Tracking and Grouping

Raquel Urtasun

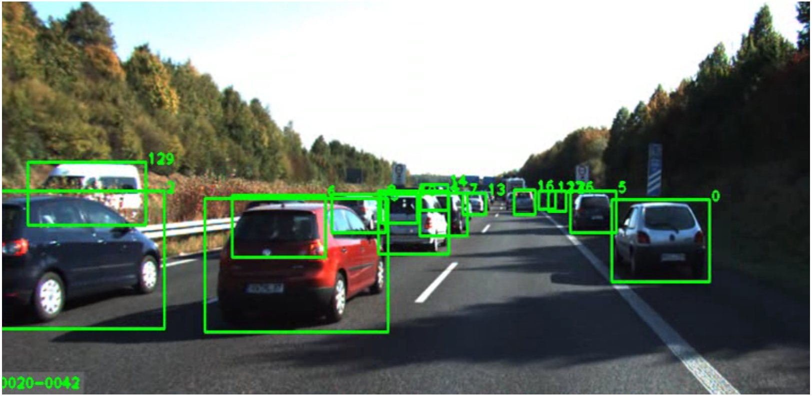
TTI Chicago

March 7, 2013

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

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

- 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

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

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

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

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

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

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

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

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

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

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

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

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

- Which can be reformulated as

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

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

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

- Which can be reformulated as

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

- 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

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

- 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?

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

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

- 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

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.

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.
- 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

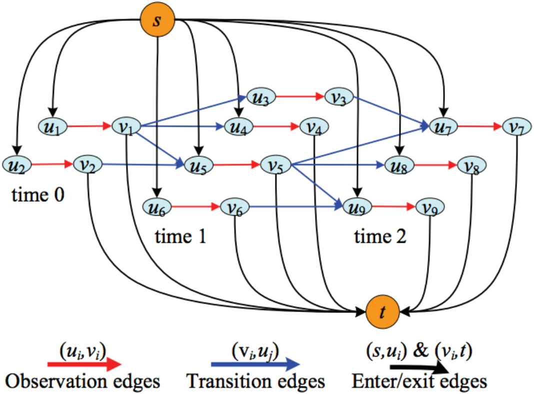
$$\min_{\mathcal{T}} \quad \sum_i C_{en,i} f_{en,i} + \sum_{i,j} C_{i,j} f_{i,j} + \sum_i C_{ex,i} f_{ex,i} + \sum_i C_i f_i$$
$$s.t. \quad f_{en,i} + \sum_j f_{j,i} = f_i = f_{ex,i} + \sum_j f_{i,j} \qquad \forall i$$

- 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

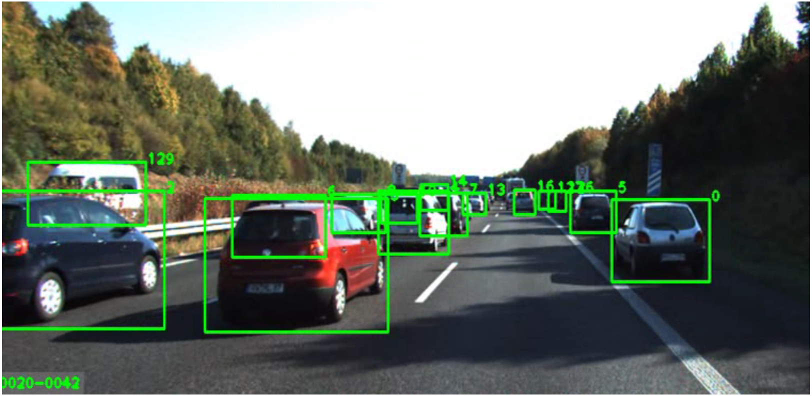
  - Construct the graph $G(V, E, C, f)$ from observation set $\mathcal{X}$
  - Start with empty flow
  - WHILE ($f(G)$ can be augmented)
    - Augment $f(G)$ by one.
    - Find the min cost flow by the algorithm of [12].
    - IF (current min cost < global optimal cost)

      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

Gestalt "Theory"

There exist a variety of factors in grouping

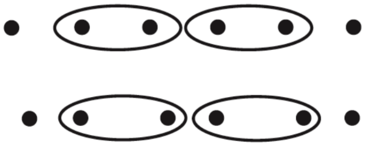
- **Proximity:** Tokens that are nearby tend to be grouped together

- **Similarity:** Similar tokens tend to be grouped together

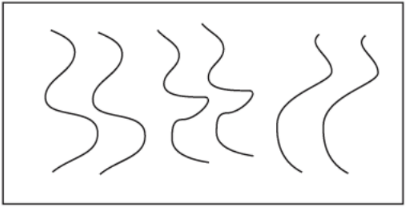
Gestalt "Theory"

There exist a variety of factors in grouping

- **Common fate:** Tokens with coherent motion tend to be grouped together
- **Common region:** Tokens that lie inside the same closed region tend to be group together



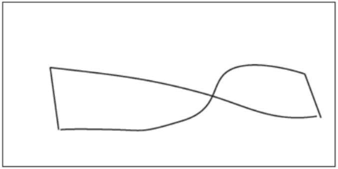
- **Parallelism:** Parallel curves or tokens tend to be grouped together



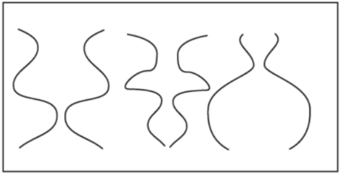
Gestalt "Theory"

There exist a variety of factors in grouping

- **Closure:** Tokens or curves that tend to lead to closed curves tend to be close together



- **Symmetry:** Curves that lead to symmetric groups are typically grouped together



Gestalt "Theory"

There exist a variety of factors in grouping

- **Continuity:** Tokens than lead to continuous (with a relax notion of continuity) curves tend to be grouped
- **Familiar Configuration:** Tokens that, when grouped, lead to a familiar object tend to be grouped together

Effects of Grouping

- Grouping makes you see hallucinate contours

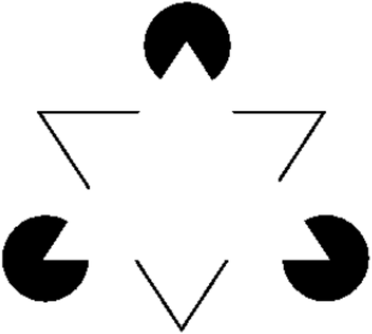


Figure: Kanizsa Triangle

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

Motivation of clustering

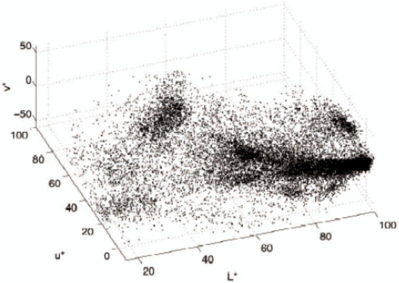


Figure: Illustration from Comanciu and Meer

Example of grouping techniques

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

Simple K-means

- Find three clusters in this data

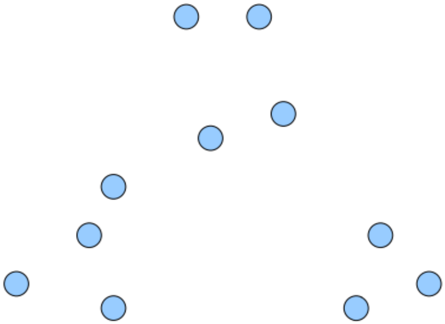


Figure: From M. Tappen

Simple K-means

- Find three clusters in this data

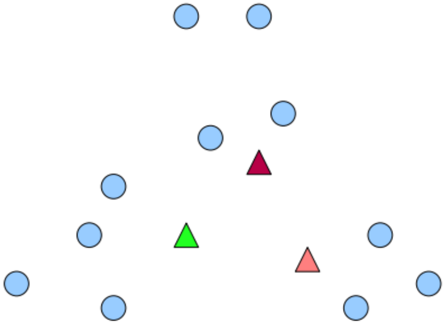


Figure: From M. Tappen

Simple K-means

- Find three clusters in this data

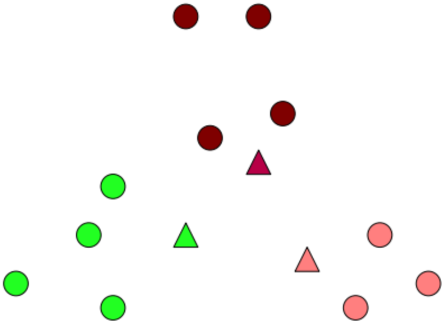


Figure: From M. Tappen

Simple K-means

- Find three clusters in this data

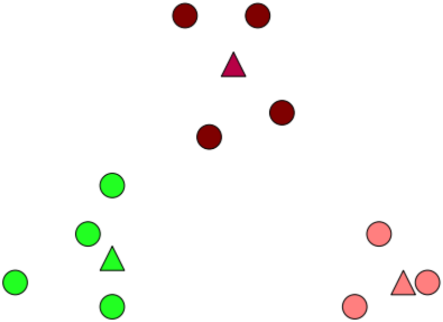


Figure: From M. Tappen

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

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

- 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 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**

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 **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

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

- 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?

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

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

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

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

- 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

- We can define the total energy of a pixel as

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

- 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

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

- 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 $\mathbf{c}$

- 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?

Joint Segmentation and Depth Estimation

- We can define the total energy of a pixel as

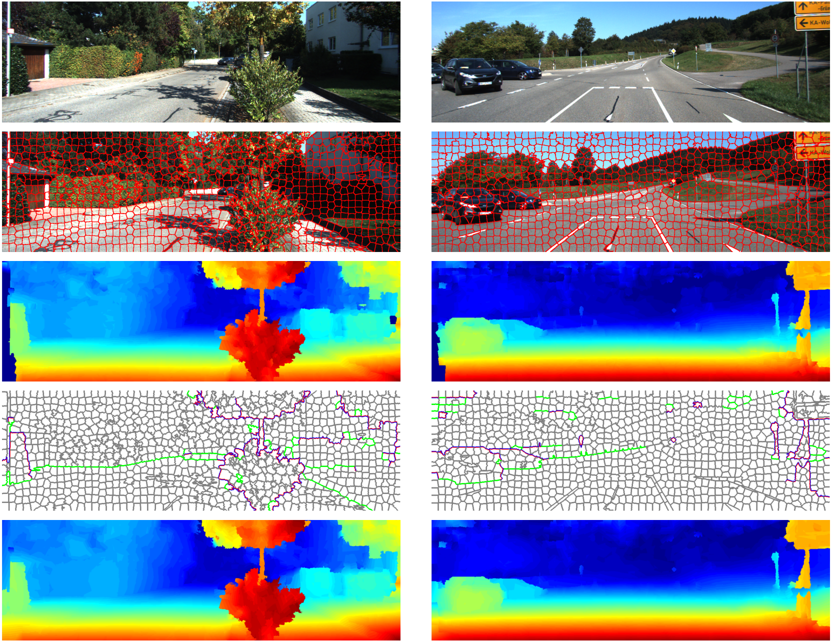
$$E(p) = E^{l,r}_{\mathrm{col}}(\mathbf{p}, c_{s_p}, \theta_{s_p}) + \lambda_{\mathrm{pos}} E_{\mathrm{pos}}(\mathbf{p}, \mu_{s_p}) + \lambda_{\mathrm{disp}} E^{l,r}_{\mathrm{disp}}(\mathbf{p}, \theta_{s_p}),$$

- The problem of joint unsupervised segmentation and flow estimation becomes

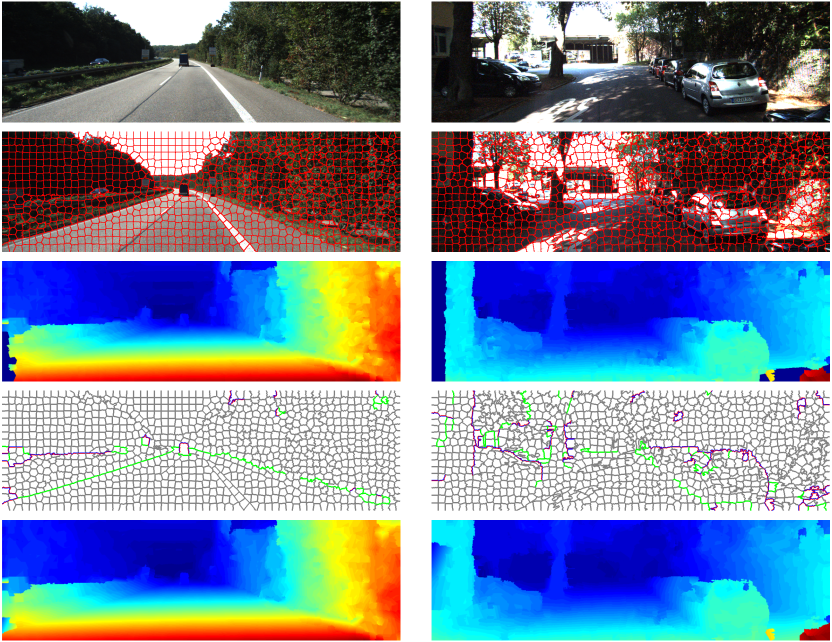
$$\min_{\Theta, \mathbf{S}, \mu, \mathbf{c}} \sum_{\mathbf{p}} E(\mathbf{p}, s_p, \theta_{s_p}, \mu_{s_p}, c_{s_p}).$$

- 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?

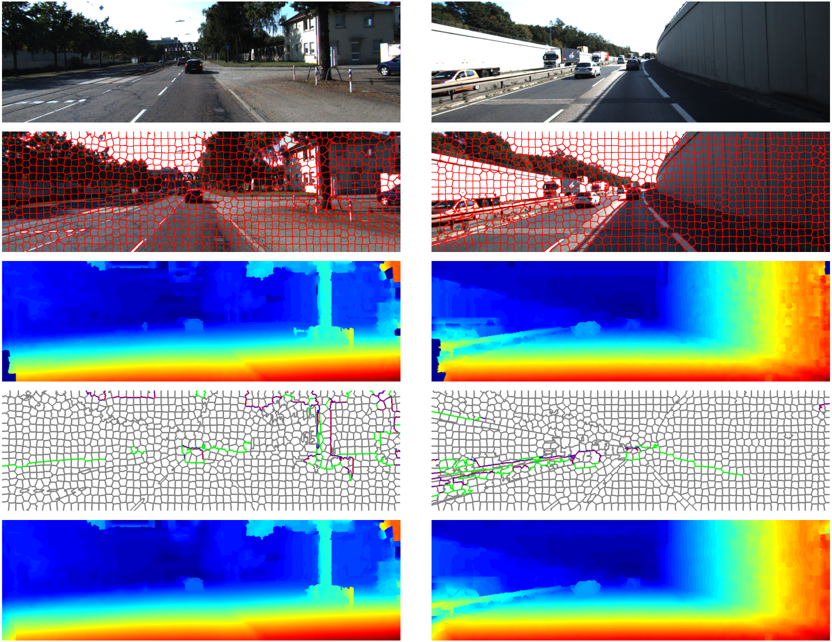
[K. Yamaguchi, D. McAllester and R. Urtasun, CVPR13]



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

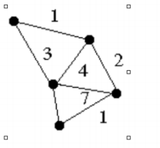


[K. Yamaguchi, D. McAllester and R. Urtasun, CVPR13]



Example of grouping techniques

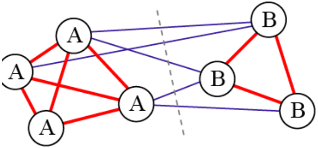
- K-means style clustering, e.g., SLIC superpixels
- Normalized cuts
- Graph-based superpixels
- Mean-shift
- Watershed transform



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

Weight Matrix: W

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



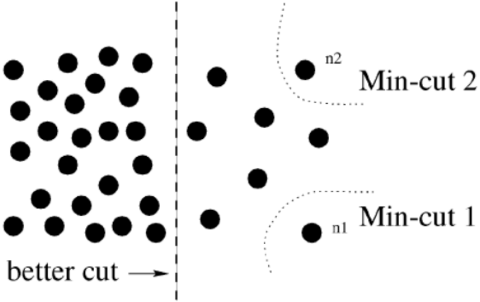
- The cut separate the nodes into two groups

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

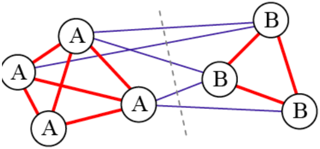
Normalized Cuts

[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

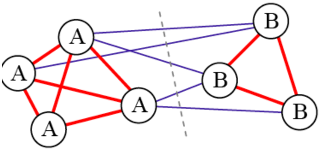
Normalized Cuts

[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

Normalized Cuts

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

Normalized Cuts

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

Normalized Cuts

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

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

relaxing $\mathbf{y}$ to be real-value

Normalized Cuts

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

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

  relaxing $\mathbf{y}$ to be real-value
- $\mathbf{D} - \mathbf{W}$ is the Laplacian

Normalized Cuts

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

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

  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 radius $||p_i - p_j||_2 < r$, 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 radius $||p_i - p_j||_2 < r$, and **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

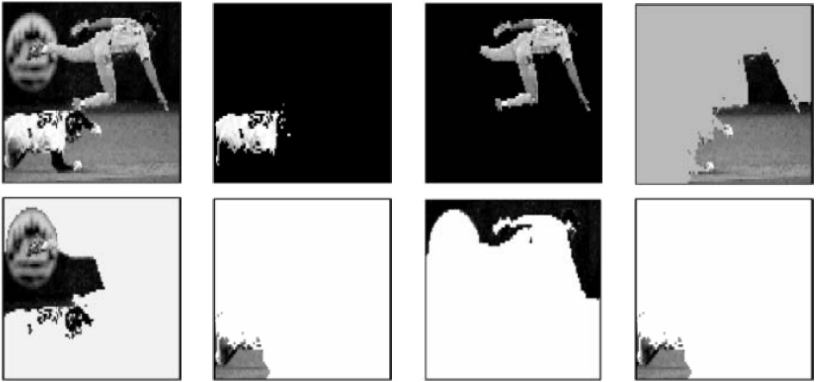


Figure: Shi and Malik N-Cuts

Example of grouping techniques

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

Graph-based Superpixels

- Construct a graph that has as many nodes as pixels
- Define edges between neighboring pixels, with whatever definition of neighboring

Graph-based Superpixels

- Construct a graph that has as many nodes as pixels
- Define edges between neighboring pixels, with whatever definition of neighboring
- Define the weight between neighbors to be the dissimilarity between them (a non-negative measure)

Graph-based Superpixels

- Construct a graph that has as many nodes as pixels
- Define edges between neighboring pixels, with whatever definition of neighboring
- Define the weight between neighbors to be the dissimilarity between them (a non-negative measure)
- Let $G(V, E)$ be the graph, we want to segment V into components $(C_1, \cdots, C_r)$

Graph-based Superpixels

- Construct a graph that has as many nodes as pixels
- Define edges between neighboring pixels, with whatever definition of neighboring
- Define the weight between neighbors to be the dissimilarity between them (a non-negative measure)
- Let $G(V, E)$ be the graph, we want to segment V into components $(C_1, \cdots, C_r)$
- Felzenswald and Hutterlocker defined a simple greedy algorithm which can be shown to have some interesting global properties

Graph-based Superpixels

- Construct a graph that has as many nodes as pixels
- Define edges between neighboring pixels, with whatever definition of neighboring
- Define the weight between neighbors to be the dissimilarity between them (a non-negative measure)
- Let $G(V, E)$ be the graph, we want to segment V into components $(C_1, \cdots, C_r)$
- Felzenswald and Hutterlocker defined a simple greedy algorithm which can be shown to have some interesting global properties

Algorithm

1. Sort E into $\pi = (o_1, \cdots, o_m)$ by non-decreasing weights
2. Start with segmentation S^0, where each vertex is its own component (i.e., as many superpixels as pixels)

Algorithm

1. Sort E into $\pi = (o_1, \cdots, o_m)$ by non-decreasing weights
2. Start with segmentation S^0, where each vertex is its own component (i.e., as many superpixels as pixels)
3. Repeat step 4 for $q = 1, \cdots, q = m$

Algorithm

1. Sort E into $\pi = (o_1, \cdots, o_m)$ by non-decreasing weights
2. Start with segmentation S^0, where each vertex is its own component (i.e., as many superpixels as pixels)
3. Repeat step 4 for $q = 1, \cdots, q = m$
4. Construct S^q given S^{q-1} as follows. Let $o_q = (v_i, v_j)$. If v_i and v_j are disjoint components of S^{q-1} and $w(o_q)$ is small compared to the internal difference of both components of S^{q-1}, then merge the two components. Otherwise do nothing $S^q = S^{q-1}$

Algorithm

1. Sort E into $\pi = (o_1, \cdots, o_m)$ by non-decreasing weights
2. Start with segmentation S^0, where each vertex is its own component (i.e., as many superpixels as pixels)
3. Repeat step 4 for $q = 1, \cdots, q = m$
4. Construct S^q given S^{q-1} as follows. Let $o_q = (v_i, v_j)$. If v_i and v_j are disjoint components of S^{q-1} and $w(o_q)$ is small compared to the internal difference of both components of S^{q-1}, then merge the two components. Otherwise do nothing $S^q = S^{q-1}$

The internal difference is defined as the largest weight in the minimum spanning tree of the component.

$$Int(C) = \max_{e \in MST(C,E)} w(e)$$

Algorithm

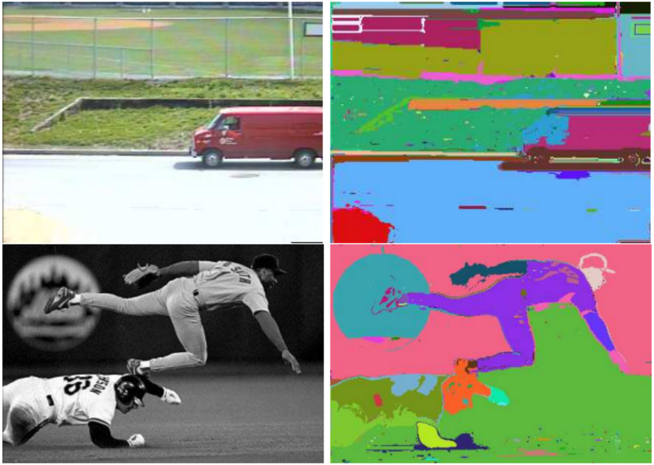
1. Sort E into $\pi = (o_1, \cdots, o_m)$ by non-decreasing weights
2. Start with segmentation S^0, where each vertex is its own component (i.e., as many superpixels as pixels)
3. Repeat step 4 for $q = 1, \cdots, q = m$
4. Construct S^q given S^{q-1} as follows. Let $o_q = (v_i, v_j)$. If v_i and v_j are disjoint components of S^{q-1} and $w(o_q)$ is small compared to the internal difference of both components of S^{q-1}, then merge the two components. Otherwise do nothing $S^q = S^{q-1}$

The internal difference is defined as the largest weight in the minimum spanning tree of the component.

$$Int(C) = \max_{e \in MST(C,E)} w(e)$$

Results

[P. Felzenszwald and D. Huttenlocher, IJCV04]

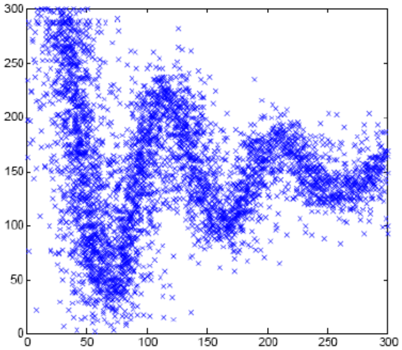


Example of grouping techniques

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

Basics of Kernel Density Estimation

- We have a bunch of points drawn from some distribution
- What's the distribution that generated these points?



[Source: M. Tappen]

Parametric vs Non-Parametric

- We can fit a parametric distribution, e.g., mixture of Gaussians
- KDE idea: Use the data to define the distribution

Parametric vs Non-Parametric

- We can fit a parametric distribution, e.g., mixture of Gaussians
- KDE idea: Use the data to define the distribution
  - If I were to draw more samples from the same probability distribution, then those points would probably be close to the points that I have already drawn

Parametric vs Non-Parametric

- We can fit a parametric distribution, e.g., mixture of Gaussians
- KDE idea: Use the data to define the distribution
  - If I were to draw more samples from the same probability distribution, then those points would probably be close to the points that I have already drawn
  - Build distribution by putting a little mass of probability around each data-point

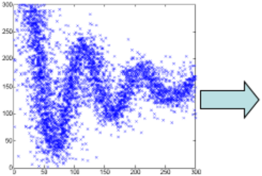
[Source: M. Tappen]

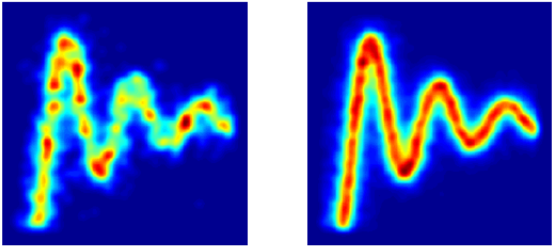
Parametric vs Non-Parametric

- We can fit a parametric distribution, e.g., mixture of Gaussians
- KDE idea: Use the data to define the distribution
  - If I were to draw more samples from the same probability distribution, then those points would probably be close to the points that I have already drawn
  - Build distribution by putting a little mass of probability around each data-point

[Source: M. Tappen]

Example





(a) 2000 Samples (b) 20000 Samples

Figure 2-2: Kernel density estimates of the density function shown in Figure 2-1(a). Figure (a) shows the estimate found with a relatively small number of samples. It is uneven and does not approximate the true density well. (b) With more samples, the estimate of the density improves significantly.

[Source: M. Tappen]

KDE

- We approximate the density by

$$\hat{f}(\mathbf{x}) = \frac{1}{n} \sum_{i=1}^{n} K_H(\mathbf{x} - \mathbf{x}_i)$$

with $\mathbf{x}_i$ the points, and $K_H(\mathbf{x} - \mathbf{x}_i)$ the kernel

- Gaussian kernel is typically used

KDE

- We approximate the density by

$$\hat{f}(\mathbf{x}) = \frac{1}{n} \sum_{i=1}^{n} K_H(\mathbf{x} - \mathbf{x}_i)$$

  with $\mathbf{x}_i$ the points, and $K_H(\mathbf{x} - \mathbf{x}_i)$ the kernel

- Gaussian kernel is typically used
- Alternative way to think about this, put 1 wherever you have a sample and convolve with a Gaussian

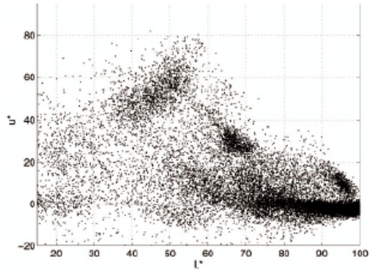
KDE

- We approximate the density by

$$\hat{f}(\mathbf{x}) = \frac{1}{n}\sum_{i=1}^{n} K_H(\mathbf{x} - \mathbf{x}_i)$$

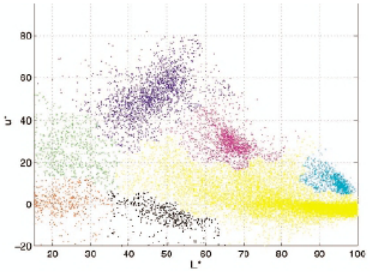
with $\mathbf{x}_i$ the points, and $K_H(\mathbf{x} - \mathbf{x}_i)$ the kernel

- Gaussian kernel is typically used
- Alternative way to think about this, put 1 wherever you have a sample and convolve with a Gaussian



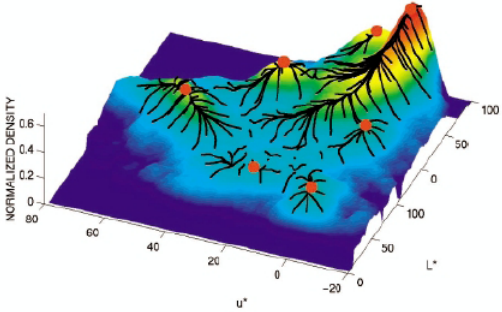
u*
80
60
40
20
0
−20
20
30
40
50
60
70
80
90
100
L*

(a)



u*
80
60
40
20
0
−20
20
30
40
50
60
70
80
90
100
L*

(b)

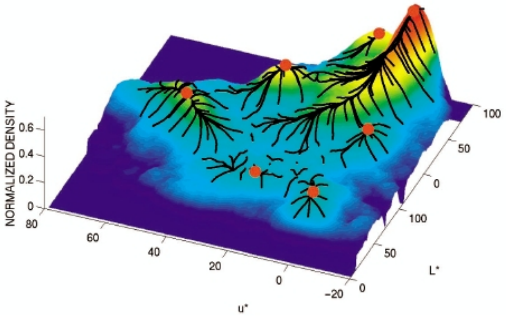


NORMALIZED DENSITY
0.6
0.4
0.2
0
80
60
40
20
0
−20
u*
100
50
0
100
50
0
L*

(c)

What is mean-shift

- The density will have peaks (also called modes)
- If we started at point and did gradient-ascent, we would end up at one of the modes
- Cluster based on which mode each point belongs to



[Source: M. Tappen]

No need for gradient ascent

- A set of iterative steps can be taken that will monotonically converge to a mode
- No worries about step sizes

No need for gradient ascent

- A set of iterative steps can be taken that will monotonically converge to a mode
- No worries about step sizes
- This is an adaptive gradient ascent, for each iteration

$$\mathbf{y}_{j+1} = \frac{\sum_{i=1}^{n} \mathbf{x}_i g(||\frac{\mathbf{y}_j - \mathbf{x}_i}{h}||_2^2)}{\sum_{i=1}^{n} g(||\frac{\mathbf{y}_j - \mathbf{x}_i}{h}||_2^2)}$$

with $g = \frac{d}{du}k(u)$, and $k(\mathbf{x}) = C\sum_{i=1}^{n} k(||\frac{\mathbf{y}_j - \mathbf{x}_i}{h}||_2^2)$

No need for gradient ascent

- A set of iterative steps can be taken that will monotonically converge to a mode
- No worries about step sizes
- This is an adaptive gradient ascent, for each iteration

$$\mathbf{y}_{j+1} = \frac{\sum_{i=1}^{n} \mathbf{x}_i g(||\frac{\mathbf{y}_j - \mathbf{x}_i}{h}||_2^2)}{\sum_{i=1}^{n} g(||\frac{\mathbf{y}_j - \mathbf{x}_i}{h}||_2^2)}$$

with $g = \frac{d}{du}k(u)$, and $k(\mathbf{x}) = C \sum_{i=1}^{n} k(||\frac{\mathbf{y}_j - \mathbf{x}_i}{h}||_2^2)$

- Why is this the update?

No need for gradient ascent

- A set of iterative steps can be taken that will monotonically converge to a mode
- No worries about step sizes
- This is an adaptive gradient ascent, for each iteration

$$\mathbf{y}_{j+1} = \frac{\sum_{i=1}^{n} \mathbf{x}_i g(||\frac{\mathbf{y}_j - \mathbf{x}_i}{h}||_2^2)}{\sum_{i=1}^{n} g(||\frac{\mathbf{y}_j - \mathbf{x}_i}{h}||_2^2)}$$

  with $g = \frac{d}{du}k(u)$, and $k(\mathbf{x}) = C\sum_{i=1}^{n} k(||\frac{\mathbf{y}_j - \mathbf{x}_i}{h}||_2^2)$
- Why is this the update?
- This procedure gives you one mode, how to get all?

No need for gradient ascent

- A set of iterative steps can be taken that will monotonically converge to a mode
- No worries about step sizes
- This is an adaptive gradient ascent, for each iteration

$$\mathbf{y}_{j+1} = \frac{\sum_{i=1}^{n} \mathbf{x}_i g(||\frac{\mathbf{y}_j - \mathbf{x}_i}{h}||_2^2)}{\sum_{i=1}^{n} g(||\frac{\mathbf{y}_j - \mathbf{x}_i}{h}||_2^2)}$$

  with $g = \frac{d}{du}k(u)$, and $k(\mathbf{x}) = C\sum_{i=1}^{n} k(||\frac{\mathbf{y}_j - \mathbf{x}_i}{h}||_2^2)$
- Why is this the update?
- This procedure gives you one mode, how to get all?
- Start from each point, and record the clusters

No need for gradient ascent

- A set of iterative steps can be taken that will monotonically converge to a mode
- No worries about step sizes
- This is an adaptive gradient ascent, for each iteration

$$\mathbf{y}_{j+1} = \frac{\sum_{i=1}^{n} \mathbf{x}_i g(||\frac{\mathbf{y}_j - \mathbf{x}_i}{h}||_2^2)}{\sum_{i=1}^{n} g(||\frac{\mathbf{y}_j - \mathbf{x}_i}{h}||_2^2)}$$

  with $g = \frac{d}{du}k(u)$, and $k(\mathbf{x}) = C \sum_{i=1}^{n} k(||\frac{\mathbf{y}_j - \mathbf{x}_i}{h}||_2^2)$
- Why is this the update?
- This procedure gives you one mode, how to get all?
- Start from each point, and record the clusters
- For segmentation use whatever feature representation you want for $\mathbf{x}_i$

[Source: M. Tappen]

No need for gradient ascent

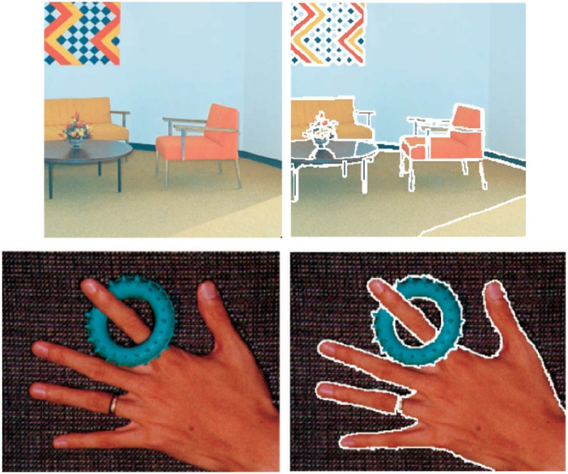
- A set of iterative steps can be taken that will monotonically converge to a mode
- No worries about step sizes
- This is an adaptive gradient ascent, for each iteration

$$\mathbf{y}_{j+1} = \frac{\sum_{i=1}^{n} \mathbf{x}_i g(||\frac{\mathbf{y}_j - \mathbf{x}_i}{h}||_2^2)}{\sum_{i=1}^{n} g(||\frac{\mathbf{y}_j - \mathbf{x}_i}{h}||_2^2)}$$

  with $g = \frac{d}{du}k(u)$, and $k(\mathbf{x}) = C\sum_{i=1}^{n} k(||\frac{\mathbf{y}_j - \mathbf{x}_i}{h}||_2^2)$
- Why is this the update?
- This procedure gives you one mode, how to get all?
- Start from each point, and record the clusters
- For segmentation use whatever feature representation you want for $\mathbf{x}_i$

[Source: M. Tappen]

Results



[Source: M. Tappen]