Linear Operators

- $D$ is a linear operator iff:
  
  $$D(\alpha f_1 + \beta f_2) = \alpha D(f_1) + \beta D(f_2)$$

  Where $f_1$ and $f_2$ are images,
  
  and $\alpha$ and $\beta$ are scalar multipliers

- Not a linear operator (why?):
  
  $$g = D(f) = af + b$$
Kernel Operators

- Kernel \( h \) = “small image”
  - Often 3x3 or 5x5
- Correlated with a “normal” image \( f \)
- Implied correlation (sum of products) makes a kernel an operator. A linear operator.
- Note: This use of correlation is often mislabeled as convolution in the literature.
- Any linear operator applied to an image can be approximated with correlation.

Kernels for Derivatives

- Task: estimate partial spatial derivatives
- Solution: numerical approximation
  - \[ \frac{f(x + 1) - f(x)}{1} \]
    - Really Bad choice: not even symmetric
  - \[ \frac{f(x + 1) - f(x - 1)}{2} \]
    - Still a bad choice: very sensitive to noise
- We need to blur away the noise (only blur orthogonal to the direction of each partial):
  - \( \frac{\partial f}{\partial x} = \frac{1}{6} \left[ \begin{array}{ccc} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{array} \right] \otimes f \) or \( \frac{\partial f}{\partial x} = \frac{1}{8} \left[ \begin{array}{ccc} -1 & 0 & 1 \\ -2 & 0 & 2 \\ -1 & 0 & 1 \end{array} \right] \otimes f \)
  - The Sobel kernel is center-weighted
Derivative Estimation #2: Use Function Fitting

- Think of the image as a surface
  - The gradient then fully specifies the orientation of the tangent planes at every point, and vice-versa.
- So, fit a plane to the neighborhood around a point
  - Then the plane gives you the gradient
- The concept of fitting occurs frequently in machine vision.
  Ex:
  - Gray values
  - Surfaces
  - Lines
  - Curves
  - Etc.

Derivative Estimation: Derive a 3x3 Kernel by Fitting a Plane

- If you fit by minimizing squared error, and you use symbolic notation to generalize, you get:
  - A headache
  - The kernel that we intuitively guessed earlier:

\[
\begin{bmatrix}
-1 & 0 & 1 \\
-1 & 0 & 1 \\
1 & 6 & 1 \\
-1 & 0 & 1
\end{bmatrix}
\]
Vector Representations of Images

- Also called lexicographic representations
- Linearize the image
  - Pixels have a single index (that starts at 0)

![Image of Lexicographic Index](image)

Vector listing of pixel values

<table>
<thead>
<tr>
<th>F₀</th>
<th>F₁</th>
<th>F₂</th>
<th>F₃</th>
</tr>
</thead>
<tbody>
<tr>
<td>7</td>
<td>4</td>
<td>6</td>
<td>1</td>
</tr>
<tr>
<td>3</td>
<td>5</td>
<td>9</td>
<td>0</td>
</tr>
<tr>
<td>8</td>
<td>1</td>
<td>4</td>
<td>5</td>
</tr>
<tr>
<td>2</td>
<td>0</td>
<td>7</td>
<td>2</td>
</tr>
</tbody>
</table>

Change of coordinates

Vector Representations of Kernels

- Can also linearize a kernel
- Linearization is unique for each pixel coordinate and for each image size.
  - For pixel coordinate (1,2) (i.e. pixel F₀) in our image:

  \[
  h = \begin{bmatrix}
  -3 & 1 & 2 \\
  -5 & 4 & 6 \\
  -7 & 9 & 8 \\
  \end{bmatrix}
  \]

  \[
  Hₜ = \begin{bmatrix}
  0 & 0 & 0 & 0 & -3 & 1 & 2 & 0 & -5 & 4 & 6 & 0 & -7 & 9 & 8 & 0 \\
  \end{bmatrix}
  \]

  \[
  Hₙ = \begin{bmatrix}
  0 & 0 & 0 & 0 & -3 & 1 & 2 & 0 & -5 & 4 & 6 & 0 & -7 & 9 & 8 \\
  \end{bmatrix}
  \]

  \[
  H = \begin{bmatrix}
  -3 & 0 & 0 \\
  1 & 0 & 0 \\
  2 & 0 & 0 \\
  0 & 0 & 0 \\
  -5 & 0 & 0 \\
  4 & 1 & -3 \\
  6 & 2 & 1 \\
  -7 & 9 & 4 \\
  8 & 6 & 4 \\
  0 & 0 & 0 \\
  0 & -7 & 0 \\
  0 & 9 & -7 \\
  0 & 8 & 9 \\
  0 & 0 & 8 \\
  \end{bmatrix}
  \]

- Can combine the kernel vectors for each of the pixels into a single lexicographic kernel matrix (H)
- H is circulant (columns are rotations of one another). Why?
Convolution in Lexicographic Representations

- Convolution becomes matrix multiplication!
- Great conceptual tool for proving theorems
- $H$ is almost never computed or written out

Basis Vectors for (Sub)Images

- Carefully choose a set of basis vectors (image patches) on which to project a sub-image (window) of size $(x,y)$
  - Is this lexicographic?
- The basis vectors with the largest coefficients are the most like this sub-image.
- If we choose meaningful basis vectors, this tells us something about the sub-image

### Cartesian Basis Vectors
$$u_1 = \begin{bmatrix} 1 & \sqrt{2} & 1 \\ 0 & 0 & 0 \\ -1 & \sqrt{2} & -1 \end{bmatrix}$$
$$u_2 = \begin{bmatrix} 0 & -1 & 0 \\ \sqrt{2} & 0 & -\sqrt{2} \\ 0 & 1 & 0 \end{bmatrix}$$
$$u_3 = \begin{bmatrix} 0 & 0 & -1 \\ -1 & 0 & -1 \\ -\sqrt{2} & 1 & 0 \end{bmatrix}$$

### Frei-Chen Basis Vectors
$$u_1 = \begin{bmatrix} \sqrt{2} & 1 & 0 \\ 0 & 1 & 0 \\ -1 & 0 & 1 \end{bmatrix}$$
$$u_2 = \begin{bmatrix} 0 & 1 & 0 \\ 1 & 0 & 1 \\ 0 & 1 & 0 \end{bmatrix}$$
$$u_3 = \begin{bmatrix} -1 & \sqrt{2} & 0 \\ 0 & 1 & 0 \\ -\sqrt{2} & 1 & 1 \end{bmatrix}$$
$$u_4 = \begin{bmatrix} -1 & 0 & \sqrt{2} \\ 0 & 0 & 1 \\ -1 & -1 & 0 \end{bmatrix}$$
**Edge Detection (VERY IMPORTANT)**

- Image areas where:
  - Brightness changes suddenly =
  - Some derivative has a large magnitude
- **Often occur at object boundaries!**
- Find by:
  - Estimating partial derivatives with kernels
  - Calculating magnitude and direction from partials

![Easy to Find](Positive step edge)
![Positive roof edge](Positive roof edge)
![Positive ramp edges](Positive ramp edges)
![Noisy Positive Edge](Noisy Positive Edge)

![Harder To Find](Negative step edge)
![Negative roof edge](Negative roof edge)
![Negative ramp edges](Negative ramp edges)
![Noisy Negative Edge](Noisy Negative Edge)

11

**Edge Detection**

Diatom image (left) and its gradient magnitude (right). ([http://bigwww.epfl.ch/thevez/raz/differentials/](http://bigwww.epfl.ch/thevez/raz/differentials/))

\[
\nabla f = \begin{bmatrix} \frac{\partial f}{\partial x} \\ \frac{\partial f}{\partial y} \end{bmatrix} = \begin{bmatrix} G_x \\ G_y \end{bmatrix}^T
\]

\[
|\nabla f| = \sqrt{G_x^2 + G_y^2} = \text{Edge Strength}
\]

\[
\angle \nabla f = \tan^{-1}\left(\frac{G_x}{G_y}\right)
\]

Then **threshold** the gradient magnitude image

**Detected edges are:**
- Too thick in places
- Missing in places
- Extraneous in places
Convolving w/ Fourier

- Sometimes, the fastest way to convolve is to multiply in the frequency domain.
- Multiplication is fast. Fourier transforms are not.
- The Fast Fourier Transform (FFT) helps.
- Pratt (Snyder ref. 5.33) figured out the details.
  - Complex tradeoff depending on both the size of the kernel and the size of the image.

*For almost all image sizes

For kernels $\leq 7\times 7$, normal (spatial domain) convolution is fastest*.

For kernels $\geq 13\times 13$, the Fourier method is fastest*.

---

Image Pyramids

- A series of representations of the same image.
- Each is a 2:1 subsampling of the image at the next "lower level.
  - Subsampling = averaging = down sampling.
  - The subsampling happens across all dimensions!
  - For a 2D image, 4 pixels in one layer correspond to 1 pixel in the next layer.
- To make a Gaussian pyramid:
  1. Blur with Gaussian.
  2. Down sample by 2:1 in each dimension.
  3. Go to step 1.

Increasing Scale
Scale Space

- Multiple levels like a pyramid
- Blur like a pyramid
- **But don’t subsample**
  - All layers have the same size
- **Instead:**
  - Convolve each layer with a Gaussian of variance $\sigma$.
  - $\sigma$ is the “scale parameter”
  - Only large features are visible at high scale (large $\sigma$).

Quad/Oc Trees

- Represent an image
- Homogeneous blocks
- Inefficient for storage
  - Too much overhead
- Not stable across small changes
- But: Useful for representing scale space.
Gaussian Scale Space

- Large scale = only large objects are visible
  - Increasing $\sigma \rightarrow$ coarser representations
- Scale space causality
  - Increasing $\sigma \rightarrow$ # extrema should not increase
  - Allows you to find “important” edges first at high scale.
- How features vary with scale tells us something about the image
- Non-integral steps in scale can be used
- Useful for representing:
  - Brightness
  - Texture
  - PDF (scale space implements clustering)

How do People Do It?

- Receptive fields
- Representable by Gabor functions
  - 2D Gaussian +
  - A plane wave
- The plane wave tends to propagate along the short axis of the Gaussian
- But also representable by Difference of offset Gaussians
  - Only 3 extrema
Canny Edge Detector

1. Use kernels to find at every point:
   - Gradient magnitude
   - Gradient direction
2. Perform Nonmaximum suppression (NMS) on the magnitude image
   - This thins edges that are too thick
   - Only preserve gradient magnitudes that are maximum compared to their 2 neighbors in the direction of the gradient

Canny Edge Detector, contd.

- Edges are now properly located and 1 pixel wide
- But noise leads to false edges, and noise+blur lead to missing edges.
  - Help this with 2 thresholds
  - A high threshold does not get many false edges, and a low threshold does not miss many edges.
  - Do a “flood fill” on the low threshold result, seeded by the high-threshold result
    - Only flood fill along isophotes
Reminders

- Quiz 4 next class
- HW2 due Monday night, Feb. 11
  - Help stops at 5pm on 11th
  - You can submit up until midnight
  - Please CC your Pitt or CMU email address when you submit, as your “proof” of submission.