# Lecture 11: GrabCut and MaxFlow/MinCut

- An example of MRF's is Grab Cut (slide presentation). This is an MRF defined over the image lattice. At each lattice site there is a binary-valued variable which can take value 0 or 1. 1 indicates the foreground object and 0 the background. The purpose is to segment the foreground object, given a rough initialization (which can be done by hand).
- This initialization is used to learn unary potentials for the MRF. These are done by learning the statistics of image features within the (initialized) foreground and background regions. These features could be the colour of image pixels. Represented by mixtures of Gaussian distributions.
- The binary terms require that neighboring image pixels are likely to have the same label (i.e. both foreground or background). So a penalty is paid by the energy function if neighboring pixels are assigned different labels. But this penalty is reduced if there is a large image gradient between the two pixels.
- The inference algorithm is Graph-Cut (see next slides). In some cases, for a binary problem like this, it can be shown to converge to the global minimum of the energy function.
- Notes that this is a local interaction (see previous lecture) and so it will bias towards having short boundaries. The algorithm can be run in several stages. Initialize, estimate feature statistics for initial foreground and background. Re-estimate foreground and background (by minimizing the energy). Recompute feature statistics and repeat.

# GraphCut: Energy Minimization and Minimal Cuts

- Energy minimization (binary variables) can be re-formulated an finding the minimal cut which can be done by max-flow algorithms (later slide – bug here – fix why cost is directional – $x_1 = 1$ and $x_j = 0$.
- Write the energy as a function of binary-valued variables $x_i \in \{0, 1\}$: $E(\{x_i\}) = \sum_{ij} a_{ij} x_i x_j + \sum_i a_i x_i + c$. This can be re-expressed by introducing two new nodes $s$ and $t$ with fixed values $x_s = 0$ and $x_t = 1$:

$$E(\{x_i\}) = -\sum_{ij} a_{ij} x_i (1 - x_j) + \sum_{ij} a_{ij} x_i + \sum_i a_i x_i + c,$$

$$= \sum_{ij} a'_{ij} x_i (1 - x_j) + \sum_i a'_i x_i + c, \quad \text{with } a'_{ij} = -a_{ij}, \; a'_i = \sum_j a_{ij} + a_i$$

$$= \sum_{ij} a'_{ij} x_i (1 - x_j) + \sum_{i: a'_i > 0} a'_i x_i (1 - x_s) + \sum_{i: a'_i} |a'_i| (1 - x_i) x_t + c', \quad (10)$$

- with $c' = c + \sum_{i: a'_i < 0} a'_i$.
- The energy is now expressed in terms which are only non-zero if the neighbouring nodes take different values. This defines a *min-cut* problem – split $\{x_i\}$ into two sets $X_0 = \{i : x_i = 0\}$ and $X_1 = \{i : x_i = 1\}$. The only penalties are paid across the cut.

# GraphCut: Max Flow Algorithms

- We introduce notation. Graph $G = (V, E)$ where $V$ are the vertices and $E$ is the edges. The edges $(u, v) \in E$ have non-negative capacity $c(u, v) \geq 0$. Source $s$ and sink $t$.
- A *flow* $f : V \times R \mapsto R$ is required to obey the following constraints.
- (1). *capacity*: $\forall u, v \in V$, we require $f(u, v) \leq c(u, v)$
- (2). skew-symmetry: $\forall u, v \in V$, we require $f(u, v) = -f(v, u)$
- (3). *flow conservation*: $\forall u \in V/\{s, t\}$, we require $\sum_{v \in V} f(u, v) = 0$.
- The *total value* of the flow $f$ is $|f| = \sum_v f(s, v)$.
- A *cut* $(S, T)$ us a partition of nodes $V$ into sets $S$ and $T = V - S$ with $s \in S$ and $t \in T$. The *net flow* across the cut $(S, T)$ is $f(S, T)$. The *capacity* of the cut is $c(S, T)$.
- If $f$ is a flow in the network $(V, E)$ then $f$ is a maximum flow if, and only if, $|f| = c(S, T)$ for some cut.
- So the value of any flow is bounded above by the capacity of any cut. So *the maximum possible flow is given by the minimum cut*. Which minimizes the energy.
- Intuitively – the capacity is the restriction in size of pipes that the water can flow down. The size of the pipes determines the maximum amount of flow. The capacities are specified by the re-formulation of the energy.