How to Fit A Line?

\[ y = mx + c \]
How to Fit A Line?

- Least squares Fit (over constraint)
- RANSAC (constraint)
- Hough Transform (under constraint)

Least Squares Fit

- Standard linear solution to estimating unknowns.
  - If we know which points belong to which line
  - Or if there is only one line

\[ y = mx + c = f(x, m, c) \]

Minimize \[ E = \sum_i [y_i - f(x_i, m, c)]^2 \]

Take derivative wrt \( m \) and \( c \) set to 0
**Line Fitting**

\[ y = mx + c \]

\[ y_1 = mx_1 + c \]
\[ y_2 = mx_2 + c \]
\[ \vdots \]
\[ y_n = mx_n + c \]

\[
\begin{bmatrix}
  y_1 \\
  y_2 \\
  \vdots \\
  y_n
\end{bmatrix}
= 
\begin{bmatrix}
  x_1 & 1 \\
  x_2 & 1 \\
  \vdots & 1 \\
  x_n & 1
\end{bmatrix}
\begin{bmatrix}
  m \\
  c
\end{bmatrix}
\Rightarrow B = AD
\]

\[
A^T B = A^T A D
\]
\[
(A^T A)^{-1} A^T B = (A^T A)^{-1} (A^T A) D
\]
\[
D = (A^T A)^{-1} A^T B
\]

Alper Yilmaz, Mubarak Shah, Fall 2011
UCF

---

**RANSAC: Random Sampling and Consensus**

1. Randomly select the minimum number of constraints to estimate the solution.

2. Find the error between the estimated solution and all data points. If the error is less than the tolerance, then quit, else go to (1).

   - Randomly select two points to fit a line
   - Find the error between the estimated solution and all other points. If the error is less than tolerance, then quit, else go to step (1).

Alper Yilmaz, Mubarak Shah, Fall 2011
UCF
Line Fitting: Segmentation

- Several Lines
- How do we Know which points belong to which lines?

Line Fitting: Hough Transform

- Line equation
  \[ y = mx + c \]
  \( m \) is slope, \( c \) is \( y \)-intercept

- Rewrite this equation
  \[ c = (-x)m + y \]

- For particular edge point \( i \) this becomes
  \[ c = (-x_i)m + y_i \]

- This is an equation of a line in \((c,m)\) space.
Line Fitting: Hough Transform

\[ c = (-x_i)m + y_i \]

Hough Transform Algorithm for Fitting Straight Lines

1. Quantize the parameter space \( P[c_{\text{min}}, \ldots, c_{\text{max}}, m_{\text{min}}, \ldots, m_{\text{max}}] \).
2. For each edge point \((x, y)\) do
   for \( m = m_{\text{min}}, \ldots, m_{\text{max}} \) do
   \( c = (-x)m + y \),
   \( P[c, m] = P[c, m] + 1 \).
3. Find the local maxima in the parameter space.

Figure 4.2: Hough transform algorithm for fitting straight lines.
Polar Form of Equation of Line

\[ c_i = (-x)m_j + y \]

Problematic for vertical lines, \( m \) and \( c \) grow to infinity

\[ p = x \cos \theta + y \sin \theta \]

Use \( \theta \) from gradient

Hough Transform for Polar Form of Equation of Line

1. Quantize the parameter space \( P[\theta_{min}, \ldots, \theta_{max}, \rho_{min}, \ldots, \rho_{max}] \).
2. For each edge point \((x, y)\) do
   \[ p = x \cos \theta + y \sin \theta, \]
   \[ P[\theta, \rho] = P[\theta, \rho] + 1. \]
3. Find the local maxima in the parameter space.

Figure 4.1: Hough transform algorithm using polar form of equation of straight line.
Line Fitting Examples

- Ideal
- Noisy
- Very noisy

Noise Factor

This is the number of votes that the real line of 20 points gets with increasing noise.
Noise Factor

as the noise increases in a picture without a line, the number of points in the max cell goes up, too

Difficulties

- What is the increments for $\theta$ and $p$.
  - too big? We cannot distinguish between different lines
  - too small? noise causes lines to be missed
Circle Fitting

- Similar to line fitting
  - Three unknowns
    \[(x - x_0)^2 + (y - y_0)^2 - r^2 = 0\]
- Construct a 3D accumulator array \(A\)
  - Dimensions: \(x_0, y_0, r\)
- Fix one of the parameters loop for the others
- Increment corresponding entry in \(A\).
- Find the local maxima in \(A\).

More Practical Circle Fitting

- Use the tangent direction \(\theta\) at the edge point
  - \(x_0, y_0\) given \(x, y, r\)
    \[x_0 = x - r \cos \theta\]
    \[y_0 = y - r \sin \theta\]
1. Quantize the parameter space
   \[ P[x_{\min}, \ldots, x_{\max}, y_{\min}, \ldots, y_{\max}, r_{\min}, \ldots, r_{\max}] \].

2. For each edge point \((x, y)\) do
   For \(r = r_{\min}, r \leq r_{\max}, r + \epsilon\)
   \[ x_0 = x - r \cos \theta \]
   \[ y_0 = y - r \sin \theta \]
   \[ P[x_0, y_0, r] = P[x_0, y_0, r] + 1 \]

3. Find the local maxima in the parameter space.

Figure 1.6: Rough transform algorithm for fitting circle using polar form of equation of a circle.

Examples
Generalized Hough Transform

- Used for shapes with **no** analytical expression
- Requires training
  - Object of known shape
  - Generate model
    - R-table
- Similar approach to line and circle fitting during detection

Generating R-table

- Compute centroid
- For each edge compute its distance to centroid
  \[
  r = \begin{bmatrix} x' \\ y' \end{bmatrix} = \begin{bmatrix} x - x_0 \\ y - y_0 \end{bmatrix}
  \]
- Find edge orientation (gradient angle)
- Construct a table of angles and \( r \) values
Generating R-table

| Φ₁ | r₁, r₂, r₃ ...
| Φ₂ | r₁₄, r₂₁, r₂₃ ...
| Φ₃ | r₄₁, r₄₂, r₃₃ ...
| Φ₄ | r₁₀, r₁₂, r₁₃ ...

Detecting shape

- known
  - Edge points (x, y)
  - Gradient angle at every edge point θ
  - R-table of shape need to be determined
- For each edge point find θ store it in corresponding row of R-table
- Create an accumulator array of 2D (x, y)
  - Increment columns of Φ
1. Quantize the parameter space \( P_{\alpha \min \ldots \alpha \max, \beta \min \ldots \beta \max} \).

2. For each edge point \((x, y)\) do
   compute \(\phi(x, y)\)
   for each table entry for \(\phi\) do
   \[
   \begin{align*}
   x' &= x + x' \\
   y' &= y + y'
   \end{align*}
   \]
   \(P_{\alpha, \beta} - P_{\alpha, \beta} + 1\).

3. Find the local maxima in the parameter space.

Figure 4.8: Generalized Hough transform algorithm.

Rotation and Scale Invariance

- Rotation around Z-axis
  \[
  \begin{align*}
  x' &= x \cos \alpha - y \sin \alpha \\
  y' &= x \sin \alpha + y \cos \alpha
  \end{align*}
  \]

- Scaling
  \[
  \begin{align*}
  x' &= sx \\
  y' &= sy
  \end{align*}
  \]

- Rotation+scaling
  \[
  \begin{align*}
  x' &= s(x \cos \alpha - y \sin \alpha) \\
  y' &= s(x \sin \alpha + y \cos \alpha)
  \end{align*}
  \]
Rotation and Scale Invariance

- Replace equations 4.13 and 4.14 in Algorithm 4.8 by and loop for scale and rotation angles.

\[
\begin{align*}
x_c &= x + s_x(x' \cos \theta + y' \sin \theta) \\
y_c &= y + s_y(-x' \sin \theta + y' \cos \theta)
\end{align*}
\]