Recap
Pyramids
Recap

Gaussian Pyramids

- Smooth rows 1, 3, 5, ... N by 1D Gaussian
  - Select filtered rows
- Smooth columns 1, 3, 5, ... N by 1D Gaussian
  - Select filtered columns
- Create image ¼th of original image size

Recap

Laplacian Pyramid

- Synthesis (Coding)
  - Compute Gaussian pyramid
  - Compute Laplacian pyramid

- Analysis (Decoding)
  - Compute Gaussian pyramid from Laplacian pyramid
  - \( g_1 \) is reconstructed image

\[
\begin{align*}
L_4 &= g_4 - \text{EXPAND}[g_4] \\
L_3 &= g_3 - \text{EXPAND}[g_3] \\
L_2 &= g_2 - \text{EXPAND}[g_2] \\
L_1 &= g_1 - \text{EXPAND}[g_1] \\
g_4 &= L_4 \\
g_3 &= \text{EXPAND}[g_4] + L_3 \\
g_2 &= \text{EXPAND}[g_3] + L_2 \\
g_1 &= \text{EXPAND} [g_2] + L_1
\end{align*}
\]
Constructing Laplacian Pyramid

- Compute Gaussian pyramid
  \( g_k, g_{k-1}, g_{k-2}, \ldots, g_2, g_1 \)

- Compute Laplacian pyramid as follows:
  
  \[
  \begin{align*}
  L_k &= g_k - \text{EXPAND}(g_{k-1}) \\
  L_{k-1} &= g_{k-1} - \text{EXPAND}(g_{k-2}) \\
  L_{k-2} &= g_{k-2} - \text{EXPAND}(g_{k-3}) \\
  &\vdots \\
  L_1 &= g_1
  \end{align*}
  \]

Hough Transform
Line Fitting

- Line equation
  \[ y = mx + b \quad m \text{ is slope, } b \text{ is } y \text{- intercept} \]

- Using edge pixels
  - Compute \( b \) for every \( m \)
  \[ b_i = y - m_j x \]

- Problematic for vertical lines
  - \( m \) and \( b \) grow to infinity
**Line Fitting**

- Polar coordinate representation
  - For each point on line $\theta$ and $\rho$ are constant
  - Numerically stable for lines in any orientation

$$x \cos \theta + y \sin \theta = \rho \quad (A)$$

- Different choices of $\theta$ for constant $\rho$ gives different choices of lines

**Algorithm**

- Construct accumulator array in 2D $(\theta, \rho)$
  - Initial values 0
- Select granularity of angle $\theta$
  - For instance $10^\circ$ increments
- For every edge point
  - Compute $\rho$ using (A)
  - Increment accumulator array by one for each computed $(\theta, \rho)$ pair.
Line Fitting
**Line Fitting Examples**

### Noisy vs. Ideal

- **Ideal**: No noise, points lie exactly on the line.
- **Noisy**: Some noise, points deviate slightly from the line.
- **Very Noisy**: High noise, points deviate significantly from the line.

---

**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

![Graph showing the relationship between number of noise points and maximum number of votes.]

**Difficulties**

- What is the increments for $\theta$ and $\rho$.  
  - too big? We cannot distinguish between different lines  
  - too small? noise causes lines to be missed
- How many lines
- Which edge point belongs to which line
- Hardly ever satisfactory due to noise.
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

\[ y = ax + b = f(x, a, b) \]

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

Take derivative wrt \( a \) and \( b \) set to 0

---

Line Fitting

\[ y = ax + b \]

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

\[
A^T B = A^T AC
\]

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

\[
C = (A^T A)^{-1} A^T B
\]
Programming assignment

- Implement line fitting algorithm
- Due date October 24
- You will be given an image of the following form

Programming Assignment Reminder

- A week from today
- Submit hardcopy of the code.
- Submit program on a CD.
- Do NOT email your projects!!