# Paper Reading Seminar

Yan Wang

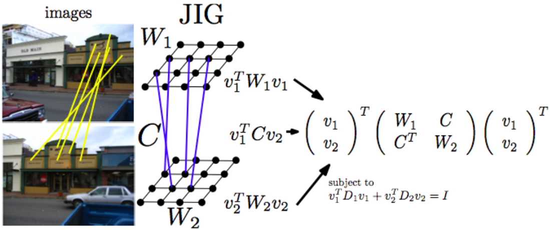
October 2, 2012

# Image Matching via Saliency Region Correspondences

- Intuition
  - Segmentation is not perfect. Weak connections $\Rightarrow$ graph-based segmentation (NCut)
  - Matching is not perfect. Weak connections $\Rightarrow$ spatial consistency check
  - Combine the two tasks together $\Rightarrow$ use a single graph to get a joint optimal
  - Encoding context in the matching process $\Rightarrow$ no need for spatial consistency check
- Outline
  - Formulation
  - Optimization
  - Features

# Formulation

- Graph formulation
  - Vertex: pixel
  - Layer: two-layer, one for each image
  - Intra-layer edges: how strongly the pixels are similar (connected) in this image
  - Inter-layer edges: how strongly the pixels are similar (connected) across images



# Formulation

- Intra-layer optimization goal

$$\max_{v} \frac{v_1^T W_1 v_1 + v_2^T W_2 v_2}{v^T D v}$$

  - Each segment should be internally consistent $\Rightarrow v_1^T W_1 v_1$. Only if $v_{1j}, v_{1k}$ both positive, will $W_{1(i,j)}$ be counted
  - $v_{ij} = \{0, 1\}^{n_i}, \sum_j v_{ij} = 1$, indicator vector for image $i$, pixel $j$, and cluster $k$, $v = [v_1^T, v_2^T]^T$
  - Normalize on the size: $\sum_j v_{1j}^T W_1 = \mathbf{1}^T W_1 = D_1$

# Formumation

- Inter-layer optimization goal

$$\max_{v} \frac{v_1^T C v_2}{v^T D v}$$

  - "Co-saliency" regions (where $v_{ij} = v_{ik}$) should be similar
  - Also normalize on size
  - Use "context" to refine segmentation as well as matching

# Formulation and optimization

- Final optimization goal

$$F(v, C) = \mathrm{IntraIS}(v, C) + \mathrm{InterIS}(v, C)$$

Note $C$ (matching) will affect segmentation result $v$

- Relaxation for optimizaiton
  - $v$ to real vector
  - EM iterative optimization (details not read)

# Features

- MSER detector + SIFT descriptor
- Intra-image $W$
  - $x, y$ are considered in the same segment $\Leftrightarrow$ no edges with large magnitude spatially separate them
  - Edge detection with large magnitude $\Rightarrow$ get $W$
- Inter-image $C$
  - 1. Feature detection: also consider the ellipse (orientation/scale) from the detector
  - 2. Feature matching
  - 2.0 Simplest pixel-wise matching: Gaussian kernel + descriptor + ellipse matrix

$$m_{x,y}(p,q) = e^{-\|d_p - d_q\|^2/\sigma_i^2} e^{-\|H_p(x) - H_q(y)\|^2/\sigma_p^2}$$

# Features

- Inter-image $C$
  - 2. Feature matching
  - 2.1 Adopt patch-wise feature matching score as pixel-wise score for $C$

$$M_{x,y} = \max\{m_{x,y}(p, q) \mid p \in P, q \in Q, x \in R_p, y \in R_q\}$$

  - 2.2 Compute patch-wise similarity matrix $M$, and then normalize to $C$

$$D_1^{-1/2} C D_2^{-1/2} = P \circ M$$

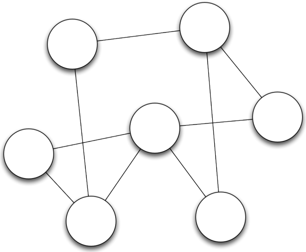
(Not really understanding this... They use MSER detector to extract sparse interest points, why can they still get a dense matching score?)

# Distributed Computer Vision Algorithms Through Distributed Averaging

- Intuition: make linear algebra algorithms distributed + CV applications
- Outline
  - Basic distributed algorithms
  - Linear algebra algorithms
  - Applications in CV

# Basic distributed algorithm

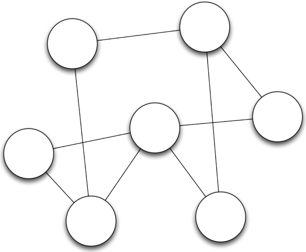
- Basic assumption
  - Several computation nodes with limited computation power as well as battery
  - Not fully connected: info can only propagated among neighbors



# Basic distributed algorithm

- Averaging a number $x$
  - Iterative algorithm: averaging among neighbors

$$x(t+1) = x(t) + \epsilon \sum_{j \in N_i} \Big( x_j(t) - x_i(t) \Big)$$



  - Easily extended to vectors/matrices

# Basic distributed algorithm

- Get the min/max

$$x_i(t+1) = \min_{j \in \{N_i \bigcup i\}} x_j(t)$$

- Covariance
  - $A = \begin{bmatrix} A_1^T & A_2^T & \cdots & A_N^T \end{bmatrix}^T \in \mathbb{R}^{n \times m}$
  - Compute $C = \frac{1}{N} A^T A$

$$C = \frac{1}{N} \sum_i A_i^T A_i$$

  - Compute $C_i = A_i^T A_i$, and then average among all the nodes
  - Only meaning for when $\text{rank}(C) \ll n$

# Linear algebra algorithms

- SVD

$$A = U\Sigma V^T$$

  - Compute $C = A^T A$
  - Locally compute C's SVD: $C = V(\frac{1}{N}\Sigma^2)V^T$
  - Recover $U_i$ locally: $U_i = A_i V \Sigma^{-1}$

# Linear algebra algorithms

- PCA
  - Compute the average of $A \Rightarrow$ do data centralization
  - Compute covariance matrix $C = A^T A$
  - Local SVD decomposition: $C = U \Sigma U^T \Rightarrow$ get the basis
  - Compute the new representation: $\hat{A} = A^T U$

# Linear algebra algorithms

- Least square estimation
  - $x_0 = \arg\min_x \|Ax - b\|^2$
  - $x_0 = \arg\min_x \|A^T Ax - A^T b\|^2$ (is this true?)
  - $A^T A$ and $A^T b$ easy to compute distributedly
  - Can be solved locally
  - (Not very clear)

# Applications in CV

- Point triangulation
- Linear pose estimation
- Structure from Motion

# Coherency Sensitive Hashing

- A simple introduction
- Find ANN for local patches
- Naive baseline: brute-force match in the feature space
- PatchMatch: like Genetic Algorithm
  - Hold a candidate pool
  - If two patches in two images are similar, include nearby patches in the pool
  - Randomly add patch pairs in the pool to get rid of (too) local minima.

# Approach

- Hashing + more sophisticated candidate expansion
  - Hashing: LSH + 2D Walsh Hadamard Kernel
  - Candidate expansion
    - $g_A(a) = g_B(b) \Rightarrow$ include $b$ in Cand($a$)
    - $b \in \text{Cand}(a_1), g_A(a_1) = g_A(a_2) \Rightarrow$ include $b$ in Cand($a_2$)
    - $b_1 \in \text{Cand}(a_2), g_B(b_1) = g_B(b_2) \Rightarrow$ include $b_2$ in Cand($a_2$)
    - Include spatial neighbor pairs in the candidates
  - Still need to do verifications on the candidate set