Speeding Up Belief Propagation for Early Vision

Daniel Huttenlocher
MSRI Low Level Vision Workshop
February, 2005

Joint work with Pedro Felzenszwalb
Overview

- Markov random field (MRF) models are broadly useful for low level vision
  - Framework for expressing tradeoff between spatial coherence and fidelity to data

- Substantial recent advances in algorithms for MRF models on grid graph
  - Two main approaches: graph cuts [BVZ01], loopy belief propagation (LBP) [WF01]

- Present three speedup techniques for LBP
  - Resulting methods hundreds of times faster than conventional techniques
Low Level Vision Problems

- Estimate label at each pixel
  - Stereo: disparity
  - Restoration: intensity
  - Segmentation: layers, regions
  - Optical flow: motion vector
Pixel Labeling Problem

- Find good assignment of labels to sites
  - Set $\mathcal{L}$ of $k$ labels
  - Set $\mathcal{S}$ of $n$ sites
  - Neighborhood system $\mathcal{N} \subseteq \mathcal{S} \times \mathcal{S}$ between sites
    - Consider case of (four connected) grid graph

- Undirected graphical model
  - Graph $\mathcal{G} = (\mathcal{S}, \mathcal{N})$
  - Discrete random variable $x_i$ over $\mathcal{L}$ at each site $i$
  - First order models
    - Maximal cliques in $\mathcal{G}$ of size 2
Form of Posterior

- Observations $o$
- Posterior distribution of labelings given observations
  \[ \Pr(x|o) \propto \Pr(o|x)\Pr(x) \]
- For first order model, prior factors as
  \[ \Pr(x) \propto \prod_{(i,j) \in \mathcal{N}} V(x_i, x_j) \]
- Further assume likelihood factors
  \[ \Pr(x|o) \propto \prod_{i \in S} D_i(x_i) \prod_{(i,j) \in \mathcal{N}} V(x_i, x_j) \]
Estimation Problems

- Marginal probability at each node
  \[ \Pr(x_i|o) \]

- Maximize posterior (MAP)
  \[ \arg\max_x \prod_{i \in S} D_i(x_i) \prod_{(i,j) \in \mathcal{N}} V(x_i, x_j) \]

- Neither problem computationally tractable
  - NP hard for grid graph with 3 or more labels

- Various methods for approximate solution
  - Annealing, variational techniques, graph cuts using \( \alpha \)-expansion, loopy belief propagation, ...
Belief Propagation

- Iterative local update technique
  - Message passing, “nosy neighbor”

- Two forms
  - Sum product for estimating marginals
  - Max product for MAP estimation

- Exact solution when no loops in graph

- Update messages until “convergence” then compute distribution at each node
  - Sum product for marginals
  - Max product then max at each node for MAP
Sum Product

- At each step node $j$ sends each neighbor a message, in parallel
  - Node $j$’s view of $i$’s labels
    \[ m_{j\rightarrow i}(x_i) = \sum_{x_j} (D_j(x_j) \cdot V(x_j, x_i) \cdot \prod_{k \in \mathcal{N}(j) \setminus i} m_{k\rightarrow j}(x_j)) \]
- After $T$ iterations compute belief at each node
  - Using messages from neighbors and local data
    \[ b_j(x_j) = D_j(x_j) \cdot \prod_{i \in \mathcal{N}(j)} m_{i\rightarrow j}(x_j) \]
Max Product

- Min sum form with cost functions $D', V'$ proportional to negative log potentials
- Message updates
  
  $$m'_{j\rightarrow i}(x_i) = \min_{x_j}(D'_j(x_j) + V'(x_j, x_i) + \sum_{k \in \mathcal{N}(j) \setminus i} m'_{k\rightarrow j}(x_j))$$

- After $T$ iterations compute label minimizing value at each node
  
  $$\arg\min_{x_j} (D'_j(x_j) + \sum_{i \in \mathcal{N}(j)} m'_{i\rightarrow j}(x_j))$$

  - Simple approach of separately minimizing at each node can be problematic
Three Techniques

- Memory requirements of BP large
  - Using bipartite form of graph can halve usage

- For vision problems $V(x_i, x_j)$ generally a function of difference between labels
  - Enables computation of (discrete) messages in linear rather than quadratic time

- Number of iterations generally proportional to diameter of graph
  - Propagate information across grid
  - Using multi-grid methods can reduce to small constant number
Bipartite Graph ("Red-Black")

- Checkerboard pattern on grid defines a bipartite graph, $V=A \cup B$
- Alternating message updates of sets $A, B$ yields messages $\overline{m}$ nearly same as $m$
  - Update messages from $A$ on odd iterations and from $B$ on even iterations
  - Then can show by induction when $t$ odd (even)
    $$\overline{m}^t_{i \rightarrow j} = \begin{cases} m^t_{i \rightarrow j} & \text{if } i \text{ in } A \ (i \text{ in } B) \\ m^{t-1}_{i \rightarrow j} & \text{otherwise} \end{cases}$$
  - Converges to same fixed point with half as many updates and half as much memory
Fast Message Updates

- Pairwise term $V$ measuring label difference
- Sum product
  - Express as a convolution
  - $O(k \log k)$ algorithm using the FFT
  - Linear-time approximation algorithms for Gaussian models
- Min sum (max product)
  - Express as a min convolution
  - Linear time algorithms for common models using distance transforms and lower envelopes
Sum Product Message Passing

- When $V(x_i, x_j) = \rho(x_i - x_j)$ can write message update as convolution
  $$m_{j \rightarrow i}(x_i) = \sum_{x_j} (\rho(x_j - x_i) \cdot h(x_j))$$
  $$= \rho * h$$
  - Where $h(x_j) = D_j(x_j) \prod_{k \in \mathcal{N}(j) \setminus i} m_{k \rightarrow j}(x_j)$

- Thus FFT can be used to compute in $O(k \log k)$ time for $k$ values
  - Still somewhat large constants

- For $\rho$ a (mixture of) Gaussian(s) do faster
Fast Gaussian Convolution

- A box filter has value 1 in some range
  \[ b_w(x) = \begin{cases} 1 & \text{if } 0 \leq x \leq w \\ 0 & \text{otherwise} \end{cases} \]

- A Gaussian can be approximated by repeated convolutions with a box filter
  - Application of central limit theorem, convolving pdf’s tends to Gaussian
  - In practice, 4 convolutions [Wells, PAMI 86]
    \[ b_{w_1}(x) \ast b_{w_2}(x) \ast b_{w_3}(x) \ast b_{w_4}(x) \approx G_\sigma(x) \]
  - Choose widths \( w_i \) such that \( \sum_i (w_i^2 - 1)/12 \approx \sigma^2 \)
Convolution Using Box Sum

- Thus can approximate $G_\sigma(x) \star h(x)$ by cascade of box filters:
  $$b_{w_1}(x) \star (b_{w_2}(x) \star (b_{w_3}(x) \star (b_{w_4}(x) \star h(x))))$$
- Compute each $b_w(x) \star f(x)$ in time independent of box width $w$ – sliding sum
  - Each successive shift of $b_w(x)$ w.r.t. $f(x)$ requires just one addition and one subtraction
- Overall computation just a few operations per label, $O(k)$ with very low constant
Max Product Message Passing

- Can write message update as
  \[ m'_{j \rightarrow i}(x_i) = \min_{x_j}(\rho'(x_j - x_i) + h'(x_j)) \]
  - Where \( h'(x_j) = D'_j(x_j) \sum_{k \in N(j) \setminus i} m'_{k \rightarrow j}(x_j) \)
  - Formulation using minimization of costs, proportional to negative log probabilities

- Convolution-like operation over min, + rather than \( \sum, \times \) [FH00,FHK03]
  - No general fast algorithm like FFT
  - Certain important special cases in linear time
Commonly Used Pairwise Costs

- Potts model \( \rho'(x) = \begin{cases} 0 & \text{if } x=0 \\ d & \text{otherwise} \end{cases} \)
- Linear model \( \rho'(x) = cx \)
- Quadratic model \( \rho'(x) = cx^2 \)
- Truncated models
  - Truncated linear \( \rho'(x) = \min(d, c|x|) \)
  - Truncated quadratic \( \rho'(x) = \min(d, cx^2) \)
- Min convolution can be computed in linear time for any of these cost functions
Potts Pairwise Model

- Substituting in to min convolution
  \[ m'_{j \rightarrow i}(x_i) = \min_{x_j}(\rho'(x_j-x_i) + h'(x_j)) \]
  can be written as
  \[ m'_{j \rightarrow i}(x_i) = \min(h'(x_i), \min_{x_j} h'(x_j) + d) \]
- No need to compare pairs \( x_i, x_j \)
  - Compute min over \( x_j \) once, then compare result with each \( x_i \)
- \( O(k) \) time for \( k \) labels
  - No special algorithm, just rewrite expression to obtain alternative (fast) computation
Linear Pairwise Model

- Substituting in to min convolution yields
  \[ m'_{j \rightarrow i}(x_i) = \min_{x_j}(c|x_j-x_i| + h'(x_j)) \]
- Similar form to the L_1 distance transform
  \[ \min_{x_j}(|x_j-x_i| + 1(x_j)) \]
  - Where \( 1(x) = \begin{cases} 0 & \text{when } x \in P \\ \infty & \text{otherwise} \end{cases} \)
    is an indicator function for membership in P
- Distance transform measures L_1 distance to nearest point of P
  - Can think of computation as lower envelope of cones, one for each element of P
Using the $L_1$ Distance Transform

- Linear time algorithm
  - Traditionally used for indicator functions, but applies to any sampled function

- Forward pass
  - For $x_j$ from 1 to $k-1$
    \[ m(x_j) \leftarrow \min(m(x_j), m(x_{j-1})+c) \]

- Backward pass
  - For $x_j$ from $k-2$ to 0
    \[ m(x_j) \leftarrow \min(m(x_j), m(x_{j+1})+c) \]

- Example, $c=1$
  - $(3,1,4,2)$ becomes $(3,1,2,2)$ then $(2,1,2,2)$
Quadratic Pairwise Model

- Substituting in to min convolution yields
  \[ m'_{j \rightarrow i}(x_i) = \min_{x_j}(c(x_j - x_i)^2 + h'(x_j)) \]
- Again similar form to distance transform
- Compute lower envelope of parabolas
  - Each value of \( x_j \) defines a quadratic constraint, parabola rooted at \((x_j, h(x_j))\)
  - In general can be done in \(O(k \log k)\) [DG95]
  - Here parabolas are same shape and ordered, so \(O(k)\)
Lower Envelope of Parabolas

- Quadratics ordered $x_1 < x_2 < \ldots < x_n$
- At step $j$ consider adding $j$-th one to LE
  - Maintain two ordered lists
    - Quadratics currently visible on LE
    - Intersections currently visible on LE
  - Compute intersection of $j$-th quadratic with rightmost visible on LE
    - If right of rightmost intersection add quadratic and intersection
    - If not, this quadratic hides at least rightmost quadratic, remove and try again
Running Time of Lower Envelope

- Consider adding each quadratic just once
  - Intersection and comparison constant time
  - Adding to lists constant time
  - Removing from lists constant time
    - But then need to try again

- Simple amortized analysis
  - Total number of removals $O(k)$
    - Each quadratic, once removed, never considered for removal again

- Thus overall running time $O(k)$
static float *dt(float *f, int n) {
    float *d = new float[n], *z = new float[n];
    int *v = new int[n], k = 0;
    v[0] = 0; z[0] = -INF; z[1] = +INF;
    for (int q = 1; q <= n-1; q++) {
        float s = ((f[q]+c*square(q)) (f[v[k]]+c*square(v[k]))
                    /(2*c*q-2*c*v[k]));
        while (s <= z[k]) {
            k--; s  = ((f[q]+c*square(q))-(f[v[k]]+c*square(v[k])))
                        /(2*c*q-2*c*v[k]);    }
        k++; v[k] = q; z[k] = s; z[k+1] = +INF; }
    k = 0;
    for (int q = 0; q <= n-1; q++) {
        while (z[k+1] < q)
            k++; d[q] = c*square(q-v[k]) + f[v[k]];  }
    return d;}

**Code for Quadratic Pairwise Model**
Combined Pairwise Models

- Truncated models
  - Compute un-truncated message $m'$
  - Truncate using Potts-like computation on $m'$ and original function $h'$
    $$\min(m'(x_i), \min_{x_j} h'(x_j) + d)$$

- More general combinations
  - Min of any constant number of linear and quadratic functions, with or without truncation
    - E.g., multiple “segments”
Fast Message Update Methods

- Efficient computation without assuming form of (discrete) distributions
  - Requires prior to be based on differences between labels rather than their identities

- Sum product
  - $O(k \log k)$ message updates for arbitrary discrete distributions over $k$ labels using FFT
  - $O(k)$ when pairwise clique potential a mixture of Gaussians using box sums

- Max product
  - $O(k)$ for commonly used clique potentials
A Multi Grid Technique

- Number of message passing iterations $T$ generally proportional to diameter of grid
  - Propagate information across the grid

- Use hierarchical approach to make independent of graph diameter
  - Previous work does this by changing the graph, building quad-tree with no loops [W02]

- Our approach is to define a hierarchy of problems with original graph structure
  - Initialize messages based on coarser levels
Hierarchy of Grids

- Consider min sum case, rewrite minimization in terms of grid $\Gamma$

\[
E(x) = \sum_{(i,j) \in \Gamma} D_{ij}(x_{i,j}) + \sum_{(i,j) \in \Gamma \setminus C} V(x_{i,j} - x_{i+1,j}) + \sum_{(i,j) \in \Gamma \setminus R} V(x_{i,j} - x_{i,j+1})
\]

- Where $C, R$ last row and column of grid

- Can define family of grids $\Gamma^0, \Gamma^1, \ldots$
  - An element of $\Gamma^\ell$ corresponds to $\varepsilon \times \varepsilon$ block of pixels, where $\varepsilon = 2^\ell$
  - Labeling $x^\ell$ of $\Gamma^\ell$ assigns the pixels in each block a single label (from same set $\mathcal{L}$)
Problem Hierarchy

- Minimization problem at each level of the hierarchy

\[ E^\ell(x^\ell) = \sum_{(i,j) \in \Gamma^\ell} D^\ell_{ij}(x^\ell_{i,j}) + \sum_{(i,j) \in \Gamma^\ell \setminus \ell} V^\ell(x^\ell_{i,j} - x^\ell_{i+1,j}) + \sum_{(i,j) \in \Gamma^\ell \setminus \ell} V^\ell(x^\ell_{i,j} - x^\ell_{i,j+1}) \]

- Multi grid: final messages at one level as initial condition for next level, and so on
  - Small number of iterations if initial conditions close to final value
Hierarchical Data Term

- Finite element approach
- Assigning label \( \alpha \) to block \((i,j)\) at level \( \ell \) equivalent to assigning \( \alpha \) to each pixel in block

\[
D_{ij}^\ell(\alpha) = \sum_{0 \leq u < \varepsilon} \sum_{0 \leq v < \varepsilon} D_{\varepsilon i + u, \varepsilon j + v}(\alpha)
\]

- Sum costs for all pixels in block
- Corresponds to product of probabilities, likelihood of observing pixels given label \( \alpha \)
- Captures preference for multiple labels
Hierarchical Discontinuity Term

- Boundary between blocks length $\varepsilon$
  - Sum along boundary
- Separation between blocks $\varepsilon$
  - Finite difference, divide by separation

\[ V^\ell(\alpha - \beta) = \varepsilon V \left( \frac{\alpha - \beta}{\varepsilon} \right) \]

- Produces different form depending on $V$
  - Linear, $V^\ell(x) = c|x|$
  - Quadratic, $V^\ell(x) = cx^2 / \varepsilon$
Multi Grid Method

- Number of levels in hierarchy proportional to log image diameter
  - So propagation time small constant at top
- Same label set at each level
  - In contrast to pyramid methods
- In practice converges after a few iterations
  - Note each iteration just 1/3 more work than standard single level
Illustrative Results for Restoration

- Image restoration using MRF with truncated quadratic discontinuity cost
  - Not practical with conventional techniques, message updates $256^2$
- Quadratic data term with no penalty for masked pixels
- Powerful formulation now practical
  - Largely abandoned except for small label sets

Gaussian noise and mask
Illustrative Results for Stereo

- Truncated linear cost functions
  \[ D_i(x_i) = \min(d_b, |L(p_{i1}, p_{i2}) - R(p_{i1} - x_i, p_{i2})|) \]
  \[ V(x_i, x_j) = \min(d_s, |x_i - x_j|) \]
  - Runs in under a second for 30 disparity levels
- Same accuracy as slower methods
  - 12th in Middlebury benchmark (graph cuts 15th)
Extensions

- Fast message updates for max product in other cases
  - Discontinuity cost any convex function
    - Or truncated
  - Label set a multi-dimensional grid
    - E.g., flow vectors
  - Label sets not a regular grid
  - Possibly other “structured” label sets

- Additional labels such as occluded state for stereo can also be handled
  - Including penalty for length of occluded runs
Summary

- Fast methods for loopy belief propagation
  - Hundreds of times faster than previous methods
  - For discrete label space with potential functions based on differences between pairs of labels
  - Does not require parametric form of distributions

- Exact methods, not heuristic pruning or variational techniques
  - Except linear time Gaussian convolution which has (arbitrarily) small fixed approximation error

- Fast in practice, simple to implement
  - Code at http://people.cs.uchicago.edu/~pff/bp/