CSCI 497P/597P: Computer Vision

Lecture 2: Image transformations and filtering
CSCI 497P/597P: Computer Vision

Lecture 2: Image transformations and filtering
Goals

• Know how to filter (v) an image by cross-correlating it with a given filter (n)/kernel/weights

• Get a feel for some of the image processing operations that can be accomplished using filtering.

• Know how to handle image borders when filtering:
  • output sizes: full / same / valid
  • out-of-bounds values: zeros, reflection, replication
Last time

Written as a function, we can *transform* the image function to create altered functions (images):

\[ g(x,y) = f(x,y) + 20 \]

(transforming the range)

\[ g(x,y) = f(-x,y) \]

(transforming the domain)
Last time

Written as a function, we can transform the image function to create altered functions (images):

\[ g(x, y) = f(x, y) + 20 \]

(transforming the range) 

\[ g(x, y) = f(-x, y) \]

(transforming the domain) (increase brightness)
Last time

Written as a function, we can *transform* the image function to create altered functions (images):

\[
g(x, y) = f(x, y) + 20
\]

(increase brightness)

\[
g(x, y) = f(-x, y)
\]

(flip horizontally)

(transforming the range)

(transforming the domain)
Last time

Make $f(x, y)$ 2 times bigger?

$g(x, y) = \frac{1}{2} f\left(2x, 2y\right)$

Make $F$ (a HxWx3 image) redder?

Increase contrast?
Real images aren't perfect

Real images are not only sampled, but they often have noise: unwanted variations in measured intensity value.

Causes of noise (incomplete list):
- electronic variations in sensor chip
- analog-to-digital quantization
- film grain
- cosmic rays

$f(x, y)$
Real images aren't perfect

Real images are not only sampled, but they often have noise: unwanted variations in measured intensity value.

$f(x, y)$

Causes of noise (incomplete list):
• electronic variations in sensor chip
• analog-to-digital quantization
• film grain
• cosmic rays

often, we can assume that noise is random
Real images aren't perfect

Real images are not only sampled, but they often have noise: unwanted variations in measured intensity value.

Causes of noise (incomplete list):
- electronic variations in sensor chip
- analog-to-digital quantization
- film grain
- cosmic rays

often, we can assume that noise is random

(other times, we can't but we do anyway)

\[ f(x, y) \]
Denoising

Scenario: you have a camera and this motionless scene. How can you get a less noisy image?

(let’s assume that noise is random)
Denoising

Scenario: you have a camera and this motionless scene. How can you get a less noisy image?

(let's assume that noise is random)

Ideally: average multiple images together

\[ g(x, y) = \frac{1}{n} \sum_{i=0}^{n} f_i(x, y) \]
Example: Denoising

Scenario: you're simply given a noisy image. Can you reduce the noise? (still assume that noise is random)

Some function, e.g., average
Example: Denoising

Scenario: you're simply given a noisy image. Can you reduce the noise? (still assume that noise is random)

Next best thing, a heuristic: nearby pixels are often the same (ideal) color, so average neighboring pixels together.

```
10  5  3
  4  5  1
  1  1  7
```
some function, e.g., average
Example: Denoising

Scenario: you're simply given a noisy image. Can you reduce the noise?  
(still assume that noise is random)

Next best thing, a heuristic: nearby pixels are often the same (ideal) color, so average neighboring pixels together.

This is an example of filtering. We're taking the mean of the neighborhood, so it's called mean filtering.
Filtering

\[ g = f \otimes w \]

Output image

Input image

Weights, or filter, or kernel

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

\[ g = f \otimes w \]

output image

input image

weights, or filter, or kernel

\[
\begin{array}{ccc}
1 & 1 & 1 \\
1 & 1 & 1 \\
0 & 1 & 1 \\
\end{array}
\]
Filtering: Let's play.

Colab notebook playground
(also linked from today's lecture on the course webpage):

https://colab.research.google.com/drive/1KasDni0Km_9HVuQXdARIQh3GS2uchVAZ?usp=sharing

1. Notebook demo
2. In groups: answer the 6 problems in your group's Google Doc.
Mean filtering: Mathily

\[ g(x, y) = \sum_{i=-1}^{1} \sum_{j=-1}^{1} \left( \frac{1}{9} f(x + i, y + j) \right) \]

one output pixel = average (3x3 neighborhood of input pixels)
Mean filtering: Mathily

\[ g(x, y) = \sum_{i=-1}^{1} \sum_{j=-1}^{1} \frac{1}{9} f(x + i, y + j) \]

one output pixel = average (3x3 neighborhood of input pixels)
Generalize!

From a 3x3 mean filter to any size mean filter

\[ g(x, y) = \sum_{i=-1}^{1} \sum_{j=-1}^{1} \frac{1}{9} f(x + i, y + j) \]

one output pixel = average (3x3 neighborhood of input pixels)
Generalize!
From a 3x3 mean filter to any size mean filter

\[ g(x, y) = \sum_{i=-k}^{k} \sum_{j=-k}^{k} \frac{1}{(2k + 1)^2} f(x + i, y + j) \]

this makes sure we **average** all values in the neighborhood
Generalize!

From a 3x3 mean filter to any size mean filter

\[
g(x, y) = \sum_{i=-k}^{k} \sum_{j=-k}^{k} \frac{1}{(2k+1)^2} f(x + i, y + j)
\]

this makes sure we **average** all values in the neighborhood

Let's generalize to a **weighted average**.

Also store weights in a 2D array (as in the playground): \(w(i,j)\)

for convenience, \((0,0)\) is at the center
Generalize!

To a **weighted average**.

\[
g(x, y) = \sum_{i=-k}^{k} \sum_{j=-k}^{k} w(i, j) f(x + i, y + j)
\]

this makes sure we **average** all values in the neighborhood.

Also store weights in a 2D array (as in the demo): \( w(i,j) \)

for convenience, \((0,0)\) is at the center.
Cross-Correlation

We've just derived the cross-correlation operator.

\[ g(x, y) = \sum_{i=-k}^{k} \sum_{j=-k}^{k} w(i, j) f(x + i, y + j) \]

We write this as:

\[ g = f \ast w \]

weights, or filter, or kernel
Cross-Correlation

We've just derived the cross-correlation operator.

\[ g(x, y) = \sum_{i=-k}^{k} \sum_{j=-k}^{k} w(i, j) f(x + i, y + j) \]

We write this as:

\[ g = f \otimes w \]

output image
input image
weights, or filter, or kernel
Computing Cross-Correlation

\[ g = f \otimes w \]

output image
input image

weights, or filter, or kernel

Naive pseudocode:

\[
\begin{align*}
\text{for } x = 0 \text{ to } w: \\
\quad \text{for } y = 0 \text{ to } h: \\
\quad \quad \text{for } i \text{ in } -k \text{ to } k: \\
\quad \quad \quad \text{for } j \text{ in } -k \text{ to } k: \\
\quad \quad \quad \quad \text{out}[x,y] \leftarrow w[i,j] \ast \text{ in}[x+i, y+j]
\end{align*}
\]
Computing Cross-Correlation

\[ g = f \otimes w \]

output image

input image

weights, or filter, or kernel

Naive pseudocode:

for \( x = 0 \) to \( w \):
    for \( y = 0 \) to \( h \):
        for \( i \) in \(-k\) to \( k\):
            for \( j \) in \(-k\) to \( k\):
                \[ \text{out}[x,y] += w[i,j] \times \text{in}[x+i, y+j] \]
Questions remain

- What happens at the edges?
- What properties does this operator have?
- What can and can't this operator do?
### Handling Edges - Padding Modes

Possible "padding modes":

**Zeros:**

<p>| | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td></td>
<td></td>
<td>90</td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
<td></td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>10</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>10</td>
<td>20</td>
<td>30</td>
<td>0</td>
<td>20</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>10</td>
<td>30</td>
<td>40</td>
<td>30</td>
<td>20</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

**Replicate:**

<p>| | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>5</td>
<td>10</td>
<td>20</td>
<td>30</td>
<td>40</td>
<td>30</td>
<td>20</td>
<td>10</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>10</td>
<td>20</td>
<td>10</td>
<td>40</td>
<td>30</td>
<td>20</td>
<td>10</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

**Reflect:**

<p>| | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>10</td>
<td>20</td>
<td>30</td>
<td>30</td>
<td>20</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>10</td>
<td>20</td>
<td>20</td>
<td>0</td>
<td>10</td>
<td>0</td>
<td>20</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>
Handling Edges - Output Sizes

"Valid" (3x3)
Handling Edges - Output Sizes

"Valid" (5x5)

<p>| | | | | | | | | | |</p>
<table>
<thead>
<tr>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
<th></th>
</tr>
</thead>
<tbody>
<tr>
<td>90</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>10</td>
<td>20</td>
<td>20</td>
<td>20</td>
<td>10</td>
<td>40</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>10</td>
<td>20</td>
<td>30</td>
<td>0</td>
<td>20</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>10</td>
<td>0</td>
<td>30</td>
<td>40</td>
<td>30</td>
<td>20</td>
<td>10</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>10</td>
<td>20</td>
<td>30</td>
<td>40</td>
<td>30</td>
<td>20</td>
<td>10</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>10</td>
<td>20</td>
<td>10</td>
<td>40</td>
<td>30</td>
<td>20</td>
<td>10</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>10</td>
<td>20</td>
<td>30</td>
<td>30</td>
<td>20</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>10</td>
<td>20</td>
<td>20</td>
<td>0</td>
<td>10</td>
<td>0</td>
<td>20</td>
<td>0</td>
</tr>
<tr>
<td>0</td>
<td>0</td>
<td>0</td>
<td>10</td>
<td>10</td>
<td>10</td>
<td>0</td>
<td>0</td>
<td>0</td>
<td>0</td>
</tr>
</tbody>
</table>

input image

output size
Handling Edges - Output Sizes

"Same"
Handling Edges - Output Sizes

"Full" (3x3)
Handling Edges - Output Sizes

"Full" (5x5)

output size

input image