Machine Vision

Lecture Set – 07

Contours

Huei-Yung Lin

Robot Vision Lab

5/11/2023

- Homework #5 will be given later.
- Homework #4 due TODAY

Contours

- What is a contour in computer vision?
  - Edges must be linked into a representation for a region boundary
  - They can be open or closed
    - Closed contours correspond to region boundaries, the pixels inside can be found by region filling
    - Open contours could be part of region boundaries, or line segments of drawing or handwriting, etc.
  - Gaps on region boundaries mainly caused by low contrast between regions (thus no edges detected)

Representation of Contours

- Criteria for good contour representations
  - Efficiency: A simple, compact representation
  - Accuracy: Fit the image features accurately
  - Effectiveness: Suitability for the operations performed in later stages of the applications
- The accuracy of a contour representation is determined by
  - The form of curve used to model the contour
  - The performance of the curve fitting algorithm
  - The accuracy of the estimates of edge location

Contour Representation

- Two contour representations:
  - Ordered list of edges
    - A very simple representation
    - As accurate as the location estimates for the edges
    - Not compact, may not provide effective representation for further image analysis
  - Fitting curve
    - Gives more accuracy (reduce errors by averaging)
    - More efficient for further processing
      - Determine the orientation and length, etc.

Some Definitions

- Interpolation and approximation
  - A curve interpolates a list of points if the curve passes through them
  - Approximation is fitting a curve to a list of points with the curve passing close to the points, but not necessarily passing exactly through the points
- An edge list is an ordered set of edge points or fragments
- A contour is an edge list or the curve that has been used to represent the edge list
- A boundary is the closed contour that surrounds a region

Geometry of Curves

- Planar curves can be represented by
  - The explicit form: $y = f(x)$
  - The implicit form: $f(x,y) = 0$
  - The parametric form: $(x(u),y(u))$ for some u
- For the parametric form, let $\mathbf{p}_1 = (x(u_1),y(u_1))$ and $\mathbf{p}_2 = (x(u_2),y(u_2))$ be two points on a curve, then
  - The length of the curve is given by $$\int_{u_1}^{u_2} \sqrt{\left(\frac{dx}{du}\right)^2 + \left(\frac{dy}{du}\right)^2}\, du$$
  - The unit tangent vector is given by $$\mathbf{t}(u) = \frac{\mathbf{p}'(u)}{|\mathbf{p}'(u)|}$$
  - The normal to the curve is given by $\mathbf{n}(u) = \mathbf{p}''(u)$

Digital Curves

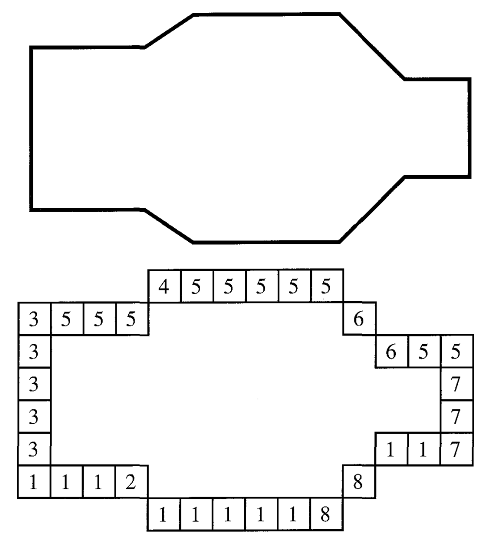
- Difficult to compute a digital curve's slope and curvature due to 45 degrees increments
- One way: via the non-adjacent points in the edge list
  - Let $\mathbf{p}_i = (x_i, y_i)$ be the coordinates of edge i in the edge list
    - k-slope: the direction vector between points that are k edge points apart
    - Left k-slope: from $\mathbf{p}_{i-k}$ to $\mathbf{p}_i$
    - Right k-slope: from $\mathbf{p}_i$ to $\mathbf{p}_{i+k}$
    - k-curvature: the difference between the left and right k-slopes
- Length of digital curve is given by

$$S = \sum_{i=2}^{n} \sqrt{(x_i - x_{i-1})^2 + (y_i - y_{i-1})^2}$$

Chain Codes

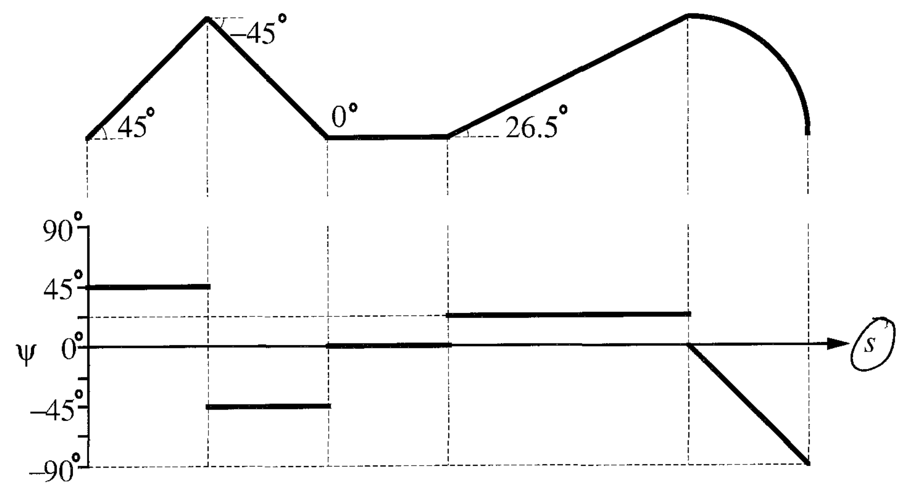
- Chain codes:
  - Used to record the list of edge points along a contour
  - Specifies the direction of a contour at each edge in the edge list
- The chain code: an edge list by the coordinates of the first edge and the list of chain codes leading to subsequent edges
- Properties of chain codes
  - $n \times 45^{\circ}$ rotation of object: Adding n mode 8 to original chain code
- Chain code's directions:

2	3	4
1		5
8	7	6



Slope Representation

- Slope representation of a contour: Ψ-s plot
  - s is the segment
  - Ψ is the slope angle



- Slope Density Function
  - The histogram of the slopes along a contour
  - It can be used for recognition
    - Correlating the slope density function of a model contour with the slope density function extracted from an image
    - This gives the orientation of the object

Curve Fitting

- Curve models for fitting edge points
  - Line segments
    - Used for the scene consisting of straight lines
  - Circular arcs
    - Used for estimating curvature
  - Conics sections
    - Used to represent lines, circular, elliptic and hyperbolic arcs
  - Cubic splines
    - Used to model smooth curves
- Two problems in fitting algorithms
  - What method?
  - How accurate?

Error Measures for Curve Fitting

- Let d_i be the distance of edge point i from a line
- Commonly used "measures" for curve fitting algorithms:
  - Maximum absolute error $\mathrm{MAE} = \max_i |d_i|$
    - Measures how much the points deviate in the worst case
  - Mean squared error (MSE) $\mathrm{MSE} = \frac{1}{n}\sum_{i=1}^{n} d_i^2$
    - Gives an overall measure of the deviation
  - Normalized maximum error $\varepsilon = \frac{\max_i |d_i|}{S}$
    - The ratio of the MAE to the length of the curve
  - Number of sign changes in the error
    - Indicates how appropriate the curve model is (how?)
  - Ratio of curve length to end point distance
    - Measures how complex the curve is (why?)

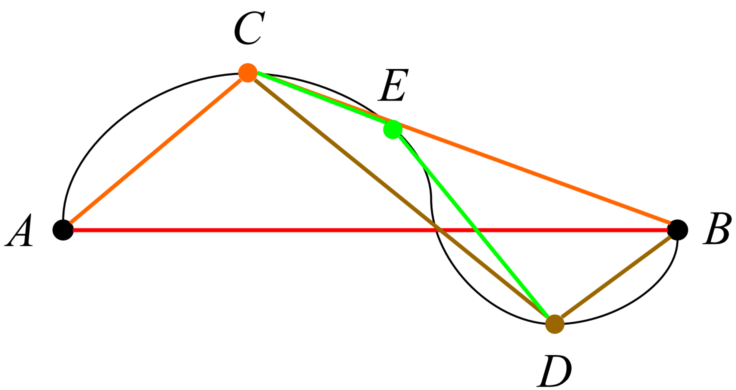
5/18/2023

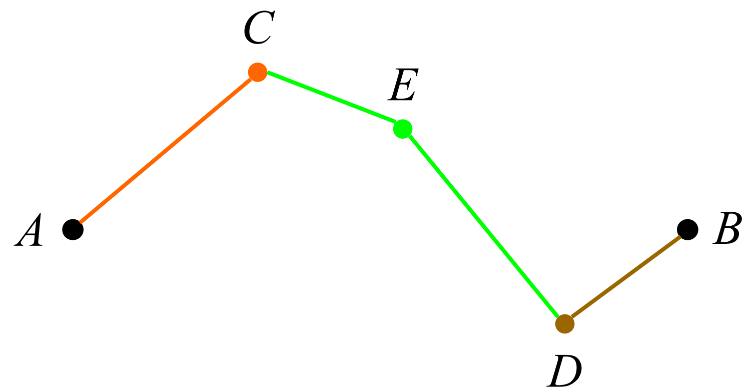
Polyline Representation

- Polyline representation
  - A sequence of line segments joined end to end
  - Fits the edge list of the contour with a sequence of line segments
  - Interpolates a selected subset of edge points in the edge list
  - *The ends of each line segment are edge points in the edge list*
  - A polyline algorithm takes as input an ordered list of edge points $\{(x_1, y_1), \ldots, (x_n, y_n)\}$
- There are two approaches to fitting polylines:
  - Top-down splitting
  - Bottom-up merging

Polyline Splitting

- The top-down splitting algorithm recursively adds vertices, starting with an initial curve
- Splitting method for polylines:
  - Normalized maximum error and threshold

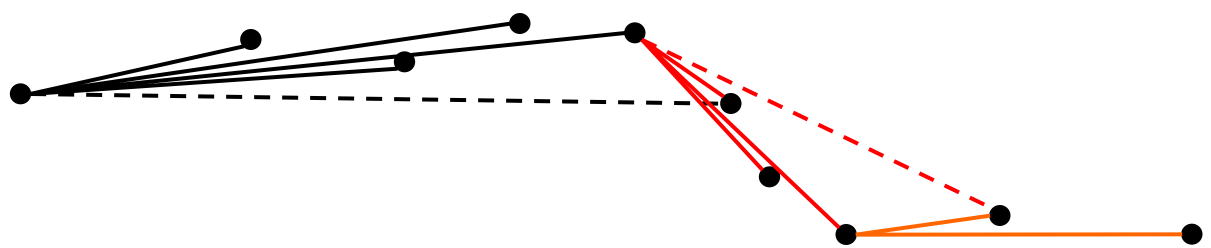




- Segment splitting is also called recursive subdivision

Segment Merging

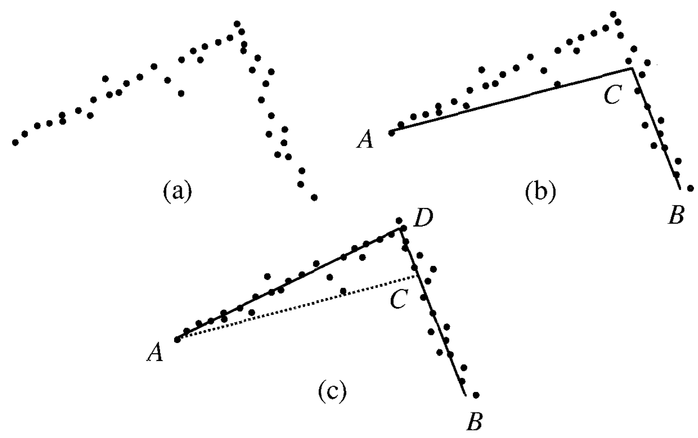
- In segment merging, edge points are added to line segment as the edge list is traversed
- New segments are started when the edge points deviate too far from the line segment
  - Sequential least-squares measure



- The merge approach is also called bottom-up approach to polyline fitting

Spilt and Merge

- The top-down method of recursive subdivision and the bottom-up method of merging can be combined as the split and merge algorithm
- The basic idea is to interleave split and merge process
  - After recursive subdivision, replace adjacent segments by a single one with less normalized error
  - After segment merging, split the new segment if necessary



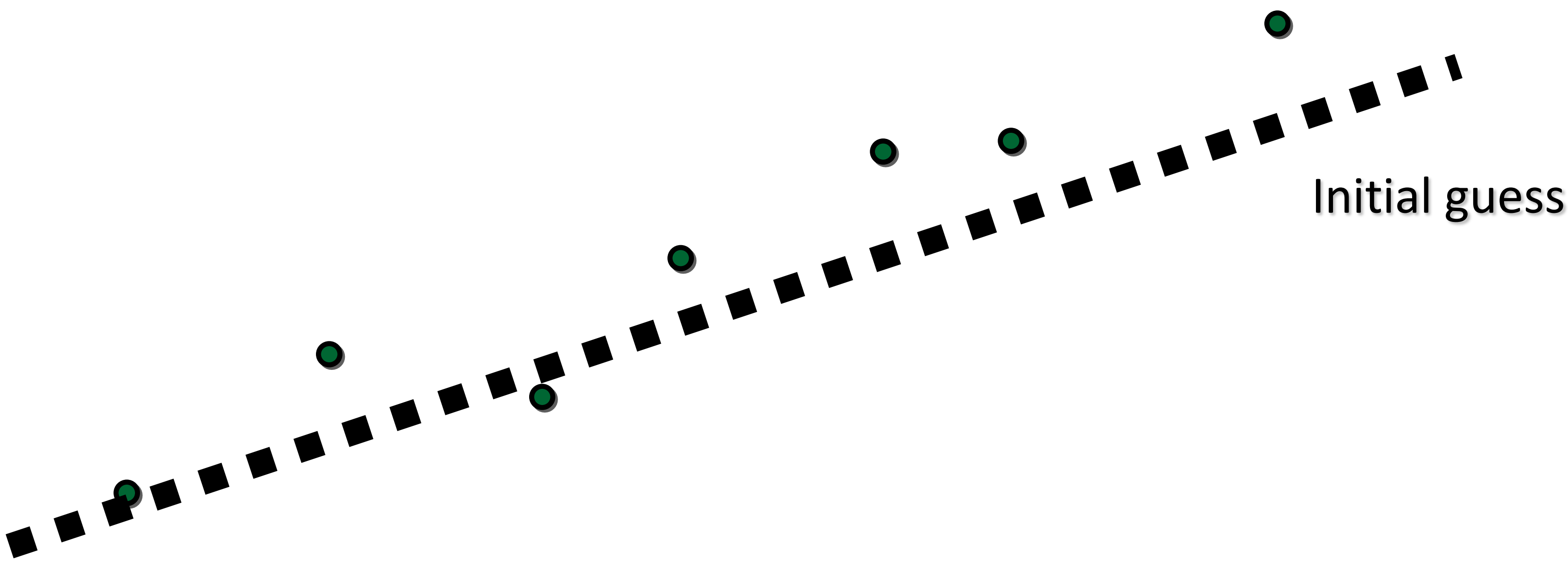
Hop-Along Algorithm

- Hop-Along algorithm for polyline fitting
  - Start with the first *k* edges from the list
  - Fit a line segment between the first and last edges
  - If the error is too large, shorten the sublist to the point of maximum error, return to step 2
  - If the line fit succeeds, compare the orientation of the current line segment with that of the previous line segment
    - If the lines have similar orientations, replace the two line segments with a single one
  - Make the current line segment the previous segment and advance the window of edges so that there are *k* edges in the sublist, return to step 2
- The algorithm considers only a "short run" of edges, thus it is more efficient

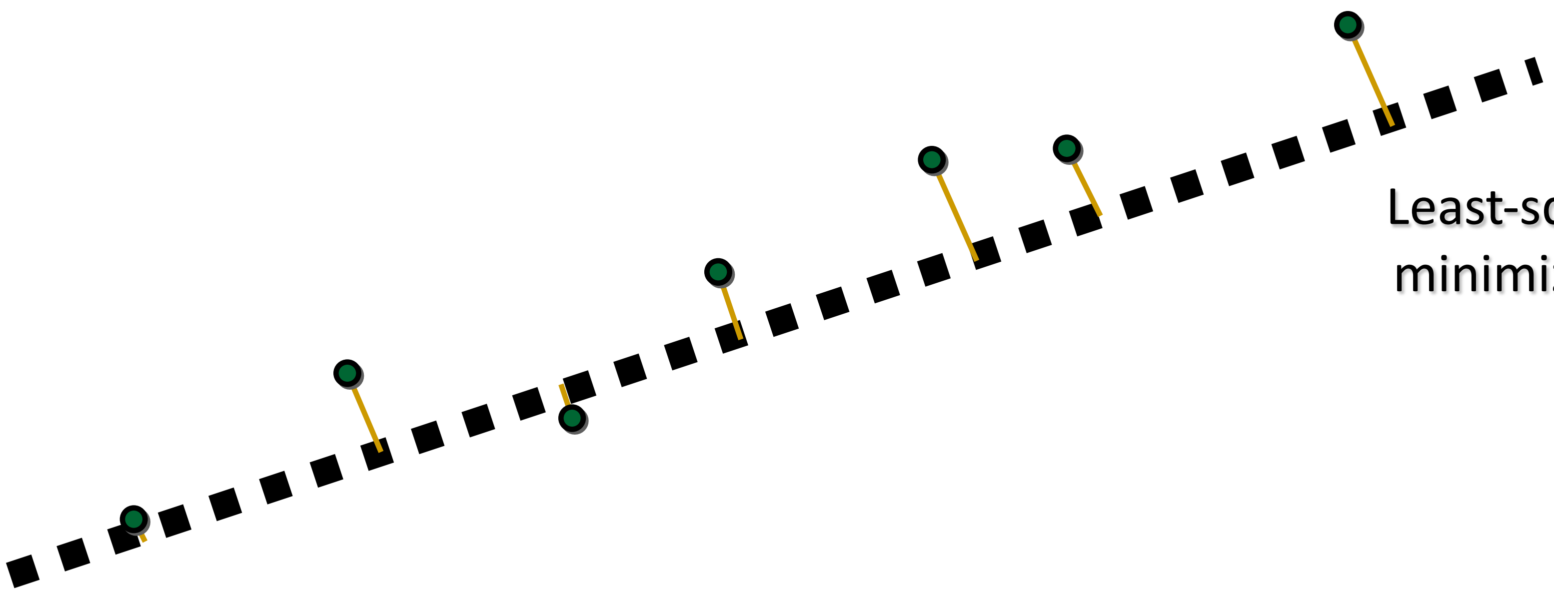
Line Fitting

- What is line fitting?
  - Used to find straight line features in an image
- Why use line fitting?
  - Output of "Hough transform" often not accurate enough
- How?
  - Follows edge extraction and linking
  - Use as an initial guess for fitting

Fitting Lines

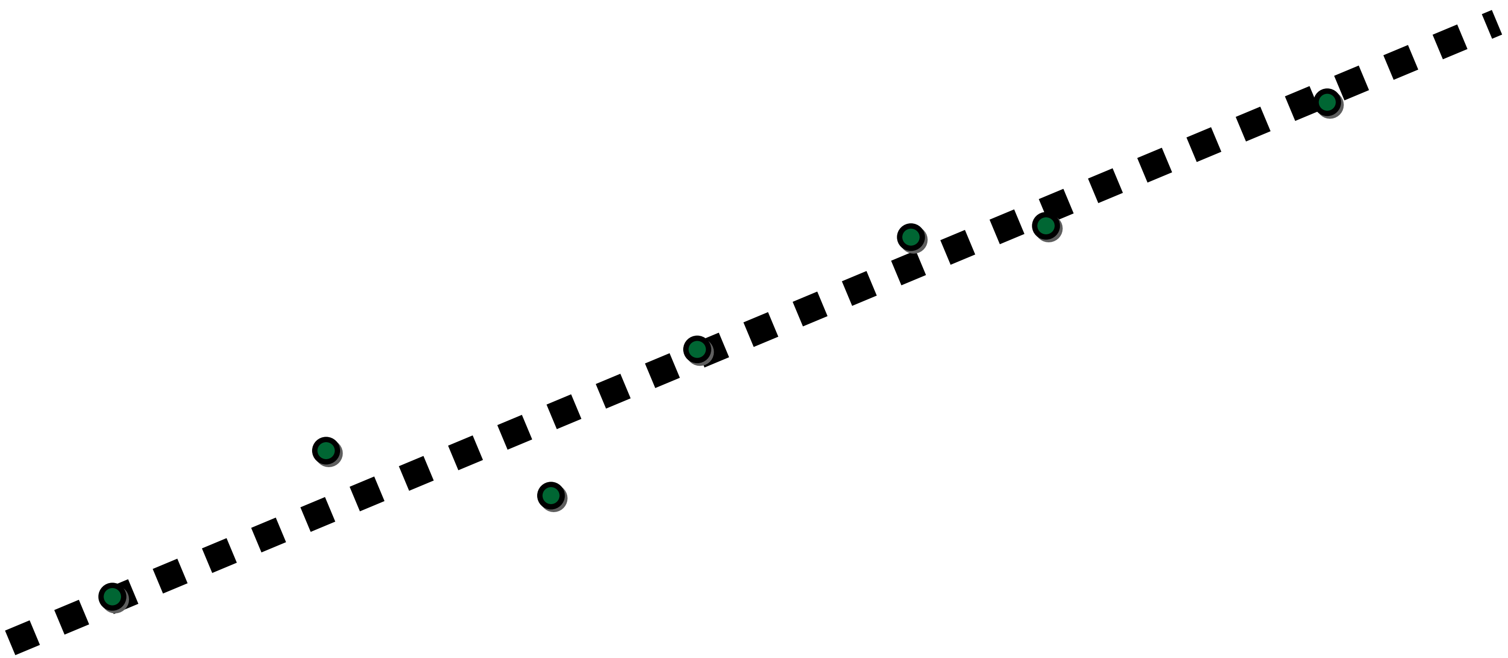


Fitting Lines



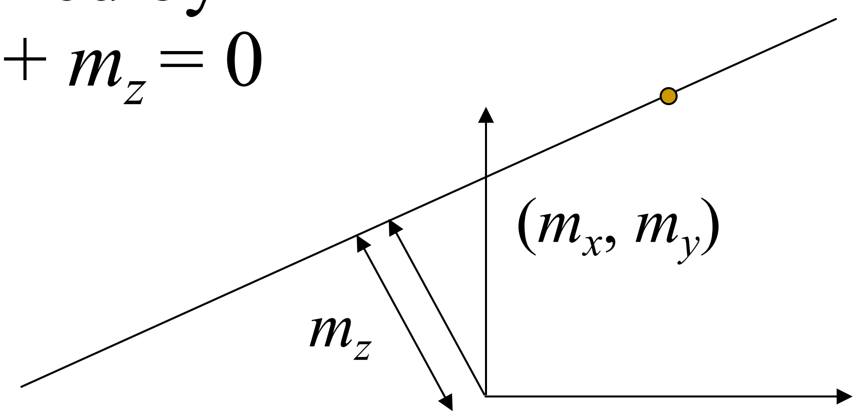
Least-squares minimization

Fitting Lines



Parameterizing Lines

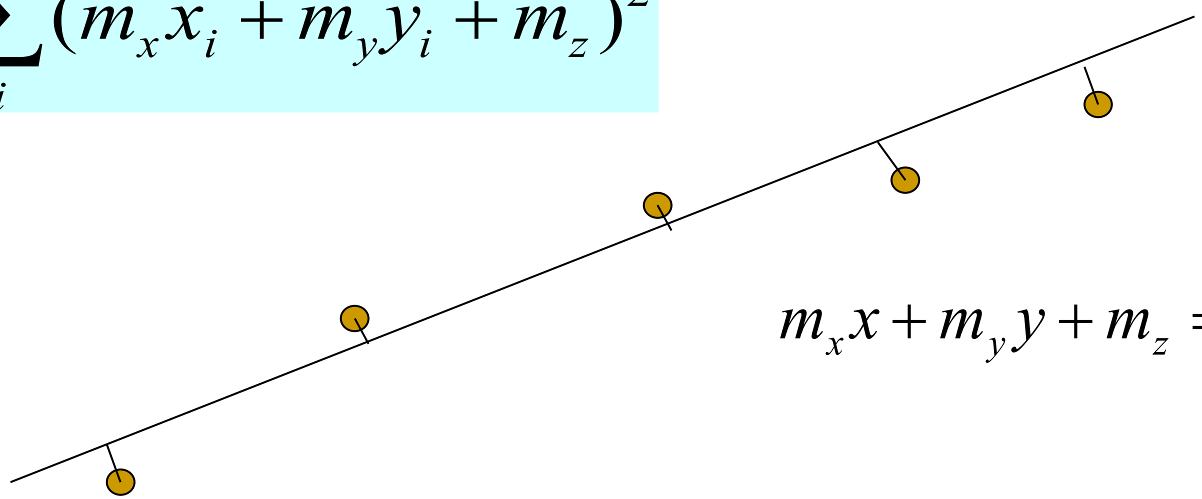
- Most popular line fitting algorithms minimize vertical point-to-line distance
- For our purposes lines in the image will be parameterized by a vector of 3 numbers (m_x, m_y, m_z) where: $m_x^2 + m_y^2 = 1$
  - Normal vector
- Points on the line are defined by the equation: $m_x x + m_y y + m_z = 0$
  - It may be a different coordinate system



Least Squares Fitting

- Given a set of points (x_i, y_i), the goal of the fitting procedure is to find the parameters (m_x, m_y, m_z) which represent the best line
- The best fit is defined in terms of the parameters which minimize the sum of the squared residuals

$$\min \sum_i (m_x x_i + m_y y_i + m_z)^2$$



$$m_x x + m_y y + m_z = 0$$

Least Squares Fitting

- Step I
  - Compute the centroid of the point set
  - Change coordinates such that the new centroid is (0,0)

$$\bar{x} = \frac{1}{n}\sum x_i, \bar{y} = \frac{1}{n}\sum y_i$$

$$x_i' = x_i - \bar{x}, y_i' = y_i - \bar{y}$$

- Step II
  - Solve for (m_x, m_y) by minimizing the following quadratic form

$$\mathrm{err}(m_x, m_y) = \sum_i (m_x x_i' + m_y y_i')^2$$

$$= \begin{pmatrix} m_x & m_y \end{pmatrix} \begin{pmatrix} \sum_i x_i'^2 & \sum_i x_i' y_i' \\ \sum_i x_i' y_i' & \sum_i y_i'^2 \end{pmatrix} \begin{pmatrix} m_x \\ m_y \end{pmatrix}$$

  - Solve for m_z

$$m_z = -(m_x \bar{x} + m_y \bar{y})$$

Issues

- The main problem with least squares fitting techniques is that they are heavily influenced by outliers

- Solutions to this problem include
  - Robust weighting measures
  - Iteratively re-weighted least squares
  - Least median squares

Fitting Ellipses

- An ellipse is a type of conic section defined by the equations

$$ax^2 + bxy + cy^2 + dx + ey + f = 0$$

$$b^2 - 4ac < 0$$

- One approach to fitting ellipses is to find a choice of parameters which minimizes the sum of squares of the residuals

$$e = \sum_i \left(ax_i^{\;2} + bx_i y_i + cy_i^{\;2} + dx_i + ey_i + f\right)^2$$

- More will be told after Active Contours (if time permitted)

Circular Arcs

- Line segments from approximation of edge lists can be replaced by circular arcs
  - Fitting circular arcs through the end points of two or more line segments, i.e. fitting the vertices of polyline
  - This gives piecewise constant curvature
- The implicit equation for a circle with radius r and center (x_0, y_0) is given by $(x-x_0)^2+(y-y_0)^2=r^2$
- The center (x_0, y_0) and radius r are uniquely determined by three points $\mathbf{p}_1 = (x_1, y_1)$, $\mathbf{p}_2 = (x_2, y_2)$, $\mathbf{p}_3 = (x_3, y_3)$
- The error in fitting is defined as the distance to the circular arc

Goodness of Circular Fitting

- When to use circular fitting instead of polyline?
- If the ratio of the length of the contour to the distance between the first and last end points is more than a threshold

- The circular arc is fit between the first and last end points and one other point

Methods for Circular Fitting

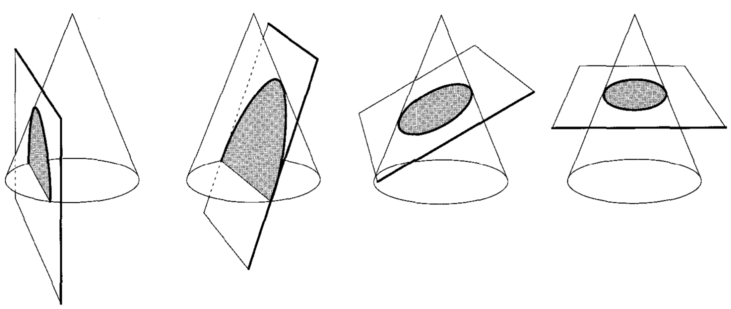
- Methods for circular fitting depend on "how the middle point is chosen"
  - Use the polyline vertex that is farthest from the line joining the first and last end points
  - Use the edge point that is farthest from the line joining the first and last end points
  - Use the polyline vertex that is in the middle of the sequence of vertices between the first and last end points
  - Use the edge point that is in the middle of the list of edges between the first and last end points

Criteria for Circular Fitting Result

- The maximum absolute error (MAE) between edge points and circular arcs is below a threshold
- The number of sign changes for the errors is large

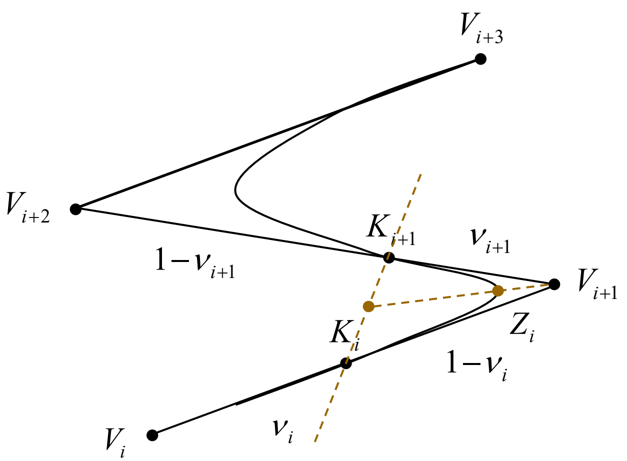
Conic Sections

- Line segments can also be replaced by conic sections
  - The implicit form: $f(x,y) = ax^2 + 2hxy + by^2 + 2ex + 2gy + c = 0$
  - Defined geometrically by intersecting a cone with a plane
  - Three different types: hyperbola, parabola, ellipse
- Conics can be fit between three vertices in the polyline
  - Knot: the locations where conics are joined
  - Conic spline: a sequence of conics that are joined end to end, with "equal tangents" at the knots



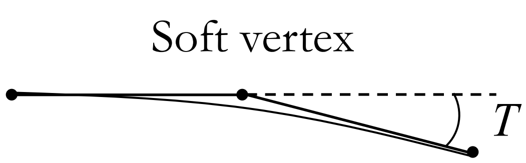
Conic Approximation

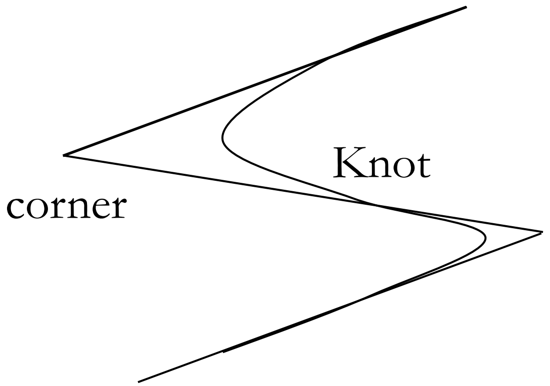
- Each conic section in a conic spline is defined by 2 end points (V_i, V_{i+2}), 2 tangents, 1 additional point
  - Knots K_i, K_{i+1} are defined as $K_i = (1-\nu_i)\, V_i + \nu_i\, V_{i+1}$, where $0 \le \nu_i \le 1$
  - The additional point Z_i is defined as $Z_i = \gamma_i\, V_{i+1} + (1-\gamma_i)(K_i+K_{i+1})/2$
- Special cases:
  - If $\nu_{i+1} = 0$, then the segment $(K_i, K_{i+1}) = (K_i, V_{i+1})$
  - If $\nu_i = 1$ and $\nu_{i+1} = 0$, then $K_i = K_{i+1} = V_{i+1}$ $\Rightarrow$ There is a corner!



Conic Fitting Algorithm

- Starting with a polyline and classified the vertices as corners, soft vertices, or knots
  - Soft vertices:
    - Have angles near 180° (i.e., almost a straight line!)
    - The adjacent line segments may be replaced with a conic section
  - Corners:
    - Have vertex angles above $180^\circ + T$ or below $180^\circ - T$
    - Adjacent line segments will not be replaced with a conic
  - Knots:
    - On a line segment and determined by soft vertices at both ends
    - Two conic sections must be joined at the knot





Spline Curves

- Spline
  - A function represented using piecewise polynomials
  - Also made from any class of functions joined end to end
  - Examples: line segments, circular arcs, conic sections
  - The most common form: cubic spline
  - Uses both <u>positions</u> and <u>orientations</u> of (edge) points
- Applications of spline
  - Used to fit data points in data analysis
  - Used to represent free-form curve in CG and CAD
  - Used for curve representation in CV if no simpler model is adequate

Geometric & Parametric Equivalences

- Geometric equivalence
  - Two curves are geometrically equivalent if they trace the same set of points (or correspond to the same shape)
- Parametric equivalence
  - Two curves are parametrically equivalent if their equations are identical (same parametric representation)
- Parametric equivalence is *stronger* than geometric equivalence
  - Two curves can be geometrically equivalent but have different parametric representations! (noise, etc.)
  - Very similar curve shapes do not imply the same (or very close parametric representation)
  - Geometric equivalence must be used for object-model comparison or recognition

Curve Approximation

- Curve fitting interpolates the curve through a subset of the edges
  - Polyline, circular arcs, conic sections, spline curves
- Curve approximation does not force curve to pass through particular edges and gives higher accuracy
  - Least-squares regression
    - When the data points (edge points) are very reliable
  - Robust regression
    - When the data points could contain some grouping error
  - Cluster analysis techniques: Hough transform
    - When the grouping of edges is very unreliable (scattered edge points)

5/20/2023

Curve Fitting

- The curve fitting problem is a regression problem with the curve modeled by the equation with p parameters:

$$f(x, y; a_1, a_2, \cdots, a_p) = 0$$

- The problem is to fit the curve model to a set of edge points $\{(x_1,y_1), (x_2,y_2), \ldots, (x_n,y_n)\}$
- p unknown curve parameters can be solved by p equations with p data points for noise-free case
- In real applications, more data points and equations will be used to solve the overdetermined system

- Least-squares regression: errors are normally distributed
- Robust regression: data points contain some outliers

Detecting Lines

- What is the difference between line detection and edge detection?
  - Edges = local
  - Lines = non-local
- Line detection usually performed on the output of an edge detector

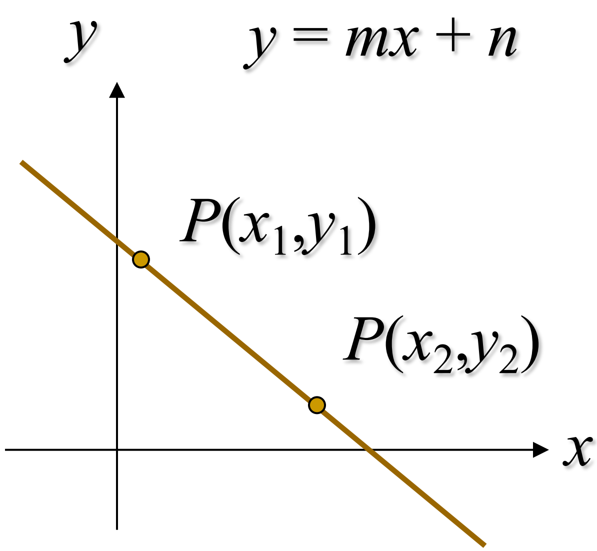
Detecting Lines

- Several different approaches:
  - For each possible line, check whether the line is present: "brute force"
  - Given detected edges, record lines to which they might belong: "Hough transform + voting"
  - Given guess for approximate location of a line, refine that guess: "fitting"
- Second method (Hough transform) is efficient for finding unknown lines, but not always accurate

Line Detection

- Mathematical model of a line:

$$y = mx + n$$

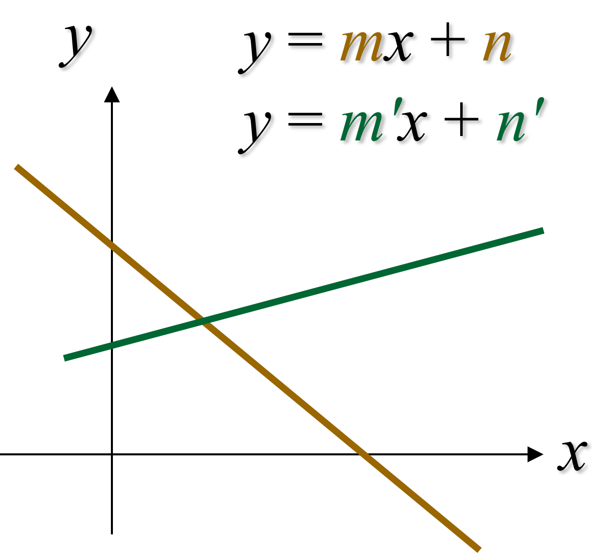


$$y_1 = mx_1 + n$$

$$y_2 = mx_2 + n$$

$$y_N = mx_N + n$$

Image and Parameter Spaces



$$y_1 = mx_1 + n$$

$$y_2 = mx_2 + n$$

$$y_N = mx_N + n$$

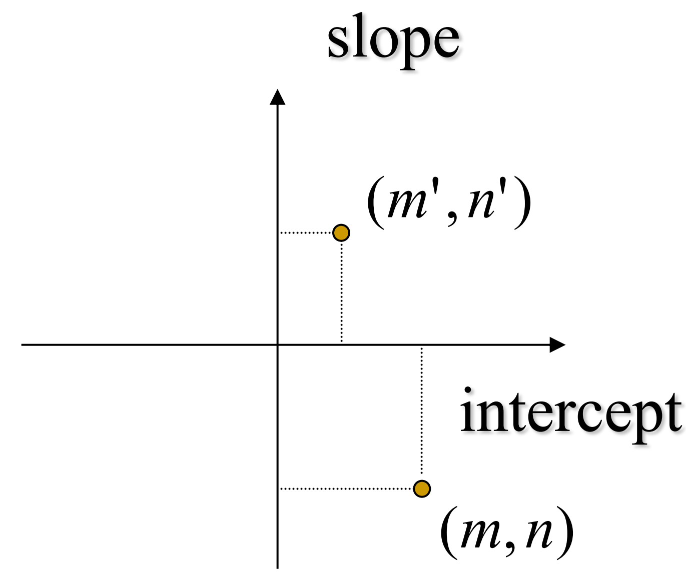


Image Space

Parameter Space

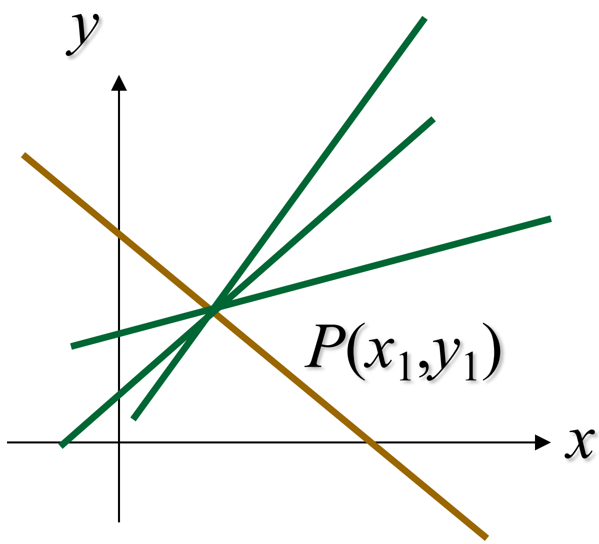
Line in Image Space ~ Point in Parameter Space

Looking at it Backwards …

Image space

Fix (m,n), vary (x,y) - Line $\qquad y = mx + n$

Fix (x_1,y_1), vary (m,n) – Lines thru a Point $\qquad y_1 = mx_1 + n$



Looking at it Backwards …

Parameter space

$y_1 = mx_1 + n$ Can be re-written as: $n = -x_1 m + y_1$

Fix $(-x_1, y_1)$, vary (m,n) - Line

$n = -x_1 m + y_1$

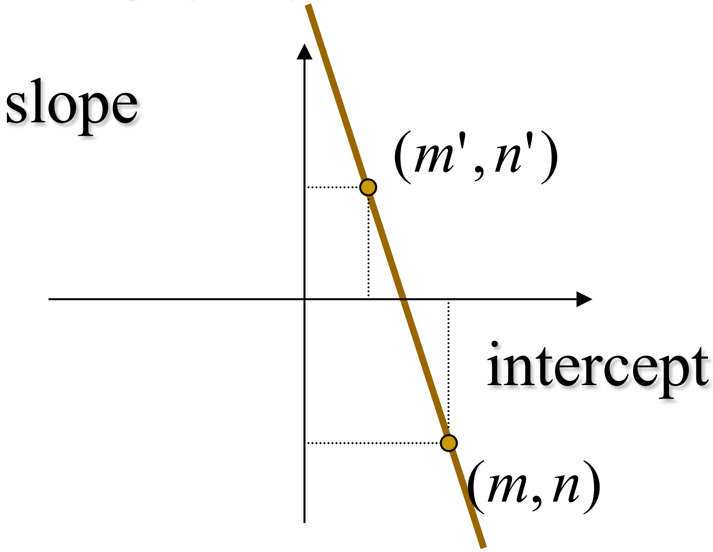


Image & Parameter Spaces

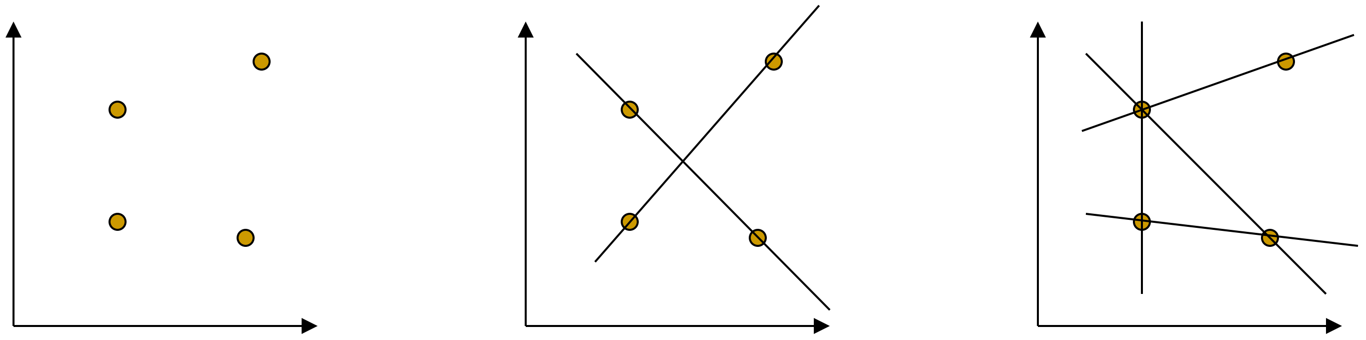
- Image Space
  - Lines
  - Points
  - Collinear points
- Parameter Space
  - Points
  - Lines
  - Intersecting lines

Hough Transform

- General idea: transform from image coordinates to parameter space of features
  - Map a difficult pattern problem into a simple peak detection problem
  - Need parameterized model of features
  - For each pixel, determine all parameter values that might have given rise to that pixel; vote
  - At end, look for peaks in parameter space
- This approach is a voting scheme based on accumulating evidence in a parameter space

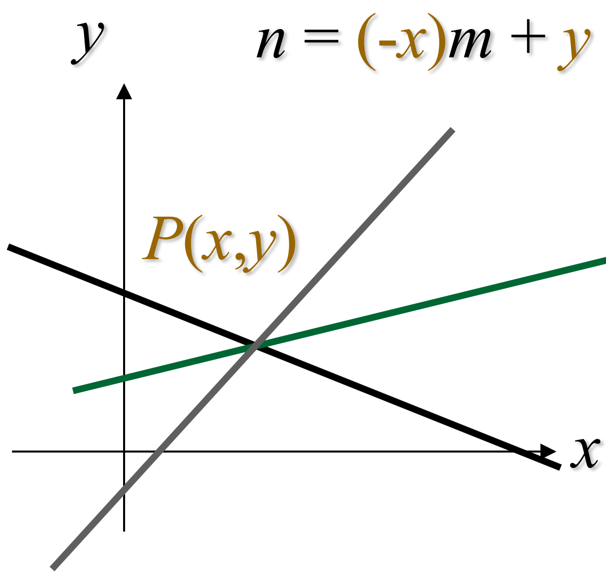
Hough Transform for Lines

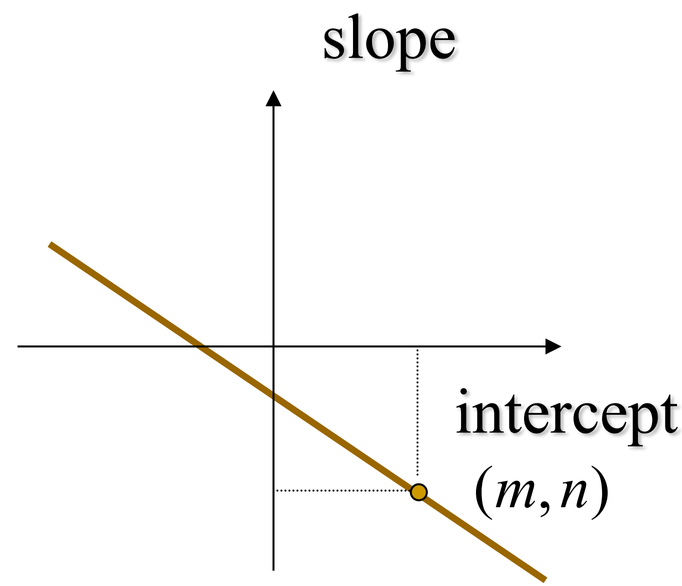
- Each input measurement indicates its contribution to a globally consistent solution
- Here this problem is under constrained
  - Generic line: $y = ax+b$
  - Parameters: a and b



Hough Transform Technique

- Given an edge point, there is an infinite number of lines passing through it (vary *m* and *n*)
  - These lines can be represented as a line in parameter space





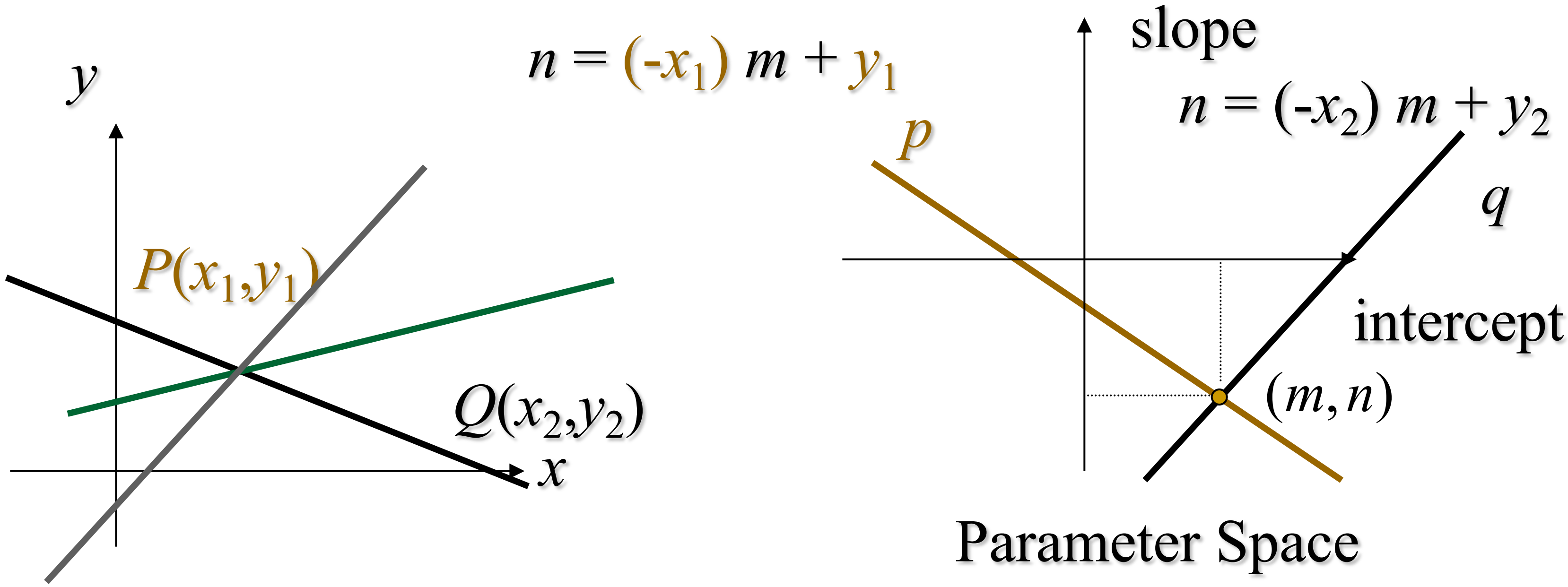
Parameter Space

5/25/2023

- No Class Next Tuesday.

Hough Transform Technique

- Given a set of collinear edge points, each of them have associated a line in parameter spaces
  - These lines intersect at the point (m,n) corresponding to the parameters of the line in the image space



Hough Transform Technique

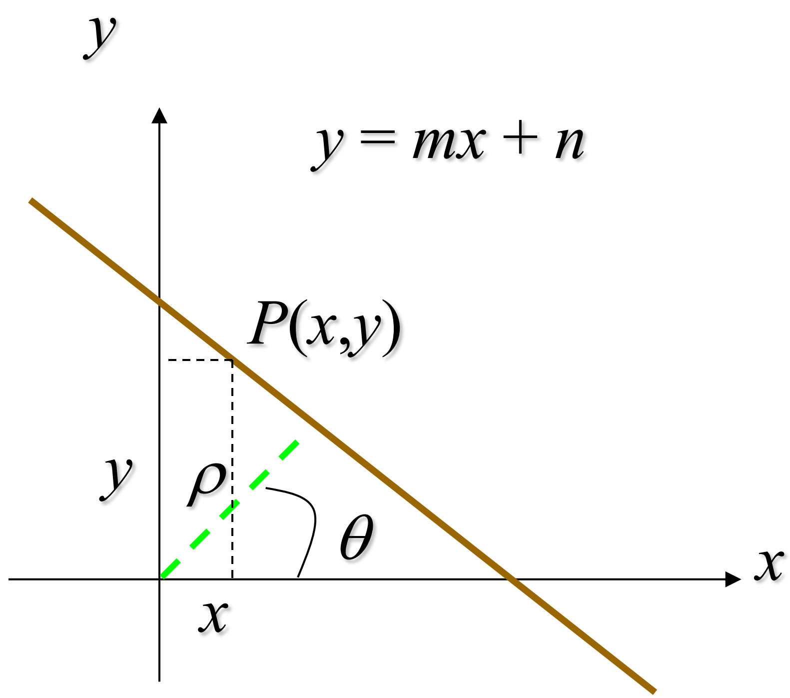
- At each point of the (discrete) parameter space, count how many lines pass through it
  - Use an array of counters
  - Can be thought as a "parameter image"
- The higher the count, the more edges are collinear in the image space
  - Find a peak in the counter array
  - This is a "bright" point in the parameter image
  - It can be found by thresholding

Practical Issues

- The slope of the line is $-\infty < m < \infty$
  - The parameter space is infinite
- The representation $y = mx + n$ does not express lines of the form $x = k$

Solution

- Use the "normal" equation of a line:



$$\rho = x\cos\theta + y\sin\theta$$

θ is the line orientation

ρ is the distance between the origin and the line

New Parameter Space

- Use the parameter space (ρ, θ)
- The new space is finite
  - $0 < \rho < D$, where D is the image diagonal
  - $0 < \theta < 2\pi$
- The new space can represent all lines
  - $y = k$ is represented with $\rho = k,\ \theta = 90^\circ$
  - $x = k$ is represented with $\rho = k,\ \theta = 0^\circ$
- A Point in Image Space is now represented as a sinusoid
  - $\rho = x\cos\theta + y\sin\theta$

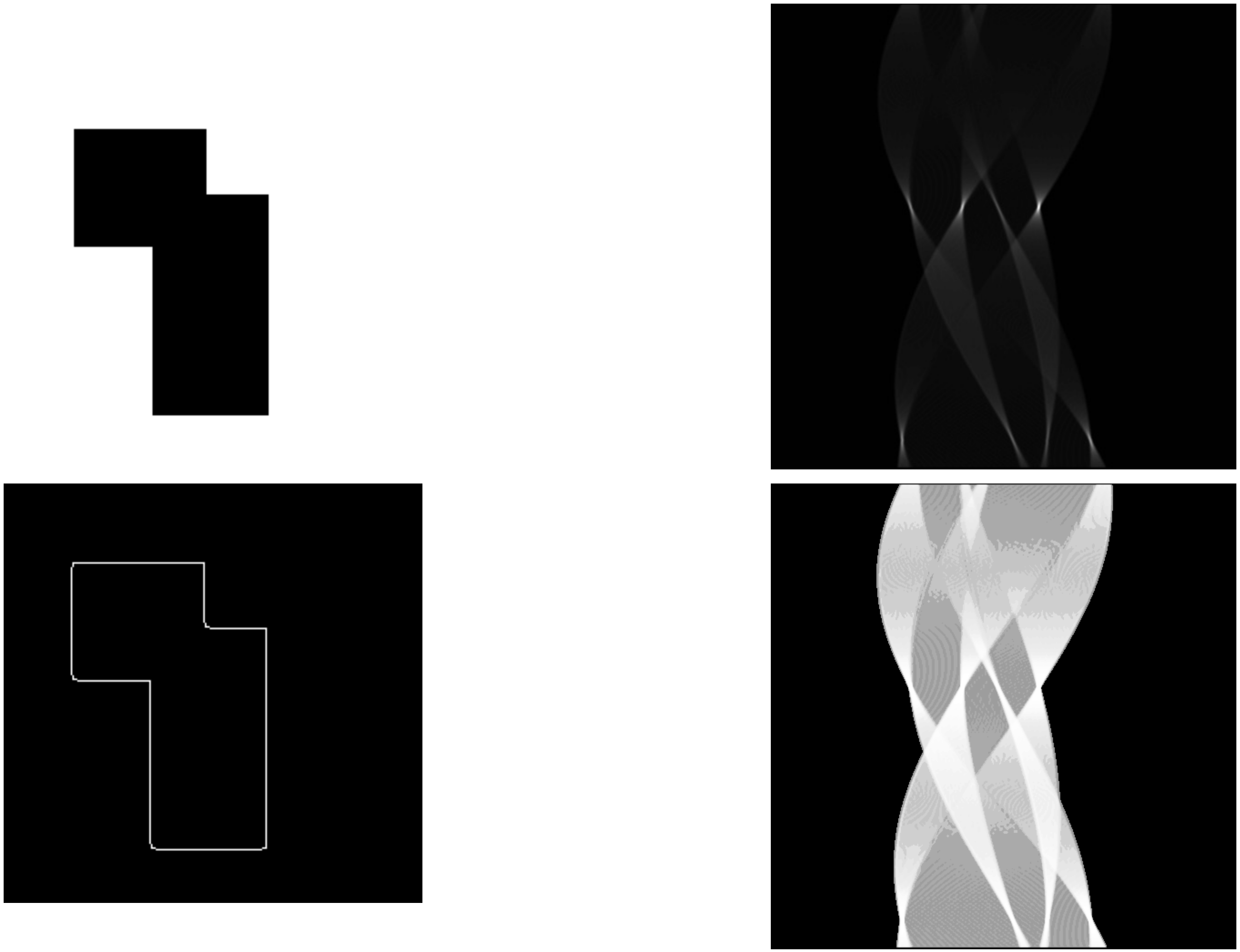
Hough Transform Algorithm

- Input is an edge image ($E(i, j) = 1$ for edgels)
  - Discretize θ and ρ in increments of θ_d and ρ_d
  - Let A(R,T) be an array of integer accumulators, initialized to 0
  - For each pixel $E(i, j) = 1$ and $h = 1, 2, \ldots$ T do
    - $\rho = i \cos(h\ \theta_d) + j \sin(h\ \theta_d)$
    - Find closest integer k of the element of ρ_d, corresponding to ρ
    - Increment counter A(h,k) by one
  - Find all local maxima in A(R,T) > threshold
- Output is a set of pairs (ρ_d, θ_d) describing the lines detected in E in polar form

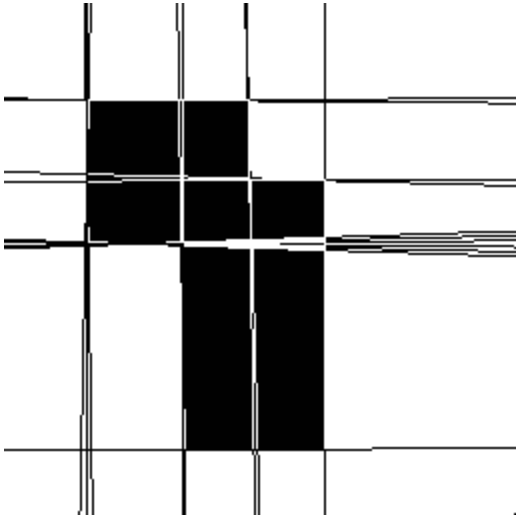
Hough Transform Speed Up

- If we know the orientation of the edge - usually available from the edge detection step
  - We fix θ in the parameter space and increment only one counter!
  - We can allow for orientation uncertainty by incrementing a few counters around the "nominal" counter

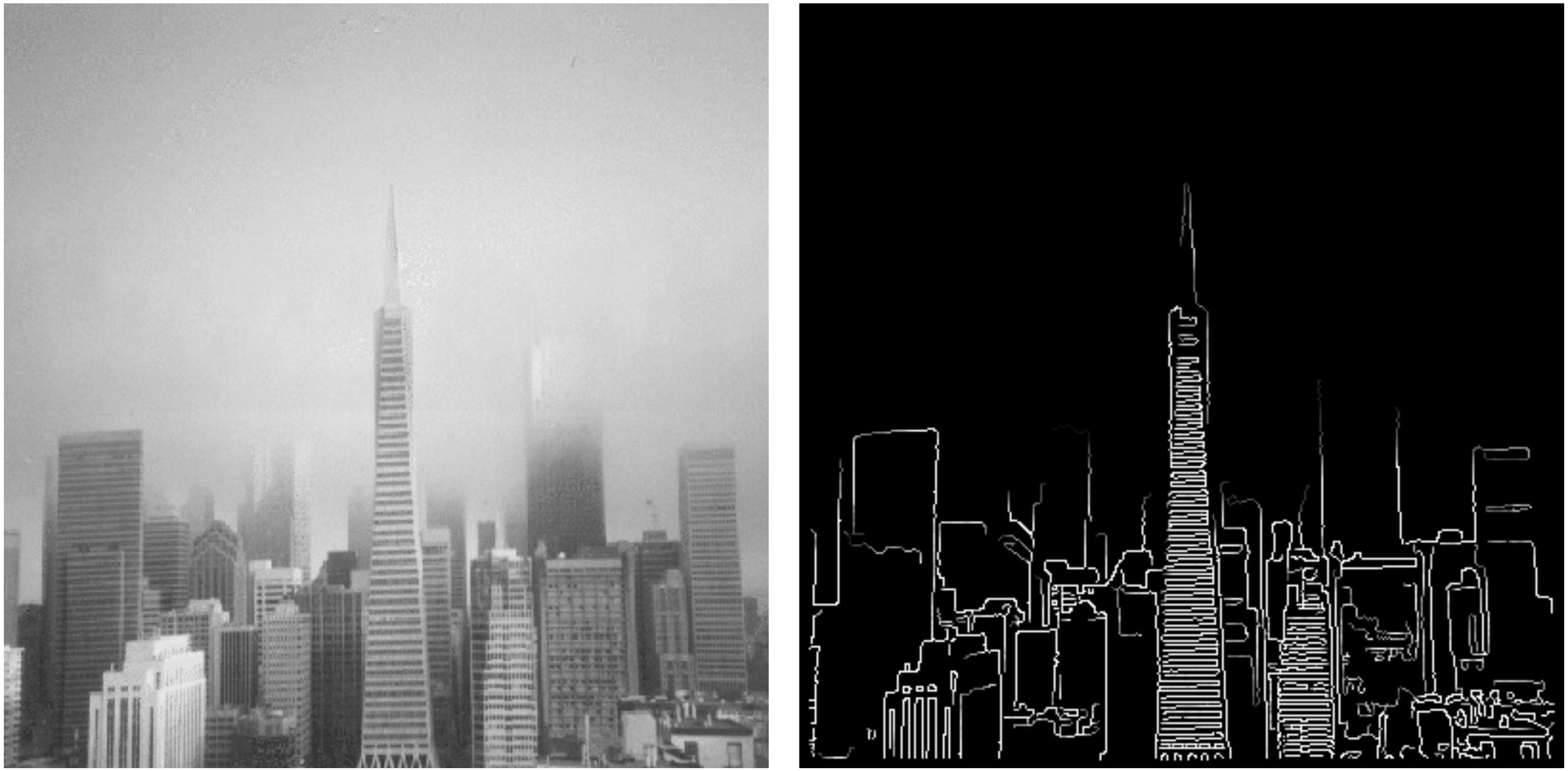
Example



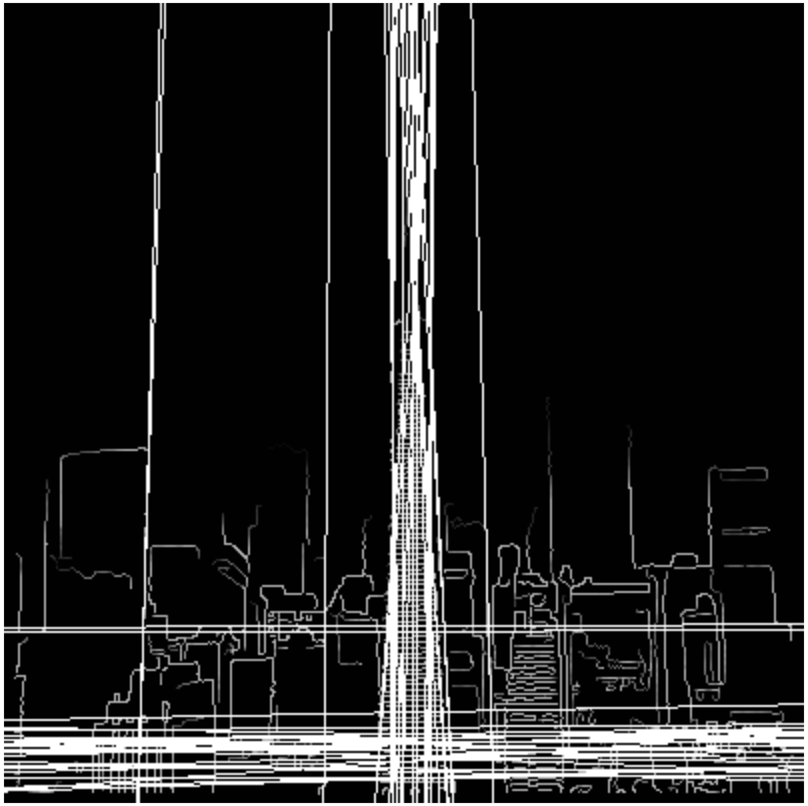
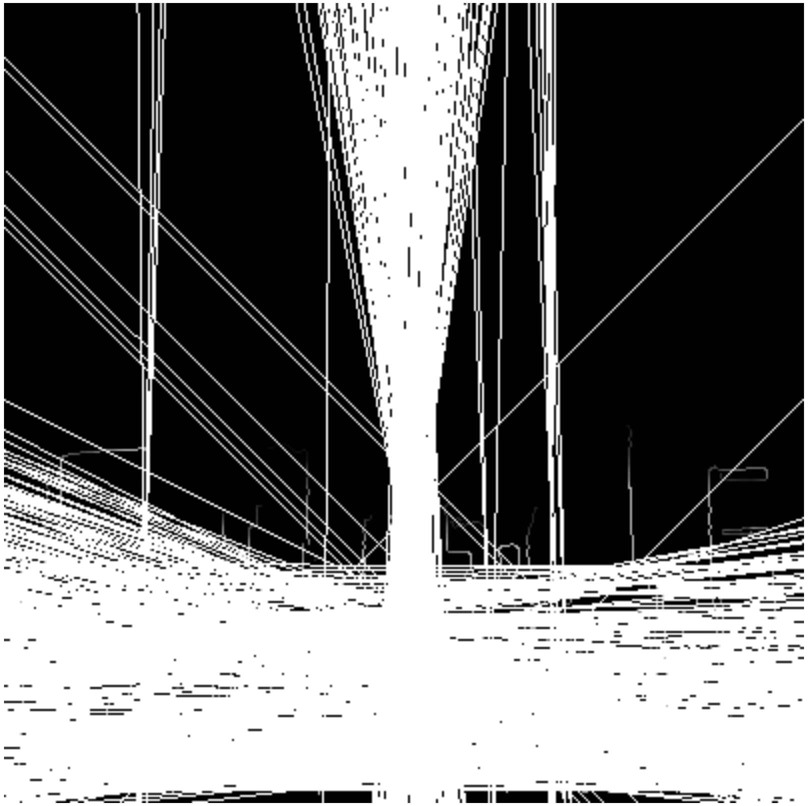
Example



Real Example



Real Example

Demo

- https://www.aber.ac.uk/~dcswww/Dept/Teaching/CourseNotes/current/CS34110/hough.html
- https://gmarty.github.io/hough-transform-js/
- http://liquiddandruff.github.io/hough-transform-visualizer/

Reading

- Chapter 6 of Jain's book