Edge Detection in Images

- Finding the contour of objects in a scene
Edge Detection in Images

- What is an object?
  It is one of the goals of computer vision to identify objects in scenes.

Edges have different sources.
What is an Edge
- Lets define an edge to be a discontinuity in image intensity function.
- Edge Models
  - Step Edge
  - Ramp Edge
  - Roof Edge
  - Spike Edge

Detecting Discontinuities
- Discontinuities in signal can be detected by computing the derivative of the signal.
Differentiation and convolution

- Recall
  \[
  \frac{\partial f}{\partial x} = \lim_{\varepsilon \to 0} \left( \frac{f(x + \varepsilon) - f(x)}{\varepsilon} \right)
  \]

- Now this is linear and shift invariant, so must be the result of a convolution.

- We could approximate this as
  \[
  \frac{\partial f}{\partial x} = \frac{f(x_{n+1}) - f(x)}{\Delta x}
  \]
  (which is obviously a convolution with Kernel [1 -1]; it’s not a very good way to do things, as we shall see)

Finite Difference in 2D

\[
\frac{\partial f(x,y)}{\partial x} = \lim_{\varepsilon \to 0} \left( \frac{f(x + \varepsilon, y) - f(x, y)}{\varepsilon} \right)
\]

\[
\frac{\partial f(x,y)}{\partial y} = \lim_{\varepsilon \to 0} \left( \frac{f(x, y + \varepsilon) - f(x, y)}{\varepsilon} \right)
\]

**Definition**

\[
\frac{\partial f(x,y)}{\partial x} = \frac{f(x_{n+1}, y_m) - f(x_n, y_m)}{\Delta x}
\]

\[
\frac{\partial f(x,y)}{\partial y} = \frac{f(x_n, y_{m+1}) - f(x_n, y_m)}{\Delta x}
\]

**Discrete Approximation**

**Convolution Kernels**
Finite differences

\[ I_x = I * [1 \ -1] \]

\[ I_y = I * [1 \ -1] \]

Frequency Response of Differential Kernel

\[ f(x) \xrightarrow{\text{Fourier Transform}} F(f)(u) \]

\[ \frac{\partial}{\partial x} f(x) \xrightarrow{\text{Fourier Transform}} uF(f)(u) \]
Noise

- Simplest noise model
  - independent
  - stationary additive
  - Gaussian noise
  - the noise value at each pixel is given by an independent draw from the same normal probability distribution

- Issues
  - this model allows noise values that could be greater than maximum camera output or less than zero
  - for small standard deviations, this isn’t too much of a problem - it’s a fairly good model
  - independence may not be justified (e.g. damage to lens)
  - may not be stationary (e.g. thermal gradients in the ccd)

Finite differences and noise

- Finite difference filters respond strongly to noise
  - obvious reason: image noise results in pixels that look very different from their neighbours
  - Generally, the larger the noise the stronger the response

- What is to be done?
  - intuitively, most pixels in images look quite a lot like their neighbours
  - this is true even at an edge; along the edge they’re similar, across the edge they’re not
  - suggests that smoothing the image should help, by forcing pixels different to their neighbours (=noise pixels?) to look more like neighbours
Finite differences responding to noise

Increasing noise ->
(this is zero mean additive gaussian noise)

Smoothing reduces noise

- Generally expect pixels to “be like” their neighbours
  - surfaces turn slowly
  - relatively few reflectance changes
- Generally expect noise processes to be independent from pixel to pixel
- Implies that smoothing suppresses noise, for appropriate noise models
- Scale
  - the parameter in the symmetric Gaussian
  - as this parameter goes up, more pixels are involved in the average
  - and the image gets more blurred
  - and noise is more effectively suppressed
The effects of smoothing
Each row shows smoothing with gaussians of different width; each column shows different realisations of an image of gaussian noise.

Classical Operators

Prewitt’s Operator

\[
\begin{bmatrix}
1 & 1 \\
1 & 1 \\
1 & 1 \\
\end{bmatrix}
\rightarrow
\begin{bmatrix}
1 & -1 \\
\end{bmatrix}
\]

\[
\text{Smooth}
\]

\[
\begin{bmatrix}
1 & 1 & 1 \\
1 & 1 & 1 \\
\end{bmatrix}
\rightarrow
\begin{bmatrix}
1 \\
-1 \\
\end{bmatrix}
\]

\[
\text{Differentiate}
\]

\[
\begin{bmatrix}
1 & 0 & -1 \\
1 & 0 & -1 \\
1 & 0 & -1 \\
\end{bmatrix}
\]

\[
\begin{bmatrix}
1 & 1 & 1 \\
0 & 0 & 0 \\
-1 & -1 & -1 \\
\end{bmatrix}
\]
Classical Operators

Sobel’s Operator

\[
\begin{pmatrix}
1 & 1 \\
2 & 2 \\
1 & 1
\end{pmatrix} \rightarrow \begin{pmatrix} [1 & -1] \\
1 & 0 & 1
\end{pmatrix}
\]

Smooth \hspace{2cm} Differentiate

\[
\begin{pmatrix}
1 & 2 & 1 \\
1 & 2 & 1 \\
1 & 2 & 1
\end{pmatrix} \rightarrow \begin{pmatrix} [1 & -1] \\
1 & 0 & 1
\end{pmatrix}
\]

Gaussian Filter

\[
G_\sigma(x, y) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right)
\]

\[
H(i, j) = \frac{1}{\sqrt{2\pi\sigma^2}} \exp\left(-\frac{(i-k-1)^2 + (j-k-1)^2}{2\sigma^2}\right)
\]

where \(H(i, j)\) is \((2k+1) \times (2k+1)\) array
Suggested Reading

- Chapter 8, David A. Forsyth and Jean Ponce, "Computer Vision: A Modern Approach"
- Chapter 4, Emanuele Trucco, Alessandro Verri, "Introductory Techniques for 3-D Computer Vision"
- Chapter 2, Mubarak Shah, “Fundamentals of Computer Vision”