination probability P_{exit}, and transition probability $P_{link}(\mathbf{x}_{k_{i+1}}|\mathbf{x}_{k_i})$

$$\begin{aligned} P(\mathcal{T}_k) &= P(\{\mathbf{x}_{k_0}, \cdots, \mathbf{x}_{k_{l_k}}\}) \\ &= P_{ent}(\mathbf{x}_{k_0}) p_{link}(\mathbf{x}_{k_1}|\mathbf{x}_{k_0}) \cdots p_{link}(\mathbf{x}_{k_{l_k}}|\mathbf{x}_{k_{l_k}}) p_{exit}(\mathbf{x}_{k_{l_k}}) \end{aligned}$$

- $P(\mathbf{x}_i|\mathcal{T})$ 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

$$
\begin{aligned}
f_{en,i} &= \begin{cases} 1 & \text{if } \exists \mathcal{T}_k \in \mathcal{T}, \mathcal{T}_k \text{ starts from } \mathbf{x}_i \\ 0 & \text{otherwise.} \end{cases} \\
f_{ex,i} &= \begin{cases} 1 & \text{if } \exists \mathcal{T}_k \in \mathcal{T}, \mathcal{T}_k \text{ ends at } \mathbf{x}_i \\ 0 & \text{otherwise.} \end{cases} \\
f_{i,j} &= \begin{cases} 1 & \text{if } \exists \mathcal{T}_k \in \mathcal{T}, \mathbf{x}_j \text{is after } \mathbf{x}_i \text{ in } \mathcal{T}_k \\ 0 & \text{otherwise.} \end{cases} \\
f_i &= \begin{cases} 1 & \text{if } \exists \mathcal{T}_k \in \mathcal{T}, \mathbf{x}_i \in \mathcal{T}_k \\ 0 & \text{otherwise.} \end{cases}
\end{aligned}
$$

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

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

Useful definitions

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

$$f_{en,i} = \begin{cases} 1 & \text{if } \exists \mathcal{T}_k \in \mathcal{T}, \mathcal{T}_k \text{ starts from } \mathbf{x}_i \\ 0 & \text{otherwise.} \end{cases}$$

$$f_{ex,i} = \begin{cases} 1 & \text{if } \exists \mathcal{T}_k \in \mathcal{T}, \mathcal{T}_k \text{ ends at } \mathbf{x}_i \\ 0 & \text{otherwise.} \end{cases}$$

$$f_{i,j} = \begin{cases} 1 & \text{if } \exists \mathcal{T}_k \in \mathcal{T}, \mathbf{x}_j \text{is after } \mathbf{x}_i \text{ in } \mathcal{T}_k \\ 0 & \text{otherwise.} \end{cases}$$

$$f_i = \begin{cases} 1 & \text{if } \exists \mathcal{T}_k \in \mathcal{T}, \mathbf{x}_i \in \mathcal{T}_k \\ 0 & \text{otherwise.} \end{cases}$$

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

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

Min-cost flow problem

- We have the optimization problem

$$\min_{\mathcal{T}} - \sum_{\mathcal{T}_k \in \mathcal{T}} \log P(\mathcal{T}_k) - \sum_i \log p(\mathbf{x}_i | \mathcal{T})$$

- This can be obtained as

$$\begin{aligned} \min_{\mathcal{T}} \quad & \sum_{\mathcal{T}_k \in \mathcal{T}} \left(C_{en,k_0} f_{en,k_0} + \sum_j C_{k_j,k_{j+1}} f_{k_j,k_{j+1}} + C_{ex,k_{l_k}} f_{ex,k_{l_k}} \right) + \\ & + \sum_i \left(- \log(1-\beta_i) f_i - log \beta_i (1 - f_i) \right) \\ s.t. \quad & f_{en,i} + \sum_j f_{j,i} = f_i = f_{ex,i} + \sum_j f_{i,j} \qquad \forall i \end{aligned}$$

Min-cost flow problem

- We have the optimization problem

$$\min_{\mathcal{T}} - \sum_{\mathcal{T}_k \in \mathcal{T}} \log P(\mathcal{T}_k) - \sum_i \log p(\mathbf{x}_i | \mathcal{T})$$

- This can be obtained as

$$\begin{aligned} \min_{\mathcal{T}} \quad & \textstyle\sum_{\mathcal{T}_k \in \mathcal{T}} \left(C_{en,k_0} f_{en,k_0} + \sum_j C_{k_j,k_{j+1}} f_{k_j,k_{j+1}} + C_{ex,k_{l_k}} f_{ex,k_{l_k}} \right) + \\ & + \textstyle\sum_i \left(-\log(1-\beta_i) f_i - log \beta_i (1 - f_i) \right) \\ s.t. \quad & f_{en,i} + \textstyle\sum_j f_{j,i} = f_i = f_{ex,i} + \sum_j f_{i,j} \qquad \forall i \end{aligned}$$

- Which can be reformulated as

$$\begin{aligned} \min_{\mathcal{T}} \quad & \textstyle\sum_i C_{en,i} f_{en,i} + \sum_{i,j} C_{i,j} f_{i,j} + \sum_i C_{ex,i} f_{ex,i} + \sum_i C_i f_i \\ s.t. \quad & f_{en,i} + \textstyle\sum_j f_{j,i} = f_i = f_{ex,i} + \sum_j f_{i,j} \qquad \forall i \end{aligned}$$

Min-cost flow problem

- We have the optimization problem

$$\min_{\mathcal{T}} - \sum_{\mathcal{T}_k \in \mathcal{T}} \log P(\mathcal{T}_k) - \sum_i \log p(\mathbf{x}_i | \mathcal{T})$$

- This can be obtained as

$$\begin{aligned} \min_{\mathcal{T}} \quad & \textstyle\sum_{\mathcal{T}_k \in \mathcal{T}} \left(C_{en,k_0} f_{en,k_0} + \sum_j C_{k_j,k_{j+1}} f_{k_j,k_{j+1}} + C_{ex,k_{l_k}} f_{ex,k_{l_k}} \right) + \\ & + \textstyle\sum_i \left(-\log(1-\beta_i) f_i - log \beta_i (1 - f_i) \right) \\ s.t. \quad & f_{en,i} + \textstyle\sum_j f_{j,i} = f_i = f_{ex,i} + \sum_j f_{i,j} \qquad \forall i \end{aligned}$$

- Which can be reformulated as

$$\begin{aligned} \min_{\mathcal{T}} \quad & \textstyle\sum_i C_{en,i} f_{en,i} + \sum_{i,j} C_{i,j} f_{i,j} + \sum_i C_{ex,i} f_{ex,i} + \sum_i C_i f_i \\ s.t. \quad & f_{en,i} + \textstyle\sum_j f_{j,i} = f_i = f_{ex,i} + \sum_j f_{i,j} \qquad \forall i \end{aligned}$$

- What are the relationships between the costs and the probabilities we had before?

Min-cost flow problem

- We have the optimization problem

$$\min_{\mathcal{T}} - \sum_{\mathcal{T}_k \in \mathcal{T}} \log P(\mathcal{T}_k) - \sum_i \log p(\mathbf{x}_i | \mathcal{T})$$

- This can be obtained as

$$\min_{\mathcal{T}} \quad \sum_{\mathcal{T}_k \in \mathcal{T}} \left(C_{en,k_0} f_{en,k_0} + \sum_j C_{k_j,k_{j+1}} f_{k_j,k_{j+1}} + C_{ex,k_{l_k}} f_{ex,k_{l_k}} \right) +$$
$$+ \sum_i \left(-\log(1-\beta_i) f_i - log \beta_i (1 - f_i) \right)$$
$$s.t. \quad f_{en,i} + \sum_j f_{j,i} = f_i = f_{ex,i} + \sum_j f_{i,j} \qquad \forall i$$

- Which can be reformulated as

$$\min_{\mathcal{T}} \quad \sum_i C_{en,i} f_{en,i} + \sum_{i,j} C_{i,j} f_{i,j} + \sum_i C_{ex,i} f_{ex,i} + \sum_i C_i f_i$$
$$s.t. \quad f_{en,i} + \sum_j f_{j,i} = f_i = f_{ex,i} + \sum_j f_{i,j} \qquad \forall i$$

- What are the relationships between the costs and the probabilities we had before?

Mapping to Min cost-flow network

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

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

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

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

Mapping to Min cost-flow network

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

$$\min_{\mathcal{T}} \quad \sum_i C_{en,i} f_{en,i} + \sum_{i,j} C_{i,j} f_{i,j} + \sum_i C_{ex,i} f_{ex,i} + \sum_i C_i f_i$$
$$s.t. \quad f_{en,i} + \sum_j f_{j,i} = f_i = f_{ex,i} + \sum_j f_{i,j} \qquad \forall i$$

- For every observation $\mathbf{x}_i \in \mathcal{X}$ 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

$$\min_{\mathcal{T}} \quad \sum_i C_{en,i} f_{en,i} + \sum_{i,j} C_{i,j} f_{i,j} + \sum_i C_{ex,i} f_{ex,i} + \sum_i C_i f_i$$
$$s.t. \quad f_{en,i} + \sum_j f_{j,i} = f_i = f_{ex,i} + \sum_j f_{i,j} \qquad \forall i$$

- For every observation $\mathbf{x}_i \in \mathcal{X}$ 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 $p_{link}(\mathbf{x}_j|\mathbf{x}_i) \neq 0$, 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

$$\begin{aligned} \min_{\mathcal{T}} \quad & \sum_i C_{en,i} f_{en,i} + \sum_{i,j} C_{i,j} f_{i,j} + \sum_i C_{ex,i} f_{ex,i} + \sum_i C_i f_i \\ s.t. \quad & f_{en,i} + \sum_j f_{j,i} = f_i = f_{ex,i} + \sum_j f_{i,j} \qquad \forall i \end{aligned}$$

- For every observation $\mathbf{x}_i \in \mathcal{X}$ 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 $p_{link}(\mathbf{x}_j | \mathbf{x}_i) \neq 0$, 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

Mapping to Min cost-flow network

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

$$\begin{aligned} \min_{\mathcal{T}} \quad & \sum_i C_{en,i} f_{en,i} + \sum_{i,j} C_{i,j} f_{i,j} + \sum_i C_{ex,i} f_{ex,i} + \sum_i C_i f_i \\ s.t. \quad & f_{en,i} + \sum_j f_{j,i} = f_i = f_{ex,i} + \sum_j f_{i,j} \qquad \forall i \end{aligned}$$

- For every observation $\mathbf{x}_i \in \mathcal{X}$ 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 $p_{link}(\mathbf{x}_j|\mathbf{x}_i) \neq 0$, 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

$$\min_{\mathcal{T}} \quad \sum_i C_{en,i} f_{en,i} + \sum_{i,j} C_{i,j} f_{i,j} + \sum_i C_{ex,i} f_{ex,i} + \sum_i C_i f_i$$
$$s.t. \quad f_{en,i} + \sum_j f_{j,i} = f_i = f_{ex,i} + \sum_j f_{i,j} \qquad \forall i$$

- For every observation $\mathbf{x}_i \in \mathcal{X}$ 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 $p_{link}(\mathbf{x}_j|\mathbf{x}_i) \neq 0$, 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.

Mapping to Min cost-flow network

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

$$\min_{\mathcal{T}} \quad \sum_i C_{en,i} f_{en,i} + \sum_{i,j} C_{i,j} f_{i,j} + \sum_i C_{ex,i} f_{ex,i} + \sum_i C_i f_i$$
$$s.t. \quad f_{en,i} + \sum_j f_{j,i} = f_i = f_{ex,i} + \sum_j f_{i,j} \qquad \forall i$$

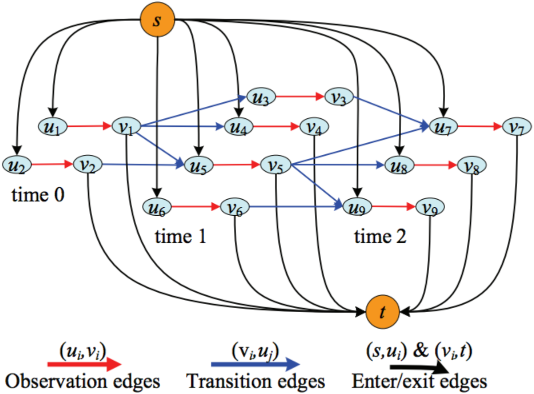
- For every observation $\mathbf{x}_i \in \mathcal{X}$ create two nodes u_i, v_i, and an arc with cost $c(u_i, v_j) = C_i$ and flow f_i.
- Add arcs $c(s, u_i) = C_{en,i}$ and flow $f_{en,i}$, as well as $c(t, u_i) = C_{ex,i}$ and flow $f_{ex,i}$
- For every transition $p_{link}(\mathbf{x}_j|\mathbf{x}_i) \neq 0$, 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.

Mapping to Min cost-flow network

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

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

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



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 $\mathcal{O}(\log n)$

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

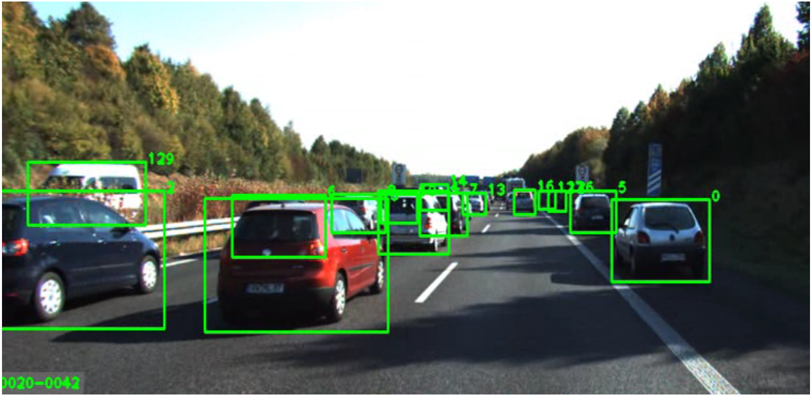
  - Construct the graph $G(V, E, C, f)$ from observation set $\mathcal{X}$
  - Start with empty flow
  - WHILE ($f(G)$ can be augmented)
    - Augment $f(G)$ by one.
    - Find the min cost flow by the algorithm of [12].
    - IF (current min cost < global optimal cost)

      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 $\mathcal{O}(\log n)$

Tracking Results

[L. Zhang, Y. Li and R. Nevatia, CVPR08]



- 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
- Wathershed transform
- Mean-shift

Simple K-means

- Find three clusters in this data

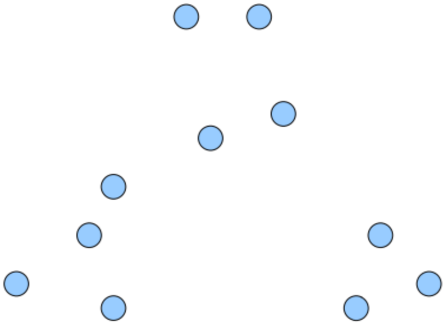


Figure: From M. Tappen

Simple K-means

- Find three clusters in this data

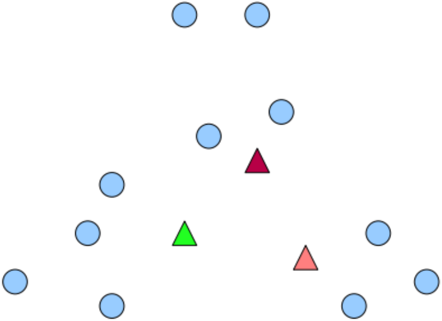


Figure: From M. Tappen

Simple K-means

- Find three clusters in this data

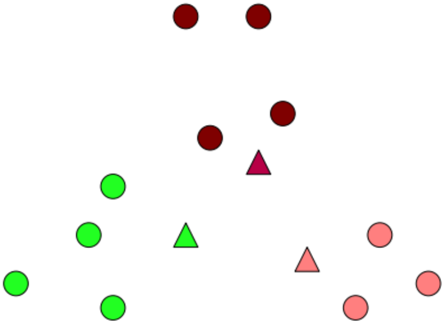


Figure: From M. Tappen

Simple K-means

- Find three clusters in this data

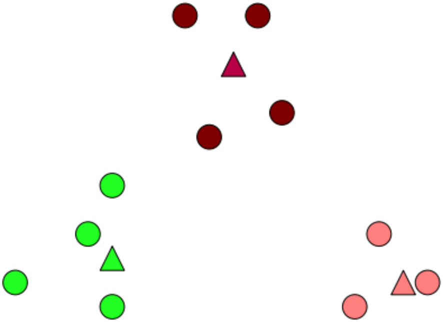


Figure: From M. Tappen

K-means style algorithms

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

K-means style algorithms

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

K-means style algorithms

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

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

K-means style algorithms

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

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

- The problem becomes

$$\min_{\mathbf{S},\mu,\mathbf{c}} \sum_{\mathbf{p}} E(\mathbf{p}, s_p, \mu_{s_p}, c_{s_p}).$$

K-means style algorithms

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

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

- The problem becomes

$$\min_{\mathbf{S},\mu,\mathbf{c}} \sum_{\mathbf{p}} E(\mathbf{p}, s_p, \mu_{s_p}, c_{s_p}).$$

K-means style algorithms

- We can define the total energy of a pixel as

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

- The problem becomes

$$\min_{\mathbf{S},\mu,\mathbf{c}} \sum_{\mathbf{p}} E(\mathbf{p}, s_p, \mu_{s_p}, c_{s_p}).$$

- Simple iterative algorithm:
  - Solve for the assignments $\mathbf{S}$
  - Solve in parallel for the positions μ and appearances $\mathbf{c}$

K-means style algorithms

- We can define the total energy of a pixel as

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

- The problem becomes

$$\min_{\mathbf{S},\mu,\mathbf{c}} \sum_{\mathbf{p}} E(\mathbf{p}, s_p, \mu_{s_p}, c_{s_p}).$$

- Simple iterative algorithm:
  - Solve for the assignments **S**
  - Solve in parallel for the positions μ and appearances **c**
- Is this easy to do?

K-means style algorithms

- We can define the total energy of a pixel as

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

- The problem becomes

$$\min_{\mathbf{S},\mu,\mathbf{c}} \sum_{\mathbf{p}} E(\mathbf{p}, s_p, \mu_{s_p}, c_{s_p}).$$

- Simple iterative algorithm:
  - Solve for the assignments **S**
  - Solve in parallel for the positions μ and appearances **c**
- Is this easy to do?

Results

[R. Achanta and A. Shaji and K. Smith and A. Lucchi and P. Fua and S. Susstrunk, PAMI12]

Joint Segmentation and Depth Estimation

- Let $\mathbf{S} = \{s_1, \cdots, s_m)$ 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 $\mathbf{S} = \{s_1, \cdots, s_m)$ 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 $\mathbf{c} = \{c_1, \cdots, c_m\}$ as the mean appearance descriptor.

Joint Segmentation and Depth Estimation

- Let $\mathbf{S} = \{s_1, \cdots, s_m)$ 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 $\mathbf{c} = \{c_1, \cdots, c_m\}$ as the mean appearance descriptor.
- We can define the total energy of a pixel as

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

Joint Segmentation and Depth Estimation

- Let $\mathbf{S} = \{s_1, \cdots, s_m)$ 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 $\mathbf{c} = \{c_1, \cdots, c_m\}$ as the mean appearance descriptor.
- We can define the total energy of a pixel as

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

Joint Segmentation and Depth Estimation

- Let $\mathbf{S} = \{s_1, \cdots, s_m)$ 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 $\mathbf{c} = \{c_1, \cdots, c_m\}$ as the mean appearance descriptor.
- We can define the total energy of a pixel as

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

- We can use:

$$E_{pos}(\mathbf{p}, \mu_{s_p}) = ||\mathbf{p} - \mu_{s_p}||_2^2 / g \qquad E_{col}(\mathbf{p}, c_{s_p} = (I_t(\mathbf{p}) - c_{s_p})^2$$

and

$$E_{disp}(\mathbf{p}, \theta_{s_p}) = \begin{cases} (d(\mathbf{p}, \theta_{s_p}) - \hat{d}(\mathbf{p}))^2 & \text{if } \mathbf{p} \in \mathcal{F} \\ \lambda & \text{otherwise} \end{cases}$$

Joint Segmentation and Depth Estimation

- Let $\mathbf{S} = \{s_1, \cdots, s_m)$ 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 $\mathbf{c} = \{c_1, \cdots, c_m\}$ as the mean appearance descriptor.
- We can define the total energy of a pixel as

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

- We can use:

$$E_{pos}(\mathbf{p}, \mu_{s_p}) = ||\mathbf{p} - \mu_{s_p}||_2^2 / g \qquad E_{col}(\mathbf{p}, c_{s_p} = (I_t(\mathbf{p}) - c_{s_p})^2$$

and

$$E_{disp}(\mathbf{p}, \theta_{s_p}) = \begin{cases} (d(\mathbf{p}, \theta_{s_p}) - \hat{d}(\mathbf{p}))^2 & \text{if } \mathbf{p} \in \mathcal{F} \\ \lambda & \text{otherwise} \end{cases}$$

Joint Segmentation and Depth Estimation

- We can define the total energy of a pixel as

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

- The problem of joint unsupervised segmentation and flow estimation becomes

$$\min_{\Theta, \mathbf{S}, \mu, \mathbf{c}} \sum_{\mathbf{p}} E(\mathbf{p}, s_p, \theta_{s_p}, \mu_{s_p}, c_{s_p}).$$

Joint Segmentation and Depth Estimation

- We can define the total energy of a pixel as

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

- The problem of joint unsupervised segmentation and flow estimation becomes

$$\min_{\Theta, \mathbf{S}, \mu, \mathbf{c}} \sum_{\mathbf{p}} E(\mathbf{p}, s_p, \theta_{s_p}, \mu_{s_p}, c_{s_p}).$$

- Simple iterative algorithm
  - Solve for the assignments $\mathbf{S}$
  - Solve in parallel for the planes Θ, positions μ and appearances $\mathbf{c}$

Joint Segmentation and Depth Estimation

- We can define the total energy of a pixel as

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

- The problem of joint unsupervised segmentation and flow estimation becomes

$$\min_{\Theta, \mathbf{S}, \mu, \mathbf{c}} \sum_{\mathbf{p}} E(\mathbf{p}, s_p, \theta_{s_p}, \mu_{s_p}, c_{s_p}).$$

- Simple iterative algorithm
  - Solve for the assignments **S**
  - Solve in parallel for the planes Θ, positions μ and appearances **c**
- How do we do this?

Joint Segmentation and Depth Estimation

- We can define the total energy of a pixel as

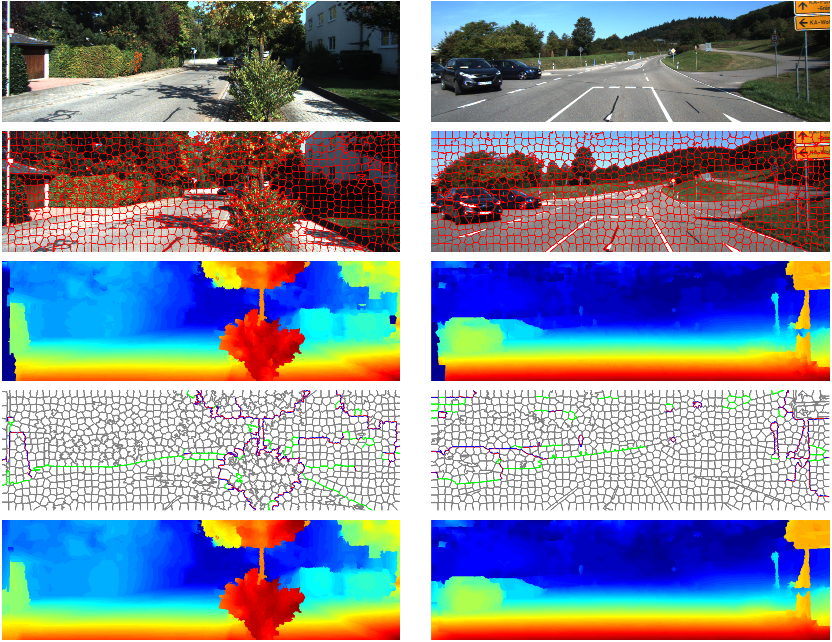
$$E(p) = E_{\text{col}}^{l,r}(\mathbf{p}, c_{s_p}, \theta_{s_p}) + \lambda_{\text{pos}} E_{\text{pos}}(\mathbf{p}, \mu_{s_p}) + \lambda_{\text{disp}} E_{\text{disp}}^{l,r}(\mathbf{p}, \theta_{s_p}),$$

- The problem of joint unsupervised segmentation and flow estimation becomes

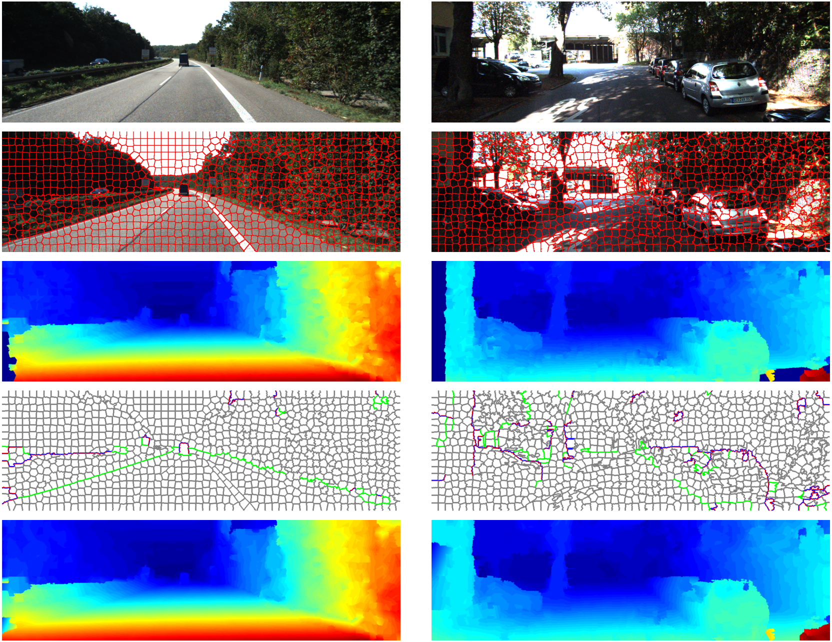
$$\min_{\Theta, \mathbf{S}, \mu, \mathbf{c}} \sum_{\mathbf{p}} E(\mathbf{p}, s_p, \theta_{s_p}, \mu_{s_p}, c_{s_p}).$$

- Simple iterative algorithm
  - Solve for the assignments $\mathbf{S}$
  - Solve in parallel for the planes Θ, positions μ and appearances $\mathbf{c}$
- How do we do this?

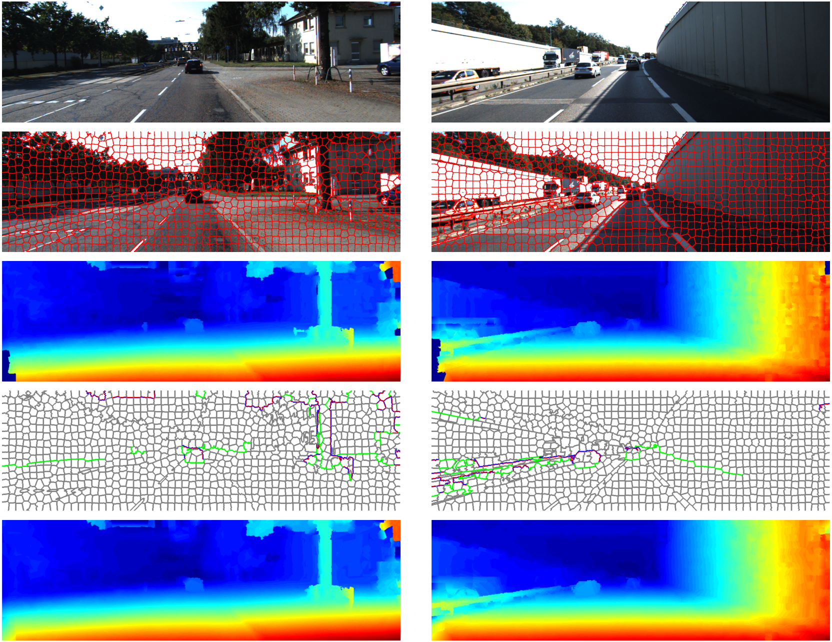
[K. Yamaguchi, D. McAllester and R. Urtasun, CVPR13]



[K. Yamaguchi, D. McAllester and R. Urtasun, CVPR13]



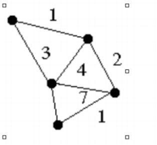
[K. Yamaguchi, D. McAllester and R. Urtasun, CVPR13]



Techniques we will see

- K-means style clustering, e.g., SLIC superpixels
- Normalized cuts
- Graph-based superpixels
- Wathershed transform
- Mean-shift

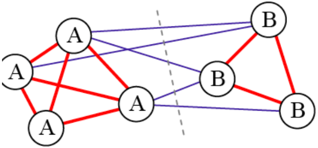
Segmentation as a mincut problem



$$\begin{bmatrix} 0 & 1 & 3 & \infty & \infty \\ 1 & 0 & 4 & \infty & 2 \\ 3 & 4 & 0 & 6 & 7 \\ \infty & \infty & 6 & 0 & 1 \\ \infty & 2 & 7 & 1 & 0 \end{bmatrix}$$

Weight Matrix: W

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



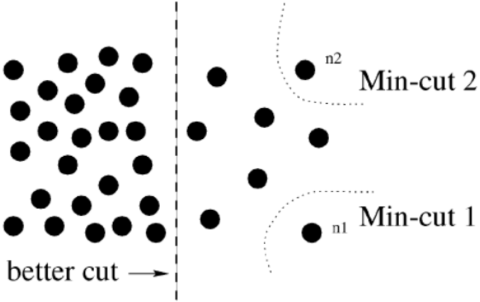
- The cut separate the nodes into two groups

Minimun Cuts

- 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

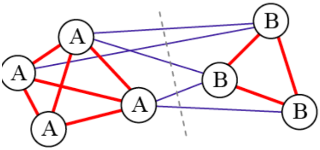
Normalized Cuts

[J. Shi and J. Malik, PAMI00]

- Better measure is the normalized cuts

$$N_{cut}(A,B) = \frac{cut(A,B)}{assoc(A,V)} + \frac{cut(A,B)}{assoc(B,V)}$$

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



	A	B	sum
A	$assoc(A,A)$	$cut(A,B)$	$assoc(A,V)$
B	$cut(B,A)$	$assoc(B,B)$	$assoc(B,V)$
sum	$assoc(A,V)$	$assoc(B,v)$	

- 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 $\mathbf{x}$ be an indicator vector, with $x_i = 1$ if $x_i \in A$, and $x_i = -1$ otherwise. Let $\mathbf{d} = \mathbf{W}1$ be the row sums of the symmetric matrix $\mathbf{W}$, and $\mathbf{D} = diag(\mathbf{d})$ 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 $\mathbf{d} = \mathbf{W}1$ be the row sums of the symmetric matrix $\mathbf{W}$, and $\mathbf{D} = diag(\mathbf{d})$ be the corresponding diagonal matrix.
- Shi and Malik, compute the cut by solving

$$\min_{\mathbf{y}} \frac{\mathbf{y}^T(\mathbf{D} - \mathbf{W})\mathbf{y}}{\mathbf{y}^T\mathbf{D}\mathbf{y}}$$

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

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 $\mathbf{d} = \mathbf{W}1$ be the row sums of the symmetric matrix $\mathbf{W}$, and $\mathbf{D} = diag(\mathbf{d})$ be the corresponding diagonal matrix.
- Shi and Malik, compute the cut by solving

$$\min_{\mathbf{y}} \frac{\mathbf{y}^T(\mathbf{D} - \mathbf{W})\mathbf{y}}{\mathbf{y}^T\mathbf{D}\mathbf{y}}$$

  with $\mathbf{y} = ((1 + \mathbf{x}) - b(1 - \mathbf{x}))/2$ is a vector with all 1's and -b's such that $\mathbf{y} \cdot \mathbf{d} = 0$, by relaxing $\mathbf{y}$ to be real value.
- $\mathbf{D} - \mathbf{W}$ is the Laplacian

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 $\mathbf{d} = \mathbf{W}1$ be the row sums of the symmetric matrix $\mathbf{W}$, and $\mathbf{D} = diag(\mathbf{d})$ be the corresponding diagonal matrix.
- Shi and Malik, compute the cut by solving

$$\min_{\mathbf{y}} \frac{\mathbf{y}^T(\mathbf{D} - \mathbf{W})\mathbf{y}}{\mathbf{y}^T\mathbf{D}\mathbf{y}}$$

  with $\mathbf{y} = ((1 + \mathbf{x}) - b(1 - \mathbf{x}))/2$ is a vector with all 1's and -b's such that $\mathbf{y} \cdot \mathbf{d} = 0$, by relaxing $\mathbf{y}$ to be real value.
- $\mathbf{D} - \mathbf{W}$ is the Laplacian

Solving for the cut

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

$$(\mathbf{D} - \mathbf{W})\mathbf{y} = \lambda \mathbf{D}\mathbf{y}$$

- This is a normal eigenvalue problem

$$(\mathbf{I} - \mathbf{N})\mathbf{z} = \lambda \mathbf{z}$$

with $\mathbf{N} = \mathbf{D}^{1/2}\mathbf{W}\mathbf{D}^{1/2}$ is the normalized affinity matrix, and $\mathbf{z} = \mathbf{D}^{1/2}\mathbf{y}$.

Solving for the cut

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

$$(\mathbf{D} - \mathbf{W})\mathbf{y} = \lambda \mathbf{D}\mathbf{y}$$

- This is a normal eigenvalue problem

$$(\mathbf{I} - \mathbf{N})\mathbf{z} = \lambda \mathbf{z}$$

with $\mathbf{N} = \mathbf{D}^{1/2}\mathbf{W}\mathbf{D}^{1/2}$ is the normalized affinity matrix, and $\mathbf{z} = \mathbf{D}^{1/2}\mathbf{y}$.

- 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

$$(\mathbf{D} - \mathbf{W})\mathbf{y} = \lambda \mathbf{D}\mathbf{y}$$

- This is a normal eigenvalue problem

$$(\mathbf{I} - \mathbf{N})\mathbf{z} = \lambda \mathbf{z}$$

with $\mathbf{N} = \mathbf{D}^{1/2}\mathbf{W}\mathbf{D}^{1/2}$ is the normalized affinity matrix, and $\mathbf{z} = \mathbf{D}^{1/2}\mathbf{y}$.

- 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
$$(\mathbf{D} - \mathbf{W})\mathbf{y} = \lambda \mathbf{D}\mathbf{y}$$
- This is a normal eigenvalue problem
$$(\mathbf{I} - \mathbf{N})\mathbf{z} = \lambda \mathbf{z}$$
with $\mathbf{N} = \mathbf{D}^{1/2}\mathbf{W}\mathbf{D}^{1/2}$ is the normalized affinity matrix, and $\mathbf{z} = \mathbf{D}^{1/2}\mathbf{y}$.
- 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
$$w_{i,j} = \exp\left(-\frac{||\mathbf{F}_i - \mathbf{F}_j||_2^2}{\sigma_f^2} - \frac{||p_i - p_j||_2^2}{\sigma_s^2}\right)$$
for pixels within a radious $||p_i - p_j||_2 < r$, and $\mathbf{F}$ 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
$$(\mathbf{D} - \mathbf{W})\mathbf{y} = \lambda \mathbf{D}\mathbf{y}$$
- This is a normal eigenvalue problem
$$(\mathbf{I} - \mathbf{N})\mathbf{z} = \lambda \mathbf{z}$$
with $\mathbf{N} = \mathbf{D}^{1/2}\mathbf{W}\mathbf{D}^{1/2}$ is the normalized affinity matrix, and $\mathbf{z} = \mathbf{D}^{1/2}\mathbf{y}$.
- 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
$$w_{i,j} = \exp\left(-\frac{||\mathbf{F}_i - \mathbf{F}_j||_2^2}{\sigma_f^2} - \frac{||p_i - p_j||_2^2}{\sigma_s^2}\right)$$
for pixels within a radious $||p_i - p_j||_2 < r$, and $\mathbf{F}$ is a feature vector with color, intensities, histograms, gradients, etc.

Algorithm

[J. Shi and J. Malik, PAMI00]

1. Given an image or image sequence, set up a weighted graph $\mathbf{G} = (\mathbf{V}, \mathbf{E})$ and set the weight on the edge connecting two nodes to be a measure of the similarity between the two nodes.
2. Solve $(\mathbf{D} - \mathbf{W})\boldsymbol{x} = \lambda \mathbf{D}\boldsymbol{x}$ 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

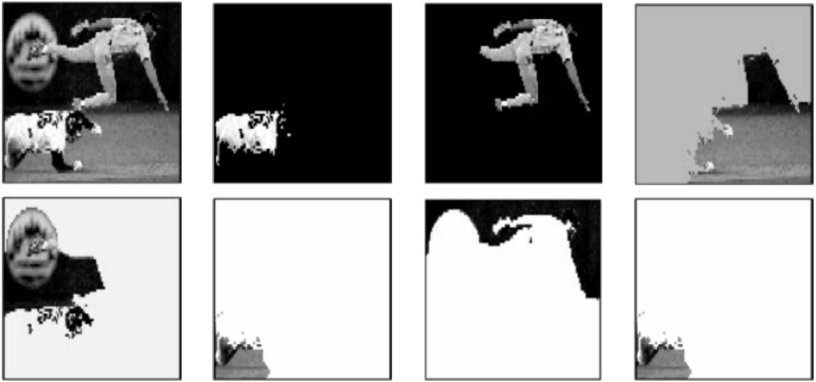


Figure: Shi and Malik N-Cuts