EE795: Computer Vision and Intelligent Systems

Spring 2012
TTh 17:30-18:45 FDH 204

Lecture 09
130219

http://www.ee.unlv.edu/~b1morris/ecg795/

Outline

- Review
  - Feature Descriptors
- Feature Matching
- Feature Tracking
- Edges
  - Canny Edge Detector
- Lines
  - Hough Transform

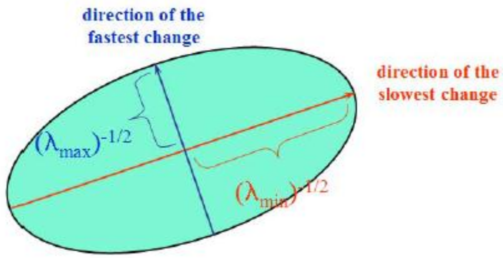
Interest Point Detection

$$E_{AC}(\Delta\boldsymbol{u}) = \sum_i w(\boldsymbol{x}_i)\,[I_0(\boldsymbol{x}_i - \Delta\boldsymbol{u}) - I_0(\boldsymbol{x}_i)]^2$$
$$= \sum_i w(\boldsymbol{x}_i)\,[\nabla I_0(\boldsymbol{x}_i) \cdot \Delta\boldsymbol{u}]^2$$
$$= \Delta\boldsymbol{u}^T A \Delta\boldsymbol{u}$$

- $\nabla I_0(\boldsymbol{x}_i)$ - image gradient
  - We have seen how to compute this
- A – autocorrelation matrix

$$A = w * \begin{bmatrix} I_x^2 & I_x I_y \\ I_y I_x & I_y^2 \end{bmatrix}$$

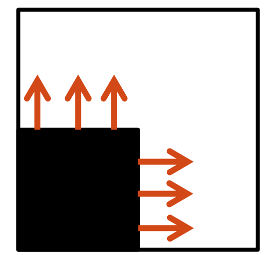
  - Compute gradient images and convolve with weight function
  - Also known as second moment matrix



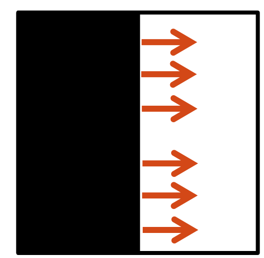
- The matrix A provides a measure of uncertainty in location of the patch
- Do eigenvalue decomposition
  - Get eigenvalues and eigenvector directions
  - Good features have both eigenvalues large
- Quantify uncertainty
  - Easiest: look for maxima in the smaller eigenvalue [Shi and Tomasi]
  - $\det(A) - \alpha\ \mathrm{trace}(A)^2$ [Harris]
  - See book for other methods

Interest Point Detection II

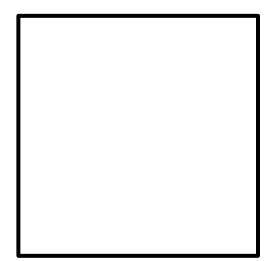
- The correlation matrix gives a measure of edges in a patch
- Corner
  - Gradient directions
    - $\begin{bmatrix}1\\0\end{bmatrix}, \begin{bmatrix}0\\1\end{bmatrix}$
  - Correlation matrix
    - $A \propto \begin{bmatrix}1 & 0\\0 & 1\end{bmatrix}$



- Edge
  - Gradient directions
    - $\begin{bmatrix}1\\0\end{bmatrix}$
  - Correlation matrix
    - $A \propto \begin{bmatrix}1 & 0\\0 & 0\end{bmatrix}$



- Constant
  - Gradient directions
    - $\begin{bmatrix}0\\0\end{bmatrix}$
  - Correlation matrix
    - $A \propto \begin{bmatrix}0 & 0\\0 & 0\end{bmatrix}$



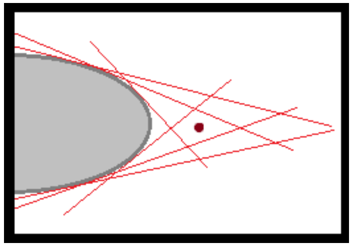
Basic Feature Detection Algorithm

1. Compute the horizontal and vertical derivatives of the image I_x and I_y by convolving the original image with derivatives of Gaussians (Section 3.2.3).
2. Compute the three images corresponding to the outer products of these gradients. (The matrix $\boldsymbol{A}$ is symmetric, so only three entries are needed.)
3. Convolve each of these images with a larger Gaussian.
4. Compute a scalar interest measure using one of the formulas discussed above.
5. Find local maxima above a certain threshold and report them as detected feature point locations.

Algorithm 4.1 Outline of a basic feature detection algorithm.

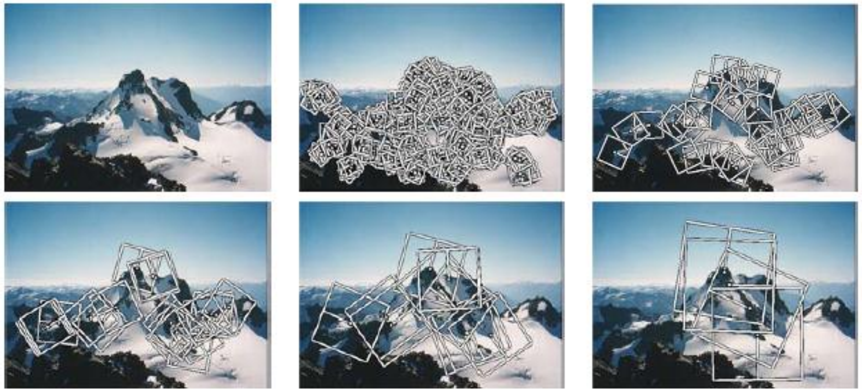
Improving Feature Detection

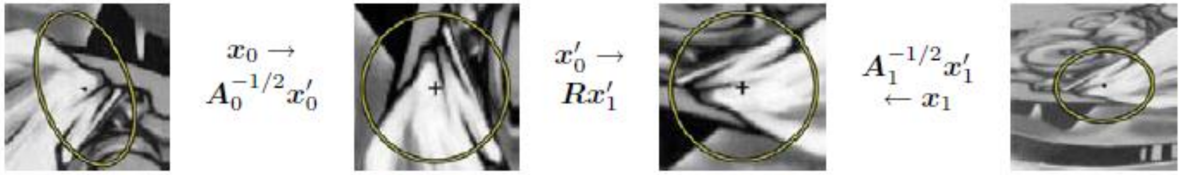
- Corners may produce more than one strong response (due to neighborhood)
  - Estimate corner with subpixel accuracy – use edge tangents
  - Non-maximal suppression – only select features that are far enough away
    - Create more uniform distribution – can be done through blocking as well
- Scale invariance
  - Use an image pyramid – useful for images of same scale
  - Compute Hessian of difference of Gaussian (DoG) image
  - Analyze scale space [SIFT – Lowe 2004]
- Rotational invariance
  - Need to estimate the orientation of the feature by examining gradient information
- Affine invariance
  - Closer to appearance change due to perspective distortion
  - Fit ellipse to autocorrelation matrix and use it as an affine coordinate frame
  - Maximally stable region (MSER) [Matas 2004] – regions that do not change much through thresholding



(a) Strongest 250

(c) ANMS 250, $r = 24$





Feature Descriptors

- Once keypoints have been detected the local appearance needs to be compactly represented
  - The representation should enable efficient matching
- Why not use the image patch itself as the descriptor?
  - The descriptor should remain the same in any image
    - Robust to photometric effects, lighting, orientation, scale, affine deformation
  - The patch intensity can be used in cases where the isn't much appearance change between images (e.g. stereo images, satellite images, video)
- The definition of descriptors to deal with the aforementioned issues is still very active

Scale Invariant Feature Transform (SIFT)

- One of the most popular feature descriptor [Lowe 2004]
  - Many variants have been developed
- Descriptor is invariant to uniform scaling, orientation, and partially invariant to affine distortion and illumination changes
- Descriptor computation:
  - Compute gradient 16×16 grid around keypoint
    - Keep orientation and down-weight magnitude by a Gaussian fall off function
      - Avoid sudden changes in descriptor with small position changes
      - Give less emphasis to gradients far from center
  - Form a gradient orientation histogram in each 4×4 quadrant
    - 8 bin orientations
    - Trilinear interpolation of gradient magnitude to neighboring orientation bins
    - Gives 4 pixel shift robustness and orientation invariance
  - Final descriptor is $4 \times 4 \times 8 = 128$ dimension vector
    - Normalize vector to unit length for contrast/gain invariance
    - Values clipped to 0.2 and renormalized to remove emphasis of large gradients (orientation is most important)

SIFT Schematic

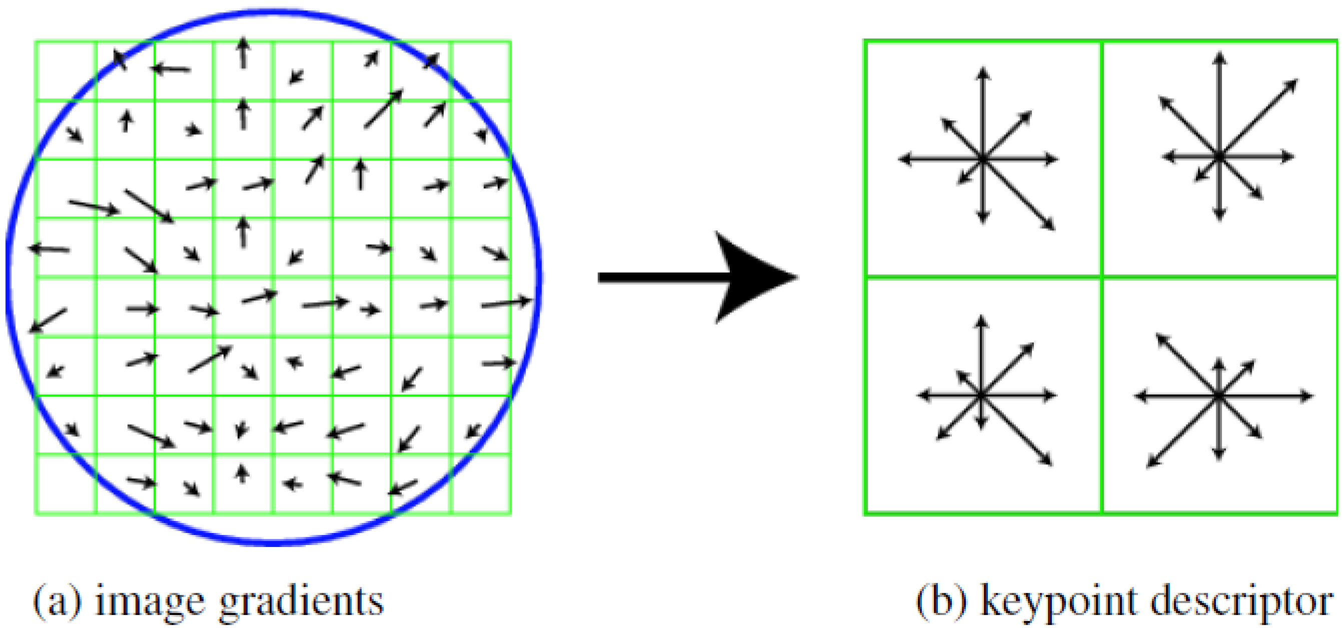
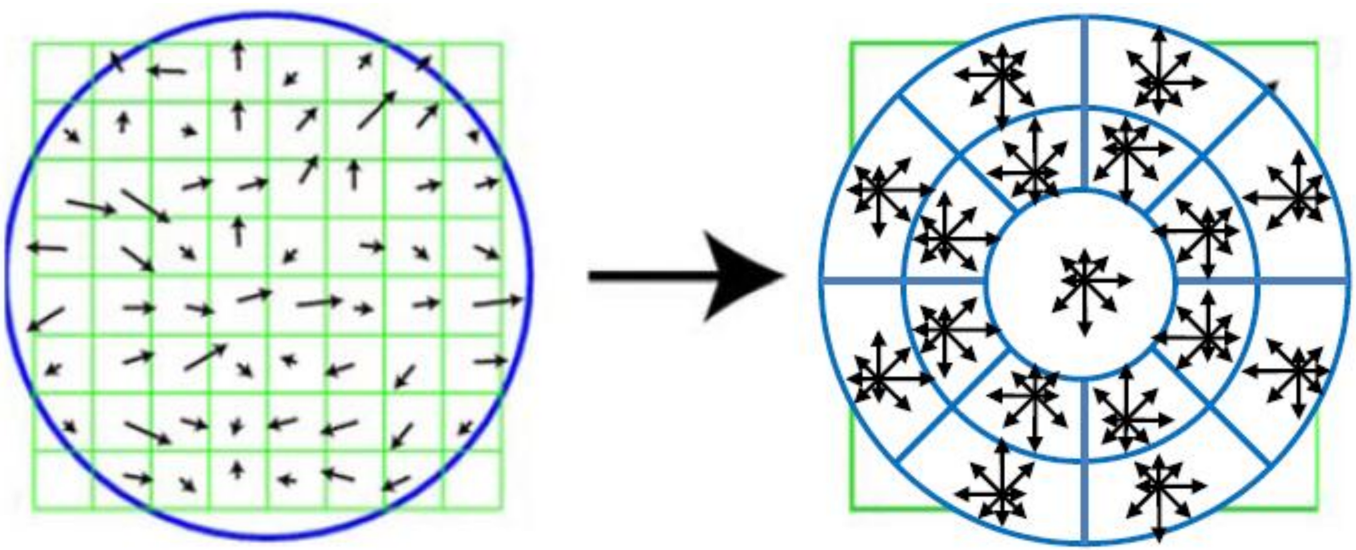


Figure 4.18 A schematic representation of Lowe's (2004) scale invariant feature transform (SIFT): (a) Gradient orientations and magnitudes are computed at each pixel and weighted by a Gaussian fall-off function (blue circle). (b) A weighted gradient orientation histogram is then computed in each subregion, using trilinear interpolation. While this figure shows an 8×8 pixel patch and a 2×2 descriptor array, Lowe's actual implementation uses 16×16 patches and a 4×4 array of eight-bin histograms.

Gradient Location-Orientation Histogram (GLOH)

- Variant on SIFT to use log-polar binning rather than 4 × 4 quadrant
  - Slightly better performance than SIFT
  - 272D histogram is projected onto 128D



(a) image gradients (b) keypoint descriptor

Figure 4.19 The gradient location-orientation histogram (GLOH) descriptor uses log-polar bins instead of square bins to compute orientation histograms (Mikolajczyk and Schmid 2005).

Other SIFT Variants

- Speeded up robust features (SURF) [Bay 2008]
  - Faster computation by using integral images (Szeliski 3.2.3 and later for object detection)
  - Popularized because it is free for non-commercial use
    - SIFT is patented
- OpenCV implements many
  - FAST
  - ORB
  - BRISK
  - FREAK
- OpenCV is maintained by Willow Garage, a robotics company
  - Emphasis on fast descriptors for real-time applications

Feature Matching

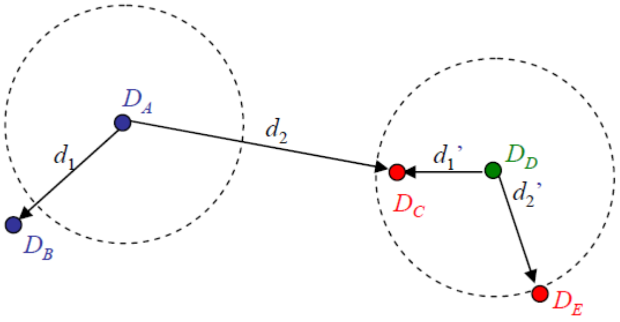
- Given descriptors from images, determine correspondences between descriptors
- Two parts to the problem
  - Matching strategy – how to select “good” correspondences
  - Efficient search – data structures and algorithms to perform matching quickly

Matching Strategy

- Generally, assume that the feature descriptor space is sufficient
  - Perform whitening of vector to concentrate on more interesting dimensions
- Use Euclidean distance as the error metric
- Set threshold to only return potential matches that are within some predefined "similarity"
  - Returns all patches from the other image that are similar enough
  - Threshold must be set appropriately to ensure matches are detected without introducing too many erroneous ones

Improved Threshold Matching

- Fixed threshold is difficult to set
  - Shouldn't expect different regions in feature space to behave the same
- Nearest neighbor matching
  - Only return the closest matching feature
  - A threshold is still required to restrict matching to "good" matches
- Nearest neighbor distance ratio
  - Adapt threshold for each feature
  - $NNDR = \frac{d_1}{d_2} = \frac{\|D_A - D_B\|}{\|D_A - D_C\|}$
    - Best if d_2 is a known not to match



Quantifying Performance

- Confusion matrix-based metrics
  - Binary {1,0} classification tasks

		actual value		
predicted outcome		p	n	total
	p'	TP	FP	P'
	n'	FN	TN	N'
	total	P	N	

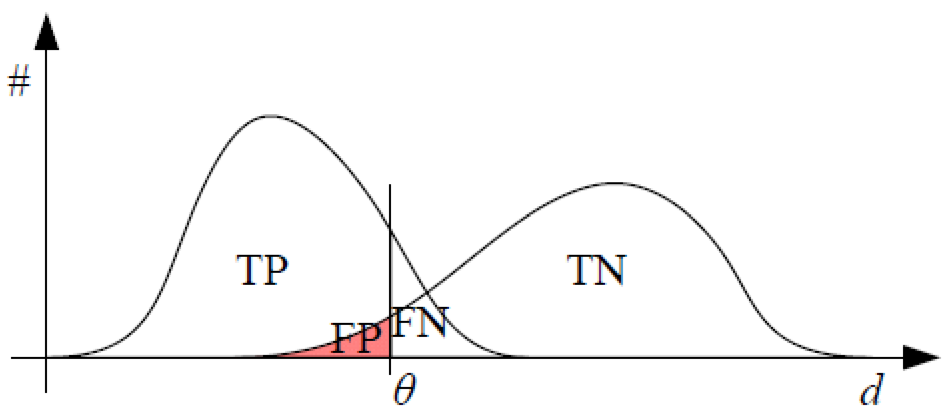
- True positives (TP) - # correct matches
- False negatives (FN) - # of missed matches
- False positives (FP) - # of incorrect matches
- True negatives (TN) - # of non-matches that are correctly rejected

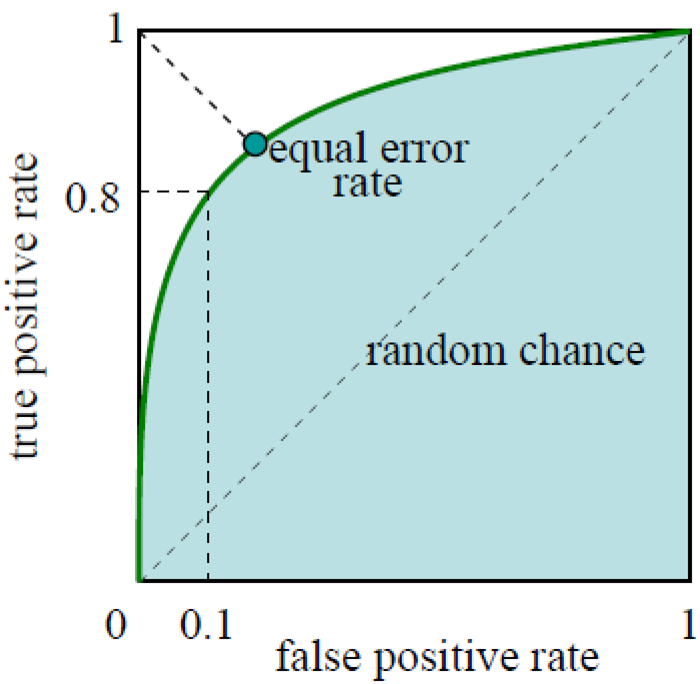
- A wide range of metrics can be defined

- True positive rate (TPR) (sensitivity)
  - $TPR = \frac{TP}{TP+FN} = \frac{TP}{P}$
  - Document retrieval → recall – fraction of relevant documents found
- False positive rate (FPR)
  - $FPR = \frac{FP}{FP+TN} = \frac{FP}{N}$
- Positive predicted value (PPV)
  - $PPV = \frac{TP}{TP+FP} = \frac{TP}{P'}$
  - Document retrieval → precision – number of relevant documents are returned
- Accuracy (ACC)
  - $ACC = \frac{TP+TN}{P+N}$

Receiver Operating Characteristic (ROC)

- Evaluate matching performance based on threshold
  - Examine all thresholds θ to map out performance curve
- Best performance in upper left corner
  - Area under the curve (AUC) is a ROC performance metric





Efficient Matching

- Straight forward matching compares all features with every other feature in every image
  - Quadratic in the number of features
- More efficient matching is possible with an indexing structure
  - Structure enables quick location of similar features
  - Can remove many potential search candidates quickly
- Popular methods are multi-dimensional trees or hash tables
  - Locality sensitive hashing, parameter-sensitive hashing
  - k-d trees

After Matching

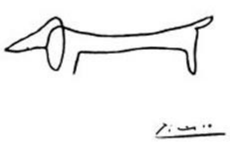
- Matching gives a list of potential correspondences
  - Must determine how to handle these maybe matches
- Different approaches depending on task
  - Object detection – enough matching points constitutes a detection
  - Image level consistency (e.g. rotation) – determine inliers/outliers to estimate image transformation
- Random sampling (RANSAC) is very popular when there is a model to fit
  - Take a small random subset of matches, compute the model, and verify on the remaining matches

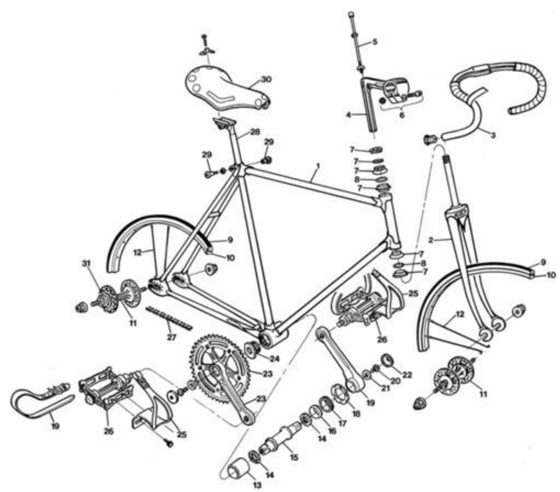
Feature Tracking

- Detect then track approach useful for video processing
- Use the same features we have already seen
- Tracking accomplished by SSD or NCC
  - Usually appearance is sufficient
- Large motions require hierarchical search strategies
  - Match in lower-resolution to provide an initial guess for speeded up search
- Must adapt the appearance model over longer time periods
  - Kanade-Lucas-Tomasi (KLT) tracker estimates affine transformation of the patch in question

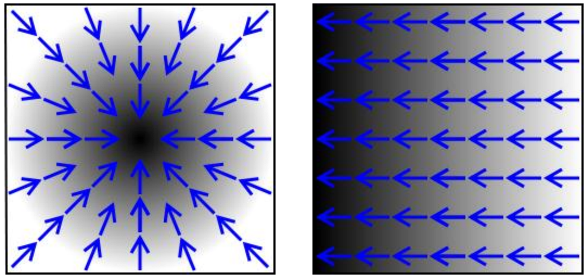
Edges

- 2D point features are good for matching
  - Limited number of "good" points
- Edges are plentiful and carry semantic significance
  - Object boundaries denoted by visible contours
  - Occur at boundaries between regions of different color, intensity, and texture
    - HoG descriptor for object recognition





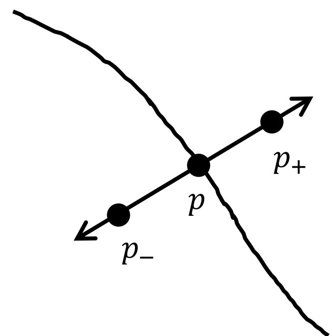
Edge Detection

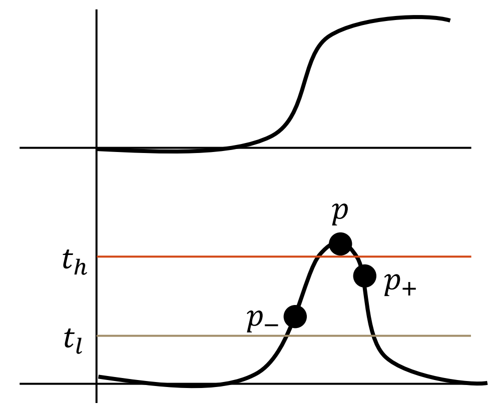


- Gradient – slope and direction
  - $J(x) = \nabla I(x) = \left(\frac{\partial I}{\partial x}, \frac{\partial I}{\partial y}\right)(x)$
    - Points in direction of steepest ascent in intensity
    - Magnitude is slope strength
    - Orientation points perpendicular to local contour
- Typically, smooth image with Gauassian before computing gradient
  - Derivative accentuates high frequency
  - $J_\sigma(x) = \nabla[G_\sigma(x) * I(x)] = \nabla[G_\sigma(x)] * I(x)$
    - $\nabla G_\sigma(x) = \left(\frac{\partial G_\sigma}{\partial x}, \frac{\partial I G_\sigma}{\partial y}\right)(x) = [-x, -y]\frac{1}{\sigma^3}\exp\left(-\frac{x^2+y^2}{2\sigma^2}\right)$
- Thinner edges are obtained with second derivatives
- Laplacian – looks for zero crossings
  - $S_\sigma(x) = \nabla \cdot J_\sigma(x) = [\nabla^2 G_\sigma(x) * I(x)]$
- Laplacian of Gaussian (LoG) kernel
  - $\nabla^2 G_\sigma(x) = \frac{1}{\sigma^3}\left(2 - \frac{x^2+y^2}{2\sigma^2}\right)\exp\left(-\frac{x^2+y^2}{2\sigma^2}\right)$
    - Separable kernel
  - Often this is approximated by a difference of Gaussians (DoG)
    - Easy to compute when doing an image pyramid

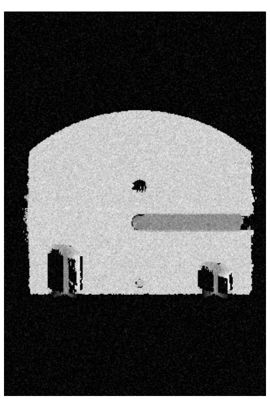
Canny Edge Detection

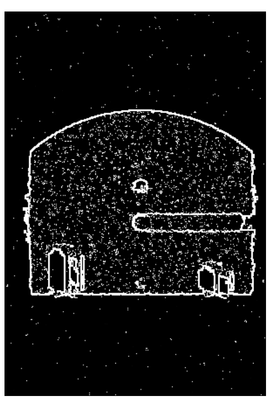
- Popular edge detection algorithm that produces a thin lines
- 1) Smooth with Gaussian kernel
- 2) Compute gradient
  - Determine magnitude and orientation (45 degree 8-connected neighborhood)
- 3) Use non-maximal suppression to get thin edges
  - Compare edge value to neighbor edgels in gradient direction

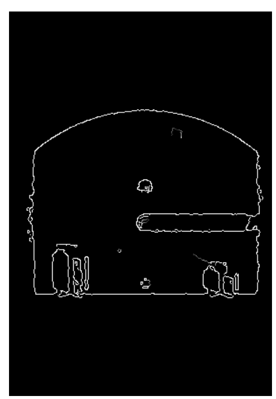




- 4) Use hysteresis thresholding to prevent streaking
  - High threshold to detect edge, low threshold to trace

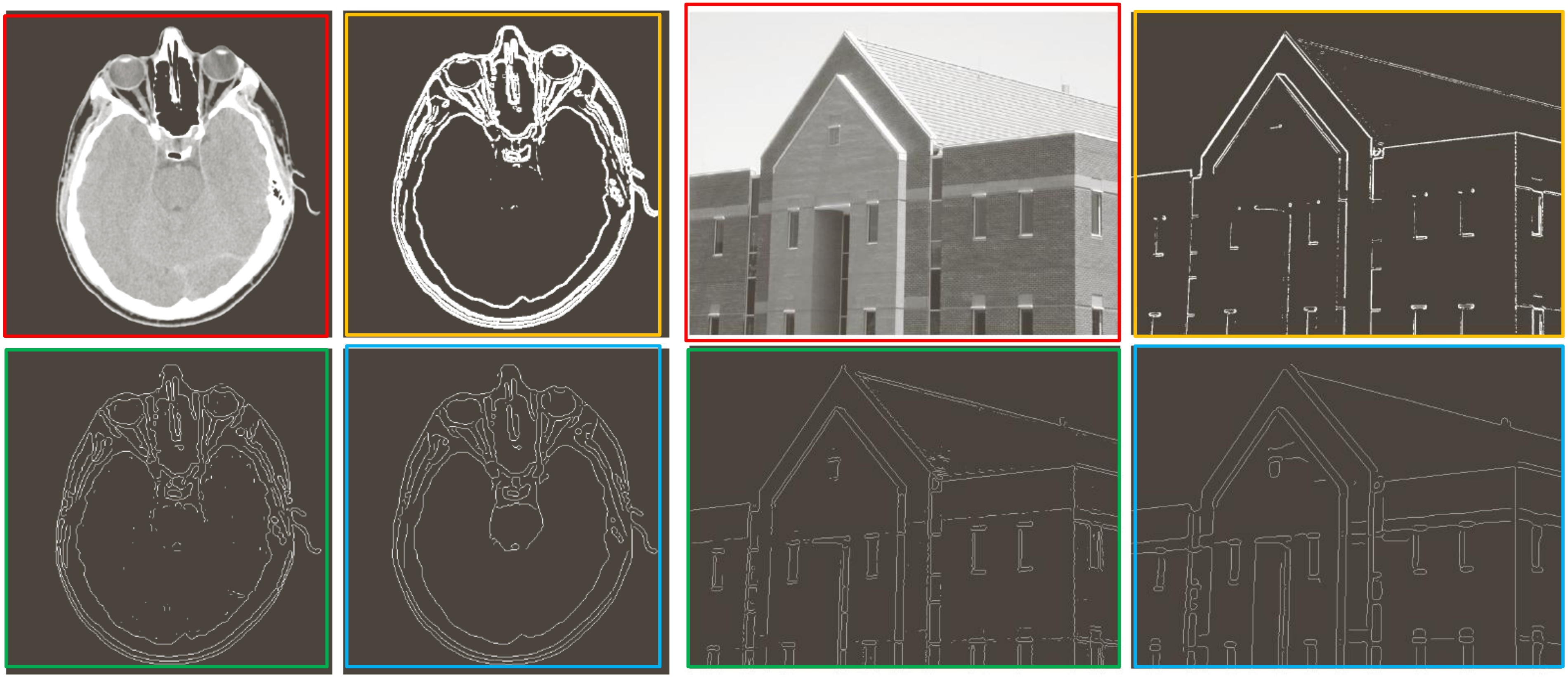

object


Sobel


Canny

http://homepages.inf.ed.ac.uk/rbf/HIPR2/canny.htm

Canny Edge Detection Results



- Original image
- Thresholded gradient of smoothed image (thick lines)
- Marr-Hildreth algorithm
- Canny algorithm (low noise, thin lines)

Lines

- Edges and curves make up contours of natural objects
  - Man-made world uses straight lines

- 3D lines can be used to determine vanishing points and do camera calibration
- Estimate pose of 3D scene

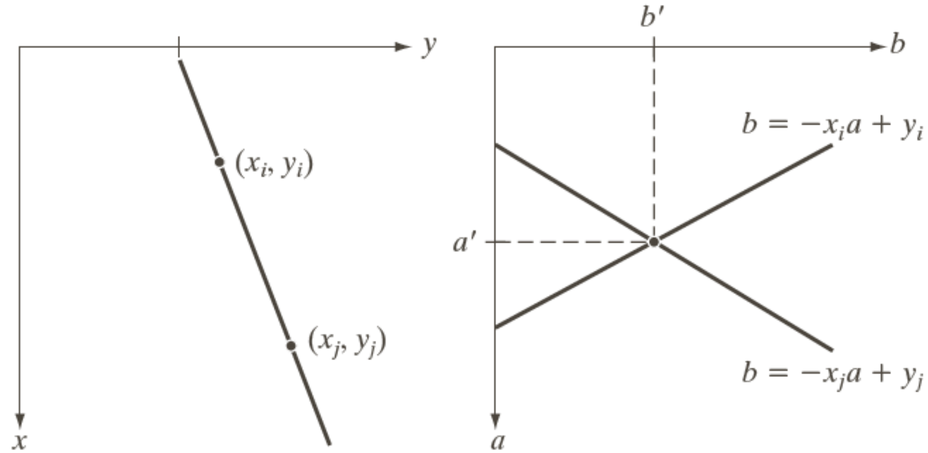
Hough Transform

- Lines in the real-world can be broken, collinear, or occluded
  - Combine these collinear line segments into a larger extended line
- Hough transform creates a parameter space for the line
  - Every pixel votes for a family of lines passing through it
  - Potential lines are those bins or accumulator values with high count
- Uses global rather than local information

- See `hough.m`, `radon.m` in Matlab

Hough Transform Insight

- Want to search for all points that lie on a line
  - This is a large search (take two points and count the number of edgels)
- Infinite lines pass through a point (x_i, y_i)
  - $y_i = ax_i + b$
- Reparameterize
  - $b = -x_i a + y_i$
  - ab-space representation has single line defined by point (x_i, y_i)



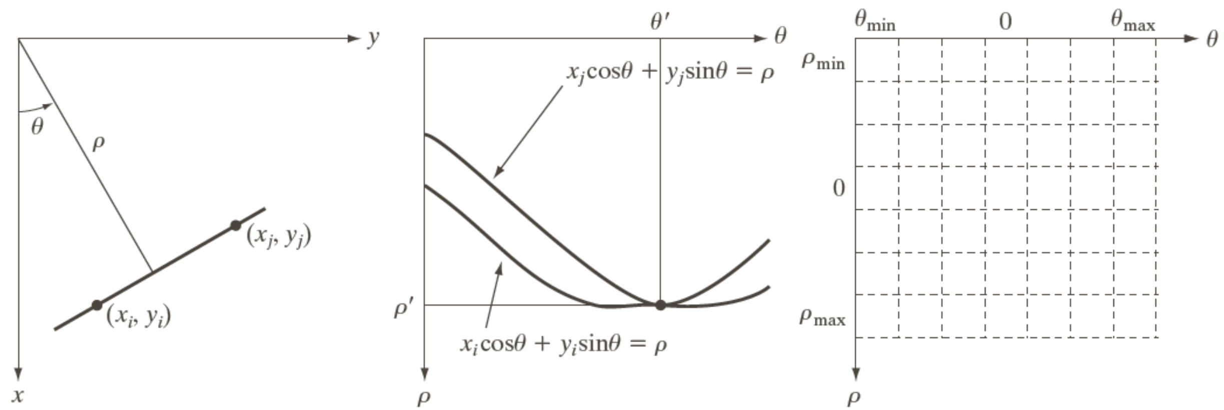
a b

FIGURE 10.31 (a) xy-plane. (b) Parameter space.

- All points on a line will intersect in parameter space
  - Divide parameter space into cells/bins and accumulate votes across all a and b values for a particular point
  - Cells with high count are indicative of many points voting for the same line parameters (a, b)

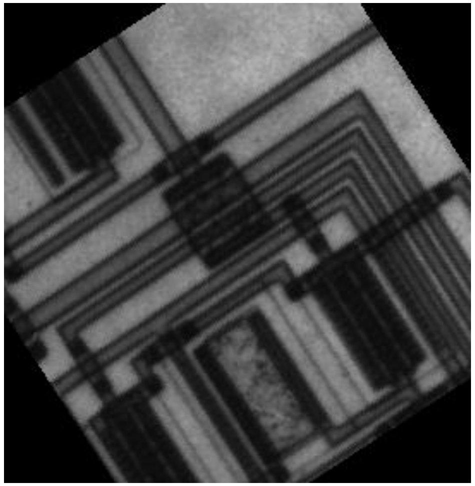
Hough Transform in Practice

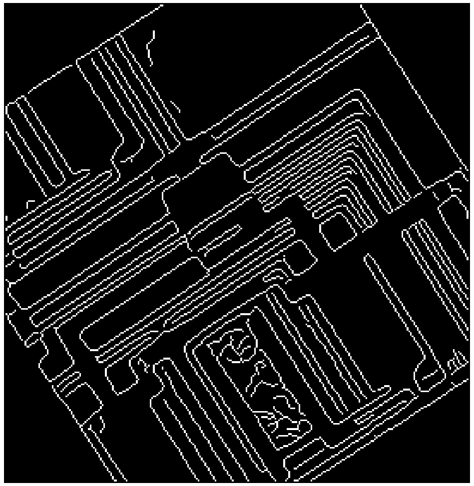
- Use a polar parameterization of a line – why?

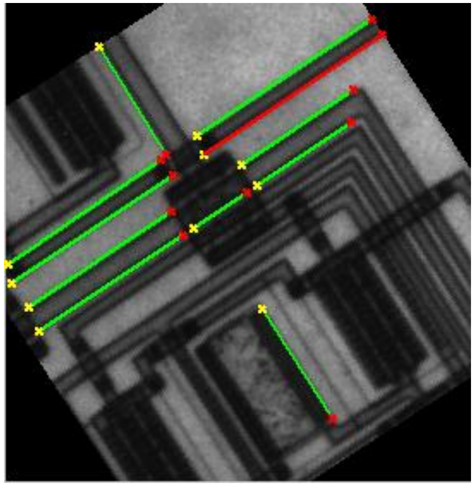


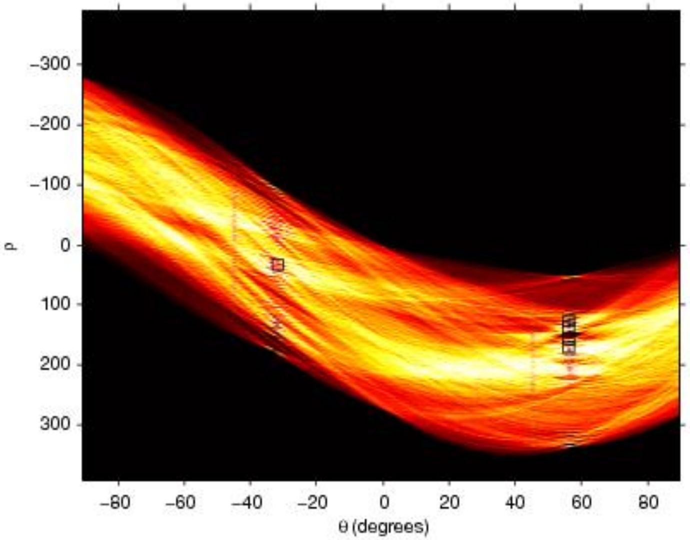
- After finding bins of high count, need to verify edge
  - Find the extent of the edge (edges do not go across the whole image)
- This technique can be extended to other shapes like circles

Hough Transform Example









http://www.mathworks.com/help/images/analyzing-images.html