Course notes, CS664, 10/7/04

- Administrivia: PS1 will be on the web this evening.

EXACT METHODS

- Consider $f : \mathcal{P} \mapsto \mathcal{L}$, find

$$\arg\min_f \sum_p D(p, f(p)) + \sum_{(p,q)\in\mathcal{N}} T[f(p) \neq f(q)]$$

  Ising model when $|\mathcal{L}| = 2$, otherwise Potts model.

- Horrible optimization problem in a high-dimensional space

- Ising model: exact solution is possible (amazing fact!)

- Potts Model is NP-hard, even on a grid

MAX FLOWS

- Max flow problem definition: weighted graph with two terminals, weights are capacities.

- A flow assigns a number to each edge subject to capacity constraints and to conservation (except at the terminals s, t).

- Notice that a max flow will "saturate" a bunch of edges (flow = capacity).

- Augmenting paths algorithm: find an unsaturated path and saturate it. Order of paths is unspecified.

- When this converges, it yields a max flow, obviously. Ford-Fulkerson method. Note that every path from source to sink goes through a saturated edge.

- It's more subtle than it looks because there are residual edges and flow cancellation, but this is a good enough description for 664.

- There are fast flow algorithms even for very large graphs. One obvious trick is to look at paths in breadth-first order, which gives Edmonds-Karp.

- There is even a flow algorithm especially designed for vision (Boykov-Kolmogorov).

MIN CUTS

- On the same kid of graph we can define the min cut problem. A cut partitions the vertices into two sets S, T such $s \in S, t \in T$.

- The min cut minimizes $\sum_{p \in S, q \in T} w_{p,q}$, the sum of the edge weights between S and T.

- Unlike max flow, this looks like an exponential problem.

- Clearly with a max flow the saturated edges form a cut. Surprisingly, they also form a *min cut*. This is the famous max flow-min cut theorem, due to Ford and Fulkerson (1953).

- When you put it all together, this is a fast way to solve a problem

that looks exponential!

GRAPH CUTS: ISING MODEL

- One of the hottest areas in vision is the use of min cuts (sometimes called "graph cuts"). See the graph cuts home page for a (Cornell-centric) list of papers.

- Original result is from 1986, and shows that the Ising model can be solved with a single min cut.

- Summary: given the vision problem, build a graph where cuts correspond to labelings and the cost of a cut is the energy of the labeling.

- Graph construction overview: terminals represent 0 or 1 (i.e. the labels). Non-terminal nodes represent pixels. We will write both s, S as 0 and t, T as 1 for convenience.

- Pixels connected to each other, as on the grid. Each pixel is connected to both terminals. Note that there are lots of short

paths in this graph!

- There are two kinds of edges (links) in the graph; n-links, between pixels, and t-links, between a terminal and a pixel. The n-links will have weight 1. The t-links are a little more subtle, as there are two different encodings (ways to view a cut as a labeling).

- Note that exactly one of the t-links to p will be cut, so cuts clearly correspond to labelings.

- The natural encoding says that p is labeled with a terminal if it is in the same set as the terminal. Under this encoding, if $p \in 0$, the t-link between p and 1 (not 0!) will have weight $D(p, 0)$.

- With this graph, the cost of a cut is the energy of the corresponding labeling. The sum of the cut t-links is the data term, and the sum of the cut n-links is the smoothness term.

- What an amazing fact!

APPLICATION OF THE ISING MODEL

- Two label problems are rare in early vision. But there is a nice example for a problem called "voxel occupancy". Sometimes called "silhouette intersection", this name confuses problem with (wrong) algorithm.

- Standard way of reconstructing 3D shape from multiple cameras. Each camera has a picture of the background (scene without the object present).

- If a pixel is very different from its background intensity, the ray corresponding to that pixel is potentially occupied.

- The set of voxels that are potentially occupied by some ray should contain the object.

- Equivalently, think of the potentially occupied pixels from a particular camera as a silhouette. We then intersect the silhouettes.

- This doesn't actually compute the right answer, even in an ideal situation (visual hull is not the convex hull)

- Early hard decisions lead to errors (holes).

- Another application: Yuri's semi-automated segmentation algorithm