EE 417 Introduction to Computer Vision

Point, Local and Global Operators

Instructor: Associate Prof. Mehmet Keskinöz

Communication Theory &Technologies (CTT) Group,
Electronics Engineering Program,
Computer Science and Engineering,
Cyber-Security Program
Faculty of Engineering and Natural Sciences

Email: keskinoz@sabanciuniv.edu

Sabancı Universitesi

Image Processing

Mapping of an image into another image

Often as *image preprocessing*, i.e. prior to image analysis

Input

Output

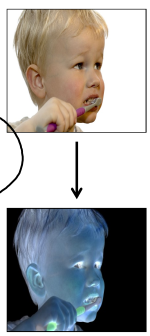
edge →

Here: Input image mapped into an image showing high-frequency information only

---

What types of image transformations can we do?

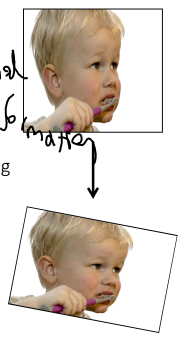
Filtering



changes pixel *values*

Spatial transformation

Warping

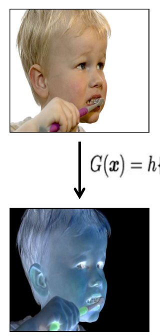


changes pixel *locations*

What types of image transformations can we do?

Filtering

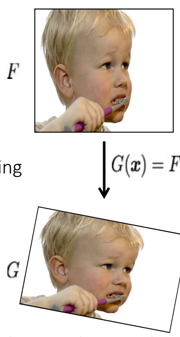
$G(x) = h\{F(x)\}$



changes *range* of image function

Warping

$G(x) = F(h\{x\})$

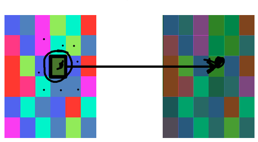


changes *domain* of image function

What types of image filtering can we do?

$I_o(x_p,y_p)=I_n(x_p,y_p)$

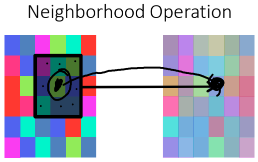
Point Operation



point processing

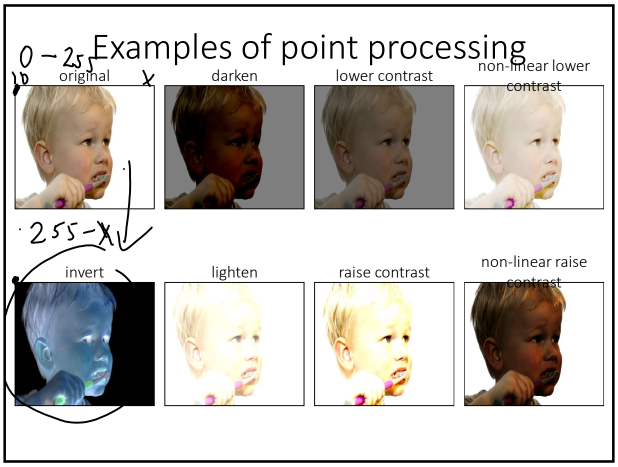
Local Operation

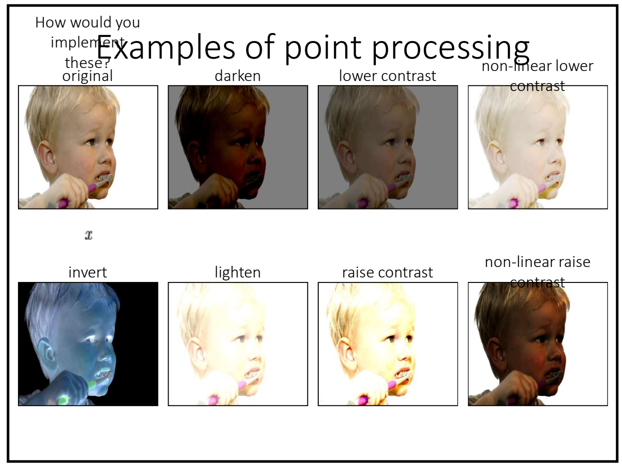
Neighborhood Operation

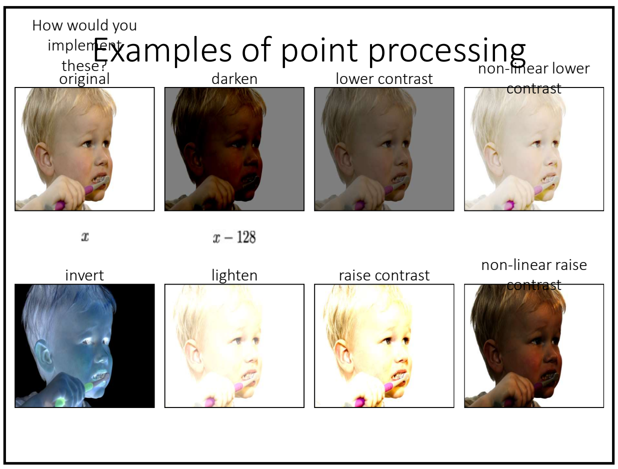


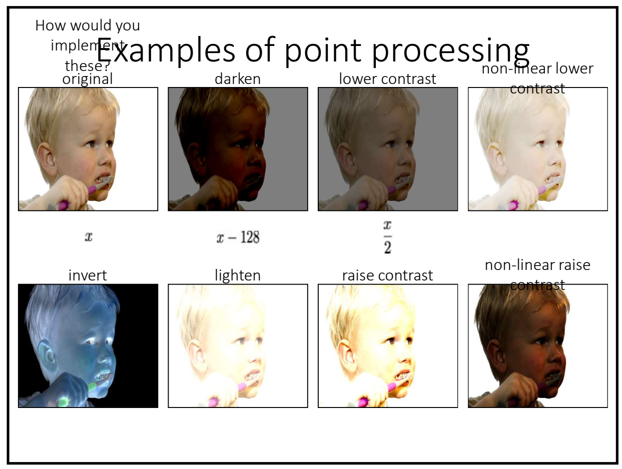
"filtering"

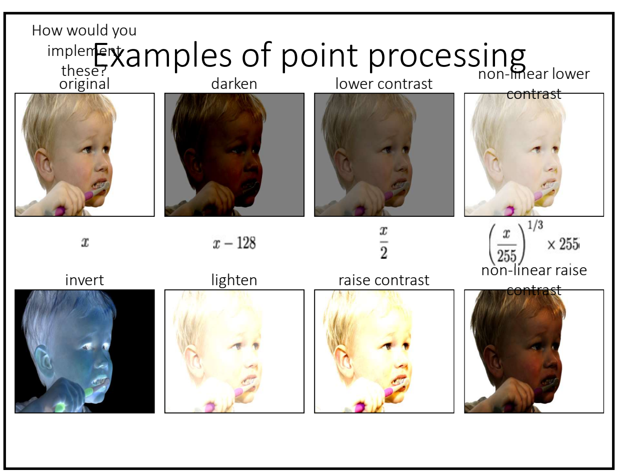
Point processing

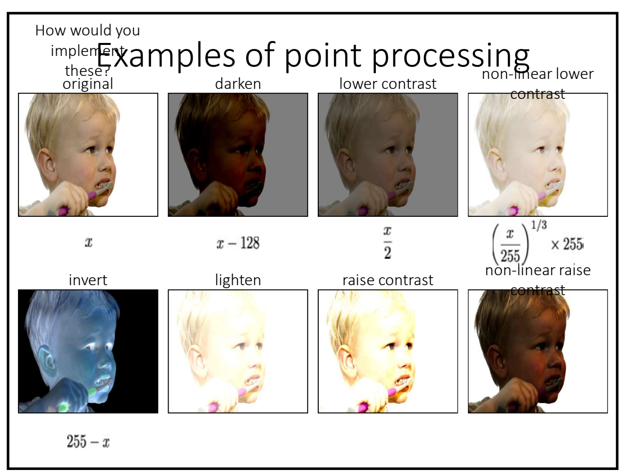

0 – 255
Examples of point processing
original
darken
lower contrast
non-linear lower contrast
x
255 – x
invert
lighten
raise contrast
non-linear raise contrast


How would you implement these?
Examples of point processing
original
darken
lower contrast
non-linear lower contrast
x
invert
lighten
raise contrast
non-linear raise contrast


How would you implement these?
Examples of point processing
original
darken
lower contrast
non-linear lower contrast
x
$x - 128$
invert
lighten
raise contrast
non-linear raise contrast


How would you implement these?
Examples of point processing
original
darken
lower contrast
non-linear lower contrast
x
$x - 128$
$\frac{x}{2}$
invert
lighten
raise contrast
non-linear raise contrast


How would you implement these?
Examples of point processing
original
darken
lower contrast
non-linear lower contrast
x
$x - 128$
$\frac{x}{2}$
$\left(\frac{x}{255}\right)^{1/3} \times 255$
invert
lighten
raise contrast
non-linear raise contrast

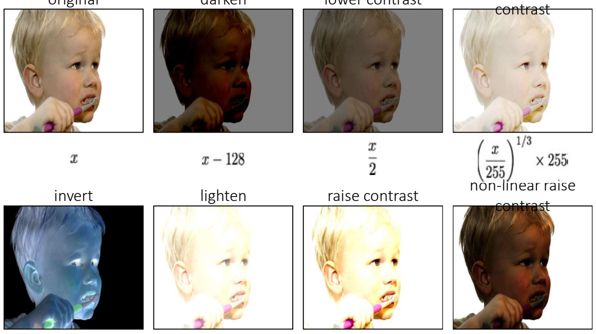

How would you implement these?
Examples of point processing
original
darken
lower contrast
non-linear lower contrast
x
$x - 128$
$\frac{x}{2}$
$\left(\frac{x}{255}\right)^{1/3} \times 255$
invert
lighten
raise contrast
non-linear raise contrast
$255 - x$

Examples of point processing

How would you implement these?

original | darken | lower contrast | non-linear lower contrast

invert | lighten | raise contrast | non-linear raise contrast



x | $x - 128$ | $\frac{x}{2}$ | $\left(\frac{x}{255}\right)^{1/3} \times 255$

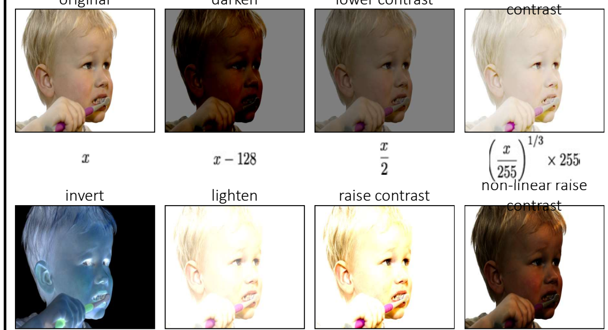
$255 - x$ | $x + 128$

Examples of point processing

How would you implement these?

original | darken | lower contrast | non-linear lower contrast

invert | lighten | raise contrast | non-linear raise contrast



x | $x - 128$ | $\frac{x}{2}$ | $\left(\frac{x}{255}\right)^{1/3} \times 255$

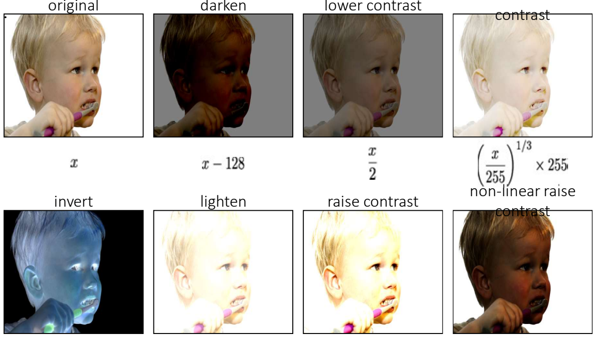
$255 - x$ | $x + 128$ | $x \times 2$

Examples of point processing

How would you implement these?

original | darken | lower contrast | non-linear lower contrast

invert | lighten | raise contrast | non-linear raise contrast



x | $x - 128$ | $\frac{x}{2}$ | $\left(\frac{x}{255}\right)^{1/3} \times 255$

$255 - x$ | $x + 128$ | $x \times 2$ | $\left(\frac{x}{255}\right)^{2} \times 255$

Many other types of point processing

camera output | image after stylistic tonemapping

[Bae et al., SIGGRAPH 2006]

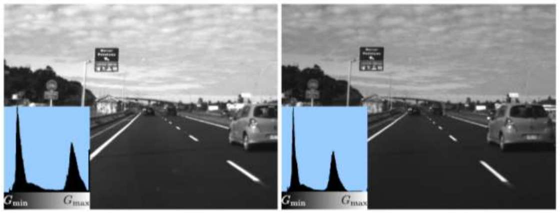
Many other types of point processing

Example: Noise Removal

Noise is unwanted data

For example, *challenges of outdoor image recording:*
Difficulties with lighting, motion blur, or sudden changes in scenes

Noise removal is one of the reasons for mapping a given image into a "better" image illustrating three examples of noise for vision-based driver-assistance



- Pair of stereo images taken time-synchronized but of different brightness; see the shown gray-level histograms
- **Goal**: Transform images such that resulting images are "as taken at uniform illumination"

- Blurring caused by rain and obstruction of view by a wiper
- **Goal:** Reduce impacts of rain and wiper when analysing this image (e.g.when calculating distance to car in front of the ego-vehicle); we could try to start with image sharpening for removing the blur

- Gaussian noise in a scene recorded at night
- **Goal:** Remove the Gaussian noise, enhance the image contrast

Point Operators

Time-efficient methods for image processing

Gradation Functions

Transform image I into image I_{new} of the same size:

Map value u at pixel location p in I by *gradation function* g onto value

$$v = g(u)$$

u
← p=(x,y)
v
← p (x,y)

at the same pixel location p in I_{new}

Change only depends on value u at location p, thus a *point operator* defined by gradation function g

Which Gradation Function?

Which function g was used to map the left into the right image?

X   255 − X

A historic photo of events in April 1960 at Korea University, Seoul, leading to an uprising against the autocratic ruler Syngman Rhee of South Korea.

$v = g(u) = 255 - u$

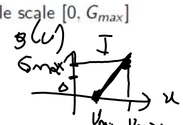
Linear Scaling

Not a histogram transform, but another example for a point operator

Image I has positive histogram values in interval $[u_{min}, u_{max}]$

Goal: Map values used in I linearly onto the whole scale $[0, G_{max}]$

$$u_{min} = \min\{I(x,y) : (x,y) \in \Omega\}$$
$$u_{max} = \max\{I(x,y) : (x,y) \in \Omega\}$$



Definition of gradation function:

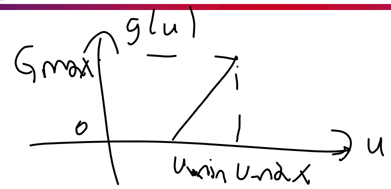
$$a = -u_{min} \quad \text{and} \quad b = \frac{G_{max}}{u_{max} - u_{min}}$$
$$g(u) = b(u + a)$$

$g(u) = ?$

Results:

Pixels having value u_{min} in I now have value 0 in I_{new}

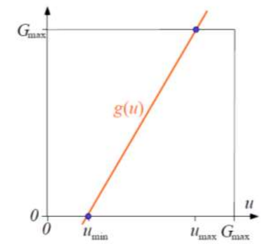
Pixels having value u_{max} in I now have value G_{max} in I_{new}



$$g(u) = \frac{G_{max}}{U_{max} - U_{min}} (u - U_{min})$$

Linear Scaling: Graph

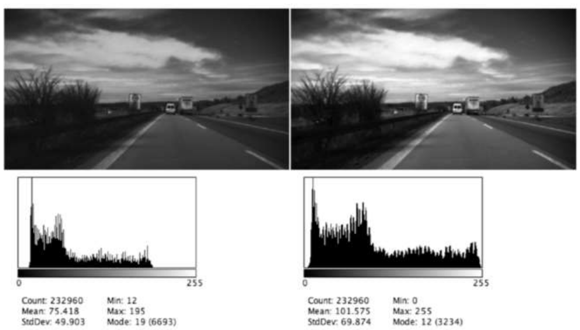
Graph of gradation function for linear scaling



Defined by being incident with points $(u_{\min}, 0)$ and $(u_{\max}, G_{\max})$: Derive the formula as given on the previous page

Linear Scaling: Example

Left: $u_{\min} = 12$ and $u_{\max} = 195$ *Right:* $u_{\min} = 0$ and $u_{\max} = 255$



Histogram Transforms

Histogram transforms are examples of point operators

Histogram Transform $I \longrightarrow g(u) \longrightarrow I_{output}$

If the goal is that I_{new} satisfies some properties defined in terms of its histogram then we speak about a *histogram transform*

Example: Histogram Equalization

Goal: Transform I such that all gray levels appear equally often in I_{new}

Ideal goal: For all $u \in \{0, 1, \ldots, G_{max}\}$

$$H_{I_{new}}(u) = \frac{N_{cols} N_{rows}}{G_{max} + 1} = \frac{|\Omega|}{G_{max} + 1} = const$$

Unfortunately: Not possible in general due to having digital images (and not continuous functions)

The Raging Bull Example

Left: Input image `RagingBull` with histogram
Right: The same image after histogram equalization

Example: A binary image would remain binary after histogram equalisation

Histogram Equalization: How To Do

Given:

$N_{cols} \times N_{rows}$ scalar image I with absolute frequencies $H_I(u)$, $0 \leq u \leq G_{max}$

Transform I into I_{new} of the same size by applying a gradation function

Use the following gradation function g $\in [0,1]$

$$g(u) = c_I(u) \cdot G_{max}$$

where c_I is the relative cumulative frequency function

Recall: What exactly is the definition of c_I?

Figure on previous page illustrates this histogram-equalization transform

In the probability theory, if X is a continuous Random variable with pdf $f_X(x)$ and cumulative distributive function $F_X(x) = P(X \le x)$

$y = F_X(x)$

uniform

Conditional Scaling

Not a histogram transform, but another example for a point operator

Goal: Map image J into image J_{new} such that J_{new} has the same mean and variance as image I

Definition of gradation function:

$$a = \mu_J \cdot \frac{\sigma_I}{\sigma_J} - \mu_I \quad \text{and} \quad b = \frac{\sigma_J}{\sigma_I}$$

$$g(u) = b(u + a)$$

Calculation of J_{new}: Replace value u at p in J by $v = g(u)$

Result: $\mu_{J_{new}} = \mu_I$ and $\sigma_{J_{new}} = \sigma_I$

Might solve noise issue (1) on Page 6 **if** brightness difference between both images is uniformly defined

Conditional Scaling: Example

Image I | Image J

Conditional scaling of J | Equalized J

Histogram Equalization

An image histogram is a graphic representation of the frequency counts of all allowable pixel intensities. Figure 2.1 shows a sample of an image with a skewed histogram.

4	1	3	2
3	1	1	1
0	1	5	2
1	1	2	2

input image

4

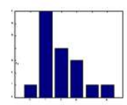


Figure 2.1 A sample image with a skewed histogram (poor intensity distribution)

u	H(u)	C(u)
0	1	1
1	7	8
2	4	12
3	2	14
4	1	15
5	1	16

$c(u) = \frac{C(u)}{16}$

relative cumulative frequency function

1. Compute the histogram of the image
2. Calculate the normalized sum of histogram
3. Transform the input image to an output image

u $\quad v = g(u) = c(u) \cdot G_{max}$

intensity	sum	normalized sum
0	1	1/16*5=0.31255 ≅ 0
1	8	2.5 = 8/16·5 ≅ 3
2	12	3.75 ≅ 4
3	14	4.375 → 4
4	15	4.6875 → 5
5	16	5.0 → 5

5	3	4	4
4	3	3	3
0	3	5	4
3	3	4	4

output image

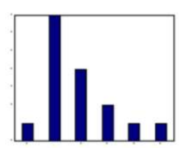


Figure 2.2 A histogram-equalized sample with histogram and normalized sum

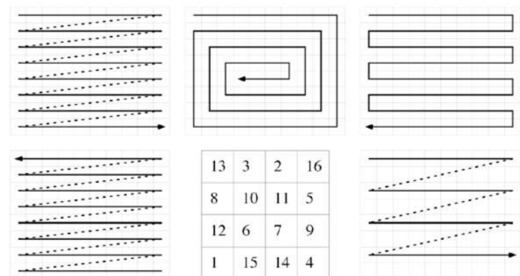
General Concept of a Local Operator

Map input image I into a new image J:

1. Given: $N_{cols} \times N_{rows}$ image I
2. Sliding window W_p of size $(2k+1) \times (2k+1)$
3. Reference point p at the center of the window
4. Reference point p moves into all possible pixel locations of I
5. At these locations we perform a *local operation*
6. Result of the operation defines new value at p

Moves of Reference Point p of Window W_p through I

Defined by a scan; upper left shows default scan:



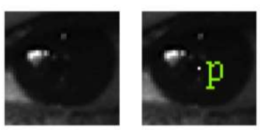
Two Examples: Local Mean and Maximum

Example: Local operation is mean $J(p) = \mu_{W_p(I)}$; for $p = (x, y)$ we have

$$\mu_{W_p(I)} = \frac{1}{(2k+1)^2} \cdot \sum_{i=-k}^{+k} \sum_{j=-k}^{+k} I(x+i, y+j)$$

Example: Local operation is the maximum; for $p = (x, y)$ we have

$$J(p) = \max\{I(x+i, y+j) : -k \le i \le k \wedge -k \le j \le k\}$$



Maximum in 41×41 input window (left) copied into p on the right
What happens if input window (i.e. p) moves one pixel to the right?

Border-pixel Strategy

Needed for windows centered at p and not completely contained in I

There is no general rule:

Define meaningful operation depending on the given local operation

Top, left: Original image. *Right*: Local maximum for $k = 3$

Linear Operators and Convolution

Linear local operators are a class of local operators

A linear local operator is defined by a *convolution* of I with a *filter kernel*

Reference point $p = (x, y)$, weights $w_{i,j} \in \mathbb{R}$, scaling factor $S > 0$:

$$J(p) = I * W(p) = \frac{1}{S} \sum_{i=-k}^{+k} \sum_{j=-k}^{+k} w_{i,j} \cdot I(x+i, y+j)$$

Array of $(2k+1) \times (2k+1)$ weights and S define the filter kernel

Example: All $w_{i,j} > 0$ and $S = \sum_{i=-k}^{+k} \sum_{j=-k}^{+k} w_{i,j}$

Common Way for Filter Kernel Representation

$$\begin{array}{|c|c|c|}\hline w_{11} & w_{12} & w_{13} \\ \hline w_{21} & w_{22} & w_{23} \\ \hline w_{31} & w_{32} & w_{33} \\ \hline \end{array} /S \qquad \begin{array}{|c|c|c|}\hline 1 & 0 & -1 \\ \hline 2 & 0 & -2 \\ \hline 1 & 0 & -1 \\ \hline \end{array} /1 \qquad \begin{array}{|c|c|c|}\hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array} /9$$

Left: General representation for a 3×3 filter kernel
Middle: Filter kernel for approximating derivation in x-direction, $S = 1$
Right: Filter kernel of a 3×3 box filter, $S = 9$

Box Filter

Local mean on Page 23 is a linear local operator known as *box filter*

All weights equal to 1, and $S = (2k+1)^2$ is the sum of all weights

Local Operators: Summary

1. Operations are limited to windows, typically of size $(2k+1) \times (2k+1)$
2. Window moves through the image following a selected scan order
3. Operation in the window is the same at all locations
4. Pixels close to the border of the image: case by case decisions
5. Results are either
   1. replacing values at reference points in input image I (*sequential local operator*); new values propagate like a "wave" over original values; windows of the local operator contain original data as well as already-processed pixel values
   2. written into a second array, leaving the original image unaltered this way (*parallel local operator*); this kind of a local operator can be implemented on parallel hardware

Point, Local, or Global Operator

Cases for $(2k+1) \times (2k+1)$ window size:

(1) $k = 0$, i.e. window is just a single pixel: A *point operator*

Example: Operators defined by a gradation function

(2) k such that whole image is covered by the window: A *global operator*

(3) If not global operator then a local operator (including point operators)

Examples: Local mean (box filter), local maximum

Linear shift-invariant image filtering

Linear shift-invariant image filtering

- Replace each pixel by a *linear* combination of its neighbors (and possibly itself).
- The combination is determined by the filter's *kernel*.
- The same kernel is *shifted* to all pixel locations so that all pixels use the same linear combination of their neighbors.

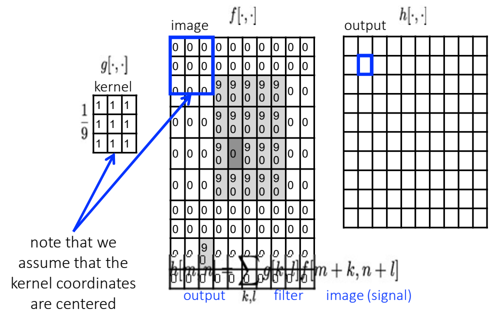
Example: the box filter

- also known as the 2D rect (not rekt) filter
- also known as the square mean filter

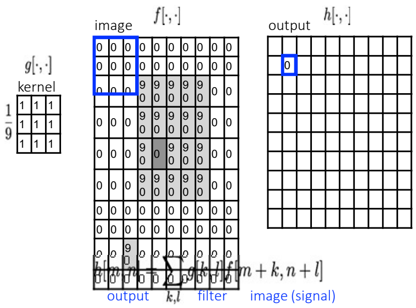
kern
el $g[\cdot,\cdot] = \frac{1}{9}\begin{array}{|c|c|c|}\hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline 1 & 1 & 1 \\ \hline \end{array}$

- replaces pixel with local average
- has smoothing (blurring) effect

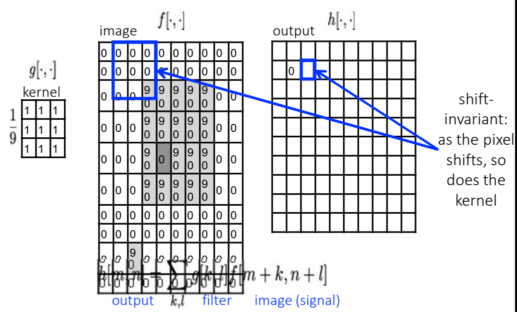
Let’s run the box filter



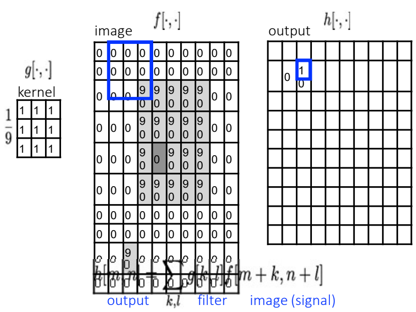
Let’s run the box filter



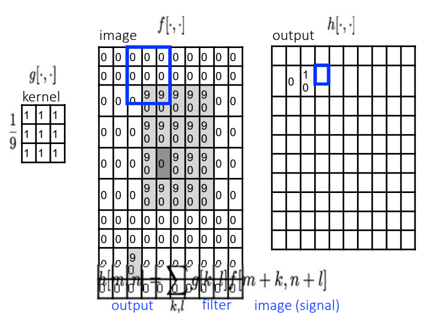
Let’s run the box filter



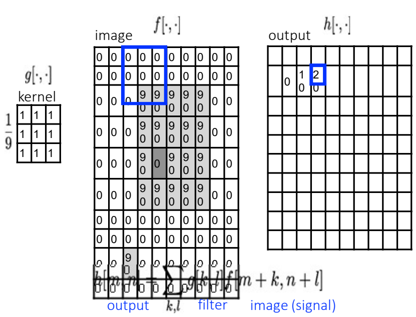
Let’s run the box filter



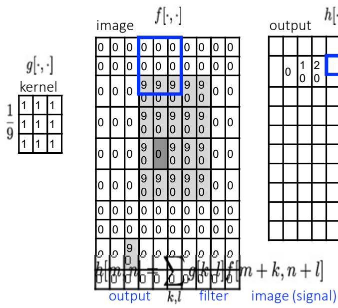
Let’s run the box filter



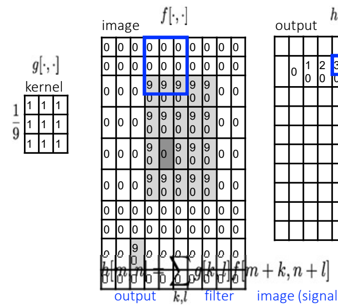
Let’s run the box filter



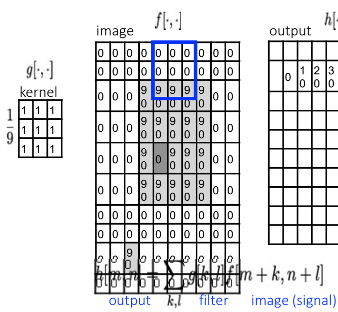
Let’s run the box filter



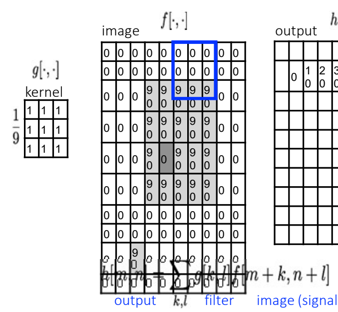
Let’s run the box filter



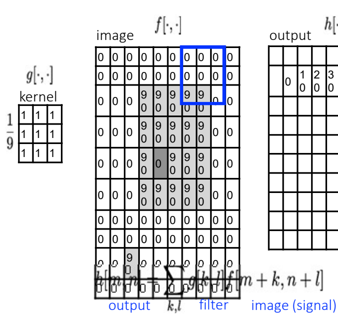
Let’s run the box filter



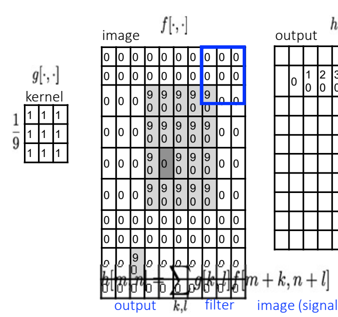
Let’s run the box filter



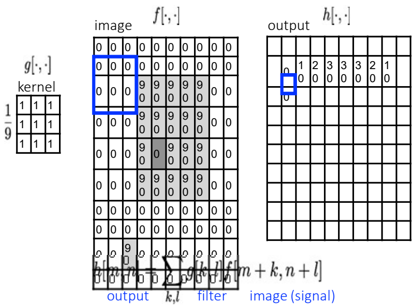
Let’s run the box filter



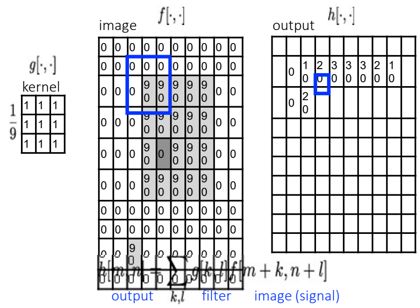
Let’s run the box filter



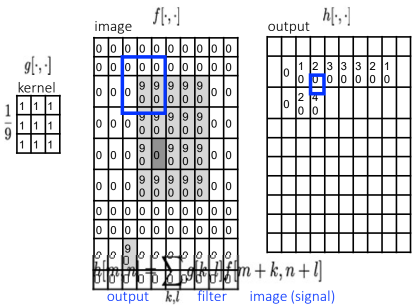
Let’s run the box filter



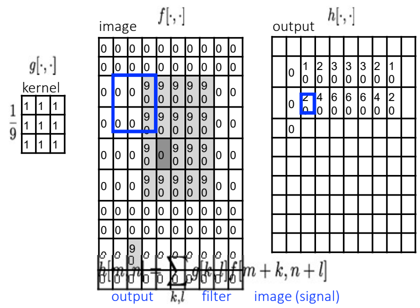
Let’s run the box filter



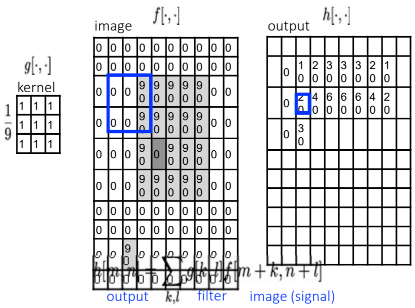
Let’s run the box filter



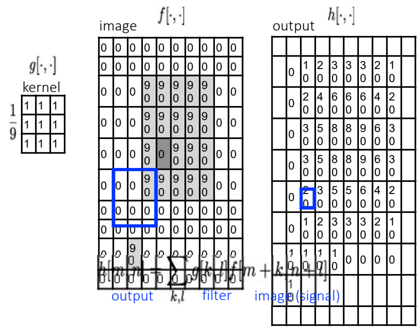
Let’s run the box filter



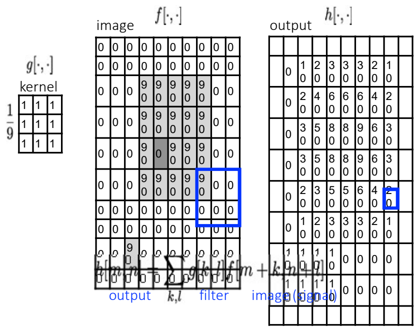
Let’s run the box filter



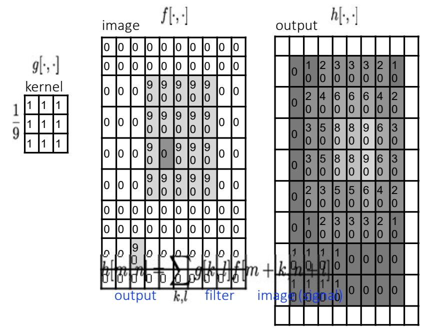
Let’s run the box filter



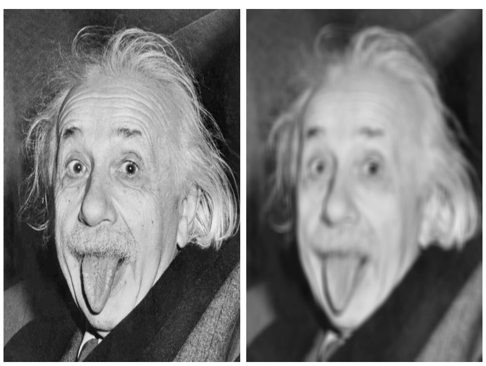
Let's run the box filter



… and the result is



Some more realistic examples



Some more realistic examples

Some more realistic examples

Convolution

Convolution for 1D continuous signals

Definition of filtering as convolution:

notice the flip

$$(f * g)(x) = \int_{-\infty}^{\infty} f(y) g(x - y) dy$$

filtered signal; filter; input signal

Convolution for 1D continuous signals

Definition of filtering as convolution:

notice the flip

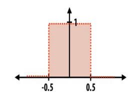
$$(f * g)(x) = \int_{-\infty}^{\infty} f(y) g(x - y) dy$$

filtered signal; filter; input signal

Consider the box filter example:

1D continuous box filter

$$f(x) = \begin{cases} 1 & |x| \leq 0.5 \\ 0 & otherwise \end{cases}$$



filtering output is a blurred version of g

$$(f * g)(x) = \int_{-0.5}^{0.5} g(x - y) dy$$

Convolution for 2D discrete signals

Definition of filtering as convolution:

notice the flip

$$(f * g)(x, y) = \sum_{i,j=-\infty}^{\infty} f(i, j) I(x - i, y - j)$$

filtered image; filter; input image

Convolution for 2D discrete signals

Definition of filtering as convolution:

notice the flip

$$(f * g)(x, y) = \sum_{i,j=-\infty}^{\infty} f(i, j) I(x - i, y - j)$$

filtered image; filter; input image

If the filt $f(i,j)$ is non-zero c $-1 \leq i, j \leq 1$, then

$$(f * g)(x, y) = \sum_{i,j=-1}^{1} f(i, j) I(x - i, y - j)$$

The kernel we saw earlier is the 3x3 matrix representation of $f(i,j)$.

Convolution vs correlation

Definition of discrete 2D convolution:

notice the flip

$$(f * g)(x, y) = \sum_{i,j=-\infty}^{\infty} f(i, j) I(x - i, y - j)$$

Definition of discrete 2D correlation:

notice the lack of a flip

$$(f * g)(x, y) = \sum_{i,j=-\infty}^{\infty} f(i, j) I(x + i, y + j)$$

- Most of the time won't matter, because our kernels will be symmetric.
- Will be important when we discuss frequency-domain filtering (lectures 5-6).

Separable filters

A 2D filter is separable if it can be written as the product of a "column" and a "row".

example: box filter

1	1	1
1	1	1
1	1	1

=

1
1
1

column

*

1	1	1

row

What is the rank of this filter matrix?

Separable filters

A 2D filter is separable if it can be written as the product of a "column" and a "row".

example: box filter

$$\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix} = \begin{bmatrix}1\\1\\1\end{bmatrix} * \begin{bmatrix}1&1&1\end{bmatrix}$$

column row

Why is this important?

Separable filters

A 2D filter is separable if it can be written as the product of a "column" and a "row".

example: box filter

$$\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix} = \begin{bmatrix}1\\1\\1\end{bmatrix} * \begin{bmatrix}1&1&1\end{bmatrix}$$

column row

2D convolution with a separable filter is equivalent to two 1D convolutions (with the "column" and "row" filters).

Separable filters

A 2D filter is separable if it can be written as the product of a "column" and a "row".

example: box filter

$$\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix} = \begin{bmatrix}1\\1\\1\end{bmatrix} * \begin{bmatrix}1&1&1\end{bmatrix}$$

column row

2D convolution with a separable filter is equivalent to two 1D convolutions (with the "column" and "row" filters).

If the image has M x M pixels and the filter kernel has size N x N:
- What is the cost of convolution with a non-separable filter?

Separable filters

A 2D filter is separable if it can be written as the product of a "column" and a "row".

example: box filter

$$\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix} = \begin{bmatrix}1\\1\\1\end{bmatrix} * \begin{bmatrix}1&1&1\end{bmatrix}$$

column row

2D convolution with a separable filter is equivalent to two 1D convolutions (with the "column" and "row" filters).

If the image has M x M pixels and the filter kernel has size N x N:
- What is the cost of convolution with a non-separable filter?
  $M^2 \times N^2$
- What is the cost of convolution with a separable filter?

Separable filters

A 2D filter is separable if it can be written as the product of a "column" and a "row".

example: box filter

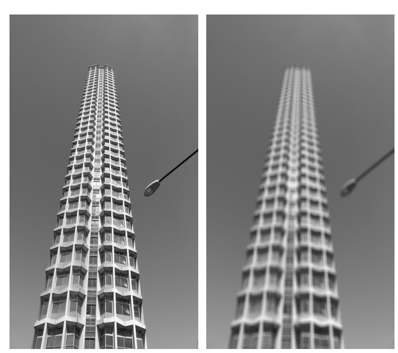
$$\begin{bmatrix}1&1&1\\1&1&1\\1&1&1\end{bmatrix} = \begin{bmatrix}1\\1\\1\end{bmatrix} * \begin{bmatrix}1&1&1\end{bmatrix}$$

column row

2D convolution with a separable filter is equivalent to two 1D convolutions (with the "column" and "row" filters).

If the image has M x M pixels and the filter kernel has size N x N:
- What is the cost of convolution with a non-separable filter?
  $M^2 \times N^2$
- What is the cost of convolution with a separable filter?
  $2 \times N \times M^2$

A few more filters



original 3x3 box filter

do you see any problems in this image?