From Last Class

- Controller operates on the ERROR between reference & output
  - Proportional
    - Reduces disturbance error
    - Non-zero steady-state error
  - Integral
    - Ensures zero steady-state error
    - Can be destabilizing
  - Derivative
    - Adds artificial damping
    - Slows response

Graphical Explanation of Convolution

Convolution & Filtering

- Image operations as convolution, given $I$, the signal/image,
  
  \[ f[m, n] = I \otimes g = \sum_{k,l} I[m - k, n - l] g[k, l] \]

- $g[k, l]$ – is the convolution kernel
- Smoothing
Linear Filtering: Warm-Up Exercise I

Original

Coefficient

Pixel offset

Filtered (no change)

(Following examples taken from B. Freeman)

Linear Filtering: Warm-Up Exercise II

Original

Coefficient

Pixel offset

Shifted

Linear Filtering: Warm-Up Exercise III

Original

Coefficient

Pixel offset

Blurred (filter applied in both dimensions)

Linear Filtering: Warm-Up Exercise V

Original

Coefficient

Pixel offset

Sharpened original
Sharpening Examples

Original

Sharpened (differences are accentuated; constant areas are left untouched).

Convolution & Filtering

- Sharpening

\[
\begin{pmatrix}
1 & 2 & 5 & 10 \\
-2 & -1 & 1 & 1
\end{pmatrix}
\]

g(x) = 1 2 3 4 5 6

- Differentiation

Properties of the Convolution

- Convolution is commutative
  \[ I \otimes g = g \otimes I \]

- Convolutions is associative
  \[ (I \otimes g) \otimes f = I \otimes (g \otimes f) \]

- Convolution is linear
  \[
  \begin{align*}
  I \otimes (g + f) &= (I \otimes g) + (I \otimes f) \\
  \alpha(I \otimes g) &= (\alpha I) \otimes g = I \otimes (\alpha g)
  \end{align*}
  \]
**Edge Detection**

- An *Edge* is where change occurs
- Change in 1D – measured by derivative
- Max change corresponds to where derivative has max magnitude

**Computing Gradients: 1st Order Derivatives**

- Compute the gradient in the X-direction:
  1) Take the image intensity difference in the X-direction
  2) Average the difference in the Y-direction (smoothing)

\[
\frac{\delta I}{\delta x}(i,j) = \frac{1}{2} \left( I(i,j+1) - I(i,j) + I(i+1,j+1) - I(i+1,j) \right)
\]

**Some Matlab Code**

```matlab
[r,c] = size(I); 
Ix = zeros(r,c); % generate an empty matrix of size r by c 
for i=1:r-1, 
    for j=1:c-1, 
        Ix(i,j) = 0.5*(I(i,j+1) - I(i,j)) + (I(i+1,j+1) - I(i+1,j)); 
    end 
end
```

**An Example**

```matlab
Ils = imcrop(lg); 
Imagesc(Ils) colormap(gray)
```
Recall

- We can compute gradient as:
  - Convolution Operation
  - Filtering Operation

\[
\begin{align*}
I(x) &= \begin{bmatrix} 1 & 2 & 5 & 5 & 10 \end{bmatrix} \\
\delta I \frac{\delta}{\delta x} (i,j) &= (I(i, j + 1) - I(i, j)) \\
g(x) &= \begin{bmatrix} 1 \ 3 \ \text{etc.} \end{bmatrix}
\end{align*}
\]
Computing Gradients

\[ \frac{\delta I}{\delta x}(i, j) = (I(i, j + 1) - I(i, j)) \]

\[ = I \otimes (\frac{\delta}{\delta x}) \]

Derivative in X-Direction

\[ \frac{\delta}{\delta x} = \begin{pmatrix} 1 & -1 \\ 1 & 1 \end{pmatrix} \]

\[ I_x = (I \otimes \frac{\delta}{\delta x}) \otimes S \]

Computing Gradients: Y-Direction

\[ \frac{\delta I}{\delta y}(i, j) = \frac{1}{2}((I(i + 1, j) - I(i, j)) + (I(i + 1, j + 1) - I(i, j + 1))) \]

\[ I_y = (I \otimes \frac{\delta}{\delta y}) \otimes S' \]

Matlab’s Conv2 Function

\[ s = [1; 1]; \]
\[ dx = [1, -1]; \]
\[ gx = \text{conv2}(\text{conv2}(I, dx, 'same'), dx, 'same'); \]

\[ I_x = (I \otimes \frac{\delta}{\delta x}) \otimes S \]
Smoothing before Differentiating

- Smoothing – Remove noise
- Does the following make a difference:
  - Image -> Smooth -> Differentiate OR
  - Image -> Differentiate -> Smooth
- Answer: NO!

Furthermore

- We can simplify this even more …

Recall the Gaussian

\[ e^{-\frac{x^2+y^2}{2\sigma^2}} \]
**Smoothed Derivative Filter**

- Rewrite:

\[
\frac{\delta}{\delta x} \otimes G = \frac{\delta G}{\delta x} \rightarrow \frac{\delta G}{\delta x} = -\frac{2x}{\sigma_x^2} G(x, y)
\]

- Illustration of gradient computation with smoothed derivative filter.

**In Summary**

- Filter out noise and compute gradient.

  - In MATLAB:
    
    ```matlab
    >> [dx,dy] = gradient(G); % G is a 2D gaussian
    >> Ix = conv2(I,dx,'same');  Iy = conv2(I,dy,'same');
    ```

**A Word of Caution**

- Important considerations when using smoothed derivative filters for image processing.