Segmentation & Clustering

Mani Golparvar-Fard
Department of Civil and Environmental Engineering
Department of Computer Science
3129D, Newmark Civil Engineering Lab
e-mail: mgolpar@illinois.edu

UNIVERSITY OF ILLINOIS AT URBANA-CHAMPAIGN
Important Notes

- Midterm Project Reports + Presentations
  - April 13

- A3 out today
  - Due April 13
Multi-Class Sign Detection
Using Google Street View

https://vimeo.com/117504040

Detection, Classification, and Mapping of U.S. Traffic Signs Using Google Street View Images for Roadway Inventory Management
Available through Mathworks

- **Computer Vision System Toolbox**

- Corner detection, including Shi & Tomasi, Harris, and FAST methods
- BRISK, MSER, and SURF detection for blobs and regions
- Extraction of BRISK, FREAK, SURF, and simple pixel neighborhood descriptors
- Histogram of Oriented Gradients (HOG) feature extraction
- Visualization of feature location, scale, and orientation

![SURF (left), MSER (center), and corner detection (right) with Computer Vision System Toolbox. Using the same image, the three different feature types are detected and results are plotted over the original image.](image-url)
Available at a web site near you…

- For most local feature detectors, executables are available online:
  - http://mi.eng.cam.ac.uk/~er258/work/fast.html
  - http://robots.ox.ac.uk/~vgg/research/affine
  - http://www.cs.ubc.ca/~lowe/keypoints/
  - http://www.cs.unc.edu/~ccwu/siftgpu/
  - http://www.vision.ee.ethz.ch/~surf
Choosing a detector

• What do you want it for?
  – Precise localization in x-y: Harris
  – Good localization in scale: Difference of Gaussian
  – Flexible region shape: MSER (maximally stable extremal regions)

• Best choice often application dependent
  – Harris-/Hessian-Laplace/DoG work well for many natural categories
  – MSER can work well for buildings and printed things

• Why choose?
  – Get more points with more detectors

• There have been extensive evaluations/comparisons
  – [Mikolajczyk et al., IJCV’05, PAMI’05]
  – All detectors/descriptors shown here work well
Example Project

- Parallel Tracking and Mapping for Small AR Workspaces

http://www.youtube.com/watch?v=Y9HMn6bd-v8
Comparison of Keypoint Detectors

<table>
<thead>
<tr>
<th>Feature Detector</th>
<th>Corner</th>
<th>Blob</th>
<th>Region</th>
<th>Rotation invariant</th>
<th>Scale invariant</th>
<th>Affine invariant</th>
<th>Repeatability</th>
<th>Localization accuracy</th>
<th>Robustness</th>
<th>Efficiency</th>
</tr>
</thead>
<tbody>
<tr>
<td>Harris</td>
<td>✓</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>+++</td>
<td>+++</td>
<td>++</td>
<td>+</td>
</tr>
<tr>
<td>Hessian</td>
<td></td>
<td>✓</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>+</td>
</tr>
<tr>
<td>SUSAN</td>
<td>✓</td>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>+</td>
</tr>
<tr>
<td>Harris-Laplace</td>
<td>✓ (✓)</td>
<td></td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>+++</td>
<td>+++</td>
<td>++</td>
<td>+</td>
</tr>
<tr>
<td>Hessian-Laplace</td>
<td>✓ (✓)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>+++</td>
<td>+++</td>
<td>+++</td>
<td>+</td>
</tr>
<tr>
<td>DoG</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>+</td>
</tr>
<tr>
<td>SURF</td>
<td>✓ (✓)</td>
<td></td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>+</td>
</tr>
<tr>
<td>Harris-Affine</td>
<td>✓ (✓)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>+++</td>
<td>+++</td>
<td>++</td>
<td>+</td>
</tr>
<tr>
<td>Hessian-Affine</td>
<td>✓ (✓)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>+++</td>
<td>+++</td>
<td>+++</td>
<td>+</td>
</tr>
<tr>
<td>Salient Regions</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>+</td>
<td>++</td>
<td>++</td>
<td>+</td>
</tr>
<tr>
<td>Edge-based</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>+++</td>
<td>+++</td>
<td>+</td>
<td>+</td>
</tr>
<tr>
<td>MSER</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>+++</td>
<td>+++</td>
<td>++</td>
<td>+</td>
</tr>
<tr>
<td>Intensity-based</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>++</td>
<td>++</td>
<td>++</td>
<td>+</td>
</tr>
<tr>
<td>Superpixels</td>
<td>✓</td>
<td>✓</td>
<td>✓ (✓)</td>
<td>✓</td>
<td>✓</td>
<td>✓</td>
<td>+</td>
<td>+</td>
<td>+</td>
<td>+</td>
</tr>
</tbody>
</table>


More examples from MATHWORKS at
http://www.mathworks.com/help/vision/ref/extractfeatures.html
Choosing a descriptor

- Again, need not stick to one

- For object instance recognition or stitching, SIFT or variant is a good choice
Things to remember

- **Keypoint detection**: repeatable and distinctive
  - Corners, blobs, stable regions
  - Harris, DoG

- **Descriptors**: robust and selective
  - Spatial histograms of orientation
  - SIFT
Outline

- K-mean clustering
- Mean-shift
- Graph-cut

Reading: Chapter 14 [FP]
Segmentation and Clustering

**Segmentation**
- Compact representation for image data in terms of a set of *components*
- Components share “common” *visual properties*
- Properties can be defined at different level of abstractions

**Clustering**
- Group together similar points and represent them with a single token
  - Token: whatever we need to group (pixels, points, surface elements, etc.)
Segmentation and Clustering

Key Challenges:

1. What makes two points/images/patches similar?

2. How do we compute an overall grouping from pairwise similarities?
Why do we cluster?

- **Summarizing data**
  - Look at large amounts of data
  - Patch-based compression or denoising
  - Represent a large continuous vector with the cluster number

- **Counting**
  - Histograms of texture, color, SIFT vectors

- **Segmentation**
  - Separate the image into different regions

- **Prediction**
  - Images in the same cluster may have the same labels
Segmentation in Computer Vision

Segmentation in Computer Vision

Segmentation in Civil Engineering


Supervised segmentation of the ground truth images

The segmentation and asset recognition results
Segmentation in Civil Engineering

Segmentation of Roadway Assets

https://vimeo.com/105275883
How do we cluster?

- **K-means**
  - Iteratively re-assign points to the nearest cluster center

- **Agglomerative clustering**
  - Start with each point as its own cluster and iteratively merge the closest clusters

- **Spectral clustering**
  - Split the nodes in a graph based on assigned links with similarity weights
Feature Space

- Every token is identified by a set of salient visual characteristics.

- For example:
  - Position
  - Color
  - Texture
  - Motion vector
  - Size, orientation (if token is larger than a pixel)
Feature Space

Source: K. Grauman
Feature space:
each token is represented by a point

\[ R = 15 \]
\[ G = 189 \]
\[ B = 2 \]
Token similarity is thus measured by distance between points ("feature vectors") in feature space.

\[ \sqrt{(p_1 - q_1)^2 + (p_2 - q_2)^2 + \cdots + (p_n - q_n)^2} = \sqrt{\sum_{i=1}^{n} (p_i - q_i)^2}. \]
Cluster together tokens with high similarity
Clustering for Summarization

**Goal:** cluster to minimize variance in data given clusters

- Preserve information

\[
\mathbf{c}^*, \delta^* = \underset{c, \delta}{\text{argmin}} \frac{1}{N} \sum_{j}^{N} \sum_{i}^{K} \delta_{ij} (\mathbf{c}_i - \mathbf{x}_j)^2
\]

Whether \(\mathbf{x}_j\) is assigned to \(\mathbf{c}_i\)
K-means

0. Initialize Cluster Centers

1. Assign Points to Clusters

2. Re-compute Means

Repeat (1) and (2)
K-means

1. Initialize cluster centers: \( c^0 \); t=0

2. Assign each point to the closest center

   \[
   \delta^t = \arg\min_{\delta} \frac{1}{N} \sum_{j} \sum_{i} \delta_{ij} (c_{i}^{t-1} - x_j)^2
   \]

3. Update cluster centers as the mean of the points

   \[
   c^t = \arg\min_{c} \frac{1}{N} \sum_{j} \sum_{i} \delta_{ij}^t (c_{i} - x_j)^2
   \]

4. Repeat 2-3 until no points are re-assigned (t=t+1)
K-means: design choices

- Initialization
  - Randomly select $K$ points as initial cluster center
  - Or greedily choose $K$ points to minimize residual

- Distance measures
  - Traditionally Euclidean, could be others

- Optimization
  - Will converge to a local minimum
  - May want to perform multiple restarts
Clustering

- K-means clustering using intensity alone and color alone
How to evaluate clusters?

- **Generative**
  - How well are points reconstructed from the clusters?

- **Discriminative**
  - How well do the clusters correspond to labels?
    - Purity
  - Note: unsupervised clustering does not aim to be discriminative
K-means demos


General:
http://home.dei.polimi.it/matteucc/Clustering/tutorial_html/AppletKM.html

Color clustering:
http://www.cs.washington.edu/research/imagedatabase/demo/kmcluster/

MATLAB implementations:
Conclusions: K-means

Pros

- Finds cluster centers that minimize conditional variance (good representation of data)
- Simple to implement, widespread application

Cons

- Prone to local minima (Sensitive to initialization)
- Need to choose $K$
- All clusters have the same parameters (e.g., distance measure is non-adaptive)
- Can be slow: each iteration is $O(KNd)$ for $N$ d-dimensional points
- Sensitive to outliers
Common similarity/distance measures

- **P-norms**
  - City Block (L1)
  - Euclidean (L2)
  - L-infinity

\[
\|x\|_p := \left( \sum_{i=1}^{n} |x_i|^p \right)^{1/p}.
\]

\[
\|x\|_1 := \sum_{i=1}^{n} |x_i|.
\]

\[
\|x\| := \sqrt{x_1^2 + \cdots + x_n^2}.
\]

\[
\|x\|_\infty := \max (|x_1|, \ldots, |x_n|).
\]

- **Mahalanobis**
  - Scaled Euclidean

\[
d(\bar{x}, \bar{y}) = \sqrt{\sum_{i=1}^{N} \frac{(x_i - y_i)^2}{\sigma_i^2}},
\]

- **Cosine distance**

\[
similarity = \cos(\theta) = \frac{A \cdot B}{\|A\| \|B\|},
\]

Here \(x_i\) is the distance between two points.
Segmentation as clustering

Cluster together tokens that share similar visual characteristics

- K-mean
- **Mean-shift**
- Graph-cut
Mean shift segmentation

- An advanced and versatile technique for clustering-based segmentation

Mean shift segmentation

- The mean shift algorithm seeks a *mode* or *local maximum of density* of a given distribution
  - Choose a search window (width and location)
  - Compute the mean of the data in the search window
  - Center the search window at the new mean location
  - Repeat until convergence
Mean shift

Region of interest

Center of mass

Mean Shift vector
Mean shift

Region of interest
Center of mass

Mean Shift vector
Mean shift

Region of interest
Center of mass
Mean Shift vector
Mean shift
Mean shift
Mean shift
Mean shift

Region of interest

Center of mass
Computing The Mean Shift

- **Simple Mean Shift procedure:**
  - Compute mean shift vector
  - Translate the Kernel window by $\mathbf{m}(\mathbf{x})$

\[
\mathbf{m}(\mathbf{x}) = \frac{\sum_{i=1}^{n} \mathbf{x}_i g\left(\frac{(||\mathbf{x} - \mathbf{x}_i||^2)}{h}\right)}{\sum_{i=1}^{n} g\left(\frac{(||\mathbf{x} - \mathbf{x}_i||^2)}{h}\right)} - \mathbf{x}
\]

$g(\mathbf{x}) = -k'(\mathbf{x})$
Real Modality Analysis

- Tessellate the space with windows
- Merge windows that end up near the same “peak” or model
Attraction basin

- **Attraction basin**: the region for which all trajectories lead to the same mode
- **Cluster**: all data points in the attraction basin of a mode
Attraction basin
Segmentation by Mean Shift

- Find features (color, gradients, texture, etc)
- Initialize windows at individual pixel locations
- Perform mean shift for each window until convergence
- Merge windows that end up near the same “peak” or mode
Mean shift segmentation results

http://www.caip.rutgers.edu/~comanici/MSPAMI/msPamiResults.html
Mean shift pros and cons

- **Pros**
  - Does not assume spherical clusters
  - Just a single parameter (window size)
  - Finds variable number of modes
  - Robust to outliers

- **Cons**
  - Output depends on window size
  - Computationally expensive
  - Does not scale well with dimension of feature space

- **Matlab Implementation**
Segmentation as clustering

Cluster together tokens that share similar visual characteristics

- K-mean
- Mean-shift
- Graph-cut
Graph-based segmentation

- Represent features and their relationships using a graph
- Cut the graph to get subgraphs with strong interior links and weaker exterior links
Images as graphs

- Node for every pixel
- Edge between every pair of pixels
- Each edge is weighted by the *affinity* or similarity of the two nodes

Source: S. Seitz
Measuring Affinity

Distance

\[
aff(x, y) = \exp\left\{ -\left( \frac{1}{2\sigma_d^2} \right) \|x - y\|^2 \right\}
\]

Intensity

\[
aff(x, y) = \exp\left\{ -\left( \frac{1}{2\sigma_i^2} \right) \|I(x) - I(y)\|^2 \right\}
\]

Color

\[
aff(x, y) = \exp\left\{ -\left( \frac{1}{2\sigma_i^2} \right) \|c(x) - c(y)\|^2 \right\}
\]
Segmentation by graph partitioning

- Break Graph into sub-graphs
  - Break links (cutting) that have low affinity
    - similar pixels should be in the same sub-graphs
    - dissimilar pixels should be in different sub-graphs

Source: S. Seitz
Segmentation by graph partitioning

- Break Graph into sub-graphs
  - Break links (cutting) that have low affinity
    - similar pixels should be in the same sub-graphs
    - dissimilar pixels should be in different sub-graphs
  - Sub-graphs represent different image segments
  - Graph-cut: technique to cut a graph optimally

Source: S. Seitz
Segmentation by graph partitioning

- **CUT**: Set of edges whose removal makes a graph disconnected
- **Cost of a cut**: sum of weights of cut edges
- **Example**: Cost of the blue cut?

Source: S. Seitz
Minimum cut

- We can do segmentation by finding the minimum cut in a graph
  - Efficient algorithms exist for doing this
- Drawback: minimum cut tends to cut off very small, isolated components

![Diagram of segmentation with minimum cut and ideal cut](image-url)
Normalized cut

- **IDEA:** normalizing the cut by component size
- The normalized cut cost is:

\[
\frac{\text{cut}(A, B)}{\text{assoc}(A, V)} + \frac{\text{cut}(A, B)}{\text{assoc}(B, V)}
\]

\[
\text{assoc}(A, V) = \text{sum of weights of all edges in } V \text{ that touch } A
\]

- The exact solution is NP-hard but an approximation can be computed by solving a generalized eigenvalue problem

J. Shi and J. Malik. Normalized cuts and image segmentation. PAMI 2000
Normalized cuts: Pro and con

- **Pros**
  - Generic framework, can be used with many different features and affinity formulations

- **Cons**
  - High storage requirement and time complexity
  - Bias towards partitioning into equal segments

*Slide from Khurram Hassan-Shafique CAP5415 Computer Vision 2003*
Normalized cuts: Results

Figure from “Image and video segmentation: the normalised cut framework”, by Shi and Malik, copyright IEEE, 1998
Normalized cuts: Results

Figure from “Normalized cuts and image segmentation,” Shi and Malik, copyright IEEE, 2000
Contour and Texture Analysis for Image Segmentation

Contour and Texture Analysis for Image Segmentation

- Using Contours to Detect and Localize Junctions in Natural Images"
  M. Maire, P. Arbelaez, C. Fowlkes, and J. Malik. CVPR 2008

Now on CUDA

http://www.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/resources.html
Efficient Graph-Based Image Segmentation

Pedro F. Felzenszwalb and Daniel P. Huttenlocher
International Journal of Computer Vision, Volume 59, Number 2, September 2004
Integrating top-down and bottom-up segmentation

Fig. 1. Evaluation of contour detectors on the Berkeley Segmentation Dataset (BSDS300) Benchmark [2]. Leading contour detection approaches are ranked according to their maximum F-measure \((\frac{2 \cdot \text{Precision} \cdot \text{Recall}}{\text{Precision} + \text{Recall}})\) with respect to human ground-truth boundaries. Iso-F curves are shown in green. Our \(g^{Fb}\) detector [3] performs significantly better than other algorithms [2], [17], [18], [19], [20], [21], [22], [23], [24], [25], [26], [27], [28] across almost the entire operating regime. Average agreement between human subjects is indicated by the green dot.

http://www.eecs.berkeley.edu/Research/Projects/CS/vision/grouping/performance.html

Fig. 2. Evaluation of segmentation algorithms on the BSDS300 Benchmark. Paired with our \(g^{Fb}\) contour detector as input, our hierarchical segmentation algorithm \(g^{Fb}-owt-ucm\) [4] produces regions whose boundaries match ground-truth better than those produced by other methods [7], [29], [30], [31], [32], [33], [34], [35].