# CS4243
AY23/24 Sem 2
github.com/jasonqiu212

## 05. Segmentation

- Goal: Separate image into coherent regions
- Idea: **Clustering** - Group similar data points together
- Challenges: What makes 2 points same/different? Choice of features (e.g. Color, Intensity, Position), Which clustering algorithm?
- **k-Means Clustering** - Iteratively re-assign points to nearest cluster center

1. Given $K$, randomly initialize the cluster centers $c_1, \dots, c_K$
2. For each point $p_i$, find the closest $c_j$ and put $p_i$ into cluster $j$
3. Set $c_j$ to be mean of points in cluster $j$
4. Repeat, if $c_j$ have changed up to some threshold

- Pros: Simple, Converges to local min.
- Cons: Setting $K$, Sensitive to initial centers (Since k-means converges to local min.), Sensitive to outliers (Can add more clusters), Assumes spherical clusters (Fix with mean-shift)

- **Simple Linear Iterative Clustering (SLIC) Superpixels**
  - **Superpixel** - Group of pixels that share common traits
    - Application: Inputs to other CV algo. since more compact representation with perceptual meaning
  - Num. of pixels: $n_{tp}$; Target num. of superpixels: $n_{sp}$
  - Initial width of each superpixel: $s = \sqrt{\frac{n_{tp}}{n_{sp}}}$
  - Features: $z = [r, g, b, x, y]$
  - Color distance: $d_c = ||\langle r_j, g_j, b_j\rangle - \langle r_i, g_i, b_i\rangle||$
  - Spatial distance: $d_s = ||\langle x_j, y_j\rangle - \langle x_i, y_i\rangle||$
  - Scaling factors: $d_{cm}$ and $d_{sm} = s$ set as max. expected values of $d_c$ and $d_s$ respectively
  - $D = \sqrt{(\frac{d_c}{d_{cm}})^2 + (\frac{d_s}{d_{sm}})^2} = \sqrt{d_c^2 + (\frac{d_s}{s})^2 c^2}$

1. Split img. into grid of size $s \times s$. Set cluster centers as lowest gradient position in $3 \times 3$ neighborhood from superpixel center to speed up convergence since initialize on value common to surrounding.
2. For each cluster center, check distance to all pixels within $2s \times 2s$ neighborhood. Assign pixels to closest checked center.
3. Update cluster centers using mean and repeat if not converged (Same as k-Means)
4. Optional: Replace superpixel region by average value to create stained glass effect

- Modification of k-Means: Not random initialization, Compute pixel's distance only to closest set of cluster centers
- Can enforce connectivity and use other features too

- **Mean-Shift Clustering** - Find local density maxima in feature space
  - **Attraction basin** - Region in feature space for which all trajectories of centroids lead to same mode
  - **Cluster** - All data points in attraction basin of a mode

1. For each data point:

(1) Define window around and get centroid
(2) Shift window to centroid
(3) Repeat until window centroid stops moving

- Segmentation with Mean Shift: Do mean shift and merge pixels in same attraction basin
- Choosing window size: Trial and error, Sample points and use avg. dist. to knn. (Num. of neighbors needs to be large enough to ensure increase in density)
  - Larger window size $\rightarrow$ Fewer clusters
- Pros: No assumptions on cluster shape, 1 parameter, Finds variable num. of modes (vs. specified $k$ in k-Means), Robust to outliers
- Cons: Choosing $h$, Slow, Scales poorly with feature space dimension
- Optimizations:
  - After each run of mean shift, assign all points within radius $r$ of end point to same cluster
  - Assign all points within radius $c < r$ of search path to mode $\rightarrow$ More aggressive, less confident

## 06. Texture

- **Texture** - Pattern with repeating elements
- **Filter Bank** - Measures variety of structures in local neighborhood and generates multi-dimensional features
  - Goal: How to represent texture?
  - Idea: Apply filters with small windows to generate statistics that summarize local patterns. Dist. in feature space bet. windows $\rightarrow$ Pixel's texture similarity
  - $d$ filters $\rightarrow$ $d$-dimensional feature vector
  - Choosing window size: Try many sizes and look for one where statistic does not change much
  - Choose filters in different scales and orientations (to solve window size problem)
  - **Gabor Filter** - Represent filter banks mathematically by combining sinusoids with exp. (Gaussian) envelope
- **Texton** - Characterizes texture by replacing each pixel with integer representing **texture type**

1. Apply filter bank to training image
2. Cluster in feature space and store cluster centers (Texton dictionary)
3. For test image, filter image with same filter bank to get feature vector for each pixel. Assign each pixel to nearest cluster. Cluster ID = Texton ID.
4. For a given region, compute **texton histograms**

- Classification: Given new img., compare hist. to other trng. samples and assign to label with most similarity
- Segmentation: Use texton histograms as a feature

- **Perceived Boundaries** - Segmentation by human
  - Idea: Texture gradients indicate boundaries well

1. For each pixel, consider a disk that is split into 2 halves with some orientation
2. Measure texture diff. bet. 2 halves via texton hist.
3. Try all orientations. Orientation with high difference suggests boundaries.

## 07. Keypoints

- Motivation: How to stitch 2 images (e.g. Panaroma)?

1. Keypoints: Find locations
2. Descriptors: Rep. surrounding regions with math
3. Do the matching

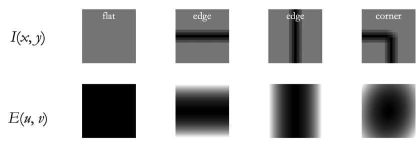
- Good keypoints are **repeatable** and **distinct**
- **Harris Corner Detection**
  - Significance: Corners have big changes in all directions when shifting window
  - Given window $W$ shifted by offset $(u, v)$:

$$E(U, v) = \sum_{(x,y)\in W} (I(x+u, y+v) - I(x, y))^2$$

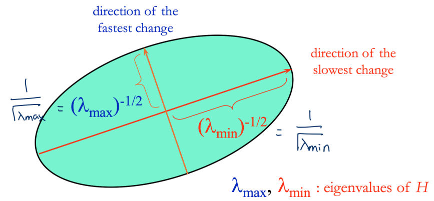
  - Assuming only small shifts (for Taylor Series Exp.):

$$E(u, v) = Au^2 + 2Buv + Cv^2 = \begin{bmatrix} u & v \end{bmatrix} \begin{bmatrix} A & B \\ B & C \end{bmatrix} \begin{bmatrix} u \\ v \end{bmatrix}$$

  - $A = \sum_{(x,y)\in W} I_x^2 \quad B = \sum_{(x,y)\in W} I_x I_y$
  - $C = \sum_{(x,y)\in W} I_y^2$
  - **2nd Moment Matrix (H)** - Middle matrix




  - $E = k$ visualized as ellipse, where $H$ controls shape
    - Eigenvectors of $H$ $\rightarrow$ Axes orientation
    - Eigenvalues of $H$ $\rightarrow$ Axes length




- **Eigenvectors** of $A$ are vectors $\mathbf{x}$ that: $A\mathbf{x} = \lambda\mathbf{x}$
- **Eigenvalue ($\lambda$)** corresponds to $\mathbf{x}$: $\det(A - \lambda I) = 0$
- Since $A = H$ is $2 \times 2$: $\lambda_{\pm} = \frac{1}{2}((h_{11} + h_{22}) \pm \sqrt{2h_{12}h_{21} + (h_{11} - h_{22})^2})$
- After getting $\lambda$s, find $\mathbf{x}$: $(A - \lambda I)\mathbf{x} = 0$
- Both $\lambda_{max}$ and $\lambda_{min}$ are large $\rightarrow$ Corner
- 'Cornerness' Score: $R = \min(\lambda_1, \lambda_2)$ (But getting $\lambda$ is slow)
- **Harris Operator**: $R = \det(H) - \kappa(\text{trace}(H))^2$
  - $\det(H) = AC - B^2 = \lambda_1\lambda_2$
  - $\text{trace}(H) = A + C = \lambda_1 + \lambda_2$
  - $R > 0 \rightarrow$ Corner, $R < 0 \rightarrow$ Edge, $R \approx 0 \rightarrow$ Flat

1. Compute gradient for each point in image
2. Compute $H$ matrix for each image window and get 'correctness' score
3. Find points with window where $R >$ threshold
4. Take points of local maxima

- **Non-Max. Suppression** - Iteratively search for max. values, then zero everything in surrounding window
  - Window size important
  - Adaptive: To prevent uneven distri. of keypoints in areas of higher contrast, pick corners which are both local max. and whose response is greater than all neighboring local max.
- In practice: $H = \sum_{(x,y)\in W} w_{x,y} \begin{bmatrix} I_x^2 & I_x I_y \\ I_x I_y & I_y^2 \end{bmatrix}$ (e.g. Convolve with Gaussian)

- **Harris Corner Invariances**
  - Purpose: If img. transf., how repeatable is detection?
  - **Equivariance** - Image transformed, and detection location undergoes similar transformation
  - **Invariance** - Image tranf., but no detection score change
  - Translation: Equivariant and invariant
  - Rotation: Equivariant and invariant
  - Photometric transformation (Assume $I' = aI + b$): Invariant to $b \neq 0$, but not invariant to $a \neq 1$
  - Scaling: Not equivariant and invariant
    - Scale of window can determine if location is keypoint $\rightarrow$ Need to scale up window by image scale
    - **Auto. Scale Selection** - When looking for keypoints, try window sizes and find scale that gives local max.

- **Laplacian of Gaussian** - Alternative keypoint detector which detects 'blobs' and is scale-sensitive
  - $\nabla^2 g = \frac{\partial^2 g}{\partial x^2} + \frac{\partial^2 g}{\partial y^2}$
  - Practice: Approx. by Difference of Gaussian for speed
  - Idea: Convolution with LoG has highest response when signal has same scale as Gaussian. Built-in scale sensitivity by varying scale $\sigma$.
  - Implementation: Fix window and kernel size; rescale img. with Gaussian blurring and downsampling

# 08. Descriptors

- Goal: Get feature vector surrounding each keypoint and measure similarity between feature vectors for matching
- Desc. should be invariant/equivariant and unique. E.g.:
  - Raw intensity: Good for exact template matching, but sensitive to lighting
  - Image gradient: Invariant to raw intensity (i.e. Lighting), but sensitive to transformations
  - Color histogram: Invariant to scale and rotation, but not sensitive to spatial layout
  - Spatial histogram: Compute color histograms over spatial cells. But not invariant to large rotations.
    - Orientation normalization: Normalize orientation of patch based on dominant image gradient
    - Save orien. angle $\theta$ w/ keypoint (e.g. Mean, mode)
  - **GIST Descriptor** - Rough spatial distribution of image gradients that is rotation invariant
    1. Divide image into $4 \times 4$ grid
    2. Apply Gabor filters (All dir. edge; $N$ filters)
    3. Compute filter response averages for each cells
    4. Size of descriptor: $4 \times 4 \times N$
- **SIFT** - Keypoint detector and descriptor
  - Detector: Uses multi-scale LoG to get scale invariance, orientation normalization for rotation invariance, and threshold for removing low-contrast and low-curvature keypoints
- **SIFT Descriptor**

1. Take $16 \times 16$-pixel window around keypoint. Partition window into $4 \times 4$ grid.
2. Compute **gradient orientations and magnitudes** for each pixel. Reweight magnitudes using Gaussian and discard pixels with low magnitude.
3. For each $4 \times 4$-pixel cell, make **histogram with 8 orientation bins**. Shift histogram binning by dominant orien. (i.e. Subtract by dom. orien.) for rotation invariance. Collapse into $1 \times 128$ vector.
4. Normalize vector to unit length

- Invariant to scale, rotation, and lighting
- Partially invariant to viewpoint (Up to $60°$)
- Quick and efficient

- **Feature Matching** - Given feature in $I_1$, how to find best match in $I_2$?

1. Define distance function that compares desc.
   - Euclidean distance: $||f_1 - f_2||$ (Can give small distances for incorrect matches)
   - Ratio distance bet. best vs. next-best: $\frac{||f_1 - f_2}{||f_1 - f_2'||}$
2. Nested loop: Find vector with min. distance in $I_2$
   - Or ratio of best vs. next-best $<$ threshold

- Evaluation: **ROC curve** by varying threshold
  - Recall vs. 1 - Specificity
  - Area Under the Curve (AUC): 1 is the best
  - **Recall** - $\frac{TP}{TP+FN}$ **Specificity** - $\frac{TN}{TN+FP}$
  - **Precision** - $\frac{TP}{TP+FP}$

# 09. Homography

- Goal: Stitch images from diff. viewpoints via projection
- When to use: Scene is planar, approx. planar (i.e. Small depth variation), or only camera rotation
- Problem: Given set of matched keypoints $\{p_i, p_i'\}$, get transformation $p' = f(p; H)$ where $H$ are parameters
- Given homography function: Convert to homogeneous coord., multiply by homo. matrix, and convert back

$$p = \begin{bmatrix} x \\ y \end{bmatrix} P = \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}; P' = HP; P' = \begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} p' = \begin{bmatrix} \frac{x'}{w'} \\ \frac{y'}{w'} \end{bmatrix}$$

- **Direct Linear Transform** - Find best estimate of $H$

1. For each matching, create $2 \times 9$ matrix $A_i$
   - $A_i = \begin{bmatrix} -x & -y & -1 & 0 & 0 & 0 & xx' & yx' & x' \\ 0 & 0 & 0 & -x & -y & -1 & xy' & yy' & y' \end{bmatrix}$
2. Concatenate into $2n \times 9$ matrix $A$
3. Compute SVD of $A = U \sum V^T$
4. Store vector of smallest singular value $h = v_{\hat{i}}$
5. Reshape to get $H$

- Assumptions: Projective model with linear transf.
- Cons: Sensitive to scaling (i.e. $P$: High res.; $P'$: Low res.) $\rightarrow$ Normalize, Outliers $\rightarrow$ Poor est. of $H$

- **RANSAC** - More robust method for est. homographies
  - Motivation: DLT easily corrupted by outliers

1. Loop $N$ times
   (1) Sample randomly num. of pts. required to fit model
   (2) Solve for model params. using samples
   (3) Score by fraction of inliers within preset threshold $\delta$ of model
2. Fit model to samples with most inliers

1. RANSAC loop
   (1) Sample 4 matches ($H$ has 8 deg. of freedom)
   (2) Compute $H$ using DLT
   (3) Inliers: Get $P''$ using $H$ and check distance to $P'$
   (4) Keep $H$ if largest number of inliers
2. Using best $H$ with most inliers, recompute using all inliers

- $\delta$: Impacts if inliers are kept (Trial and error)
- $N = \frac{\log(1-p)}{\log(1-(1-e)^s)}$ where $p$ is prob. that $\geq 1$ set of samples does not contain outliers, $e$ is prob. that point is outlier, and $s$ is num. of samples per iter.
- Can loop $N$ times or stop early when expected prob. of inliers reached, but both need prob. of outliers
- Integrate RANSAC with feature matching: Compute matches as before, add RANSAC loop and eliminate some matches that do not fit $H$

- **Warping** - Moves pixels of an image
  - Mapping: Transf. from source to destination via $f$
  - Resampling: **Splat** if map to bet. pixels, avg. if receive $> 1$ source

# 10. Optical Flow

# 11. Tracking

# 12. Deep Learning