Markov net inference

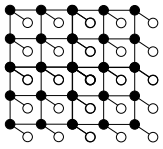
CS B553
Spring 2013

Announcements

- Assignment 2
  - Due next week

Pairwise Markov networks

- In a pairwise Markov network (aka pairwise Markov Random Field or MRF), the max clique size is 2
  - Grid graphs are an especially popular special case



MRFs for images

- MRFs have revolutionized computer vision and image processing over the last decade
- A sample application: Image reconstruction
  - Given a noisy image, estimate the original noise-free image

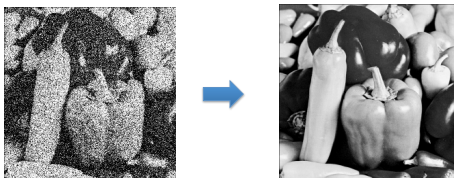


Image reconstruction

- In probabilistic terms, we're given a noisy image N and we want to estimate the original image O,

$$P(O|N) \propto P(N|O)P(O)$$

  - E.g. estimate the most likely original image,

$$O^* = \arg\max_O P(N|O)P(O)$$

- We might assume that noise at each pixel is independent given original pixel data,

$$P(N|O) = \prod_x \prod_y P(N_{x,y}|O_{x,y})$$

What about the prior, P(O)?

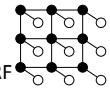
$$O^* = \arg\max_O P(N|O)P(O)$$

- Prior P(O) very difficult to define
  - For a 1 megapixel camera, there are $2^{24000000}$ possible images, so the joint distribution has this many entries!
- Intuitively, captures the fact that some images are much more likely than others

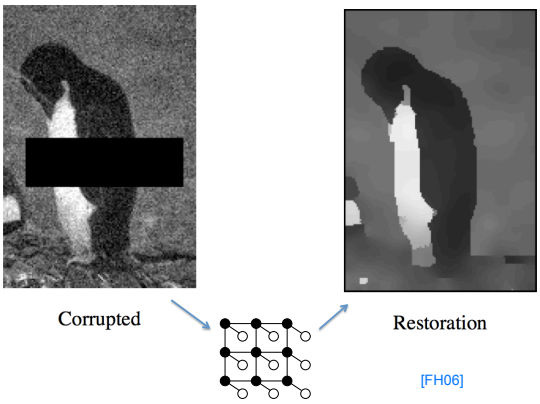
P() > P()

- Simple prior: Encourage "smoothness"
  - Neighboring pixels should generally have similar intensities

Application: Image reconstruction

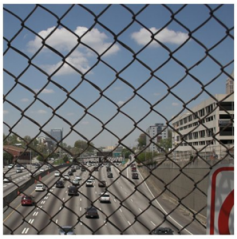


- Given a noisy image, infer original image
- Express problem naturally in terms of an MRF
  - Image is stored as a sampled function on a grid
  - We can observe noisy pixel values, and we'd like to estimate the original, clean pixel values
  - Important constraint: In images of real-world scenes, one pixel's color is correlated with that of its neighbors
  - The pairwise factors model this constraint
- Problem can be solved by doing inference on the Markov network



[FH06]

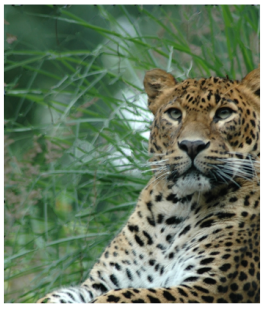
Image restoration



[Liu08]

9

Image restoration



[Liu08]

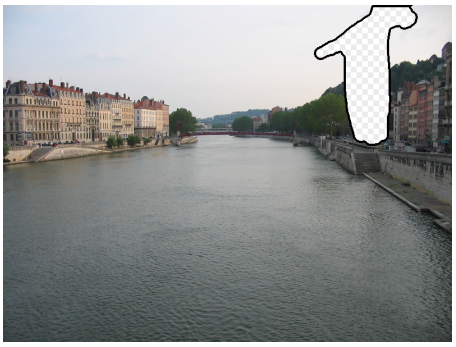
10

Image restoration

[Liu08]

11

Hays and Efros, SIGGRAPH 2007



Hays and Efros, SIGGRAPH 2007

14

Hays and Efros, SIGGRAPH 2007

Hays and Efros, SIGGRAPH 2007

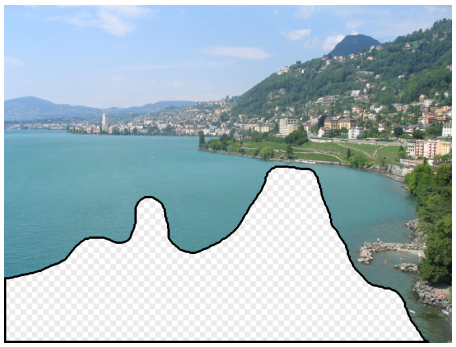
Hays and Efros, SIGGRAPH 2007

17

Hays and Efros, SIGGRAPH 2007

[Hays07]

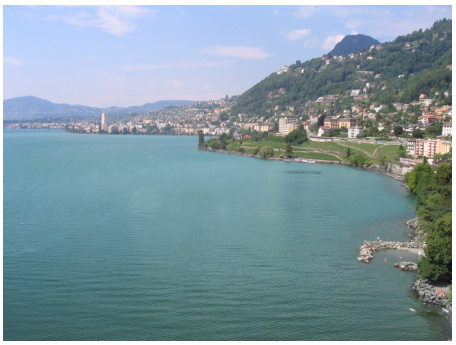
18



19

20

Hays and Efros, SIGGRAPH 2007



21

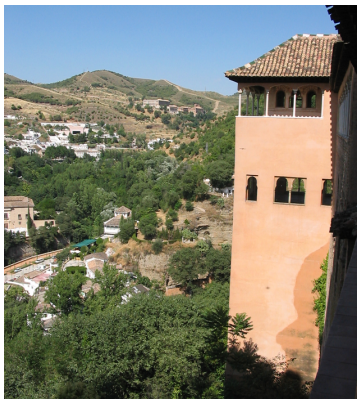
Hays and Efros, SIGGRAPH 2007

22

Hays and Efros, SIGGRAPH 2007

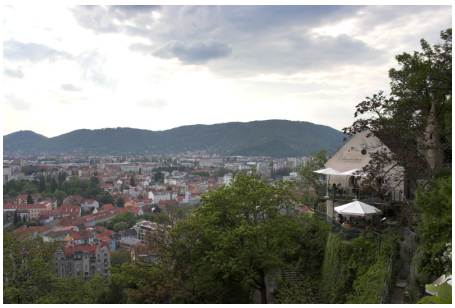
23

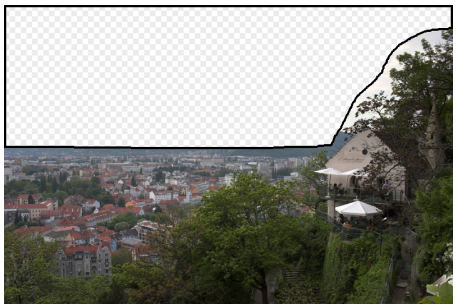
Hays and Efros, SIGGRAPH 2007



24

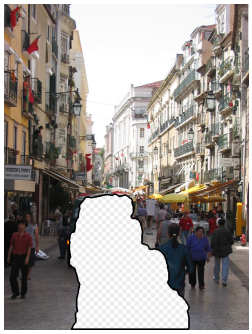
Hays and Efros, SIGGRAPH 2007





Failures

Failures

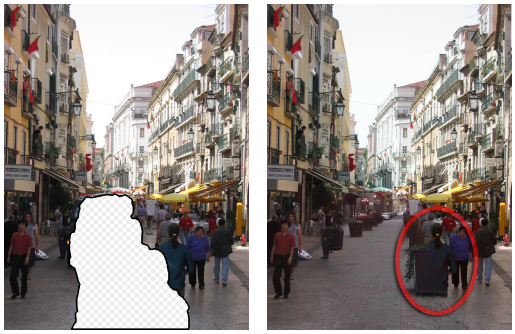


Hays and Efros, SIGGRAPH 2007

Failures

Hays and Efros, SIGGRAPH 2007

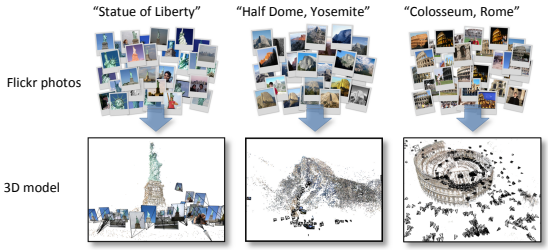
Failures



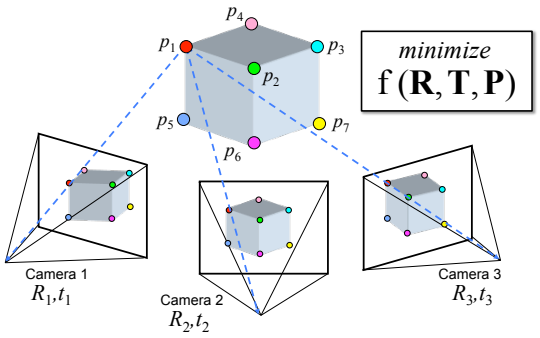
Hays and Efros, SIGGRAPH 2007

Automatic 3-D reconstruction

- From Internet photo collections [Snavely06]

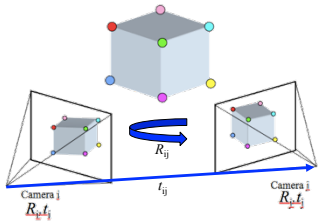


Structure from Motion



Constraints on camera *pairs*

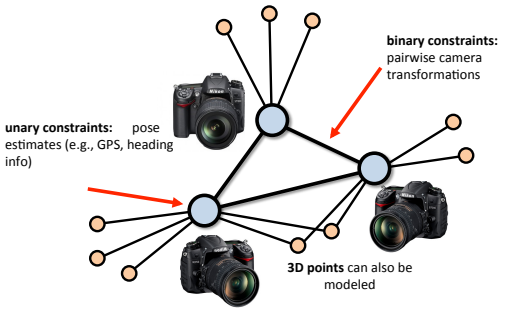
- Estimate relative pose between cameras, $\boldsymbol{R}_{ij}$ and $\boldsymbol{t}_{ij}$



- Want to solve for absolute camera poses, ($\boldsymbol{R}_i$, $\boldsymbol{t}_i$) and ($\boldsymbol{R}_j$, $\boldsymbol{t}_j$)
  - Two key constraints:

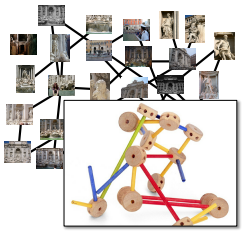
$$\mathbf{R}_{ij} = \mathbf{R}_i^\top \mathbf{R}_j \qquad \lambda_{ij}\mathbf{t}_{ij} = \mathbf{R}_i^\top(\mathbf{t}_j - \mathbf{t}_i)$$

The MRF model



Our approach

- View SfM as inference over a Markov random field, solving for all camera poses at once



- **Vertices** are cameras (or points)
- Both **pairwise** and **unary** constraints
- **Inference problem:** label each image with a camera pose, such that constraints are satisfied

Inference on Markov nets

- Of course, all of these applications (and the thousands of others…) require efficient inference
  - This is the hard part!

Variable elimination on Markov nets

1. Sort the non-query variables in an arbitrary order, Z1, Z2, … Zn
2. Initialize set of *factors* **F** to be the conditional probability distributions, P(Zi | Pa(Zi)).
3. For each i=1..n,
   a. Identify subset of factors **F'** involving Zi; these factors have some subset of variables **V** as parameters
   b. Take product of factors **F'**, producing ψ(**V**)
   c. Sum this product over all values of Zi, producing a new factor τ parameterized by **V**-{Zi}
   d. Remove **F'** from **F**, then add τ to **F**

Variable elimination on Markov nets

1. Sort the non-query variables in an arbitrary order, Z1, Z2, … Zn
2. Initialize set of *factors* **F** to be the ~~conditional probability distributions, P(Zi | Pa(Zi)).~~ factors in the Markov net, ϕ_1, ϕ_2, …
3. For each i=1..n,
   a. Identify subset of factors **F'** involving Zi; these factors have some subset of variables **V** as parameters
   b. Take product of factors **F'**, producing ψ(**V**)
   c. Sum this product over all values of Zi, producing a new factor τ parameterized by **V**-{Zi}
   d. Remove **F'** from **F**, then add τ to **F**

Inference on Markov nets

- What's the running time of Variable Elimination on a Markov network?
- Recall that, in general, we'll have a factor for every maximal clique in the Markov net
  - If the size of the largest clique is *N*, and each variable can take on *s* possible labels, then we'll need to process a factor with $O(s^N)$ possible labels
  - So the running time is at least exponential in the size of the maximal clique

Example

- What is the running time of Variable Elimination on:
  - A tree?
  - A simple cycle?
  - A grid graph?

Running time of VE

- The running time of Variable Elimination is controlled by the size of the scope of the largest factor
  - Including the original factors and the intermediate factors produced by the algorithm
  - Intermediate factors depend on elimination order
  - But it's often impossible to find a "good" elimination order
- In other words, running time is controlled by:
  - The maximal clique size of the original Markov net
  - Something about the structure of the Markov net

Cluster graph

- *Cluster graph:* An undirected graph in which:
  - Each node represents a subset of variables
  - Edge *might* connect nodes **X** and **Y** if $\mathbf{X} \cap \mathbf{Y} \neq \emptyset$
- For a given execution of Variable Elimination, we can draw a cluster graph to "visualize" the process

Variable elimination on Markov nets

1. Sort the non-query variables in an arbitrary order, Z1, Z2, … Zn
2. Initialize set of *factors* **F** to be the ~~conditional probability distributions, P(Zi | Pa(Zi)).~~ factors in the Markov net, ϕ_1, ϕ_2, …
3. For each i=1..n,
   a. Identify subset of factors **F'** involving Zi; these factors have some subset of variables **V** as parameters
   b. Take product of factors **F'**, producing $\psi(\mathbf{V})$
   c. Sum this product over all values of Zi, producing a new factor τ parameterized by **V**-{Zi}
   d. Remove **F'** from **F**, then add τ to **F**

Nodes of clique tree are values of V

Clique tree

- A particularly interesting cluster graph is that produced as follows from a run of VE:
  - Every subset **V** produced by VE is a node in the graph
  - Edge connects **X** and **Y** iff VE used $\psi(\mathbf{X})$ when computing $\psi(\mathbf{Y})$
- This graph is always a tree
  - Called a *clique tree* or *junction tree*