HW Thoughts

• Homework in 442 is a bit more flexible, open-ended, and less guided compared to 281
• A difficulty of 442 is that you learn the material and how to do more open-ended work
• This is a *skill* that you learn
• Numpy is a skill too!
HW Thoughts – Cons

- Some stuff will behave mysteriously
- Some library code you’ll call will be annoying
- You won’t really work with a specification but rather a guide
- In general, can be much more frustrating, especially at first
- Side-note: Check the homework release post.
HW Thoughts – Pros

• Life isn’t fill-in-the-blanks!
• But handling open-endedness is a skill you learn, develop, and practice. You don’t start out good at it, but you get better at it.
• Good news: Many more answers are right (unless it’s a specific math equation)
• In general, can be much more rewarding.
HW Thoughts

• We rarely expect you match exact bytes
• Look at calling code; look at code that calls you. Google funny numpy bugs.
• Print sizes, types each time you have an error.
• Try on small examples
• Work in groups
Let’s Take An Image
Let’s Fix Things

• We have noise in our image
• Let’s replace each pixel with a *weighted* average of its neighborhood
• Weights are *filter kernel*

![Weighted Filter Kernel](attachment:image.png)
What's the average of 9, 10, 12?

(a) 9  (b) 11.5  (c) 10.33  (d) 11.66
1D Case

Signal

| 10 | 12 | 9  | 11 | 10 | 11 | 12 |

Filter

| 1/3 | 1/3 | 1/3 |

Output

10.33

Done! Next?
1D Case

Signal

| 10 | 12 | 9  | 11 | 10 | 11 | 12 |

Filter

| 1/3 | 1/3 | 1/3 |

Output

| 10.33 | 10.66 |

(a) 10.66  (b) 9.33  (c) 14.2  (d) 11.33
1D Case

(a) 10.33  (b) 11.33
(c) 10  (d) 9.1

Signal

Filter

Output
1D Case

Signal

Filter

Output

10 12 9 11 10 11 12

1/3 1/3 1/3

10.33 10.66 10 10.66
1D Case

Signal

10  12  9  11  10  11  12

Filter

1/3  1/3  1/3

Output

10.33  10.66  10  10.66  11
1D Case

You lose pixels (maybe...)
Filter “sees” only a few pixels at a time
Applying a Linear Filter

<table>
<thead>
<tr>
<th>Input</th>
<th>Filter</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td>I11</td>
<td>F11</td>
<td>O11</td>
</tr>
<tr>
<td>I12</td>
<td>F12</td>
<td>O12</td>
</tr>
<tr>
<td>I13</td>
<td>F13</td>
<td>O13</td>
</tr>
<tr>
<td>I14</td>
<td>F21</td>
<td>O14</td>
</tr>
<tr>
<td>I15</td>
<td>F22</td>
<td>O21</td>
</tr>
<tr>
<td>I16</td>
<td>F23</td>
<td>O22</td>
</tr>
<tr>
<td>I21</td>
<td>F31</td>
<td>O23</td>
</tr>
<tr>
<td>I22</td>
<td>F32</td>
<td>O24</td>
</tr>
<tr>
<td>I23</td>
<td>F33</td>
<td>O31</td>
</tr>
<tr>
<td>I24</td>
<td></td>
<td>O32</td>
</tr>
<tr>
<td>I25</td>
<td></td>
<td>O33</td>
</tr>
<tr>
<td>I26</td>
<td></td>
<td>O34</td>
</tr>
<tr>
<td>I31</td>
<td></td>
<td></td>
</tr>
<tr>
<td>I32</td>
<td></td>
<td></td>
</tr>
<tr>
<td>I33</td>
<td></td>
<td></td>
</tr>
<tr>
<td>I34</td>
<td></td>
<td></td>
</tr>
<tr>
<td>I35</td>
<td></td>
<td></td>
</tr>
<tr>
<td>I36</td>
<td></td>
<td></td>
</tr>
<tr>
<td>I41</td>
<td></td>
<td></td>
</tr>
<tr>
<td>I42</td>
<td></td>
<td></td>
</tr>
<tr>
<td>I43</td>
<td></td>
<td></td>
</tr>
<tr>
<td>I44</td>
<td></td>
<td></td>
</tr>
<tr>
<td>I45</td>
<td></td>
<td></td>
</tr>
<tr>
<td>I46</td>
<td></td>
<td></td>
</tr>
<tr>
<td>I51</td>
<td></td>
<td></td>
</tr>
<tr>
<td>I52</td>
<td></td>
<td></td>
</tr>
<tr>
<td>I53</td>
<td></td>
<td></td>
</tr>
<tr>
<td>I54</td>
<td></td>
<td></td>
</tr>
<tr>
<td>I55</td>
<td></td>
<td></td>
</tr>
<tr>
<td>I56</td>
<td></td>
<td></td>
</tr>
</tbody>
</table>
Applying a Linear Filter

\[
O_{11} = I_{11}F_{11} + I_{12}F_{12} + \ldots + I_{33}F_{33}
\]
Applying a Linear Filter

Input & Filter

\[ O_{12} = I_{12}F_{11} + I_{13}F_{12} + \ldots + I_{34}F_{33} \]
Applying a Linear Filter

Input

<table>
<thead>
<tr>
<th>I11</th>
<th>I12</th>
<th>I13</th>
<th>I14</th>
<th>I15</th>
<th>I16</th>
</tr>
</thead>
<tbody>
<tr>
<td>I21</td>
<td>I22</td>
<td>I23</td>
<td>I24</td>
<td>I25</td>
<td>I26</td>
</tr>
<tr>
<td>I31</td>
<td>I32</td>
<td>I33</td>
<td>I34</td>
<td>I35</td>
<td>I36</td>
</tr>
<tr>
<td>I41</td>
<td>I42</td>
<td>I43</td>
<td>I44</td>
<td>I45</td>
<td>I46</td>
</tr>
<tr>
<td>I51</td>
<td>I52</td>
<td>I53</td>
<td>I54</td>
<td>I55</td>
<td>I56</td>
</tr>
</tbody>
</table>

Filter

<table>
<thead>
<tr>
<th>F11</th>
<th>F12</th>
<th>F13</th>
</tr>
</thead>
<tbody>
<tr>
<td>F21</td>
<td>F22</td>
<td>F23</td>
</tr>
<tr>
<td>F31</td>
<td>F32</td>
<td>F33</td>
</tr>
</tbody>
</table>

Output

How many times can we apply a 3x3 filter to a 5x6 image?
Applying a Linear Filter

\[
O_{ij} = I_{ij} \cdot F_{11} + I_{i(j+1)} \cdot F_{12} + \ldots + I_{(i+2)(j+2)} \cdot F_{33}
\]
Painful Details – Edge Cases
Convolution doesn’t keep the whole image.
Suppose $f$ is the image and $g$ the filter.

**Full** – any part of $g$ touches $f$. **Same** – same size as $f$; **Valid** – only when filter doesn’t fall off edge.

Full – any part of $g$ touches $f$. Same – same size as $f$; Valid – only when filter doesn’t fall off edge.
Painful Details – Edge Cases

What to about the “?” region?

Symm: fold sides over

Circular/Wrap: wrap around

pad/fill: add value, often 0

f/g Diagram Credit: D. Lowe
Painful Details – Does it Matter?

(I’ve applied the filter per-color channel)

Which padding did I use and why?

Input Image

Box Filtered ???

Box Filtered ???

Note – this is a zoom of the filtered, not a filter of the zoomed
Painful Details – Does it Matter?

(I’ve applied the filter per-color channel)

Input Image  Box Filtered Symm Pad  Box Filtered Zero Pad

Note – this is a zoom of the filtered, not a filter of the zoomed
Practice with Linear Filters

Original

\[
\begin{array}{ccc}
0 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 0 \\
\end{array}
\]
Practice with Linear Filters

Original

The Same!
Practice with Linear Filters

Original

\[
\begin{array}{ccc}
0 & 0 & 0 \\
0 & 0 & 1 \\
0 & 0 & 0 \\
\end{array}
\]
Practice with Linear Filters

Original

Shifted \textit{LEFT} 1 pixel

\begin{pmatrix}
0 & 0 & 0 \\
0 & 0 & 1 \\
0 & 0 & 0 \\
\end{pmatrix}
Practice with Linear Filters

Original

Slide Credit: D. Lowe
Practice with Linear Filters

Original

```
0 1 0
0 0 0
0 0 0
```

Shifted **DOWN**
1 pixel
Practice with Linear Filters

Original

\[
\begin{array}{ccc}
1/9 & 1/9 & 1/9 \\
1/9 & 1/9 & 1/9 \\
1/9 & 1/9 & 1/9 \\
\end{array}
\]
Practice with Linear Filters

Original

Blur (Box Filter)
Practice with Linear Filters

Original

\[
\begin{bmatrix}
0 & 0 & 0 \\
0 & 2 & 0 \\
0 & 0 & 0 \\
\end{bmatrix}
\]

?
Practice with Linear Filters

Original

<table>
<thead>
<tr>
<th>0</th>
<th>0</th>
<th>0</th>
</tr>
</thead>
<tbody>
<tr>
<td>0</td>
<td>2</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

Sharpened

(Accentuates difference from local average)

\[
\begin{array}{ccc}
\frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\
\frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\
\frac{1}{9} & \frac{1}{9} & \frac{1}{9} \\
\end{array}
\]
Sharpening

before

after

Slide Credit: D. Lowe
Properties – Linear

Assume: I image f1, f2 filters

**Linear:** \( \text{apply}(I,f1+f2) = \text{apply}(I,f1) + \text{apply}(I,f2) \)

I is a white box on black, and f1, f2 are rectangles

\[
A(\begin{array}{c}
\text{□} \\
\text{□} \\
\end{array}, \begin{array}{c}
\text{□} \\
\text{□} \\
\end{array} + \begin{array}{c}
\text{□} \\
\text{□} \\
\end{array}) = A(\begin{array}{c}
\text{□} \\
\text{□} \\
\end{array}, \begin{array}{c}
\text{□} \\
\text{□} \\
\end{array}) = \begin{array}{c}
\text{□} \\
\text{□} \\
\end{array}
\]

\[
A(\begin{array}{c}
\text{□} \\
\text{□} \\
\end{array}, \begin{array}{c}
\text{□} \\
\text{□} \\
\end{array}) + A(\begin{array}{c}
\text{□} \\
\text{□} \\
\end{array}, \begin{array}{c}
\text{□} \\
\text{□} \\
\end{array}) = \begin{array}{c}
\text{□} \\
\text{□} \\
\end{array} + \begin{array}{c}
\text{□} \\
\text{□} \\
\end{array} = \begin{array}{c}
\text{□} \\
\text{□} \\
\end{array}
\]

Note: I am showing filters un-normalized and blown up. They’re a smaller box filter (i.e., each entry is \(1/(\text{size}^2)\))
Properties – Shift-Invariant

Assume: I image, f filter

**Shift-invariant:** \( \text{shift}(\text{apply}(I,f)) = \text{apply}(\text{shift}(I,f)) \)

Intuitively: only depends on filter neighborhood
Painful Details – Signal Processing

Often called “convolution”. Actually cross-correlation. Source of **terrible** confusion.

Cross-Correlation  
(Original Orientation)  

Convolution  
(Flipped in x and y)
Properties of Convolution

• Any shift-invariant, linear operation is a convolution (\(\ast\))
• Commutative: \(f \ast g = g \ast f\)
• Associative: \((f \ast g) \ast h = f \ast (g \ast h)\)
• Distributes over +: \(f \ast (g + h) = f \ast g + f \ast h\)
• Scalars factor out: \(kf \ast g = f \ast kg = k (f \ast g)\)
• Identity (a single one with all zeros):

\[
\begin{array}{c}
\text{Property List: K. Grauman}
\end{array}
\]
Questions?

• Nearly everything onwards is a convolution.
• This is important to get right.
### Smoothing With A Box

Intuition: if filter touches it, it gets a contribution.

<table>
<thead>
<tr>
<th>Input</th>
<th>Filter</th>
<th>Output</th>
</tr>
</thead>
<tbody>
<tr>
<td><img src="image1.png" alt="Input Image" /></td>
<td><img src="image2.png" alt="Filter" /></td>
<td><img src="image3.png" alt="Output Image" /></td>
</tr>
</tbody>
</table>
Solution – Weighted Combination

Intuition: weight contributions according to closeness to center.

\[ \text{Filter}_{ij} \propto 1 \]

What’s this?

\[ \text{Filter}_{ij} \propto \exp \left(- \frac{x^2 + y^2}{2\sigma^2} \right) \]
Recognize the Filter?

It’s a Gaussian!

\[
Filter_{ij} \propto \frac{1}{2\pi\sigma^2} \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right)
\]
Smoothing With A Box & Gauss

Still have some speckles, but it’s not a big box

Input

Box Filter

Gauss. Filter
Gaussian Filters

σ = 1
filter = 21x21

σ = 2
filter = 21x21

σ = 4
filter = 21x21

σ = 8
filter = 21x21

Note: filter visualizations are independently normalized throughout the slides so you can see them better
Applying Gaussian Filters
Applying Gaussian Filters

Input Image
(no filter)
Applying Gaussian Filters

\[ \sigma = 1 \]
Applying Gaussian Filters

\[ \sigma = 2 \]
Applying Gaussian Filters

\[ \sigma = 4 \]
Applying Gaussian Filters

$\sigma = 8$
Picking a Filter Size

Too small filter $\rightarrow$ bad approximation
Wants size $\approx 6\sigma$ (99.7% of energy)
Left far too small; right slightly too small!

$\sigma = 8$, size = 21  $\sigma = 8$, size = 43
Runtime Complexity

Image size = \( N \times N = 6 \times 6 \)
Filter size = \( M \times M = 3 \times 3 \)

\[
\begin{array}{cccccc}
I_{11} & I_{12} & I_{13} & I_{14} & I_{15} & I_{16} \\
I_{21} & F_{11} & F_{12} & F_{13} & I_{25} & I_{26} \\
I_{31} & F_{21} & F_{22} & F_{23} & I_{35} & I_{36} \\
I_{41} & F_{31} & F_{32} & F_{33} & I_{45} & I_{46} \\
I_{51} & I_{52} & I_{53} & I_{54} & I_{55} & I_{56} \\
I_{61} & I_{62} & I_{63} & I_{64} & I_{65} & I_{66}
\end{array}
\]

for Image\( Y \) in range(\( N \)):
  for Image\( X \) in range(\( N \)):
    for Filter\( Y \) in range(\( M \)):
      for Filter\( X \) in range(\( M \)):
        ...

Time: \( O(N^2M^2) \)
Separability

Conv(vector, transposed vector) → outer product
Separability

\[
\text{Filter}_{ij} \propto \frac{1}{2\pi \sigma^2} \exp\left(-\frac{x^2 + y^2}{2\sigma^2}\right)
\]

\[
\rightarrow
\]

\[
\text{Filter}_{ij} \propto \frac{1}{\sqrt{2\pi \sigma}} \exp\left(-\frac{x^2}{2\sigma^2}\right) \frac{1}{\sqrt{2\pi \sigma}} \exp\left(-\frac{y^2}{2\sigma^2}\right)
\]
Separability

1D Gaussian $\ast$ 1D Gaussian = 2D Gaussian

Image $\ast$ 2D Gauss = Image $\ast$ (1D Gauss $\ast$ 1D Gauss )
= (Image $\ast$ 1D Gauss) $\ast$ 1D Gauss
Runtime Complexity

Image size = N x N = 6 x 6
Filter size = M x 1 = 3 x 1

What are my compute savings for a 13x13 filter?

Time: $O(N^2M)$
Why Gaussian?

Gaussian filtering removes parts of the signal above a certain frequency. Often noise is high frequency and signal is low frequency.
Where Gaussian Fails
Applying Gaussian Filters

\[ \sigma = 1 \]
### Why Does This Fail?

Means can be arbitrarily distorted by outliers

#### Signal

| 10 | 12 | 9 | 8 | 1000 | 11 | 10 | 12 |

#### Filter

| 0.1 | 0.8 | 0.1 |

#### Output

| 11.5 | 9.2 | 107.3 | 801.9 | 109.8 | 10.3 |

**What else is an “average” other than a mean?**
Non-linear Filters (2D)

\[
\begin{bmatrix}
40 & 81 & 13 & 22 \\
125 & 830 & 76 & 80 \\
144 & 92 & 108 & 95 \\
132 & 102 & 106 & 87 \\
\end{bmatrix}
\]

[040, 081, 013, 125, 830, 076, 144, 092, 108]

Sort

[013, 040, 076, 081, 092, 108, 125, 144, 830]

92

[830, 076, 080, 092, 108, 095, 102, 106, 087]

Sort

[076, 080, 087, 092, 095, 102, 106, 108, 830]

95
Applying Median Filter

Median Filter (size=3)
Applying Median Filter

Median Filter (size = 7)
Is Median Filtering Linear?

\[
\begin{bmatrix}
1 & 1 & 1 \\
1 & 1 & 2 \\
2 & 2 & 2 \\
\end{bmatrix}
\begin{bmatrix}
0 & 0 & 0 \\
0 & 1 & 0 \\
0 & 0 & 0 \\
\end{bmatrix}
= 
\begin{bmatrix}
1 & 1 & 1 \\
1 & 2 & 2 \\
2 & 2 & 2 \\
\end{bmatrix}
\]

Example from (I believe): Kristen Grauman
Some Examples of Filtering
Filtering – Sharpening

Image

- Smoothened

Details
Filtering – Sharpening

Image

Details

\[ \text{"Sharpened" } \alpha = 1 \]
Filtering – Sharpening

Image

Details

"Sharpened" $\alpha=0$

$\alpha + \alpha$
Filtering – Sharpening

Image + $\alpha$

"Sharpened" $\alpha=2$
Filtering – Sharpening

Image

Details

\[ +\alpha \]

“Sharpened” \( \alpha=0 \)
Filtering – Extreme Sharpening

Image + \alpha

"Sharpened" \alpha=10
Filtering

What’s this Filter?

Dx

Dy
Filtering – Derivatives

\[(Dx^2 + Dy^2)^{1/2}\]
Filtering – Bonus
Filtering – Counting

How many “on” pixels have 10+ neighbors within 10 pixels?
Filtering – Counting

How many “on” pixels have 10+ neighbors within 10 pixels?
Filtering – Missing Data

Oh no! Missing data!
(and we know where)

Common with many non-normal cameras (e.g., depth cameras)
Filtering – Missing Data

Image

Binary Mask

Per-element /
Filtering – Missing Data

Image

Binary Mask

Per-element /
Filtering – Missing Data

Before
Filtering – Missing Data

After
Filtering – Missing Data

After (without missing data)