Vision Systems

Lecture 3

Part 1

Linear Filtering, Correlation and Convolution

Review

- Different types of image processing operations: *point*, *local* and *global*

Review

- Different types of image processing operations: *point*, *local* and *global*
- **Question:** How do you perform histogram equalization?

Review

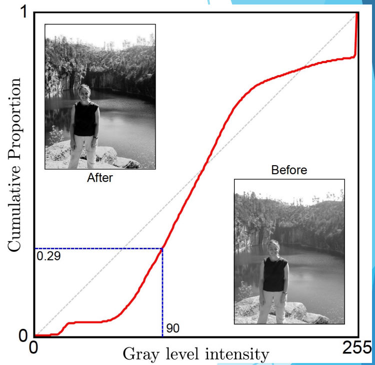
- Different types of image processing operations: *point*, *local* and *global*
- **Question:** How do you perform histogram equalization?
- Let I be the image with $M \times N$ pixels in total; I_{MAX} be the maximum image intensity value (255); $h(I)$ be the image histogram

Review

- Different types of image processing operations: *point*, *local* and *global*
- **Question:** How do you perform histogram equalization?
- Let I be the image with $M \times N$ pixels in total; I_{MAX} be the maximum image intensity value (255); $h(I)$ be the image histogram
- Integrate $h(I)$ to obtain the cumulative distribution $c(I)$, whose each value $c_k = \frac{1}{M \times N} \sum_{l=1}^{k} h(l)$

Review

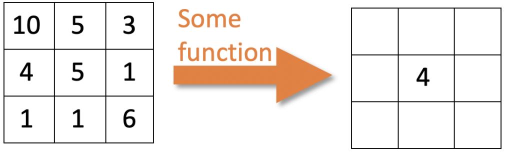
- Different types of image processing operations: *point*, *local* and *global*
- **Question:** How do you perform histogram equalization?
- Let I be the image with $M \times N$ pixels in total; I_{MAX} be the maximum image intensity value (255); $h(I)$ be the image histogram
- Integrate $h(I)$ to obtain the cumulative distribution $c(I)$, whose each value $c_k = \frac{1}{M \times N} \sum_{l=1}^{k} h(l)$
- The transformed image $\hat{I}(i,j) = I_{MAX} \times c_{p_{ij}}$
- E.g., in figure, value 90 will be mapped to $I_{MAX} \times 0.29$ (rounded off)



Credit: Simon Prince, Computer Vision: Models, Learning, and Inference, Cambridge University Press

Image Filters: Linear Filter

- **Image Filter:** Modify image pixels based on some function of a local neighbourhood of each pixel



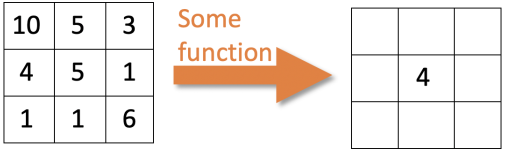
10	5	3
4	5	1
1	1	6

	4	

What's the function?

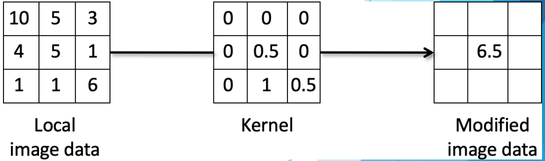
Image Filters: Linear Filter

- **Image Filter:** Modify image pixels based on some function of a local neighbourhood of each pixel



What's the function?

- **Linear Filter:** Replace each pixel by linear combination (a weighted sum) of neighbours
- Linear combination called kernel, mask or filter



Linear Filter: Cross-Correlation

- Given a kernel of size $(2k+1) \times (2k+1)$:

  Correlation defined as:

$$G(i,j) = \underbrace{\frac{1}{(2k+1)^2}}_{\substack{\text{Uniform weight} \\ \text{to each pixel}}} \underbrace{\sum_{u=-k}^{k} \sum_{v=-k}^{k} I(i+u,j+v)}_{\substack{\text{Loop over pixels in considered} \\ \text{neighbourhood around } I(i,j)}}$$

Linear Filter: Cross-Correlation

Given a kernel of size $(2k+1)\times(2k+1)$:

- **Correlation** defined as:

$$G(i,j) = \underbrace{\frac{1}{(2k+1)^2}}_{\text{Uniform weight to each pixel}} \underbrace{\sum_{u=-k}^{k}\sum_{v=-k}^{k} I(i+u,j+v)}_{\text{Loop over pixels in considered neighbourhood around } I(i,j)}$$

- **Cross-correlation** defined as:

$$G(i,j) = \sum_{u=-k}^{k}\sum_{v=-k}^{k} \underbrace{H(u,v)}_{\text{Non-uniform weights}} I(i+u,j+v)$$

Linear Filter: Cross-Correlation

Given a kernel of size $(2k+1) \times (2k+1)$:

- **Correlation** defined as:

$$G(i,j) = \underbrace{\frac{1}{(2k+1)^2}}_{\text{Uniform weight to each pixel}} \underbrace{\sum_{u=-k}^{k} \sum_{v=-k}^{k} I(i+u, j+v)}_{\text{Loop over pixels in considered neighbourhood around } I(i,j)}$$

- **Cross-correlation** defined as:

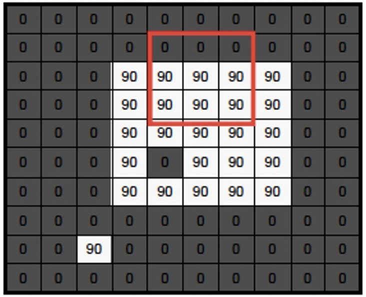
$$G(i,j) = \sum_{u=-k}^{k} \sum_{v=-k}^{k} \underbrace{H(u,v)}_{\text{Non-uniform weights}} I(i+u, j+v)$$

- Cross-correlation denoted by $G = H \otimes I$
- Can be viewed as "dot product" between local neighbourhood and kernel for each pixel
- Entries of kernel or mask $H(u,v)$ called filter co-efficients

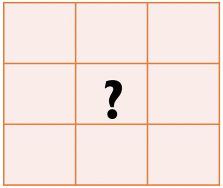
Moving Average: Linear Filter

What values belong in the kernel H for the moving average example we saw earlier?

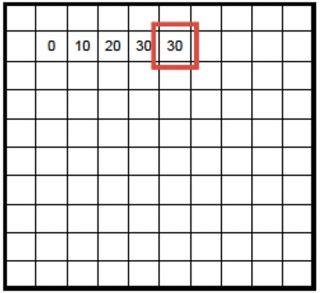
$$I(i,j) \otimes H(u,v) = G(i,j)$$



0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0



	?	



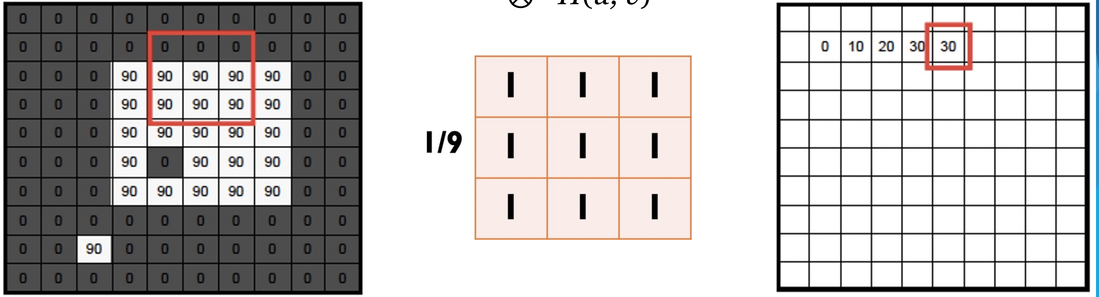
	0	10	20	30	30				

Credit: K Grauman, Univ of Texas Austin

Moving Average: Linear Filter

What values belong in the kernel H for the moving average example we saw earlier?

$$I(i,j) \otimes H(u,v) = G(i,j)$$



0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

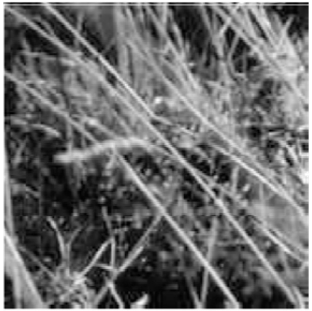
$$\frac{1}{9}\begin{bmatrix}1 & 1 & 1\\ 1 & 1 & 1\\ 1 & 1 & 1\end{bmatrix}$$

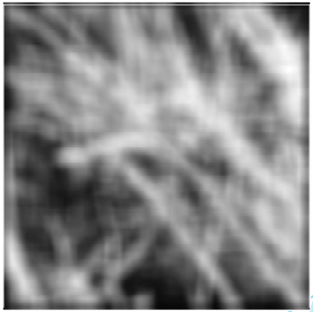
	0	10	20	30	30				

Credit: K Grauman, Univ of Texas Austin

Moving Average Filter: Example

Effect of moving average filter (also known as **box filter**):



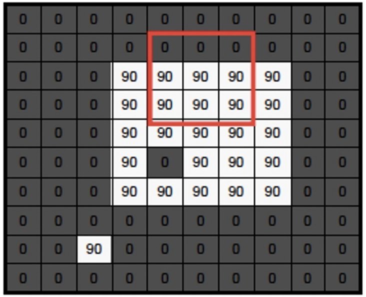


Credit: K Grauman, Univ of Texas Austin

Gaussian Average Filter

What if we want the nearest neighbouring pixels to have the most influence on the output?

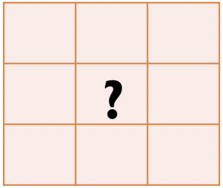
$I(i, j)$



0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

Credit: K Grauman, Univ of Texas Austin

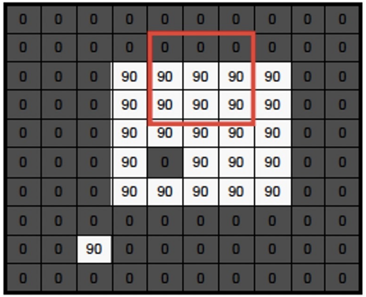
$\otimes \quad H(u, v)$



Gaussian Average Filter

What if we want nearest neighbouring pixels to have the most influence on the output?

$I(i, j)$



0	0	0	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	90	0	90	90	90	0	0
0	0	0	90	90	90	90	90	0	0
0	0	0	0	0	0	0	0	0	0
0	0	90	0	0	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0

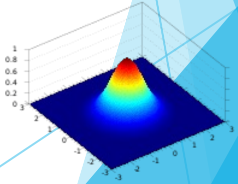
Credit: K Grauman, Univ of Texas Austin

$\otimes \quad H(u, v)$

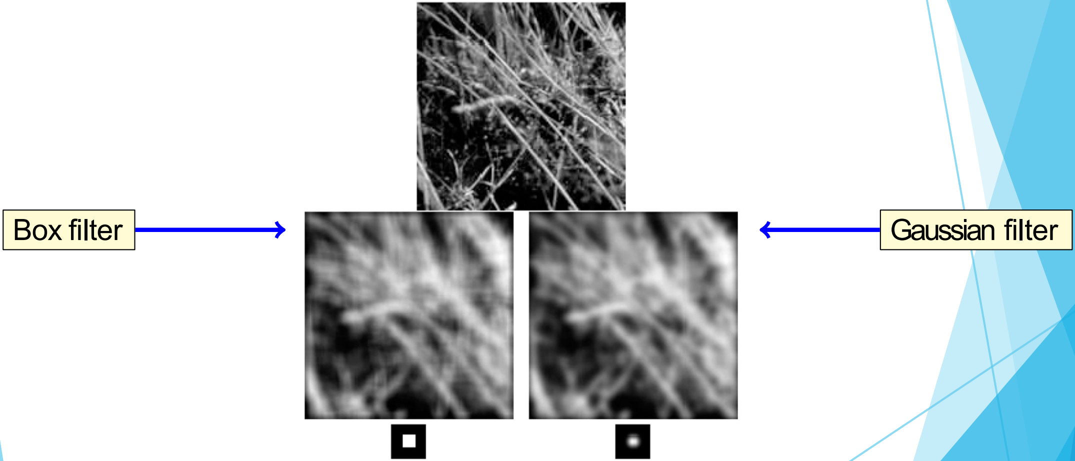
$$\frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

This kernel is an approximation of a 2D Gaussian function:

$$h(u, v) = \frac{1}{2\pi\sigma^2} \exp^{\frac{-u^2+v^2}{\sigma^2}}$$



Averaging Filters: A Comparison

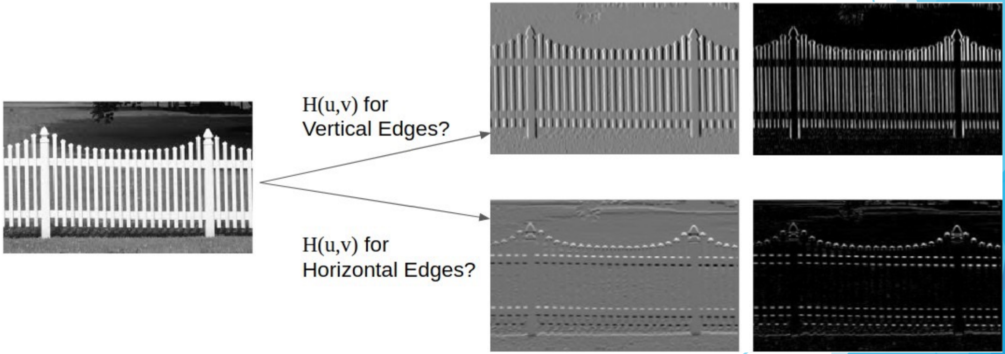


Credit: K Grauman, Univ of Texas Austin

Other Filters: The Edge Filter

What should H look like to find the edges in a given image?

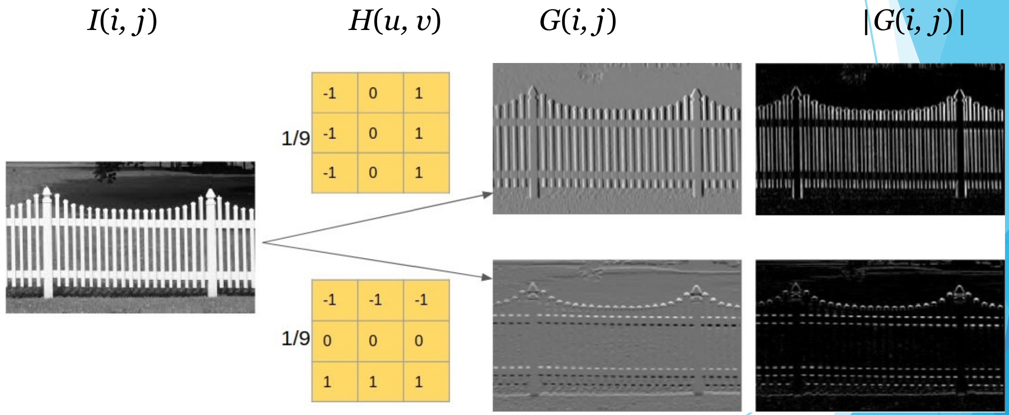
$I(i, j)$ $H(u, v)$ $G(i, j)$ $|G(i, j)|$



Credit: KiwiWorker

Other Filters: The Edge Filter

What should H look like to find the edges in a given image?



$$\frac{1}{9}\begin{bmatrix} -1 & 0 & 1 \\ -1 & 0 & 1 \\ -1 & 0 & 1 \end{bmatrix}$$

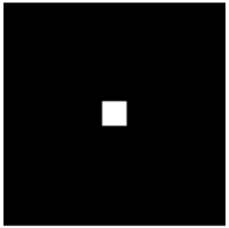
$$\frac{1}{9}\begin{bmatrix} -1 & -1 & -1 \\ 0 & 0 & 0 \\ 1 & 1 & 1 \end{bmatrix}$$

Credit: KiwiWorker

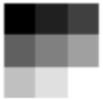
Beyond Correlation

What is the result of filtering the impulse signal (image) I with the arbitrary kernel H?

$I(i, j)$



$\otimes \quad H(u, v)$



a	b	c
d	e	f
g	h	i

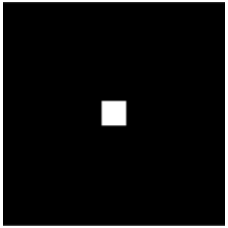
$G(i, j)$

?

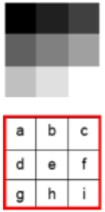
Beyond Correlation

What is the result of filtering the impulse signal (image) F with the arbitrary kernel H?

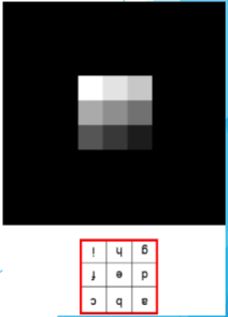
$I(i, j)$



$\otimes \quad H(u, v)$



$G(i, j)$



Introducing Convolution

Given a kernel of size $(2k+1) \times (2k+1)$:

- **Convolution** defined as:

$$G(i,j) = \sum_{u=-k}^{k} \sum_{v=-k}^{k} H(u,v) I(i-u, j-v)$$

Introducing Convolution

Given a kernel of size $(2k+1)\times(2k+1)$:

- **Convolution** defined as:

$$G(i,j) = \sum_{u=-k}^{k}\sum_{v=-k}^{k} H(u,v)I(i-u,j-v)$$

- Equivalent to flip the filter in both directions (bottom to top, right to left) and apply cross-correlation
- Denoted by $G = H * I$

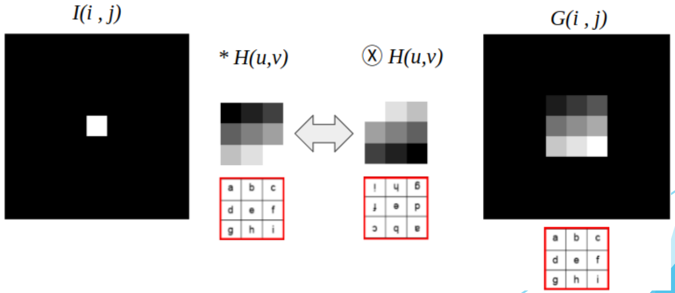
Introducing Convolution

Given a kernel of size $(2k+1)\times(2k+1)$:

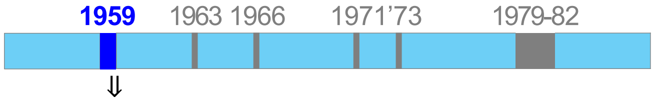
- **Convolution** defined as:

$$G(i,j) = \sum_{u=-k}^{k}\sum_{v=-k}^{k} H(u,v)\,I(i-u, j-v)$$

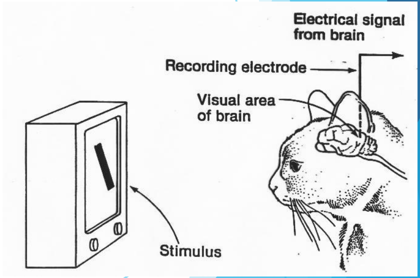
- Equivalent to flip the filter in both directions (bottom to top, right to left) and apply cross-correlation
- Denoted by $G = H * I$



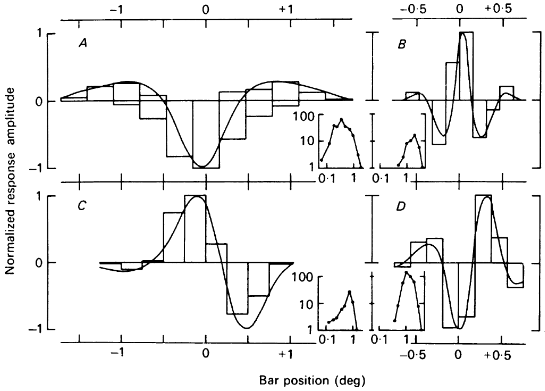
Recall: Early History



- David Hubel and Torsten Wiesel publish their work "*Receptive fields of single neurons in the cat's striate cortex*"
- Placed electrodes into primary visual cortex area of an anesthetized cat's brain
- Showed that simple and complex neurons exist, and that visual processing starts with simple structures such as oriented edges



Linear Summation in the Visual Cortex



Simple cells perform linear spatial summation over their receptive fields[1]

[1]*Movshon, Thompson and Tolhurst, Spatial Summation in the Receptive Fields of Simple Cells in the Cat's Striate Cortex, JP 1978*

Linear Shift-Invariant Operators

- Both correlation and convolution are Linear Shift-Invariant operators, which obey:
  - Linearity (or Superposition principle):

$$I \circ (h_0 + h_1) = I \circ h_0 + I \circ h_1$$

  - Shift-Invariance: shifting (or translating) a signal commutes with applying the operator

$$g(i,j) = h(i+k, j+l) \iff (f \circ g)(i,j) = (f \circ h)(i+k, j+l)$$

- Equivalent to saying that the effect of the operator is the same everywhere. Why do we need this in computer vision?

Source: Raquel Urtasun, Univ of Toronto

Linear Shift-Invariant Operators

- Both correlation and convolution are Linear Shift-Invariant operators, which obey:
  - Linearity (or Superposition principle):

$$I \circ (h_0 + h_1) = I \circ h_0 + I \circ h_1$$

  - Shift-Invariance: shifting (or translating) a signal commutes with applying the operator

$$g(i,j) = h(i+k, j+l) \iff (f \circ g)(i,j) = (f \circ h)(i+k, j+l)$$

- Equivalent to saying that the effect of the operator is the same everywhere. Why do we need this in computer vision?

Source: Raquel Urtasun, Univ of Toronto

Properties of Convolution

- **Commutative:** $a * b = b * a$
  - Conceptually no difference between filter and signal
- **Associative:** $a * (b * c) = (a * b) * c$
  - We often apply filters one after the other: (((a * b1) * b2) * b3)
  - This is equivalent to applying one cumulative filter: a * (b1 * b2 * b3)
- **Distributive over addition:** $a * (b + c) = (a * b) + (a * c)$
  - We can combine the responses of a signal over two or more filters by combining the filters
- **Scalars factor out:** $ka * b = a * kb = k(a * b)$
- **Identity:** Unit impulse $e = [\ldots, 0, 0, 1, 0, 0, \ldots]$, $a * e = a$

Properties of Convolution

- **Commutative:** $a * b = b * a$
  - Conceptually no difference between filter and signal
- **Associative:** $a * (b * c) = (a * b) * c$
  - We often apply filters one after the other: (((a * b1) * b2) * b3)
  - This is equivalent to applying one cumulative filter: a * (b1 * b2 * b3)
- **Distributive over addition:** $a * (b + c) = (a * b) + (a * c)$
  - We can combine the responses of a signal over two or more filters by combining the filters
- **Scalars factor out:** $ka * b = a * kb = k(a * b)$
- **Identity:** Unit impulse $e = [\ldots, 0, 0, 1, 0, 0, \ldots]$, $a * e = a$

Properties of Convolution

- **Commutative**: $a * b = b * a$
  - Conceptually no difference between filter and signal
- **Associative**: $a * (b * c) = (a * b) * c$
  - We often apply filters one after the other: (((a * b1) * b2) * b3)
  - This is equivalent to applying one cumulative filter: a * (b1 * b2 * b3)
- **Distributive over addition**: $a * (b + c) = (a * b) + (a * c)$
  - We can combine the responses of a signal over two or more filters by combining the filters
- **Scalars factor out**: $ka * b = a * kb = k(a * b)$
- **Identity**: Unit impulse $e = [\ldots, 0, 0, 1, 0, 0, \ldots]$, $a * e = a$

Properties of Convolution

- **Commutative**: $a * b = b * a$
  - Conceptually no difference between filter and signal
- **Associative**: $a * (b * c) = (a * b) * c$
  - We often apply filters one after the other: (((a * b1) * b2) * b3)
  - This is equivalent to applying one cumulative filter: a * (b1 * b2 * b3)
- **Distributive over addition**: $a * (b + c) = (a * b) + (a * c)$
  - We can combine the responses of a signal over two or more filters by combining the filters
- **Scalars factor out**: $ka * b = a * kb = k(a * b)$
- **Identity:** Unit impulse $e = [\ldots, 0, 0, 1, 0, 0, \ldots], a * e = a$

Separability

Convolution operator requires k^2 operations per pixel, where k is the width (and height) of a convolution kernel.

Can be costly. How can we reduce cost?

Separability

- Convolution operator requires k^2 operations per pixel, where k is the width (and height) of a convolution kernel.
- Can be costly. How can we reduce cost?
- For certain filters, can be sped up by performing a 1D horizontal convolution followed by a 1D vertical convolution, requiring $2k$ operations $\implies$ convolution kernel is **separable**.

$$K = vh^T$$

where v, h are 1D kernels, and K is the 2D kernel

Separability

- Convolution operator requires k^2 operations per pixel, where k is the width (and height) of a convolution kernel.
- Can be costly. How can we reduce cost?
- For certain filters, can be sped up by performing a 1D horizontal convolution followed by a 1D vertical convolution, requiring $2k$ operations $\Longrightarrow$ convolution kernel is **separable**.
- $$K = vh^T$$

  where v, h are 1D kernels, and K is the 2D kernel

Example 1:

$\frac{1}{16}$

1	2	1
2	4	2
1	2	1

Separability

- Convolution operator requires k^2 operations per pixel, where k is the width (and height) of a convolution kernel.
- Can be costly. How can we reduce cost?
- For certain filters, can be sped up by performing a 1D horizontal convolution followed by a 1D vertical convolution, requiring $2k$ operations $\Longrightarrow$ convolution kernel is **separable**.
- $$K = vh^T$$

  where v, h are 1D kernels, and K is the 2D kernel

Example 1:

$$\frac{1}{16}\begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline 2 & 4 & 2 \\ \hline 1 & 2 & 1 \\ \hline \end{array} \Longrightarrow v = h = \frac{1}{4}\begin{array}{|c|}\hline 1 \\ \hline 2 \\ \hline 1 \\ \hline \end{array}$$

Separability

- Convolution operator requires k^2 operations per pixel, where k is the width (and height) of a convolution kernel.
- Can be costly. How can we reduce cost?
- For certain filters, can be sped up by performing a 1D horizontal convolution followed by a 1D vertical convolution, requiring $2k$ operations $\implies$ convolution kernel is **separable**.
- $$K = vh^T$$

  where v, h are 1D kernels, and K is the 2D kernel

Example 1:

$$\frac{1}{16}\begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline 2 & 4 & 2 \\ \hline 1 & 2 & 1 \\ \hline\end{array} \implies v = h = \frac{1}{4}\begin{array}{|c|}\hline 1 \\ \hline 2 \\ \hline 1 \\ \hline\end{array}$$

Example 2:

$$\frac{1}{8}\begin{array}{|c|c|c|}\hline -1 & 0 & 1 \\ \hline -2 & 0 & 2 \\ \hline -1 & 0 & 1 \\ \hline\end{array}$$

Separability

- Convolution operator requires k^2 operations per pixel, where k is the width (and height) of a convolution kernel.
- Can be costly. How can we reduce cost?
- For certain filters, can be sped up by performing a 1D horizontal convolution followed by a 1D vertical convolution, requiring $2k$ operations $\Longrightarrow$ convolution kernel is **separable**.
- $$K = vh^T$$

  where v, h are 1D kernels, and K is the 2D kernel

Example 1:

$$\frac{1}{16}\begin{array}{|c|c|c|}\hline 1 & 2 & 1 \\ \hline 2 & 4 & 2 \\ \hline 1 & 2 & 1 \\ \hline \end{array} \Longrightarrow v = h = \frac{1}{4}\begin{array}{|c|}\hline 1 \\ \hline 2 \\ \hline 1 \\ \hline \end{array}$$

Example 2:

$$\frac{1}{8}\begin{array}{|c|c|c|}\hline -1 & 0 & 1 \\ \hline -2 & 0 & 2 \\ \hline -1 & 0 & 1 \\ \hline \end{array} \Longrightarrow v = \frac{1}{4}\begin{array}{|c|}\hline -1 \\ \hline -2 \\ \hline -1 \\ \hline \end{array} \ \& \ h = \frac{1}{2}\begin{array}{|c|}\hline 1 \\ \hline 0 \\ \hline 1 \\ \hline \end{array}$$

Separable Convolution

How can we tell if a given kernel K is separable?

Separable Convolution

How can we tell if a given kernel K is separable?

- Visual inspection

Separable Convolution

How can we tell if a given kernel K is separable?

- Visual inspection
- Analytically, look at the Singular Value Decomposition (SVD), and if only one singular value is non-zero, then it is separable.

$$K = U\Sigma V^T = \sum_i \sigma_i u_i v_i^T$$

where $\Sigma = \text{diag}(\sigma_i)$

$\sqrt{\sigma_1}u_1$ and $\sqrt{\sigma_1}v_1$ are the vertical and horizontal kernels

Source: Raquel Urtasun, Univ of Toronto

Practical Issues

Ideal size for the filter?

Practical Issues

Ideal size for the filter?

The bigger the mask:

- more neighbours contribute
- smaller noise variance of output
- bigger noise spread
- more blurring
- more expensive to compute

Practical Issues

Ideal size for the filter?

The bigger the mask:

- more neighbours contribute
- smaller noise variance of output
- bigger noise spread
- more blurring
- more expensive to compute

What about the boundaries? Do we lose information?

Practical Issues

Ideal size for the filter?

The bigger the mask

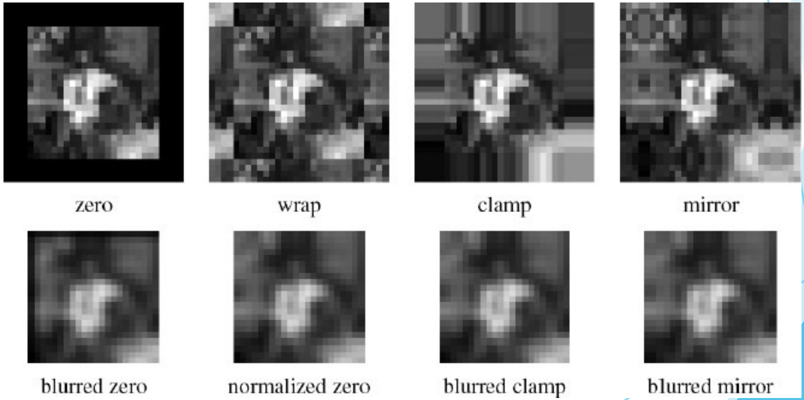
- more neighbours contribute
- smaller noise variance of output
- bigger noise spread
- more blurring
- more expensive to compute

What about the boundaries? Do we lose information?

- Without padding, we lose out on information at the boundaries.
- We can use a variety of strategies such as zero padding, wrapping around, copy the edge

Practical Issues

Different padding strategies:



Questions to Think About

- Do we then need (cross)-correlation at all?
- Are all filters always linear?

Homework

Readings

Chapter 3 (§3.1-3.3), Szeliski, *Computer Vision: Algorithms and Applications*, 2010 draft

Chapter 7 (§7.1-7.2), Forsyth and Ponce, *Computer Vision: A Modern Approach*, 2003 edition

References

- Richard Szeliski. *Computer Vision: Algorithms and Applications*. Texts in Computer Science. London: Springer-Verlag, 2011.
- David Forsyth and Jean Ponce. *Computer Vision: A Modern Approach*. 2 edition. Boston: Pearson Education India, 2015.
- *Hays, James, CS 6476 - Computer Vision (Fall 2018)*. URL:
- https://www.cc.gatech.edu/~hays/compvision/ (visited on 04/28/2020).
- *Hoiem, Derek, CS 543 - Computer Vision (Spring 2011)*. URL:
- https://courses.engr.illinois.edu/cs543/sp2017/ (visited on 04/25/2020).
- *Oliva, Aude, 6.819/6.869 - Advances in Computer Vision (Fall 2015)*. URL:
- http://6.869.csail.mit.edu/fa15/ (visited on 04/28/2020).