Welcome!

• About Me
  • (Relatively) New Faculty Member (Two Years)
  • Interested in Machine Vision and Machine Learning
  • Happy to chat with you at almost any time
  • May want to e-mail me first
  • Office Hours:
    • Monday and Wednesday 1:00PM to 2:30PM

Grading

• Problem Sets – 30%
• Take Home Exams – 30%
• Project – 40%
  • Exciting opportunity
  • CVPR Submission Deadline: Late Nov.
  • Think about working in groups

Doing the problems

• Finishing the problem sets will require access to an interpreted environment
  • MATLAB
  • Octave
  • Numerical Python
• NO COMPILED LANGUAGES!!!!
  • No C/C++
  • No Java
  • No x86 Assembler
• My Compiled Languages Rant

Environments

• MATLAB
  • Pro: Well-established package. You can find many tutorials on the net.
  • Con: Not free. If your lab does not already have it, talk to me about getting access.
• Octave
  • Free MATLAB look-alike
  • Pro: Should be able to handle anything you will do in this class
  • Con: “Should be”. I’m not sure about support in Windows

• Numerical Python
  • All the capabilities of MATLAB
  • Free!
  • Real programming language
  • Used for lots of stuff besides numerical computing
  • Cons: Documentation is a bit sparse and can be outdated
    • I can get you started – I am working on a tutorial
Math

- We will use it
- We will be talking about mathematical models of images and image formation
- This class is not about proving theorems
- My goal is to have you build intuitions about the models
- Try and visualize the computation that each equation is expressing
- Basic Calculus and Basic Linear Algebra should be sufficient

Course Text

- We will use Forsyth and Ponce
- Not required

Course Structure

- This year, we will be covering pattern recognition more deeply than in previous years
- Machine learning is critical to modern computer vision
- You need to understand it well
- Important Foundational Topics:
  - Image Processing
  - Optimization
  - Machine Learning
  - Geometry

Image Processing

- For now, we won't worry about the physical aspects of getting images
- View image as an array of continuous values

Simple Modification

- What if we wanted to blur this image?
- We could take a local average
  - Replace each pixel with the mean of an NxN pixel neighborhood surrounding that pixel.

3x3 Neighborhood

Original

Averaged
Let's represent this more generally

- Multiply corresponding numbers and add
- Template moves across the image
- Think of it as a sliding window
This is called *convolution*

- Mathematically expressed as
  \[
  R(i, j) = \sum_{m=-N}^{N} \sum_{n=-N}^{N} I(m, n)K(i - m, j - n)
  \]

Take out a piece of paper

- Let’s say \( i = 10 \) and \( j=10 \)
- Which location in \( K \) is multiplied by \( I(5,5) \)?
- \( I(5,4) \)

\[
R(i, j) = \sum_{m=-N}^{N} \sum_{n=-N}^{N} I(m, n)K(i - m, j - n)
\]

Notation

- Also denoted as
- \( R = I \ast K \)
- We “convolve” \( I \) with \( K \)
  - Not convolute!

Sliding Template View

- Take the template \( K \)

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

- Flip it

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

- Slide across image

Predict the Image

\[
\begin{array}{ccc}
0 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 0
\end{array}
\]

Predict the Image

\[
\begin{array}{ccc}
0 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 0
\end{array}
\]
What if I wanted to compute 
\[ R(i,j) = I(i+1,j) - I(i,j) \]
at every pixel?

• What would the kernel be?
• This is one discrete approximation to the derivative

What’s the problem with this derivative?

\[ [1 \ -1 \ 0] \]

Where’s the center of the derivative?

• An alternative
  \[ [-1 \ 0 \ 1] \]
Your Convolution filter toolbox

- In my experience, 90% of the filtering that you will do will be either
  - Smoothing (or Blurring)
  - High-Pass Filtering (I'll explain this later)
- Most common filters:
  - Smoothing: Gaussian
  - High Pass Filtering: Derivative of Gaussian

Gaussian Filter

- Let's assume that a \((2k+1) \times (2k+1)\) filter is parameterized from \(-k\) to \(+k\)
- The Gaussian filter has the form
  \[
  K(i,j) = \frac{1}{2\pi \sigma^2} e^{-\frac{i^2 + j^2}{2\sigma^2}}
  \]
- And looks like

Derivative of Gaussian Filter

- Take the derivative of the filter with respect to \(i\):
  \[
  \frac{\partial K(i,j)}{\partial i} = -i \sigma^2 Z e^{-\frac{i^2 + j^2}{2\sigma^2}}
  \]
- Filter looks like:

Effect of Changing \(\sigma\)

- With \(\sigma\) set to 1
- With \(\sigma\) set to 3

Practical Aspects of Computing Convolutions

- Let's blur this flat, gray image:

Practical Aspects of Computing Convolutions

- Depending on how you do the convolution in MATLAB, you could end up with 3 different images

266x266 Image 256x256 Image 246x246 Image
Border Handling

- Lets go back to the sliding template view
- What if I wanted to compute an average right here?

<table>
<thead>
<tr>
<th></th>
<th>1/9</th>
<th>1/9</th>
<th>1/9</th>
</tr>
</thead>
<tbody>
<tr>
<td>1/9</td>
<td>1/9</td>
<td>3/9</td>
<td>0</td>
</tr>
<tr>
<td>1/9</td>
<td>1/9</td>
<td>2/9</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>2</td>
<td>46</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>2</td>
<td>43</td>
</tr>
</tbody>
</table>

Practical Aspects of Computing Convolutions

- Filled in borders with zeros, computed everywhere the kernel touches
- Called "full" in MATLAB

266x266 Image

Border Handling

- Fill in border with zeros, only compute at "original pixels"
- Called "same" in MATLAB

<table>
<thead>
<tr>
<th></th>
<th>1/9</th>
<th>1/9</th>
<th>1/9</th>
</tr>
</thead>
<tbody>
<tr>
<td>1/9</td>
<td>1/9</td>
<td>1/9</td>
<td>1/9</td>
</tr>
<tr>
<td>1/9</td>
<td>1/9</td>
<td>1/9</td>
<td>1/9</td>
</tr>
<tr>
<td>1/9</td>
<td>1/9</td>
<td>1/9</td>
<td>1/9</td>
</tr>
<tr>
<td>1/9</td>
<td>1/9</td>
<td>1/9</td>
<td>1/9</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>2</td>
<td>46</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>2</td>
<td>43</td>
</tr>
</tbody>
</table>
Practical Aspects of Computing

Convolutions

- Only compute at places where the kernel fits in the image

There are other options

- The first two methods that I described fill missing values in by substituting zero
- Can fill in values with different methods
  - Reflect image along border
  - Pull values from other side
- Not supported in MATLAB’s convolution
  - Eero Simoncelli has a package that supports that kind of convolution

Going Non-Linear

- Convolution is a linear operation
  - What does that mean?
- A simple non-linear operation is the median filter
  - Will explore that filter in the first problem set.

Basic Convolution Properties

\[
\begin{align*}
  f(x) * ((g(x) * h(x))) &= (f(x) * g(x)) * h(x) \\
  (\alpha f(x)) + g(x) &= \alpha (f(x) + g(x)) \\
  f(x) * g(x) &= g(x) * f(x) \\
  f(x) * (g(x) + h(x)) &= f(x) * g(x) + f(x) * h(x)
\end{align*}
\]

- Can derive all of these with the definition of convolution
- Comes from linearity of convolution

\[
R(i,j) = \sum_{m=-N}^{N} \sum_{n=-N}^{N} I(m,n)K(i-m,j-n)
\]

Practical Use of These Properties – Image Sharpening

- Take this image and blur it

Practical Use of These Properties – Image Sharpening

- What do we get if we subtract the two?

- This is the leftover “sharp-stuff”
Let's make the image sharper

• We know

\[ \begin{array}{c}
\text{Original} \\
+ \\
\text{2 ×} \\
= \\
\text{Sharpened}
\end{array} \]

Let's boost the sharp stuff a little

\[ 2 × + = \]

Now look at the computation

• Operations
  - 1 convolution
  - 1 subtraction over the whole image
• As an equation:

\[ I \ast f + 2 (I - I \ast f) \]

Rewrite this

\[ I \ast f + 2 (I \ast \delta - I \ast f) \]

This is an identity filter or unit impulse
Basic Convolution Properties

\begin{align*}
  f(x) \ast ((g(x) + h(x)) &= (f(x) \ast g(x)) + h(x) \\
  (\alpha f(x)) \ast h(x) &= \alpha (f(x) \ast g(x)) \\
  f(x) \ast g(x) &= g(x) \ast f(x) \\
  f(x) \ast (g(x) + h(x)) &= f(x) \ast g(x) + f(x) \ast h(x)
\end{align*}

• Can derive all of these with the definition of convolution
• Comes from linearity of convolution

\[ R(i, j) = \sum_{m=-N}^{N} \sum_{n=-N}^{N} I(m, n) K(i - m, j - n) \]

Rewrite this

\begin{align*}
  \mathcal{I} \ast f + 2 (\mathcal{I} - \mathcal{I} \ast f) \\
  \mathcal{I} \ast f + 2 (\mathcal{I} \ast \delta - \mathcal{I} \ast f) \\
  \mathcal{I} \ast (f + 2\delta - 2f)
\end{align*}

Now look at the computation

\[ \mathcal{I} \ast (f + 2\delta - 2f) \]

• Can pre-compute new filter
• Operations
  1 convolution