Markov Random Fields and its Applications

Huiwen Chang
Introduction

• Markov Random Fields (MRF)
  – A kind of undirected graphical model
• To model vision problems:
  – Low level: image restoration, segmentation, texture analysis...
  – High level: object recognition and matching (structure from motion, stereo matching)...

[Images of different scenes and models]
Introduction

• Markov Random Fields (MRF) – A kind of undirected graphical model
  • To model vision problems:
    – Low level: image restoration, segmentation, texture analysis...
    – High level: object recognition and matching (structure from motion, stereo matching)...

Labeling Problem
Graphical Models

- Probabilistic graphical models
  - Nodes: random variables
  - Edges: statistical dependencies among random variables

  - Advantage
    Compact and efficient way to visualize conditional independence assumptions to represent probability distribution
Conditional Independence

Random variables $y$ and $x$ are conditionally independent given $z$ if

$$P(y, x | z) = P(y | z) P(x | z).$$
Graphical Model

• Bayesian Network
  – Directed acyclic graph:

![Bayesian Network Diagram]

• Factorization:
  \[ p(a, b, c) = p(a|c)p(b|c)p(c) \]

• Conditional Independence:
  \[ a \perp b \mid c \]
Graphical Model

• Markov Network (MRF)

\[ P(a,b,c,d) = \frac{1}{Z} \psi(a, b) \psi(b, d) \psi(d, c) \psi(c, a) \]

Potential functions over the maximal clique of the graph

Potential function \( \psi(.) > 0 \)
Graphical Model

- Markov Network (MRF)

Hammersley-Clifford theorem says such distributions that are consistent with the set of conditional independence statements & the set of such distributions that can be expressed as a factorization by the maximal cliques of the graph are identical

\[
P(a, b, c, d) = \frac{1}{Z} \psi(a, b) \psi(b, d) \psi(d, c) \psi(c, a)
\]
MAP inference

- Posterior probability of the labelling $y$ given observation $x$ is:

\[
P(y \mid x) = \frac{P(x \mid y) \cdot P(y)}{P(x)}
\]

Maximum a Posteriori (MAP) inference: $y^* = \text{argmax}_y P(y \mid x)$. 

Markov Random Field

• Posterior probability of the labelling $y$ given observation $x$ is:

$$P(y|x) = \frac{1}{Z(x)} \prod_c \varphi_c(y_c; x)$$

where $Z(x) = \sum \prod_c \varphi_c(y_c; x)$ is called the partition function.

• Since we define potential function is strictly positive, we can express them as exponentials:

$$P(y | x) = \frac{1}{Z(x)} \exp \{-E(y; x)\}$$

The energy function $E(y; x)$ usually has some structured form:

$$E(y; x) = \sum_c \psi_c(y_c; x)$$
MAP inference

• Posterior probability of the labelling $y$ given observation $x$ is:

$$P(y | x) = \frac{P(x | y) \cdot P(y)}{P(x)}$$

Maximum a Posteriori (MAP) inference: $y^* = \arg\max_y P(y | x)$.

• The most possible labeling is to minimize the Energy

$$y^* = \arg\max_y P(y | x) = \arg\min_y E(y; x).$$
Pairwise MRF

• Most common energy function for image labeling

\[ E(y; x) = \sum_c \psi_c(y_c; x) = \sum_{i \in V} \psi^U_i(y_i; x) + \sum_{ij \in E} \psi^P_{ij}(y_i, y_j) \]

Unary  
Pairwise

• Which of the energy acted as the prior?
Example MRF model: Image Denoising

- How can we retrieve the original image given the noisy one?

Original image $Y$

Noisy image $X$ (Input)
**Nodes**

- For each pixel $i$,

  - $y_i$: latent variable (value in original image)
  - $x_i$: observed variable (value in noisy image)

Simple setting: $x_i, y_i \in \{-1, 1\}$
MRF formulation

- **Edges**
  - $x_i, y_i$ of each pixel $i$ correlated
  - neighboring pixels, similar value (smoothness)
MRF formulation

- **Edges**
  - $x_i, y_i$ of each pixel $i$ correlated
  - neighboring pixels, similar value (smoothness)

\[ \phi(x_i, y_i) = -\beta x_i y_i \]
\[ \psi(y_i, y_j) = -\alpha y_i y_j \]
MRF formulation

Energy function

\[ E(y; x) = \sum_{ij} \psi(y_i, y_j) + \sum_i \phi(x_i, y_i) \]

\[ = -\alpha \sum_{ij} y_i y_j - \beta \sum_i x_i y_i \]
Optimization

Energy function

\[ E(y; x) = \sum_{ij} \psi(y_i, y_j) + \sum_i \phi(x_i, y_i) = -\alpha \sum_{ij} y_i y_j - \beta \sum_i x_i y_i \]

Iterated Conditional Modes (ICM)
- Initialize \( y_i = x_i \) for all \( i \)
- Take a \( y_i \), fix others, flip \( y_i \) if \(-y_i\) make energy lower
- Repeat until converge
Bayes' Theorem

original image

Bayes' Theorem

noise image (10%)

Bayes' Theorem

Restored by ICM (4%)
Optimization

• Iterated Conditional Modes (ICM)
• Graph Cuts (GC)
• Message Passing
  – Belief Propagation (BP)
  – Tree Reweighted (not concluded here)
• LP-relaxation
  – Cutting-plane (not concluded here)

……
Graph Cuts

• To find labeling $f$ that minimizes the energy

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

$E(f; X)$

$E_{smooth}$ neighbors

$E_{data}$ pixels

$D_p(f_p; X_p)$
Graph Cuts

• For labeling problem
  – 2 Labels: Find global minimum
    • Max-flow/min-cut algorithm
    • Boykov and Kolmogrov 2001
      – the worst case complexity $O(mn^2 |C|)$
      – Fast in practice
  – For multi-labels: Computing the global minimum is NP-hard
    • Will discuss later
Two-Label Example: Lazy snapping

- Goal: Separate foreground from background

1\textsuperscript{st} step: Use stroke to select

2\textsuperscript{nd} step: Use polygon with vertices to refine
Model the problem

\[ E(X) = \sum_{i \in \mathcal{V}} E_1(x_i) + \lambda \sum_{(i,j) \in \mathcal{E}} E_2(x_i, x_j) \]

\( E_1(x_i) \): Likelihood Energy
\( E_2(x_i, x_j) \): Prior Energy

\( X_i \): Label of node \( i \).
in \{0: \text{background}, 1: \text{foreground}\}
Likelihood Energy

• Data Term

\[ E_1(x_i = 1) = 0 \quad E_1(x_i = 0) = \infty \quad \forall i \in F \]
\[ E_1(x_i = 1) = \infty \quad E_1(x_i = 0) = 0 \quad \forall i \in B \]
\[ E_1(x_i = 1) = \frac{d_i^F}{d_i^F + d_i^B} \quad E_1(x_i = 0) = \frac{d_i^B}{d_i^F + d_i^B} \quad \forall i \in U \]

• Use color similarity with known pixel to give an energy for uncertain pixel
  – \( d_i^F \): Minimum distance to front color
  – \( d_i^B \): Minimum distance to background color
Prior Energy

- Smoothness Term

\[ E_2(x_i, x_j) = |x_i - x_j| \cdot g(C_{ij}) \]
\[ C_{ij} = ||C(i) - C(j)||^2 \quad g(\varepsilon) = \frac{1}{\varepsilon + 1} \]

- Penalty Term for boundaries
  - Only nonzero if across segmentation boundary
  - Larger if adjacent pixels have similar colors
Problem

- Too slow for “real-time” requirement!
Problem

• Too slow for “real-time” requirement!
Pre-segmentation

- Graph Cut on segment instead of pixel level
- $C_{ij}$: the mean color difference between the two segments, weighted by the shared boundary length
- Speed Comparison

<table>
<thead>
<tr>
<th>Image</th>
<th>Dimension</th>
<th>Nodes Ratio</th>
<th>Edges Ratio</th>
<th>Lag with Pre-segmentation</th>
<th>Lag without Pre-segmentation</th>
</tr>
</thead>
<tbody>
<tr>
<td>Boy</td>
<td>(408, 600)</td>
<td>10.7</td>
<td>16.8</td>
<td>0.12s</td>
<td>0.57s</td>
</tr>
<tr>
<td>Ballet</td>
<td>(440, 800)</td>
<td>11.4</td>
<td>18.3</td>
<td>0.21s</td>
<td>1.39s</td>
</tr>
<tr>
<td>Twins</td>
<td>(1024, 768)</td>
<td>20.7</td>
<td>32.5</td>
<td>0.25s</td>
<td>1.82s</td>
</tr>
<tr>
<td>Girl</td>
<td>(768, 1147)</td>
<td>23.8</td>
<td>37.6</td>
<td>0.22s</td>
<td>2.49s</td>
</tr>
<tr>
<td>Grandpa</td>
<td>(1147, 768)</td>
<td>19.3</td>
<td>30.5</td>
<td>0.22s</td>
<td>3.56s</td>
</tr>
</tbody>
</table>
Graph Cuts

- **2 Labels**: Find global minimum
  - Max-flow/min-cut algorithm
  - Fast

- **Multi-Labels**: computing the global minimum is NP-hard
  - Approximation algorithm (for some forms of energy function: smoothness energy term $V$ which is metric or semi-metric)

\[
V(\alpha, \beta) = 0 \iff \alpha = \beta, \quad \text{Identity}
\]
\[
V(\alpha, \beta) = V(\beta, \alpha) \geq 0, \quad \text{Symmetry & non-negativity}
\]
\[
V(\alpha, \beta) \leq V(\alpha, \gamma) + V(\gamma, \beta), \quad \text{Triangle inequality}
\]
Graph Cuts for multi labels

- $\alpha$-expansion(d)
  - $V$ is metric, within a known factor of global minimum

- $\alpha \beta$-swap(c)
  - $V$ is semi-metric, local minimum
Graph Cut for multi labels

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

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

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

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

3.1. Find \( \hat{f} = \arg \min E(f') \) among \( f' \) within one \( \alpha \)-expansion of \( f \)

<table>
<thead>
<tr>
<th>edge</th>
<th>weight</th>
<th>for</th>
</tr>
</thead>
<tbody>
<tr>
<td>( t^\alpha_p )</td>
<td>( \infty )</td>
<td>( p \in \mathcal{P}_\alpha )</td>
</tr>
<tr>
<td>( t^\alpha_p )</td>
<td>( D_p(f_p) )</td>
<td>( p \notin \mathcal{P}_\alpha )</td>
</tr>
<tr>
<td>( t^\alpha_p )</td>
<td>( D_p(\alpha) )</td>
<td>( p \in \mathcal{P} )</td>
</tr>
<tr>
<td>( e_{{p,a}} )</td>
<td>( V(f_p, \alpha) )</td>
<td>( {p, q} \in \mathcal{N}, f_p \neq f_q )</td>
</tr>
<tr>
<td>( e_{{a,q}} )</td>
<td>( V(\alpha, f_q) )</td>
<td>( {p, q} \in \mathcal{N}, f_p \neq f_q )</td>
</tr>
<tr>
<td>( t^\alpha_a )</td>
<td>( V(f_p, f_q) )</td>
<td>( {p, q} \in \mathcal{N}, f_p = f_q )</td>
</tr>
<tr>
<td>( e_{{p,q}} )</td>
<td>( V(f_p, \alpha) )</td>
<td>( {p, q} \in \mathcal{N}, f_p \neq f_q )</td>
</tr>
</tbody>
</table>
Multi-label Examples: Shift Map

No accurate segmentation required
Multi-label Examples: Shift Map

User Constraints
Multi-label Examples: Shift Map

Image completion/In-painting(object removing)

Input

User’s mask

Output
Multi-label Examples: Shift Map

Retargeting (content-aware image resizing)

Output

Input
Multi-label Examples: Shift Map

- Label Set: relative mapping coordinate $M$

\[
M(u,v) = (t_x, t_y)
\]

\[
R(u,v) = I(u+t_x, v+t_y) = I(x,y)
\]

Nodes: pixels

Labels: shift-map values $(t_x, t_y)$

Output: $R(u,v)$

Input: $I(x,y)$
Multi-label Examples: Shift Map

- Energy function:

\[ E(M) = \alpha \sum_{p \in R} E_d(M(p)) + \sum_{p,q \in N} E_s(M(p), M(q)) \]

  - Data term: External Editing Requirement
  - Smoothness term: Avoid Stitching Artifacts

- Data term
  - varies between different application
  - Inpainting:
    Specific input pixels can be forced not to be included in the output image by setting \( D(x,y) = \infty \)
Smoothness term

Discontinuity in the shift-map

\[ E_s(M(p), M(q)) = \begin{align*}
& (I(n_{p'}) - I(q'))^2 + (\nabla I(n_{p'}) - \nabla I(q'))^2 \\
& (I(n_{q'}) - I(p'))^2 + (\nabla I(n_{q'}) - \nabla I(p'))^2
\end{align*} \]

For p

For q

**color**

**gradient**
Multi-label Examples: Shift map

- Graph Cuts: $\alpha$-expansion
- Why design label as disparity instead of absolute coordinate of input image?
Hierarchical Solution

Gaussian pyramid on input

Shift-Map

Output
Multi-label Examples: Dense Stereo

- Label set for each pixel:
  - disparity \( d \in \{0, 1, \ldots, D\} \)
Multi-label Examples: Dense Stereo

• Data term: for pixel $p$, label $d$

\[
C(p, d) = \left( \min \{ C_{\text{fwd}}(p, d), C_{\text{rev}}(p, d), \text{const} \} \right)^2.
\]

\[
C_{\text{rev}}(p, d) = \min_{p - \frac{1}{2} \leq x \leq p + \frac{1}{2}} |I_x - I'_{p+d}|, \quad C_{\text{fwd}}(p, d) = \min_{d - \frac{1}{2} \leq x \leq d + \frac{1}{2}} |I_p - I'_{p+x}|.
\]

Left Camera Image  Right Camera Image
Multi-label Examples: Dense Stereo

- Smoothness term for neighboring pixels $p$ and $q$
  
  $$V_{p,q} = \min(K|f_p - f_q|, T)$$

- or,

  $$V_{p,q} = u_{\{p,q\}} \cdot T(f_p \neq f_q)$$

  $$U(|I_p - I_q|) = \begin{cases} 
  2K & \text{if } |I_p - I_q| \leq 5 \\
  K & \text{if } |I_p - I_q| > 5.
\end{cases}$$
Design smoothness term $V$

- Choices of $V$: not Robust

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

- Better!

$$= 1(|\alpha - \beta| > k)$$

- Potts model

- Truncated linear model

- Linear model

- Quadratic model
Results

Original Image

Initial Solution
Results

Original Image

1st Expansion
Results

Original Image

2nd Expansion
Results

Original Image

3^{rd} Expansion
Results

Original Image

Final expansion
Results

- [http://vision.middlebury.edu/stereo/eval/](http://vision.middlebury.edu/stereo/eval/)
Comments on Graph Cuts

• In practice, GraphCut $\alpha$-expansion algorithm usually outperforms $\alpha\beta$-swap method

• Limitations of GC algorithm:
  – Constraint on energy term
  – Speed
Belief Propagation (BP)

- **Belief Propagation** allows the marginals and maximizer to be computed efficiently on graphical models.

- **Sum-product** BP is a message passing algorithm that calculates the marginal distribution on a graphical model

\[
P(x_1), P(x_2), \ldots, P(x_N)
\]

- **Max-product** BP (or max-sum in log domain), is used to estimate the state configuration with maximum probability.

\[
\arg \max_{x_1, x_2, \ldots, x_N} P(x_1, x_2, \ldots, x_N)
\]

- Exhaustive search $O(|\text{state}|^N)$
Sum-product BP

\[ P(x_1 | \bar{y}) = \sum_{x_2} \sum_{x_3} P(x_1, x_2, x_3 | \bar{y}) \]

\[ P(x_1 | \bar{y}) = \frac{1}{P(\bar{y})} \sum_{x_2} \sum_{x_3} \phi_{12}(x_1, x_2) \phi_{23}(x_2, x_3) \psi_1(y_1, x_1) \psi_2(y_2, x_2) \psi_3(y_3, x_3) \]

\[ = \frac{1}{P(\bar{y})} \psi_1(y_1, x_1) \sum_{x_2} \phi_{12}(x_1, x_2) \psi_2(y_2, x_2) \sum_{x_3} \phi_{23}(x_2, x_3) \psi_3(y_3, x_3) \]

\[ = \frac{1}{P(\bar{y})} \psi_1(y_1, x_1) \sum_{x_2} \phi_{12}(x_1, x_2) \psi_2(y_2, x_2) \sum_{x_3} \phi_{23}(x_2, x_3) m_{63}(x_3) \]

\[ = \frac{1}{P(\bar{y})} \psi_1(y_1, x_1) \sum_{x_2} \phi_{12}(x_1, x_2) \psi_2(y_2, x_2) m_{32}(x_2) \]

\[ = \frac{1}{P(\bar{y})} \psi_1(y_1, x_1) m_{21}(x_1) \]
Messages

\[
P(x_1, x_2, \ldots, x_N) = \frac{1}{Z} \prod_{i=1}^{N} g_i(x_i) \prod_{<ij>} f_{ij}(x_i, x_j)
\]

- Initialize all messages to uniform (any non-negative)
- Update: message from \(i\) to \(j\), consider all messages flowing into \(i\) (except for message from \(j\)):

\[
m_{ij}^{\text{new}}(x_j) = \sum_{x_i} f_{ij}(x_i, x_j) g_i(x_i) \prod_{k \in \text{Nbd}(i) \setminus j} m_{ki}^{\text{old}}(x_i)
\]

In one round: \(O(\text{#edges} \times \text{#states})\)
Noisy neighbor rule

• BP follows the “noisy neighbor rule” (from Brendan Frey, U. Toronto).

• Every node is a house in some neighborhood.

• A message: The noisy neighbor says “given everything I’ve heard, here’s what I think is going on inside your house.”
Belief Propagation

- Once messages have converged, output the normalized belief $b$ as the marginal

\[ b_i(x_i) \propto g(x_i) \prod_{k \neq i, k \in N(i)} m_{ki}^t(x_i) \]

- Max-product: $x^*$ that maximizes $b_i(x_i)$ is individually selected for each node is the maximal configuration (compute maximum during update)
Comments on Belief Propagation

• BP gives exact solutions when the graph is a tree (ie. has no loops), but only approximates the truth in loopy graphs (no convergence guarantee)

• Scheduling
  – For a tree or chain, with proper scheduling of the message updates, it will terminate after 2 steps.
  – For a grid (e.g. stereo on pixel lattice), people often sweep in an “up-down-left-right fashion” [Tappen & Freeman]
Speed-ups

• Binary variables: Use log ratios [Mackay]
• Distance transform and multi-scale [Felzenszwalb & Huttenlocher]
• Sparse forward-backward [Pal et al]
• Dynamic quantization of state space [Coughlan & Shen]
• Higher-order factors with linear interactions [Potetz & Lee]
• GPU [Brunton et al]
Large-scale Structure from Motion

• Recent work has built 3D models from large, unstructured online image collections
  – [Snavely06], [Li08], [Agarwal09], [Frahm10], Microsoft’s PhotoSynth, ...
Pipeline

Two-view reconstruction      Incremental Bundle adjustment      Surface Reconstruction

*From Fisher Yu’s Lecture*
Incremental Bundle Adjustment

Incremental BA

- ✔️ Works very well for many scenes
- ✗ Poor scalability, much use of bundle adjustment
- ✗ Poor results if a bad seed image set is chosen
- ✗ Drift and bad local minima for some scenes
Reconstruction pipeline

- **Feature detection**
- **Feature matching**
  - Find scene points seen by multiple cameras
- **Initialization**
  - Robustly estimate camera poses and/or scene points
- **Bundle adjustment**
  - Refine camera poses $R, T$ and scene structure $P$
-Structure from motion-
Examples: Optimizing MRF using BP

- View SfM as inference over a Markov Random Field, solving for all camera poses at once
  - Vertices are images (or points)
  - Inference problem: label each image with a camera pose, such that constraints are satisfied
  - Initializes all cameras at once
  - Can avoid local minima
  - Easily parallelized
  - Up to **6x speedups** over IBA for large problems
The MRF model

• Input: set of images with correspondence

**Unary constraints:** pose estimates (e.g., GPS, heading info)

**Binary constraints:** pairwise camera transformations
Constraints on camera pairs

- Recall how to compute relative pose between camera pairs using 2-frame SfM

- Find absolute camera poses \((R_i, t_i)\) and \((R_j, t_j)\) that agree with these pairwise estimates:

\[
R_{ij} = R_i^T R_j
\]
rotation consistency

\[
\lambda_{ij} t_{ij} = R_i^T (t_j - t_i)
\]
translation direction consistency
Constraints on camera pairs

• Define robust error functions to use as pairwise potentials:

\[ d^R(R_{ij}, R_i^T R_j) \]
\[ d^R(R_a, R_b) = \rho_R(||R_a - R_b||) \]

rotation consistency $\rho_R$ is truncated quadratic

\[ d^T(t_j - t_i, R_i t_{ij}) \]
\[ d^T(t_a, t_b) = \rho(\text{angleof}(t_a, t_b)) \]

translation direction consistency
Data term: pose information

• *Noisy* absolute pose info for some cameras
  – 2D positions from geotags (GPS coordinates)
  – Orientations (tilt & twist angles) from vanishing point detection [Sinha10]
Prior pose information

• *Noisy* absolute pose info for some cameras
  – 2D positions from geotags (GPS coordinates)
  – Orientations (tilt & twist angles) from vanishing point detection

Rotations unary potential

\[ d_i^O(R_i) = d_i^\theta(R_i) + d_i^\psi(R_i) + d_i^\phi(R_i) \]

\( d^\theta, d^\psi, d^\phi \) measure robust error *wrt* prior pose estimate

Translations unary potential

\[ d_i^G(t_i) = \rho_T(|| en(g_i) - \pi(t_i)||) \]

\( g_i \) is a GPS coordinate

\( en & \pi \) project into a common coordinate system

\( \rho_T \) is a robust distance function
Overall optimization problem

• Given pairwise and unary pose constraints, solve for absolute camera poses simultaneously
  – for $n$ cameras, estimate
    \[ \mathcal{R} = (R_1, R_2, ..., R_n) \quad \text{and} \quad \mathcal{T} = (t_1, t_2, ..., t_n) \]
    so as to minimize total error over the entire graph

\[
D^R(\mathcal{R}) = \sum_{e_{ij} \in E_C} d^R(R_{ij}, R_i^T R_j) + \alpha_1 \sum_{I_i \in \mathcal{I}} d_i^O(R_i)
\]

**pairwise rotation consistency**

**unary rotation consistency**

\[
D^T(\mathcal{T}, \mathcal{R}) = \sum_{e_{ij} \in E_C} d^T(t_j - t_i, R_i t_{ij}) + \alpha_2 \sum_{I_i \in \mathcal{I}} d_i^G(t_i)
\]

**pairwise translation consistency**

**unary translation consistency**
Solving the MRF

• Use discrete loopy belief propagation
  – Reduced by solving Rotation and Translation separately
  – Up to 1,000,000 nodes (cameras and points)
  – Up to 5,000,000 edges (constraints between cameras and points)
Discrete BP: Rotations

• Parameterize viewing directions as points on unit sphere
  – Discretize into 10x10x10 = 1,000 possible labels
  – Measure rotational errors as robust Euclidean distances on sphere (to allow use of distance transform)

• Uniform Grid on the sphere: 
  http://vision.princeton.edu/code.html#icosahedron
Discrete BP: Translations

• Parameterize positions as 2D points in plane
  – Use approximation to error function
    (to allow use of distance transforms)
  – Discretize into up to 300 x 300 = 90,000 labels
Central Rome
Reconstructed images: 14,754
Edges in MRF: 2,258,416
Median camera pose difference wrt IBA: 25.0m

Our result

<table>
<thead>
<tr>
<th>Dataset</th>
<th>Our approach</th>
<th>Incremental BA</th>
</tr>
</thead>
<tbody>
<tr>
<td>Acropolis</td>
<td>0.2 hours</td>
<td>0.5 hours</td>
</tr>
<tr>
<td>Quad</td>
<td>7.7 hours</td>
<td>62 hours</td>
</tr>
<tr>
<td>Dubrovnik</td>
<td>5.5 hours</td>
<td>28 hours</td>
</tr>
<tr>
<td>CentralRome</td>
<td>13.2 hours</td>
<td>82 hours</td>
</tr>
</tbody>
</table>

Incremental Bundle Adjustment [Agarwal09]
Markov Random Fields

• In MRF: \[ E(L \mid X) = E(X \mid L) + E(L) \]

• MRF assumes that the prior distribution is independent of the measurements
  – Specifically, the prior(smoothness) energy \( E(L) \) is independent of the observations \( X \)

• Sometimes, we hope each variable is based on a set of global observation data
  – No pure prior knowledge
Why Conditional Random Fields?

- Object segmentation/recognition:
- Label set = \{tree, people, water, sky, car\}
Conditional Random Field

- Given observations $X$, $(X, L)$ is said to be a Conditional Random Field (CRF) if, the random variables/labels $L$ obey the Markov property with respect to the graph:

  $$p(L_i | X, L_{\neq i}) = p(L_i | X, L_{\text{neighbor}(i)})$$

- In MRF, we have very strict independence assumptions:

  $$p(L_i | L_{\neq i}) = p(L_i | L_{\text{neighbor}(i)})$$
CRF in supervised training problem

- Define the conditional distribution $P(y | x, \theta)$
- Parameter Learning

$$\theta^* = \arg \max_\theta \sum_i \log P(y_i | x_i, \theta)$$

Take the label $y^* = \arg \max_y P(y | x, \theta^*)$
Object Recognition

• **Possible Image Label Set, i.e.** $Y = \{\text{background}; \text{car}\}$
• Given an image, define variable: label $y \in Y$, “patches” $x = \{x_1, \ldots x_m\}$
• Training set : $n$ labeled images $(x_i, y_i)$ where each $y_i \in Y$, and each $x_i = \{x_{i,1}, \ldots x_{i,m}\}$
• $\phi(x_j) \in R^d$ : representation feature
• Hidden variable: parts $h = \{h_1, \ldots h_m\}$
• $\theta$ : parameter of the model
• Take the label $\text{argmax}_y p(y|x, \theta^*)$
Object Recognition

- We define

\[
P(y, h \mid x, \theta) = \frac{e^{\Psi(y, h, x; \theta)}}{\sum_{y', h} e^{\Psi(y', h, x; \theta)}}.
\]

where the potential function is

\[
\Psi(y, h, x; \theta) = \sum_j \phi(x_j) \cdot \theta(h_j) + \sum_j \theta(y, h_j) + \sum_{(j, k) \in E} \theta(y, h_j, h_k).
\]
Object Recognition

• The object function in training the parameter:

\[
L(\theta) = \sum_i \log P(y_i \mid x_i, \theta) - \frac{1}{2\sigma^2} \|\theta\|^2
\]

where the first term is likelihood term, and the second term is the log of a Gaussian prior with variance \(\sigma^2\)

\[\theta^* = \arg\max_{\theta} L(\theta)\]

• Optimization: gradient ascent
Using BP

- \( \text{argmax}_y p(y|x, \theta^*) \)

\[
\forall y \in \mathcal{Y}, \quad Z(y \mid x, \theta) = \sum_h \exp\{\Psi(y, h, x; \theta)\}
\]

- \( \frac{\partial L(\theta)}{\partial \theta} \)

\[
\forall y \in \mathcal{Y}, j \in 1 \ldots m, a \in \mathcal{H}, \quad P(h_j = a \mid y, x, \theta) = \sum_{h \mid h_j = a} P(h \mid y, x, \theta)
\]

\[
\forall y \in \mathcal{Y}, (j, k) \in E, a, b \in \mathcal{H}, \quad P(h_j = a, h_k = b \mid y, x, \theta) = \sum_{h \mid h_j = a, h_k = b} P(h \mid y, x, \theta)
\]
Results

<table>
<thead>
<tr>
<th>Data set</th>
<th>5 parts</th>
<th>10 parts</th>
</tr>
</thead>
<tbody>
<tr>
<td>Car Side</td>
<td>94 %</td>
<td>99 %</td>
</tr>
<tr>
<td>Car Rear</td>
<td>91 %</td>
<td>91.7 %</td>
</tr>
</tbody>
</table>
Summary

- MRF
- CRF
- Graph Cut Algorithm
  - Binary
  - Multi-label
- Belief Propagation Algorithm

- Modeling Vision Problem is hard because
  - Optimization algorithm doesn’t help you to find global maxima
  - Model itself
Thank you!