This Week's Topics

7. The Convolution Operation

8. A Unifying View

Template matching	⇔ Derivatives via WLS fitting
Image smoothing	⇔ Template matching
Image interpolation	⇔ Convolution w/ continuous smoothing function
Image differentiation	⇔ Convolution w/ derivative of a smoothing function
Image Laplacian	⇔ Difference of two Gaussian-smoothed versions of an image

Review

Sliding Window

Image Filtering

- Find gradients
- Edge detection
- Smoothing
- Convolution

Template Matching

- (Root) Sum/Mean Squared distance – next week
- Cross Correlation
- Normalized Cross Correlation – next week

Topic 7:

The Convolution Operation

You will see a lot of formulas in today's lecture.

Don't panic because either you have seen them before or you will see them again and again later on…

This lecture is purely connecting different concepts together.

Understand before memorizing them!

Filtering: signal smooth example

"signal", "image"

"filter", "mask",
"kernel", "template",
"impulse response"

$$\sum_{i=0}^{400} g_i f_i$$

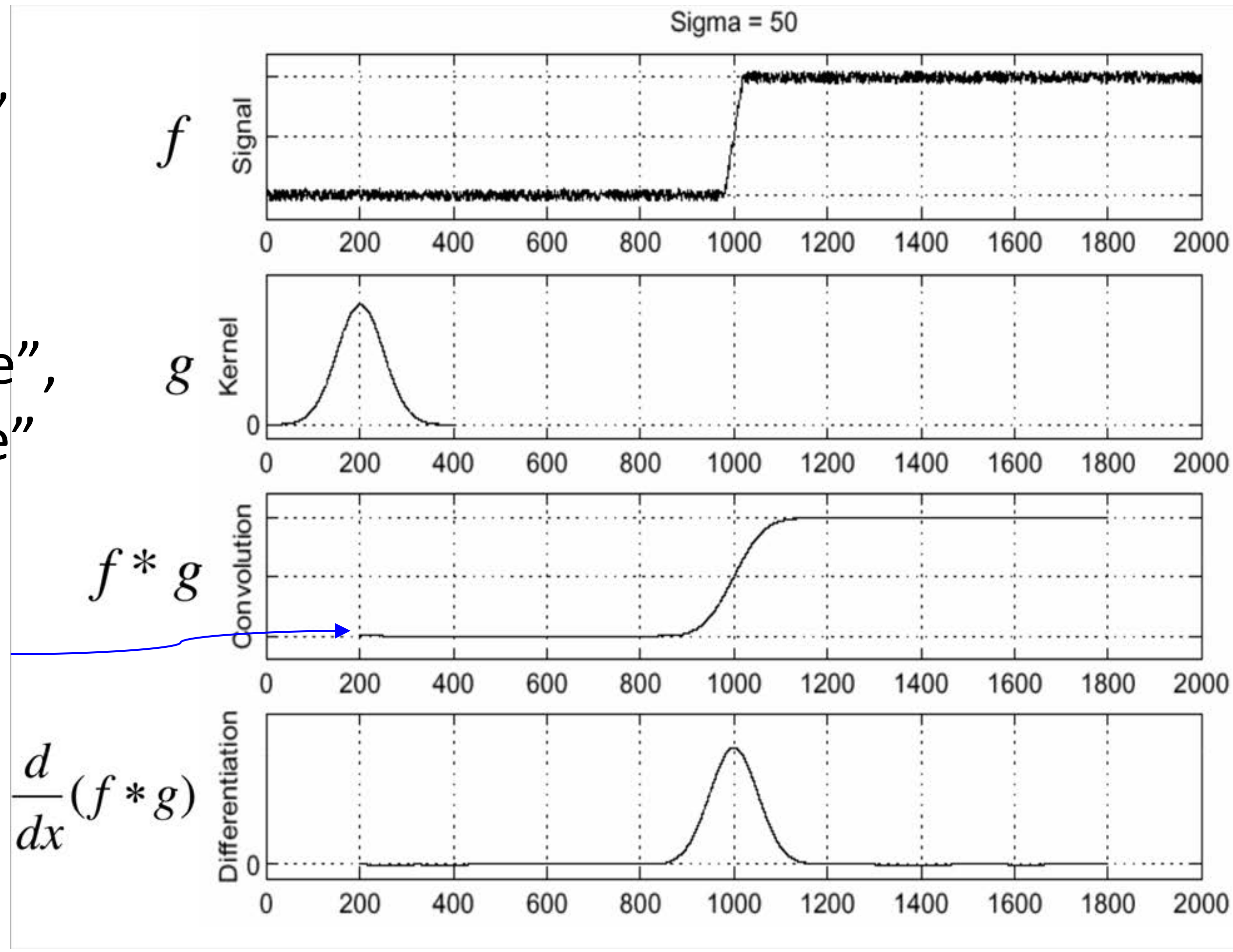


Image Smoothing Using Averaging Masks

Original Image

Image Smoothing Using Averaging Masks

Result of filtering with 3x3 Mask

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

Averaging mask a.k.a. mean filter, box filter

Image Smoothing Using Averaging Masks

Result of filtering with 5x5 Mask

1/25	1/25	1/25	1/25	1/25
1/25	1/25	1/25	1/25	1/25
1/25	1/25	1/25	1/25	1/25
1/25	1/25	1/25	1/25	1/25
1/25	1/25	1/25	1/25	1/25

Image Smoothing Using Averaging Masks

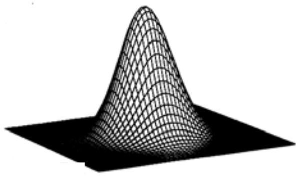
Result of filtering with 15x15 Mask

15x15 array of elements equal to 1/225

Averaging Mask vs. Gaussian Mask

box filter

gaussian



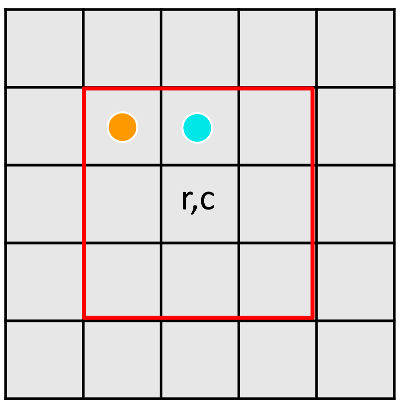
Slide by Robert Collins

Cross-Correlation vs. Convolution (discrete)

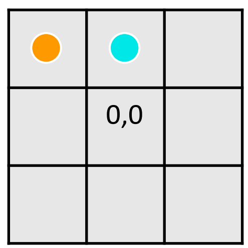
Cross-correlation (sliding inner-product)

$\forall$r, c:

$$\mathrm{CC}(I, T, r, c) = \sum_{a=-w}^{w} \sum_{b=-w}^{w} I(r+a, c+b)\, T(a,b)$$



Image



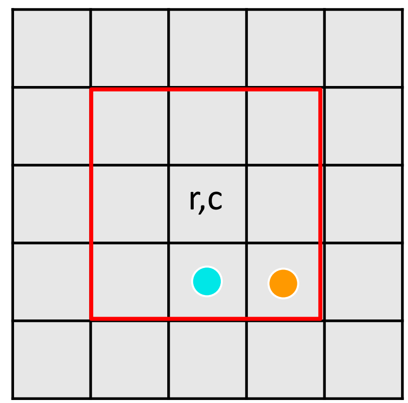
Template

Lots of useful properties.

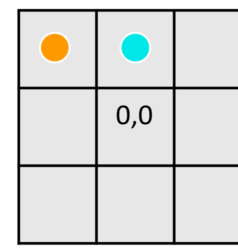
Convolution

$\forall$r, c:

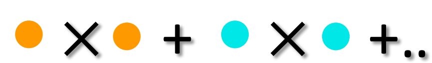
$$I * T\,(r, c) = \sum_{a=-w}^{w} \sum_{b=-w}^{w} I(r-a, c-b)\, T(a,b)$$



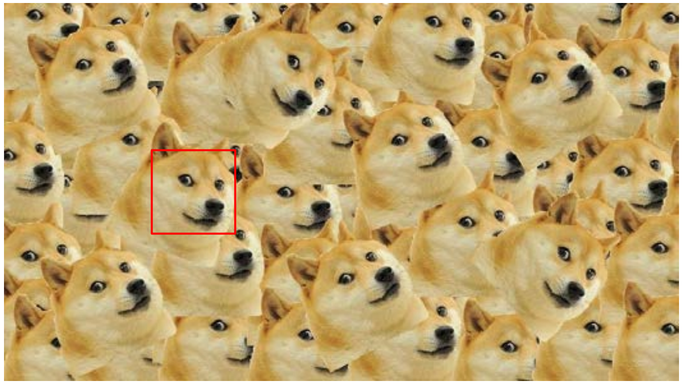
Image



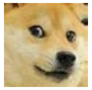
Template



Template Matching (2D)

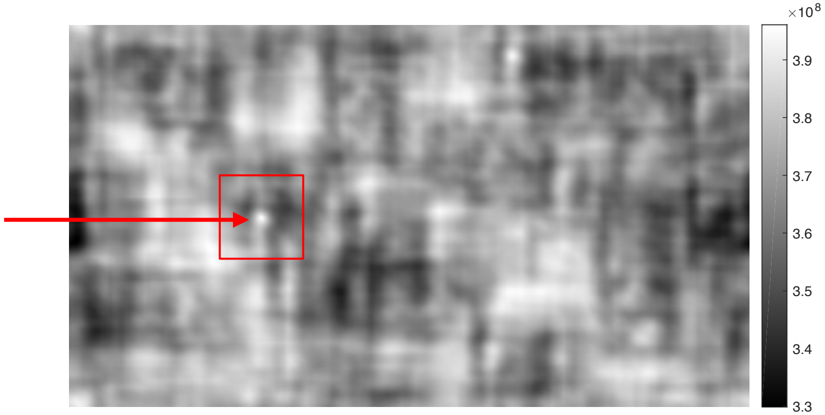


Image



Template

Maximum location



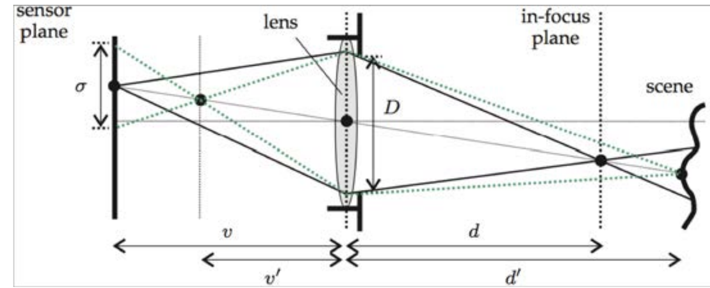
Cross-Correlation (similarity value)

Convolution in imaging

Motion blur

- Blur kernel shape ≈ path of motion

Out of focus blur



- Blur kernel is different for objects at different distance

Aperture shape

Solar eclipse 2017 by Yani

Convolution result of the eclipse (signal) and holes formed by leaves (kernel)

Cross Correlation Template Matching

Using Sliding Window Algorithm

$\Updownarrow$

Matrix Multiplication

$\Updownarrow$

The convolution Operation

Template Matching (1D)

"Sliding window" algorithm for template matching with template T

- Define a "pixel window" centered at pixel (w,r)
- Compute cross-correlation of T with patch centered at (w,r)
- "Slide" window one pixel over, so that it is centered at pixel (w+1,r)
- Repeat 1-4 until window reaches right image border

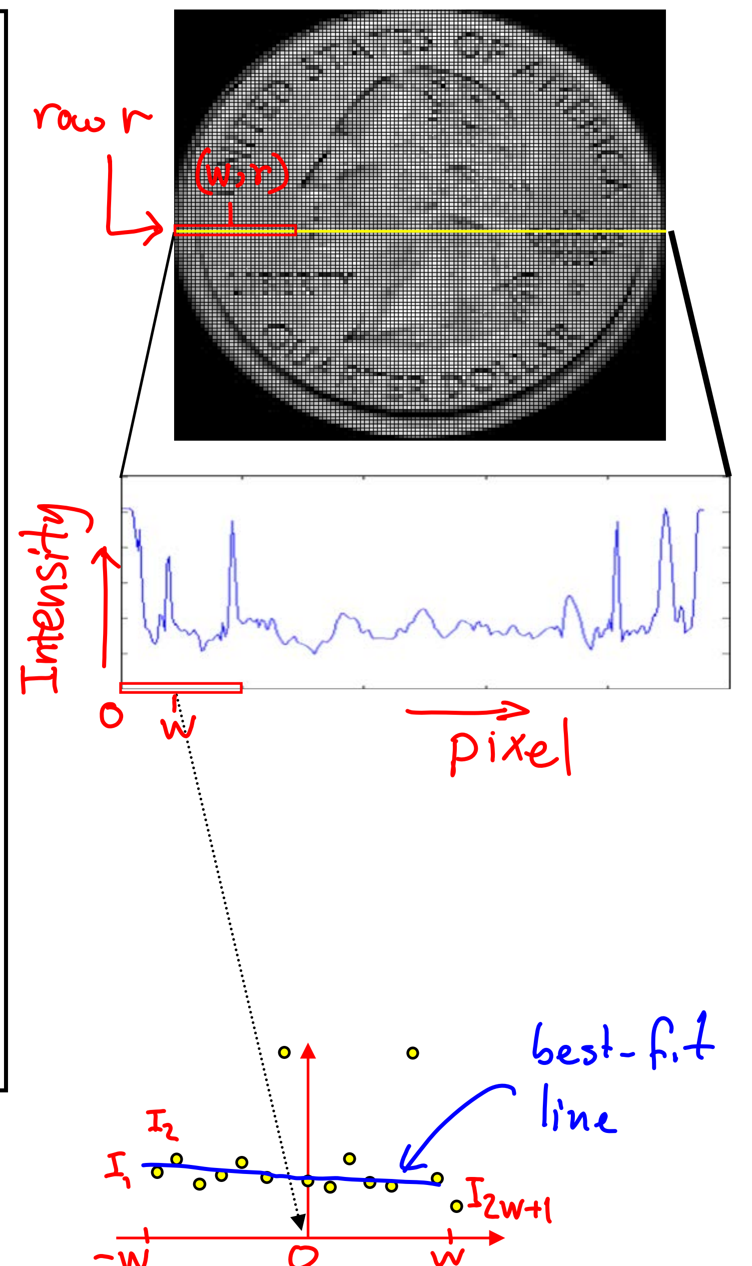


Image Cross Correlation ⇔ Matrix Multiplication

I : row r of the image (M pixels)

T : template (2w+1 pixels)

$[T_{-w}\ T_{-w+1}\ \cdots\ T_0\ \cdots\ T_w]$

cross-correlation at pixel w

$$[T_{-w}\ T_{-w+1}\ \cdots T_0 \cdots T_w\ 0 \cdots 0] \begin{bmatrix} I_0\\ I_1\\ I_2\\ \vdots\\ I_w\\ \vdots\\ I_{2w}\\ I_{2w+1}\\ \vdots\\ I_{M-1}\end{bmatrix}$$

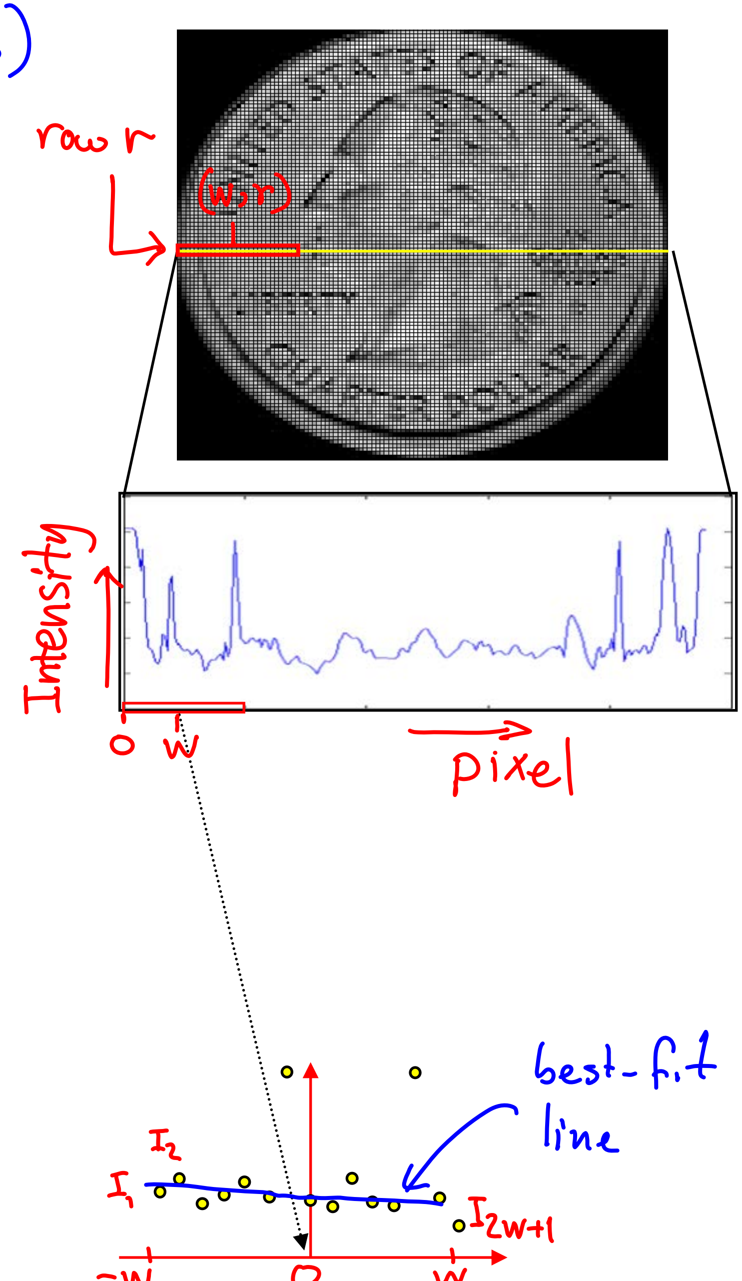


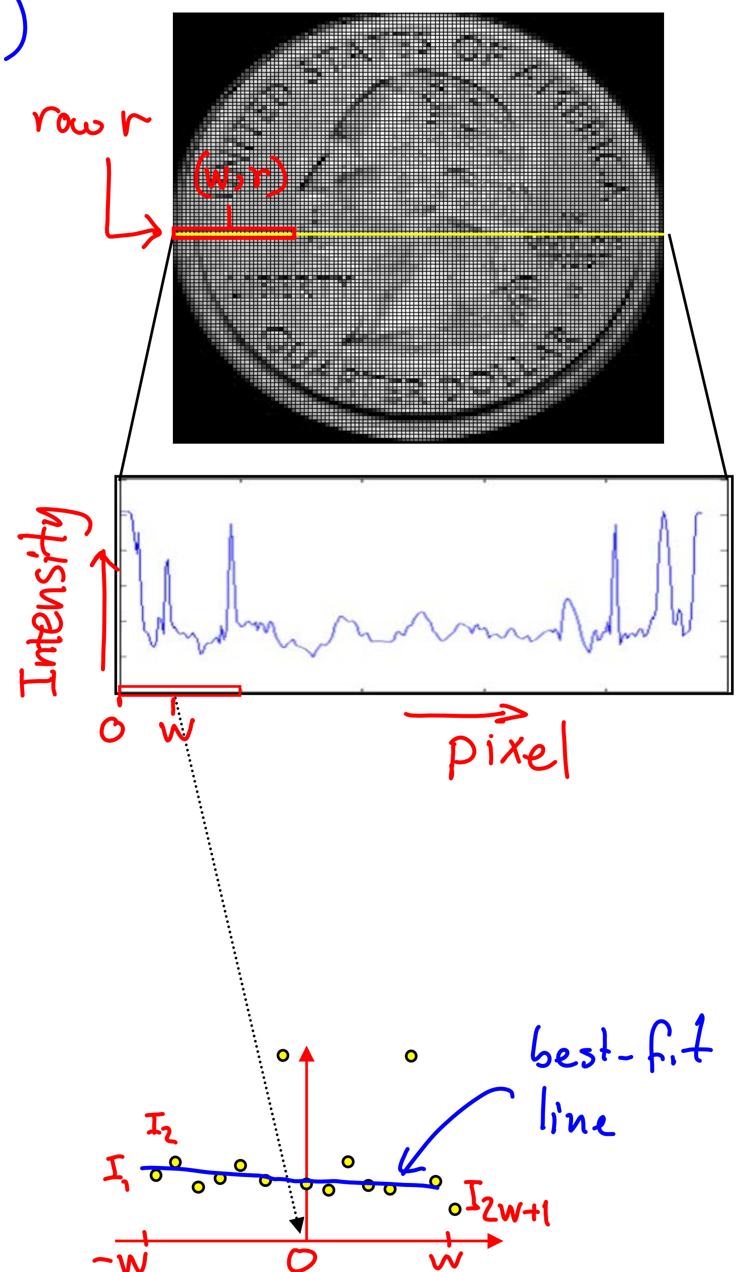
Image Cross Correlation ⇔ Matrix Multiplication

I : row r of the image (M pixels)

T : template ($2w+1$ pixels)

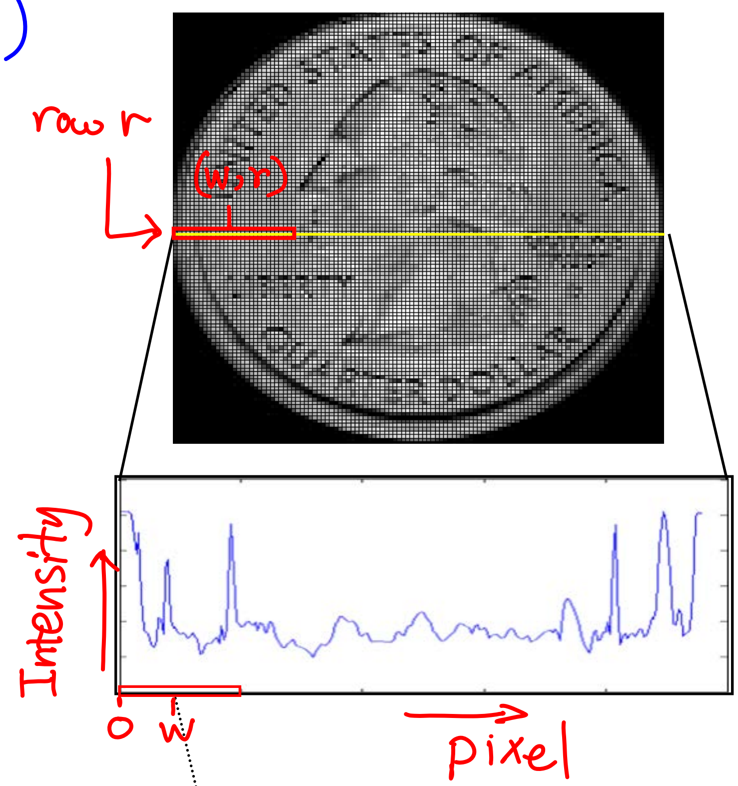
cross-correlation at pixel $w+1$

$$\begin{array}{l} \text{pixel } w \\ \text{pixel } w+1 \end{array} \begin{bmatrix} T_{-w} & T_{-w+1} & \cdots & T_0 & \cdots & T_w & 0 & \cdots & 0 \\ 0 & T_{-w} & T_{-w+1} & \cdots & T_{w-1} & T_w & \cdots & \cdots & 0 \end{bmatrix} \begin{bmatrix} I_0 \\ I_1 \\ I_2 \\ \vdots \\ I_w \\ \vdots \\ I_{2w} \\ I_{2w+1} \\ \vdots \\ I_{M-1} \end{bmatrix}$$



The Toeplitz Matrix of a Template

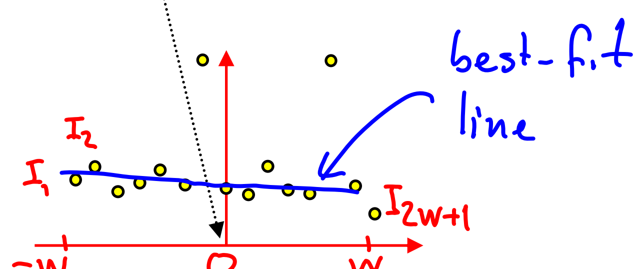
I: row r of the image (M pixels)
T: template (2w+1 pixels)



cross-correlation at pixels (w ,..., M-w-1)

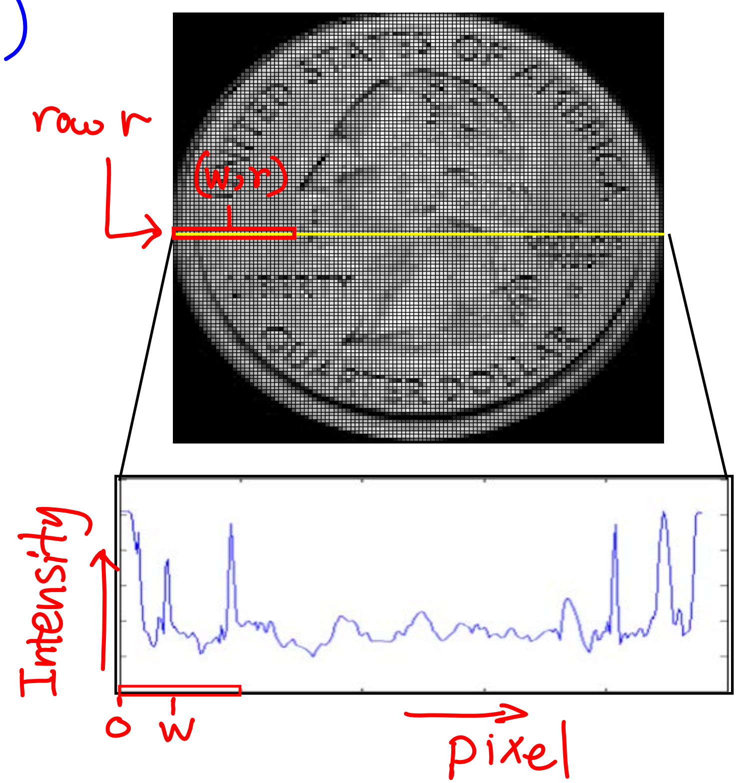
$$\begin{bmatrix} T_{-w} & T_{-w+1} & \cdots & T_0 & \cdots & T_w & 0 & \cdots & 0 \\ 0 & \ddots & & \ddots & & \ddots & 0 & \cdots & 0 \\ & & \ddots & & \ddots & & \ddots & & 0 \\ & 0 & & \ddots & & \ddots & & \ddots & \end{bmatrix} \begin{bmatrix} I_0 \\ I_1 \\ I_2 \\ \vdots \\ I_w \\ \vdots \\ I_{2w} \\ I_{2w+1} \\ \vdots \\ I_{M-1} \end{bmatrix}$$

each row is a 1-pixel right shift of the row above it
(called a TOEPLITZ matrix)



The Toeplitz Matrix of a Template

I: row r of the image (M pixels)
T: template (2w+1 pixels)



what about pixels
$0, \ldots, w-1$ & $M-w, \ldots, M-1$?

$$\begin{bmatrix} T_{-w} & T_{-w+1} & \cdots & T_0 & \cdots & T_w & 0 & \cdots & 0 \\ 0 & \ddots & & \ddots & & \ddots & 0 & \cdots & 0 \\ & & \ddots & & \ddots & & \ddots & & 0 \\ & 0 & & \ddots & & \ddots & & \ddots & \end{bmatrix} \begin{bmatrix} I_0 \\ I_1 \\ I_2 \\ \vdots \\ I_w \\ \vdots \\ I_{2w} \\ I_{2w+1} \\ \vdots \\ I_{M-1} \end{bmatrix}$$

each row is a 1-pixel right shift of the row above it
(called a TOEPLITZ matrix)

Here:
Assume I "wraps around"
i.e. I is periodic with period M (= # pixels in I)

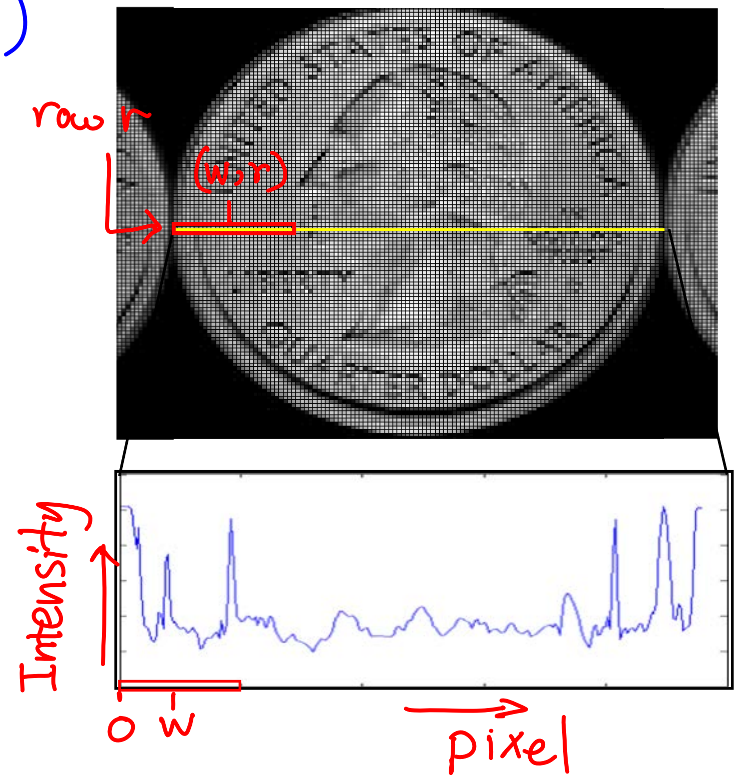
The Toeplitz Matrix of a Template

I: row r of the image (M pixels)
T: template (2w+1 pixels)

cross-correlation at pixels (w-1, ..., M-w-1)

$$\begin{bmatrix} T_{-w+1} & T_{-w+2} & \cdots & T_w & 0 & 0 & \cdots & 0 & T_{-w} \\ T_{-w} & T_{-w+1} & \cdots & T_0 & \cdots & T_w & 0 & \cdots & 0 \\ 0 & \ddots & & & \ddots & & \ddots & 0 & 0 \\ & & \ddots & & & \ddots & & \ddots & 0 \\ & 0 & & \ddots & & & \ddots & & \ddots \end{bmatrix} \begin{bmatrix} I_0 \\ I_1 \\ I_2 \\ \vdots \\ I_w \\ \vdots \\ I_{2w} \\ I_{2w+1} \\ \vdots \\ I_{M-1} \end{bmatrix}$$

each row is a 1-pixel right shift of the row above it
(called a TOEPLITZ matrix)



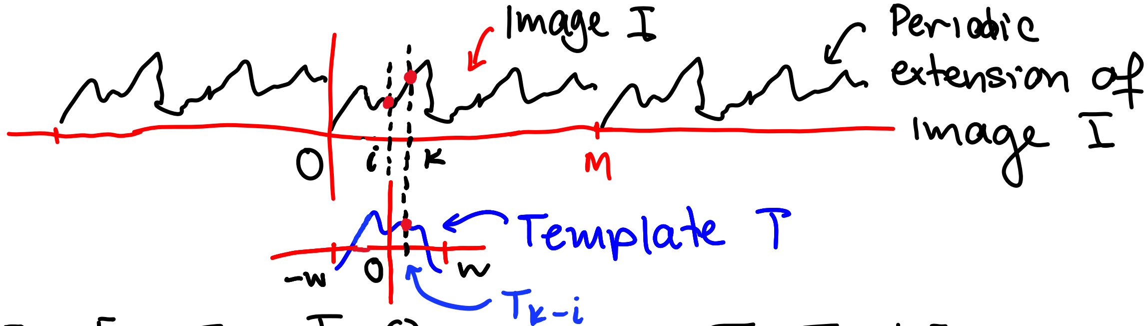
Here:
Assume I "wraps around"
i.e. I is periodic with period M (= # pixels in I)
e.g. I(0) = I(M)

Cross-Correlation Expressed as a Sum

General sum notation:

$$J_i = \sum_{k=0}^{M-1} I_k T_{k-i}$$

$0 \le i \le M-1$

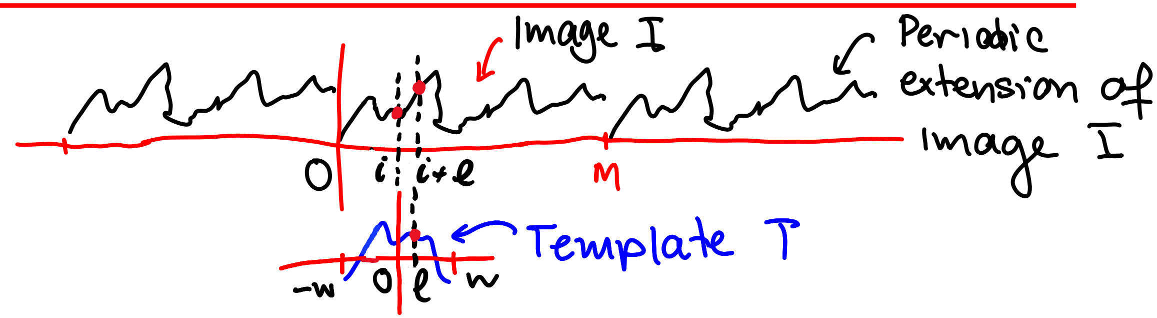


$$J = CC(I,T) = \begin{bmatrix} J_0 \\ J_1 \\ \vdots \\ J_w \\ J_{w+1} \\ \vdots \\ J_{2w} \\ \vdots \\ J_{M-1} \end{bmatrix} = \begin{bmatrix} T_0 & T_1 & \cdots & T_w & 0 & \cdots & 0 & T_{-w} & \cdots & T_{-2} & T_{-1} \\ T_{-1} & T_0 & T_1 & & T_w & & 0 & 0 & T_{-w} & \cdots & T_{-2} \\ \vdots & & & & & & & & & & \\ T_{-w} & \cdot & T_0 & \cdots & T_w & 0 & \cdot & & & & 0 \\ 0 & T_{-w} & & T_0 & \cdots & T_{w-1} & T_w & \cdot & & & 0 \\ \vdots & & & & & & & & & & \\ T_2 & \cdots & T_w & 0 & & 0 & T_{-w} & T_{-w-1} & \cdots & T_{-1} \; T_0 & T_1 \\ T_1 & T_2 & \cdots & T_w & 0 & \cdots & 0 & T_{-w} & \cdots & T_{-2} \; T_{-1} & T_0 \end{bmatrix} \begin{bmatrix} I_0 \\ I_1 \\ \vdots \\ I_w \\ I_{w+1} \\ \vdots \\ I_{2w} \\ \vdots \\ I_{M-1} \end{bmatrix}$$

Example

$$J_{i+1} = \sum_{k=0}^{M-1} I_k \cdot T_{k-i-1} = \sum_{k=0}^{M-1} I_k \cdot T_{k-(i+1)}$$

Cross-Correlation Expressed as a Sum



General sum notation:

$$J_i = \sum_{k=0}^{M-1} I_k T_{k-i}$$

$0 \le i \le M-1$

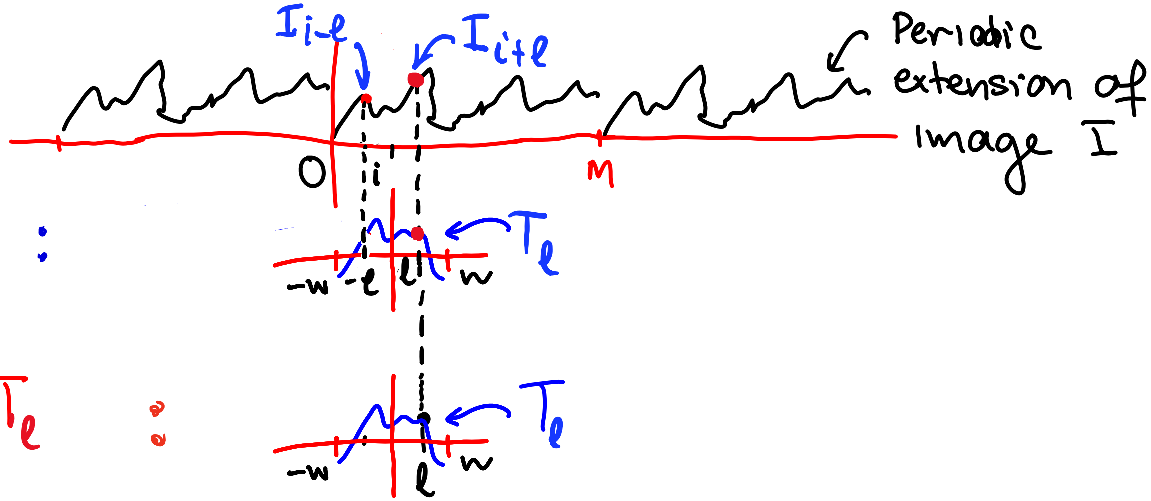
$$J = CC(I,T) = \begin{bmatrix} J_0 \\ J_1 \\ \vdots \\ J_w \\ J_{w+1} \\ \vdots \\ J_{2w} \\ \vdots \\ J_{M-1} \end{bmatrix} = \begin{bmatrix} T_0 & T_1 & \cdots & T_w & 0 & \cdots & \cdots & 0 & T_{-w} & \cdots & T_{-2} & T_{-1} \\ T_{-1} & T_0 & T_1 & & T_w & & 0 & 0 & T_{-w} & \cdots & & T_{-2} \\ & & & & & & & & & & & \\ T_{-w} & \cdot & & T_0 & \cdots & T_w & 0 & \cdot & & & & 0 \\ 0 & T_{-w} & & & T_0 & \cdots & T_{w-1} & T_w & \cdot & & & 0 \\ & & & & & & & & & & & \\ T_2 & \cdots & T_w & 0 & 0 & & T_{-w} & T_{-w+1} & \cdots & T_{-1} & T_0 & T_1 \\ T_1 & T_2 & \cdots & T_w & 0 & \cdots & 0 & T_{-w} & \cdots & T_{-2} & T_{-1} & T_0 \end{bmatrix} \begin{bmatrix} I_0 \\ I_1 \\ \vdots \\ I_w \\ I_{w+1} \\ \vdots \\ I_{2w} \\ \vdots \\ I_{M-1} \end{bmatrix}$$

Alternative sum notation:

$$J_i = \sum_{l=-w}^{w} I_{i+l} T_l$$

obtained by setting $l = k - i$, $k = i + l$

The Convolution Operation



$$J_i = \sum_{\ell=-w}^{w} I_{i+\ell} T_\ell \quad :$$

$$(I * T)_i = \sum_{\ell=-w}^{w} I_{i-\ell} T_\ell \quad :$$

Cross-correlation:

$$J_i = \sum_{\ell=-w}^{w} I_{i+\ell} T_\ell$$

Convolution:

$$(I * T)_i = \sum_{\ell=-w}^{w} I_{i-\ell} T_\ell$$

$$\begin{bmatrix} T_{-w} & T_{-w+1} & \cdots & T_0 & \cdots & T_w & 0 & \cdots & 0 \end{bmatrix}$$

$$\begin{bmatrix} T_{w} & T_{w-1} & \cdots & T_0 & \cdots & T_{-w} & 0 & \cdots & 0 \end{bmatrix}$$

$$\begin{bmatrix} I_0 \\ I_1 \\ I_2 \\ \vdots \\ I_w \\ \vdots \\ I_{2w} \\ I_{2w+1} \\ \vdots \\ I_{M-1} \end{bmatrix}$$

The Convolution Operation

- The convolution operation is one of the most fundamental operations in image (and signal) processing

- Common terminology
  - I = "image" or "signal"
  - T = "filter", "mask", "template", "impulse response", "kernel"

- Notation: $I * T$: convolution of I with mask T

$$\boxed{\text{Convolution: } \quad (I*T)_i = \sum_{\ell=-w}^{w} I_{i-\ell} T_\ell}$$

Properties (exercise: prove them)

1. For symmetric masks T convolution is equal to cross-correlation:

$$\boxed{\begin{array}{l} CC(I,T) = I * T \\ \text{when } T_\ell = T_{-\ell} \end{array}}$$

2. Commutativity:

$$\boxed{I * T = T * I}$$

3. Linearity:

$$\boxed{\begin{array}{l} (aI + bJ) * T = \\ \quad a(I*T) + b(J*T) \\ \text{for any constants } a, b \end{array}}$$

The Convolution Operation

Convolution in 2D ($M \times N$ image)

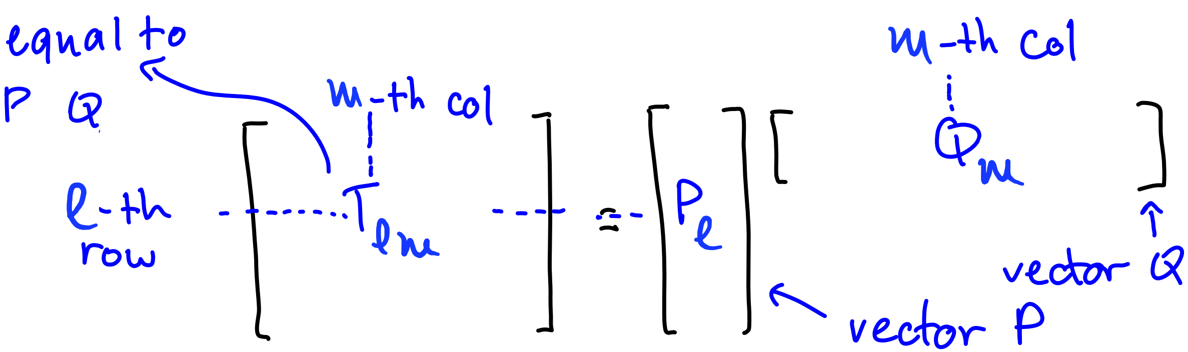
$$(I * T)_{ij} = \sum_{\ell=-w}^{w} \sum_{m=-w}^{w} I_{(i-\ell)(j-m)} T_{\ell m}$$

Very important special case:

Definition (Separable 2D mask)

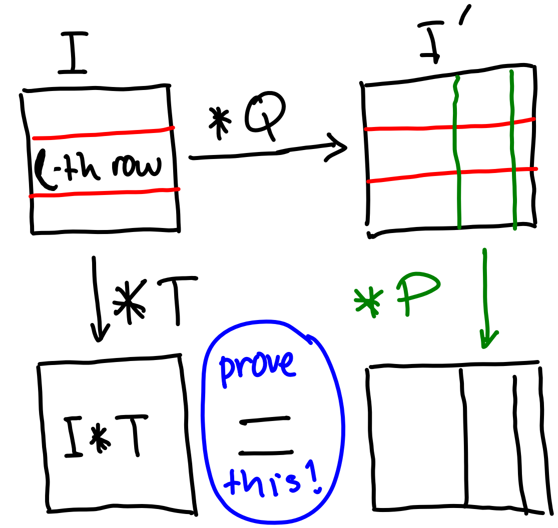
A mask T such that

$T = PQ^T$ for some vectors P, Q



Algorithm for computing $I*T$ for separable T:

① Convolve each row of I with Q

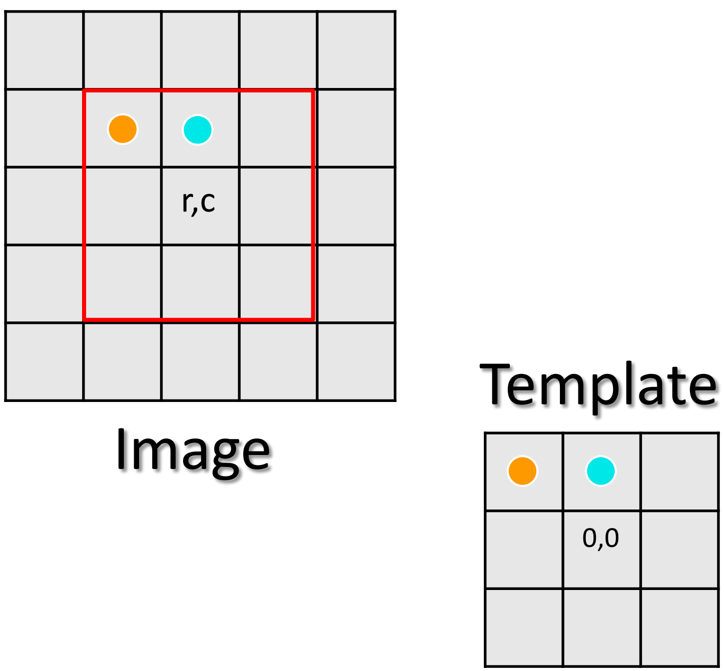


② Convolve each column of result I' with P

Cross-Correlation vs. Convolution (discrete)

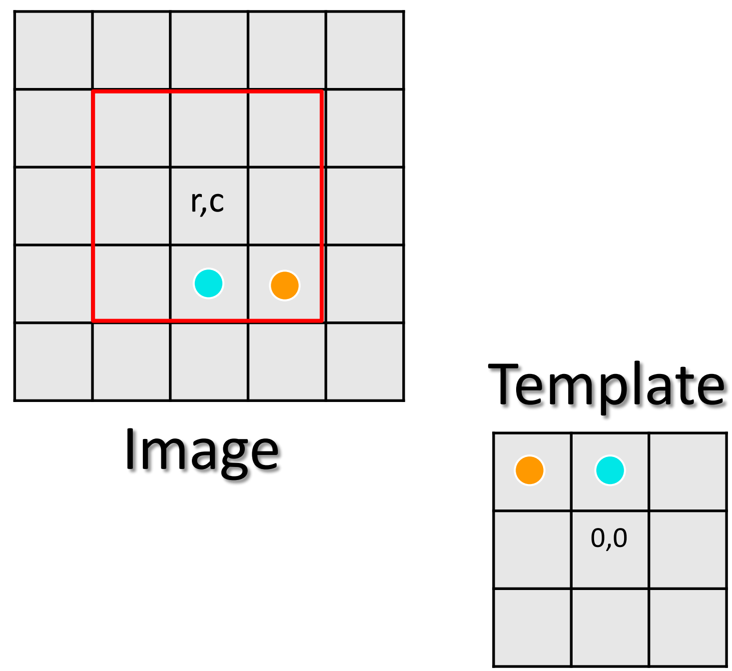
$\forall$r, c:

$$\mathrm{CC}(I, T, r, c) = \sum_{a=-w}^{w} \sum_{b=-w}^{w} I(r+a, c+b)\, T(a,b)$$



$\forall$r, c:

$$I * T\ (r, c) = \sum_{a=-w}^{w} \sum_{b=-w}^{w} I(r-a, c-b)\, T(a,b)$$



Which one is convolution?

$$\begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \; ? \; \begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \Rightarrow \begin{bmatrix} 3 & 1 & 2 \\ 6 & 4 & 5 \\ 9 & 7 & 8 \end{bmatrix}$$

$$\begin{bmatrix} 0 & 0 & 0 \\ 1 & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} \; ? \; \begin{bmatrix} 1 & 2 & 3 \\ 4 & 5 & 6 \\ 7 & 8 & 9 \end{bmatrix} \Rightarrow \begin{bmatrix} 2 & 3 & 1 \\ 5 & 6 & 4 \\ 8 & 9 & 7 \end{bmatrix}$$

I use $*$ for convolution and $\star$ for CC, and people might use other symbols (e.g. $\odot$, $\otimes$) for these operations. Always define the symbols when using them.

Topic 8:

A Unifying View:

1. Template matching	⇔	Derivatives via WLS fitting
2. Image smoothing	⇔	Template matching
3. Image interpolation	⇔	Convolution w/ continuous smoothing function
4. Image differentiation	⇔	Convolution w/ derivative of a smoothing function
5. Image Laplacian	⇔	Difference of two Gaussian-smoothed versions of an image

Topic 8:

A Unifying View:

1. Template matching ⇔ Derivatives via WLS fitting
2. Image smoothing ⇔ Template matching
3. Image interpolation ⇔ Convolution w/ continuous smoothing function
4. Image differentiation ⇔ Convolution w/ derivative of a smoothing function
5. Image Laplacian ⇔ Difference of two Gaussian-smoothed versions of an image

Review: WLS Estimation of I(x)

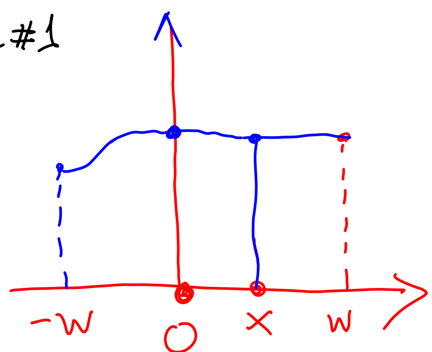
Q: How can we bias our estimate of $\frac{d}{dx}I(0)$ toward the patch center?

Patch (2w+1 pixels)

x=-w				x=0		x=2		x=w
I_1	I_2	.		I_{w+1}			...	I_{2w+1}

$$I_{(2w+1)\times 1} = X_{(2w+1)\times(n+1)}\, d_{(n+1)\times 1}$$

case #1

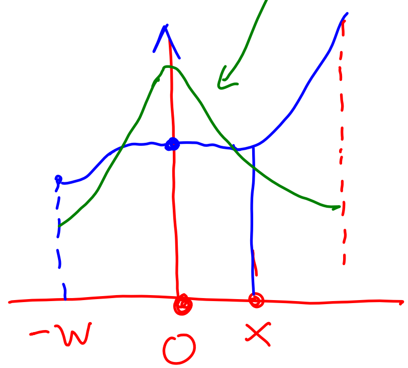


Idea: Weigh pixels near center more than pixels away from it

weight function $\Omega(x)$

(e.g. $\Omega(x) = e^{-x^2}$)

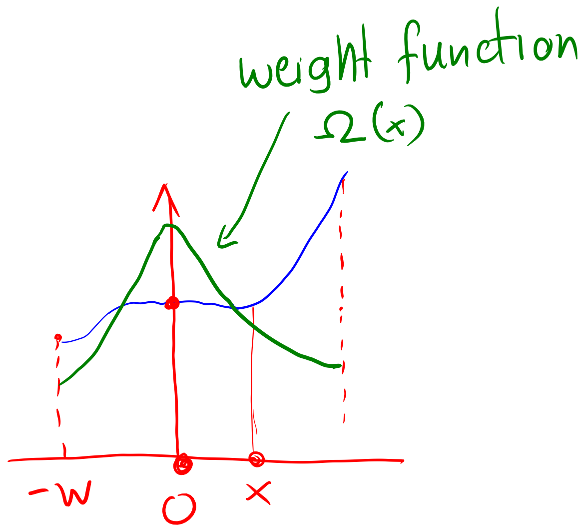
case #2



New equation for pixel x:

$$\Omega(x)\, I(x) = \Omega(x) \cdot \left[1 \quad x \quad \frac{1}{2}x^2 \quad \frac{1}{6}x^3 \quad \cdots \quad \frac{1}{n!}x^n \right] \begin{bmatrix} I(0) \\ \frac{dI}{dx}(0) \\ \frac{d^2I}{dx^2}(0) \\ \vdots \\ \frac{d^nI}{dx^n}(0) \end{bmatrix}$$

WLS Estimation ⇔ Cross Correlation



Patch (2w+1 pixels)

x=-w				x=0		x=2		x=w
I_1	I_2	.		I_{w+1}			. . .	I_{2w+1}

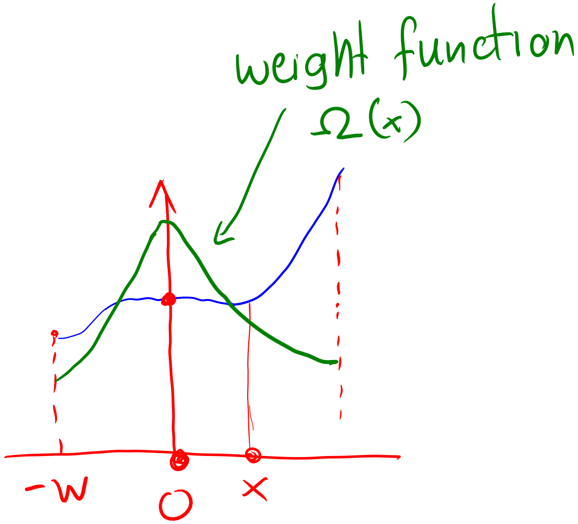
$$\underbrace{\begin{bmatrix} \Omega_1 & & 0 \\ & \ddots & \\ 0 & & \Omega_{2w+1} \end{bmatrix}}_{W} I = \begin{bmatrix} \Omega_1 & & \\ & \ddots & \\ & & \Omega_{2w+1} \end{bmatrix} X d$$

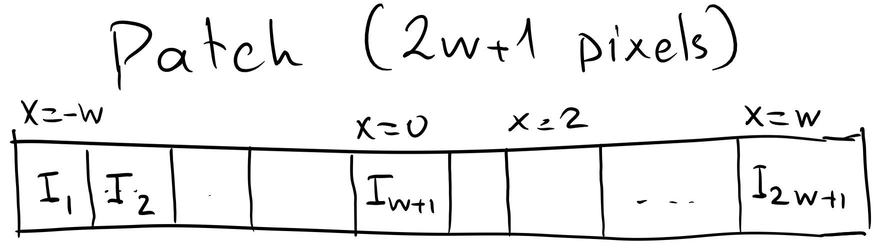
$$WXd = WI \Rightarrow (WX)^T (WX) d = (WX)^T W I$$

$$\Rightarrow d = \underbrace{\left[(WX)^T (WX)\right]^{-1} (WX)^T W}_{\text{represented as a single matrix}} I$$

Solution: $d = \begin{bmatrix} \quad \end{bmatrix} \begin{bmatrix} I_1 \\ I_2 \\ \vdots \\ I_{2w+1} \end{bmatrix}$

WLS Estimation ⇔ Cross Correlation





$$\underbrace{\begin{bmatrix} \Omega_1 & & 0 \\ 0 & \ddots & \\ & & \Omega_{2w+1} \end{bmatrix}}_{W} I = \begin{bmatrix} \Omega_1 & & \\ & \ddots & \\ & & \Omega_{2w+1} \end{bmatrix} X d$$

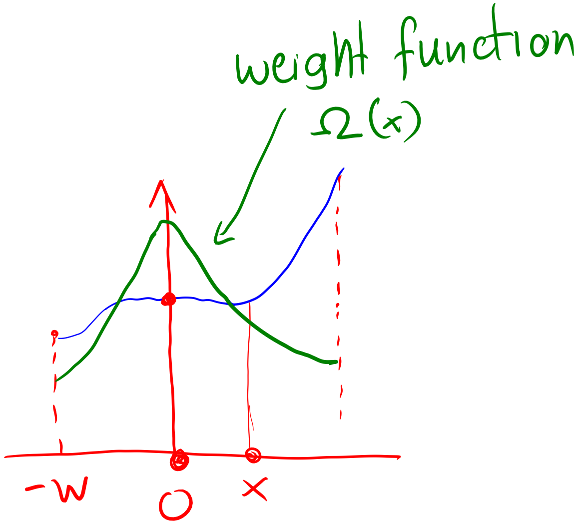
$$WXd = WI \Rightarrow (WX)^T (WX) d = (WX)^T W I$$

$$\Rightarrow d = \underbrace{\left[(WX)^T (WX)\right]^{-1} (WX)^T W}\, I$$

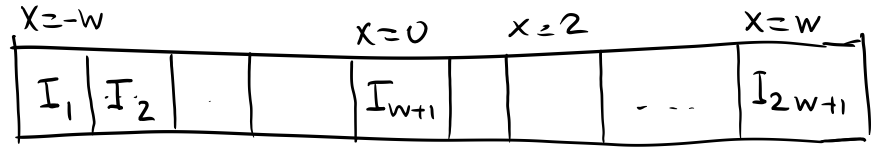
one row of this matrix

$$\text{Solution: } d = \begin{bmatrix} \boxed{(t^0)^T} \\ \vdots \\ \boxed{(t^n)^T} \end{bmatrix} \begin{bmatrix} I_1 \\ I_2 \\ \vdots \\ I_{2w+1} \end{bmatrix} = \begin{bmatrix} (t^0)^T \\ (t^1)^T \\ \vdots \\ (t^n)^T \end{bmatrix} \cdot \begin{bmatrix} I_1 \\ I_2 \\ \vdots \\ I_{2w+1} \end{bmatrix}$$

WLS Estimation ⇔ Cross Correlation



Patch ($2w+1$ pixels)



$$\begin{bmatrix} \Omega_1 & & 0 \\ & \ddots & \\ 0 & & \Omega_{2w+1} \end{bmatrix} I = \begin{bmatrix} \Omega_1 & & \\ & \ddots & \\ & & \Omega_{2w+1} \end{bmatrix} X d$$

- Constant term:
  (1st element of d)

$$\boxed{I^T \cdot t^0 = CC(I, t^0)}$$

i.e. it is computed as the cross-correlation of patch I and the "template" t^0

- Solution:

$$d = \begin{bmatrix} (t^0)^T \\ (t^1)^T \\ \vdots \\ (t^n)^T \end{bmatrix} \cdot \begin{bmatrix} I_1 \\ I_2 \\ \vdots \\ I_{2w+1} \end{bmatrix}$$

WLS Estimation ⇔ Cross Correlation

Goal: Estimate term $\frac{dI}{dx}(0)$ as the "similarity value" between image patch X_i and template t^1

"Template" patch t^1

↓

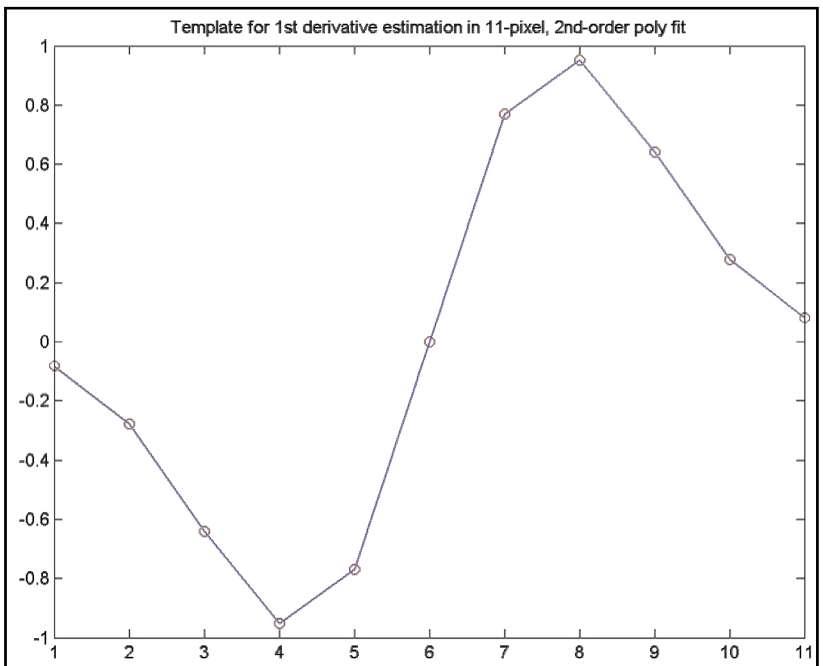


Image patch X_i

50	255	30	80	200	110	50	200	250	100	130

Similarity function

Cross-Correlation

$$CC(X_i, t') = X_i^T \cdot t'$$

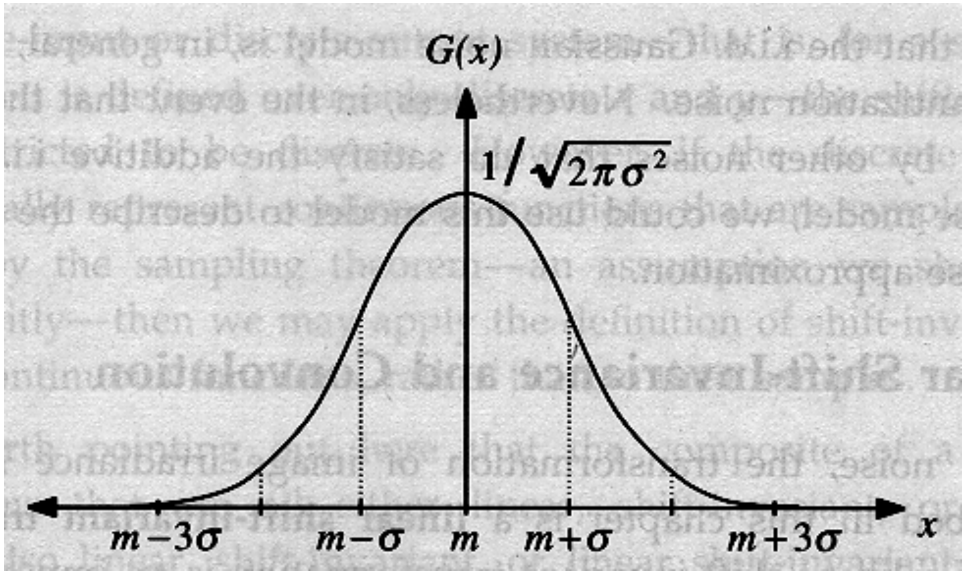
Topic 8:

A Unifying View:

1. **Template matching** ⇔ Derivatives via WLS fitting
2. Image smoothing ⇔ Template matching
3. **Image interpolation** ⇔ Convolution w/ continuous smoothing function
4. **Image differentiation** ⇔ Convolution w/ derivative of a smoothing function
5. **Image Laplacian** ⇔ Difference of two Gaussian-smoothed versions of an image

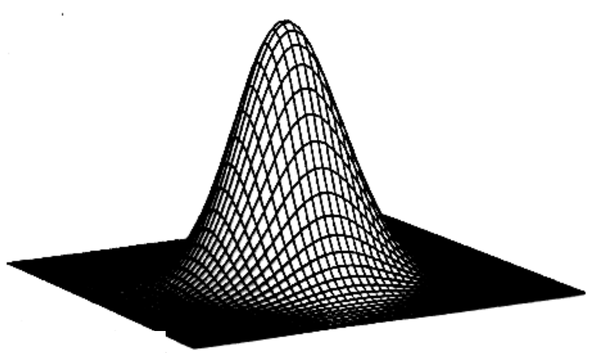
Gaussians in 1D and 2D

1D Gaussian



$$\boxed{G_\sigma(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-m)^2}{2\sigma^2}}}$$

2D Gaussian



$$\boxed{G_\sigma(r,c) = G_\sigma(r)G_\sigma(c) = \frac{1}{2\pi\sigma^2} e^{-\frac{(r-m_r)^2+(c-m_c)^2}{2\sigma^2}}}$$

Gaussian Smoothing ⇔ Cross-Correlation

Goal: Estimate constant term as the "similarity value" between image patch X_i and template T

Gaussian Template G_σ

	...				...	
$-w$		-1	0	1		w

Image patch X_i

50	255	30	80	200	110	50	200	250	100	130

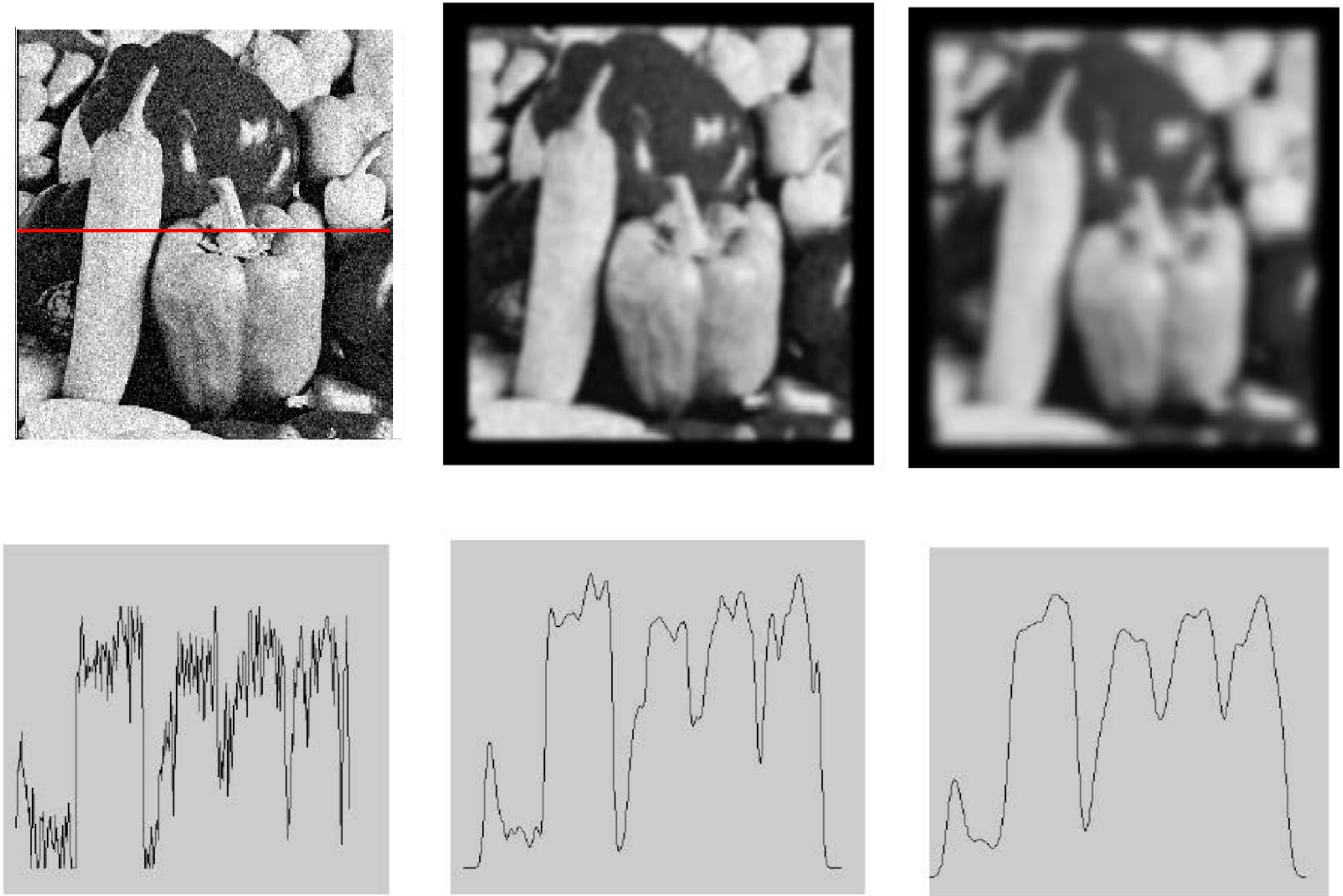
Prove this:

$CC(X_i, \bar{G}_\sigma) =$ 0^{th} order weighted least squares fit using the weight function $\Omega(x) = \sqrt{\bar{G}_\sigma(x)}$

- For smoothing operations we want to compute a weighted average, so we use the scaled template

$$\boxed{\bar{G}_\sigma = \frac{1}{\sum_{x=-w}^{w} G_\sigma(x)} G_\sigma(x)}$$

Example: Applying Gaussian Smoothing



No smoothing $\sigma = 2$ $\sigma = 4$

Topic 8:

A Unifying View:

1. **Template matching** ⇔ Derivatives via WLS fitting
2. **Image smoothing** ⇔ Template matching
3. Image interpolation ⇔ Convolution w/ continuous smoothing function
4. **Image differentiation** ⇔ Convolution w/ derivative of a smoothing function
5. **Image Laplacian** ⇔ Difference of two Gaussian-smoothed versions of an image

Image Interpolation: Definition

- Given a discrete set of (intensity) values $I_0, I_1, \ldots, I_{M-1}$ (also called "samples")
- Define a continuous function $I(x)$ that can be evaluated for any real $x \in [0, M-1]$

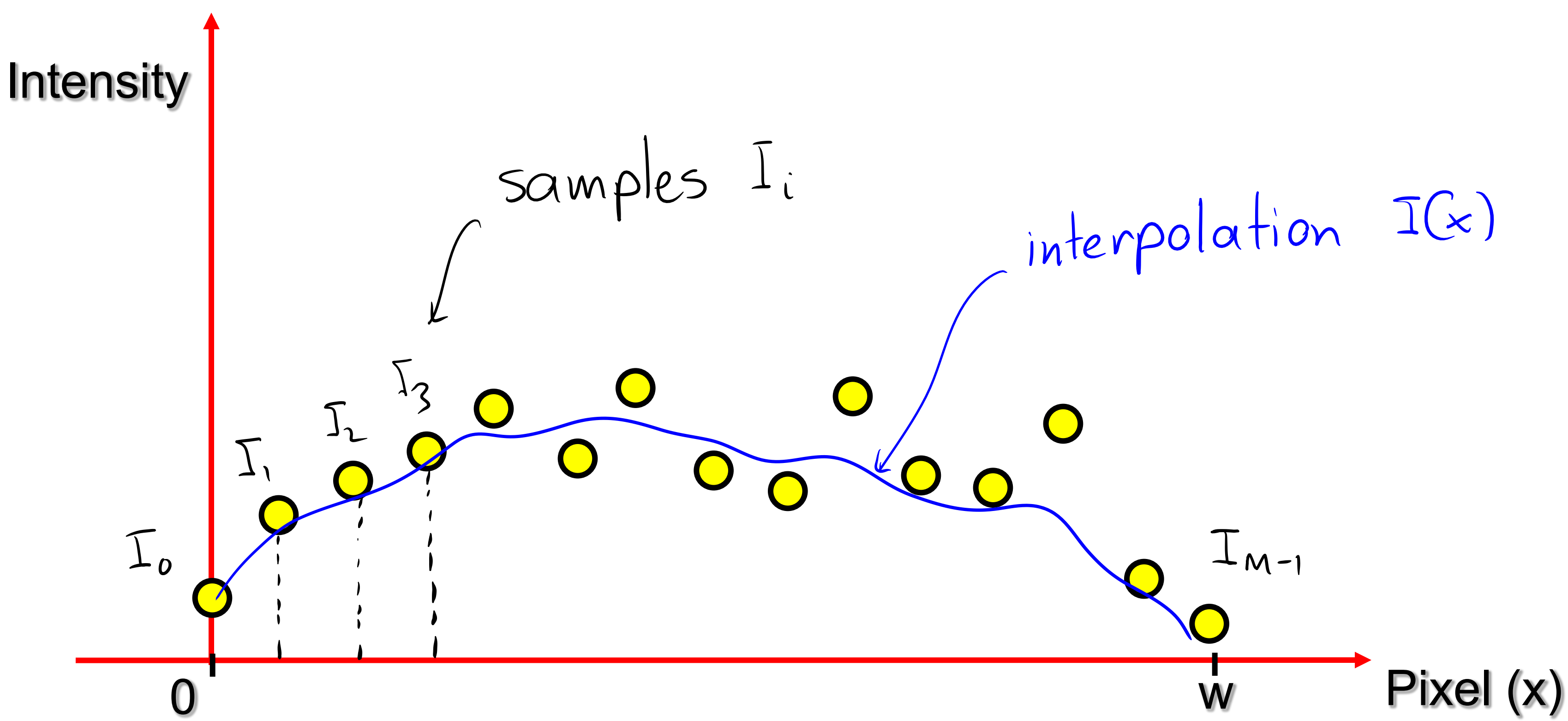


Image Interpolation: Applications

Some applications:

- Design of differentiation templates
- Image warping

Image Interpolation: Applications

Some applications:

- Design of differentiation templates
- Image warping

I_1	I_2	I_3	I_4	I_5	I_6	I_7
50	30	80	100	20	20	10

I_1	I_2	I_3	I_4	I_5	I_6	I_7	I_8	I_9	I_{10}	I_{11}	I_{12}	I_{13}	I_{14}
50	?	30	?	80	?	100	?	20	?	20	?	10	?

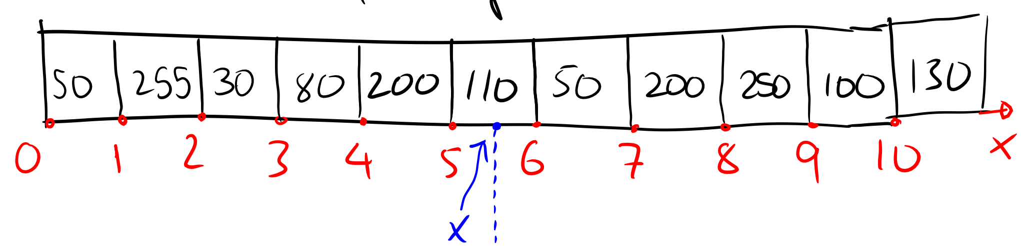
x2 Stretch

Interpolation: General Expression

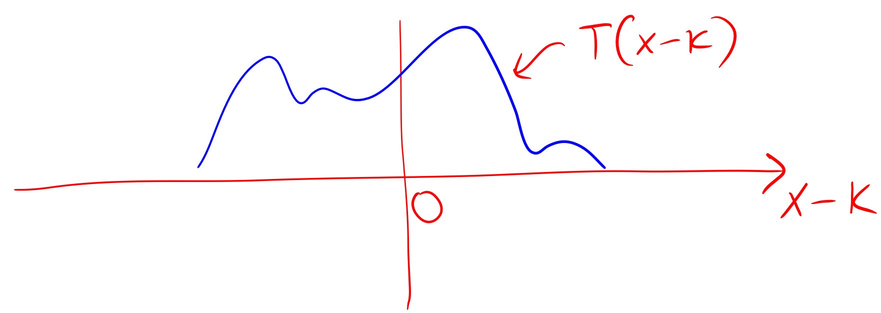
$$\boxed{(I * T)(x) = \sum_{k=0}^{M-1} \underbrace{I_k}_{\substack{\text{intensity}\\ \text{at k-th}\\ \text{pixel}}} \cdot \overbrace{T(\underbrace{x-k}_{\substack{\text{distance between x and}\\ \text{k-th pixel}}})}^{\substack{\text{contribution of k-th pixel to}\\ \text{intensity at x}}}}$$

- The result of interpolation (a function that can be evaluated at non-integer values of x)
- The general expression above allows T to be any arbitrary function defined for $x-k \in [-M+1, M-1]$
- In practice, T is chosen to satisfy additional properties (e.g. differentiability, $\int T(x)\,dx = 1$, etc)

Image I



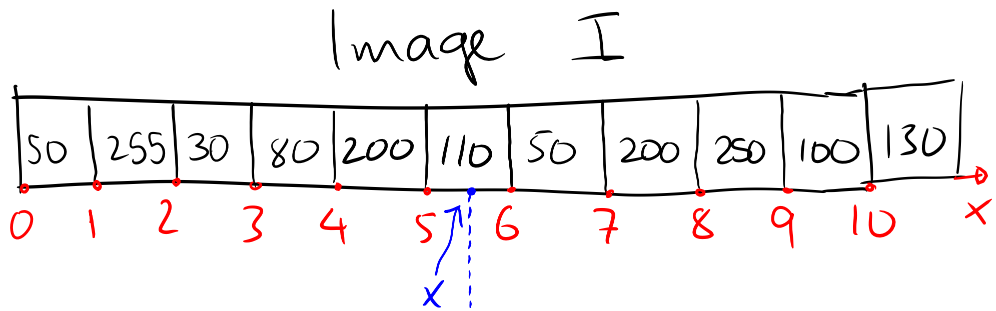
Smoothing function T a.k.a "Interpolation Kernel"



Example #1: Interpolation Using Gaussian Kernel

$$\boxed{(I * G_\sigma)(x) = \sum_{k=0}^{M-1} I_k \cdot G_\sigma(x-k)}$$

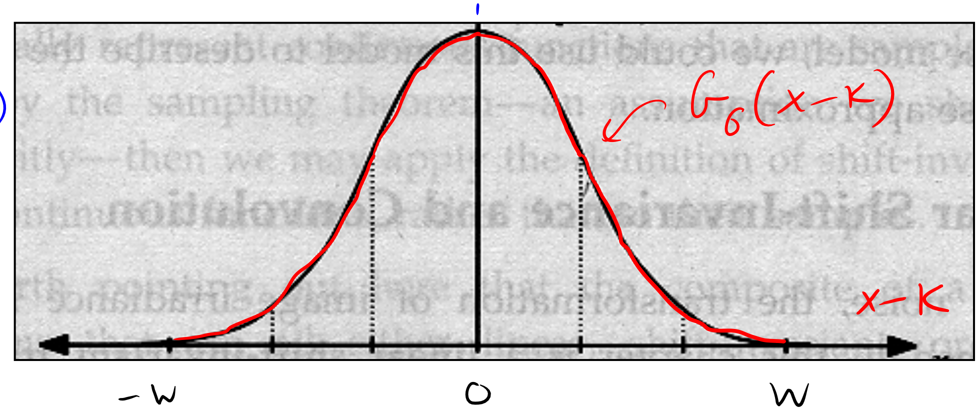
- $(I * G_\sigma)(x)$ = weighted combination of $I[0], \ldots, I[M-1]$



- $(I * G_\sigma)(x) =$ distance between the desired location x and the known pixel (5)

$$I[5] \cdot G_\sigma(x-5)$$
$$+ I[6] \cdot G_\sigma(x-6)$$
$$+ I[7] \cdot G_\sigma(x-7)$$
$$+ I[4] \cdot G_\sigma(x-4)$$
$$+ \ldots$$

Gaussian Interpolation Kernel

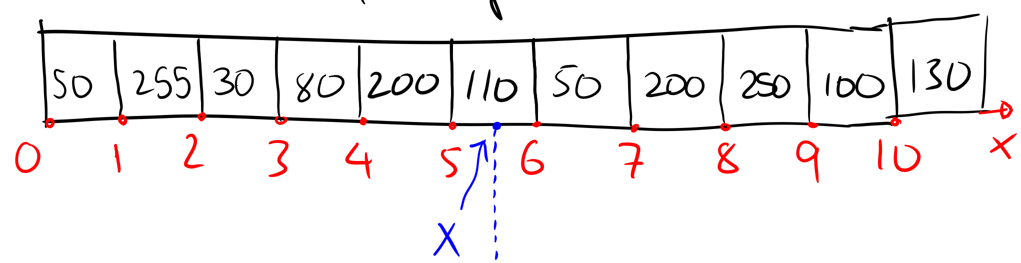


Example #2: Interpolation Using Linear Kernel

$$\boxed{(I * L)(x) = \sum_{k=0}^{M-1} I_k \cdot L(x-k)}$$

- $(I * L)(x)$ = weighted combination of $I[0], \ldots, I[M-1]$

Image I

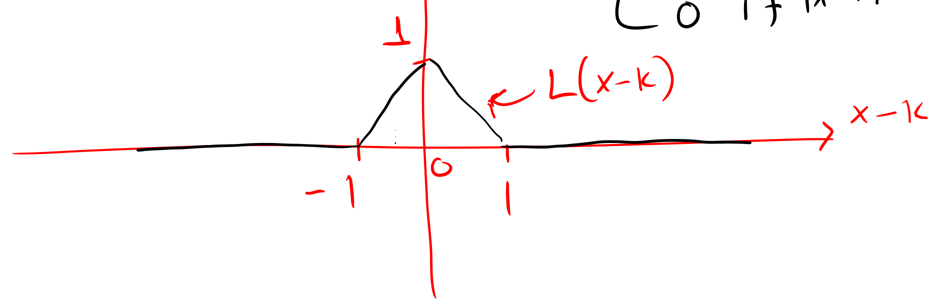


- $(I * L)(x) =$
  $\ldots +$
  $+ I[5] \cdot L(x-5)$
  $+ I[6] \cdot L(x-6)$
  $+ I[7] \cdot L(x-7) + \cdots =$ (for $x = 5 + \delta$ with $0 \leq \delta \leq 1$)

$I[5] \cdot \delta + I[6](1-\delta)$

Linear Interpolation kernel L

$$L(x-k) = \begin{cases} 1 - |x-k| & \text{if } |x-k| \leq 1 \\ 0 & \text{if } |x-k| > 1 \end{cases}$$



Topic 8:

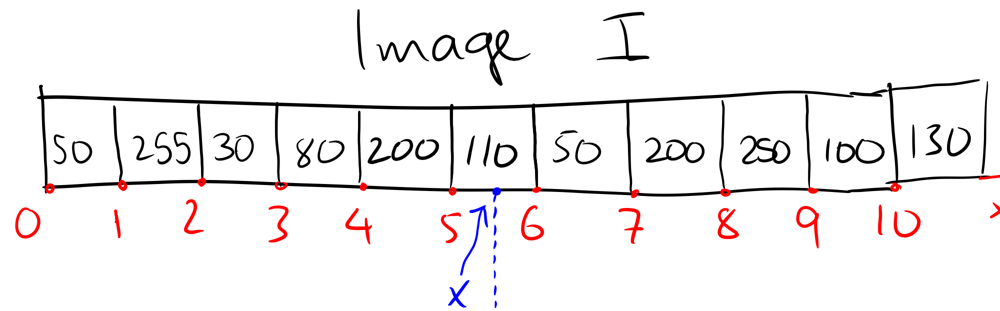
A Unifying View:

1. Template matching ⇔ Derivatives via WLS fitting
2. Image smoothing ⇔ Template matching
3. Image interpolation ⇔ Convolution w/ continuous smoothing function
4. Image differentiation ⇔ Convolution w/ derivative of a smoothing function
5. Image Laplacian ⇔ Difference of two Gaussian-smoothed versions of an image

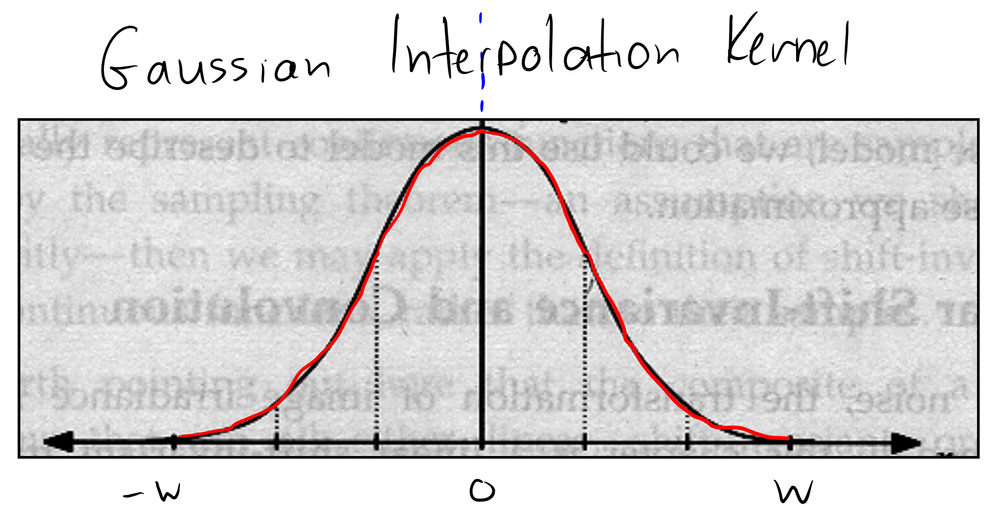
Step #1: Interpolate Using Gaussian Kernel

$$\boxed{(I * G_\sigma)(x) = \sum_{k=0}^{M-1} I_k \cdot G_\sigma(x-k)}$$

To interpolate, we evaluate the expression at continuous values x



- Since $(I * G_\sigma)(x)$ is a weighted sum of Gaussians, we can compute its derivative analytically!



Step #2: Differentiate the Interpolated Image

$$\boxed{(I * G_\sigma)(x) = \sum_{k=0}^{M-1} \underbrace{I_k}_{\text{discrete}} \cdot G_\sigma(\overbrace{x}^{\text{continuous}} - k)}$$

$$\frac{d}{dx}(I * G_\sigma)(x) =$$

$$\frac{d}{dx}\left[\sum_{k=0}^{M-1} I_k \cdot G_\sigma(x-k)\right] =$$

$$\sum_{k=0}^{M-1} I_k \cdot \frac{d}{dx} G_\sigma(x-k) \Longleftrightarrow$$

$$\boxed{\frac{d}{dx}(I * G_\sigma)(x) = \left[I * \left(\frac{d}{dx} G_\sigma\right)\right](x)}$$

Image Differentiation ⇔ Convolution w/ Gaussian Derivative

$$\boxed{(I * G_\sigma)(x) = \sum_{k=0}^{M-1} I_k \cdot G_\sigma(x-k)}$$

We can compute derivatives by applying a template that is the derivative of the Gaussian function!

$$\boxed{\frac{d}{dx}(I * G_\sigma)(x) = \left[I * \left(\frac{d}{dx} G_\sigma\right)\right](x)}$$

Convolution with the Derivative of a Gaussian

$$(I * G_\sigma)(x) = \sum_{k=0}^{M-1} I[k] \cdot G_\sigma(x-k)$$

$$G_\sigma(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{x^2}{2\sigma^2}}$$

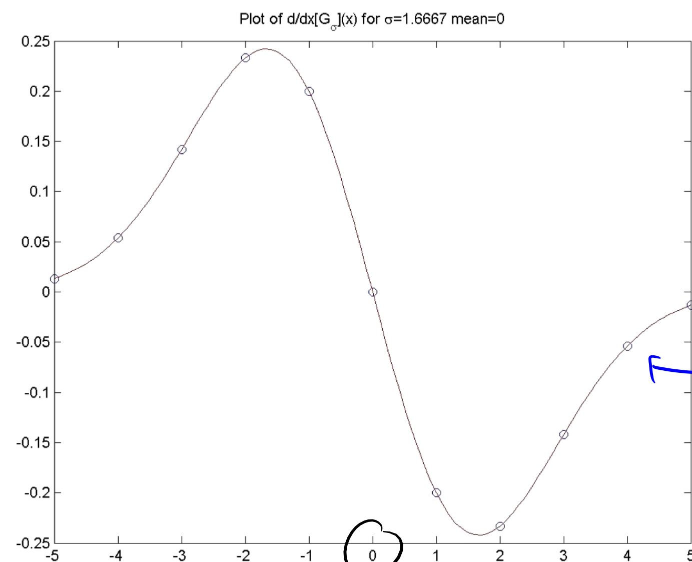
$$G'_\sigma(x) = \frac{d}{dx}\left(\frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{x^2}{2\sigma^2}}\right) = -\frac{2x}{2\sigma^2} \cdot \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{x^2}{2\sigma^2}} = -\frac{x}{\sigma^2} G_\sigma(x)$$

Differentiation mask:

$G'_\sigma(-2\sigma)$	...	$G'_\sigma(0)$	...	$G'_\sigma(2\sigma)$

$$\frac{d}{dx}(I * G_\sigma)(x) = \left[I * \left(\frac{d}{dx} G_\sigma\right)\right](x)$$

Convolution with the Derivative of a Gaussian



$G'(i)$

$$\frac{dI}{dx}(6) = [G'(-5) \cdots G'(5)] * [I_1\ I_2 \cdots I_{11}] * \frac{1}{\sum_{i=-5}^{5} G(i)}$$

Gaussian derivative evaluated at $i=1, \ldots, 11$

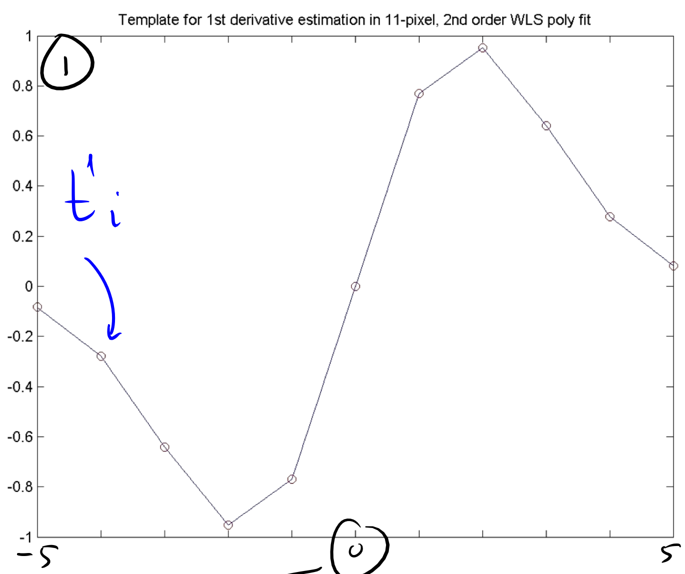
Convolution with the Derivative of a Gaussian

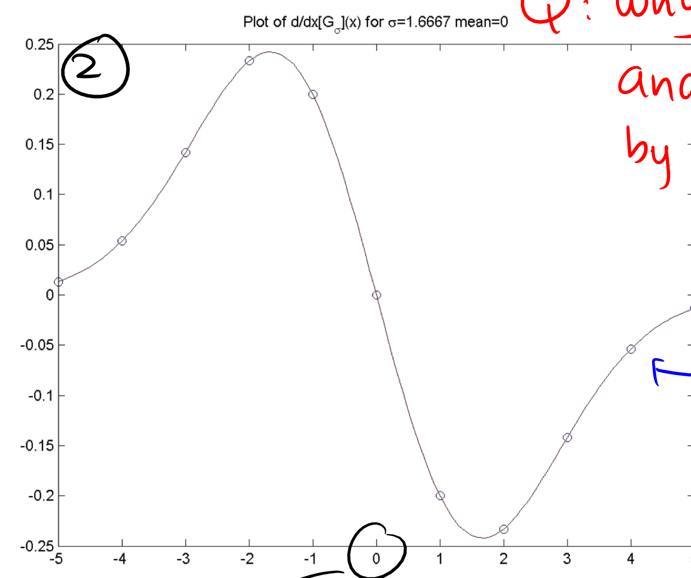
Comparing the 1D kernel for estimating $\frac{dI}{dx}$

① using WLS polyfit with Gaussian weights

② using convolution with $\frac{dG}{dx}$

Q: why are t^1 and G' related by a reflection?





$$\frac{dI}{dx}(6) = [t^1_{-5}\ t^1_{-4} \cdots t^1_{5}] \cdot \begin{bmatrix} I_1 \\ I_2 \\ I_3 \\ \vdots \\ I_{11} \end{bmatrix}$$

2nd row of polyfit matrix

$$\frac{dI}{dx}(6) = [G'(-5) \cdots G'(5)] * [I_1\ I_2 \cdots I_{11}] * \frac{1}{\sum_{i=-5}^{5} G(i)}$$

Gaussian derivative evaluated at $i=1,\ldots,11$

Convolution with the Derivative of a Gaussian

$$\boxed{(I * G_6)(x) = \sum_{k=0}^{M-1} I[k] \cdot G_6(x-k)}$$

We can compute derivatives by applying a template that is the derivative of the Gaussian function!

$$\boxed{\frac{d^2}{dx^2}(I * G_6)(x) = \left[I * \left(\frac{d^2}{dx^2} G_6\right)\right](x)}$$

Convolution with the Derivative of a Gaussian

$$(I * G_\sigma)(x) = \sum_{k=0}^{M-1} I[k] \cdot G_\sigma(x-k)$$

$$G_\sigma(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{x^2}{2\sigma^2}}$$

$$G''_\sigma(x) = \frac{d^2}{dx^2}\left(\frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{x^2}{2\sigma^2}}\right) =$$

$$= \left(\frac{x^2}{\sigma^2} - 1\right) \cdot \frac{1}{\sigma^2} \cdot G_\sigma(x)$$

Differentiation mask:

$G''_\sigma(-2\sigma)$	...	$G''_\sigma(0)$	...	$G''_\sigma(2\sigma)$

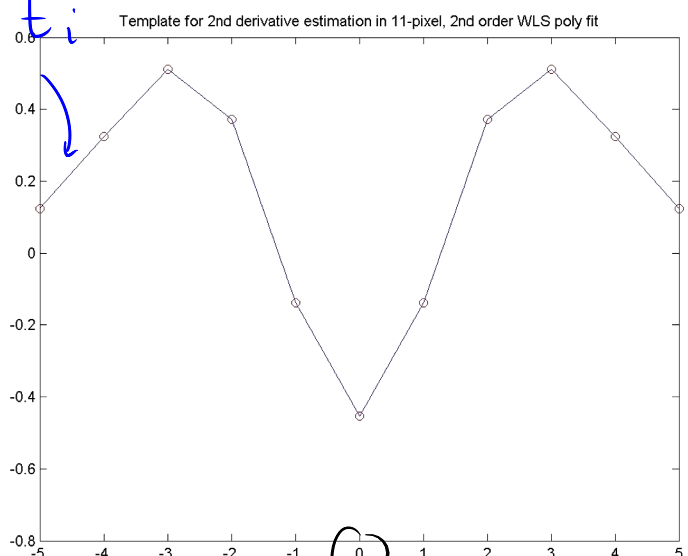
$$\frac{d^2}{dx^2}(I * G_\sigma)(x) = \left[I * \left(\frac{d^2}{dx^2} G_\sigma\right)\right](x)$$

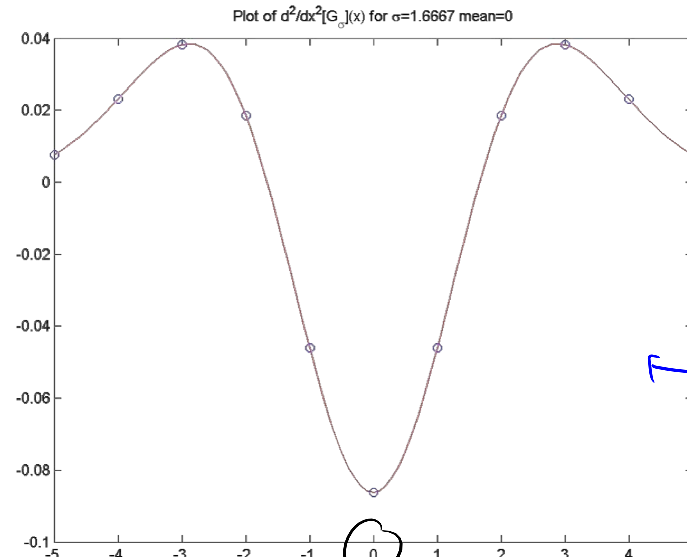
Convolution with the Derivative of a Gaussian

Comparing the 1D kernel for estimating $\frac{d^2I}{dx^2}$

- using WLS polyfit with Gaussian weights
- using convolution with $\frac{d^2G}{dx^2}$

t_i^2





$\leftarrow G''(i)$

$$\frac{d^2I}{dx^2}(6) = [t^2_{-5}\ t^2_{-4} \cdots t^2_{5}] \cdot \begin{bmatrix} I_1 \\ I_2 \\ I_3 \\ \vdots \\ I_{11} \end{bmatrix}$$

3rd row of polyfit matrix

$$\frac{d^2I}{dx^2}(6) = [G''(-5) \cdots G''(5)] * [I_1\ I_2 \cdots I_{11}]$$

Gaussian 2nd derivative evaluated at $i=1,\ldots,11$

Convolution with the Derivative of a Gaussian

- Advantages (1D images)
  - Derivative of G_σ computed analytically
  - Can control σ (scale of smoothing)
  - Can compute even higher order derivatives by convolving with $\frac{d^n}{dx^n} G_\sigma(x)$
  - Computations are always dot products (⇒ efficient)

$$\boxed{\frac{d^n}{dx^n}(I * G_\sigma)(x) = \left[I * \left(\frac{d^n}{dx^n} G_\sigma \right) \right](x)}$$

Topic 8:

A Unifying View:

1. **Template matching**	$\Leftrightarrow$	Derivatives via WLS fitting
2. **Image smoothing**	$\Leftrightarrow$	Template matching
3. **Image interpolation**	$\Leftrightarrow$	Convolution w/ continuous smoothing function
4. **Image differentiation**	$\Leftrightarrow$	Convolution w/ derivative of a smoothing function
5. Image Laplacian	$\Leftrightarrow$	**Difference of two Gaussian-smoothed versions of an image**

What does smoothing take away?

$I * G_{\sigma_1}$

What Does Smoothing Take Away?

$I * G_{\sigma_2}$

The Difference-Of-Gaussians (DOG) Filter

Difference of two Gaussian-smoothed versions of I:

$I * G_{\sigma_1} -$

$I * G_{\sigma_2} =$

$I * \underbrace{(G_{\sigma_1} - G_{\sigma_2})}_{\text{the DOG filter}}$

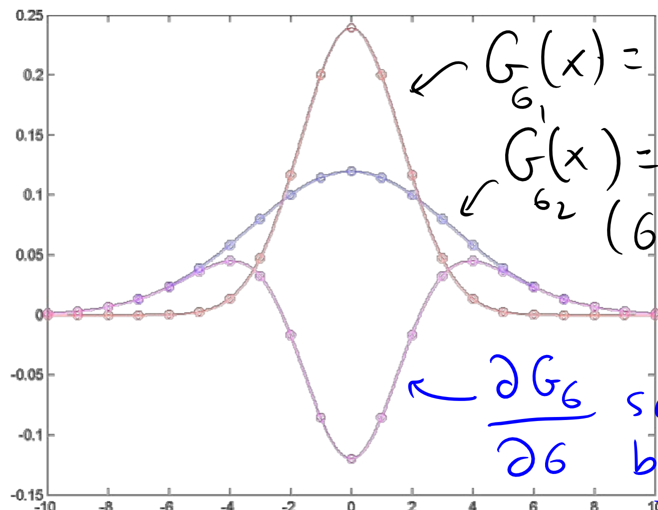
the DOG filter (just the difference between two Gaussian masks)

Equivalence of DOG and 2nd Derivative Filter

- What is $I * (G_{\sigma_1} - G_{\sigma_2})$?
- What is $G_{\sigma_1} - G_{\sigma_2}$?

To answer, consider G to be a function of both x and σ

①

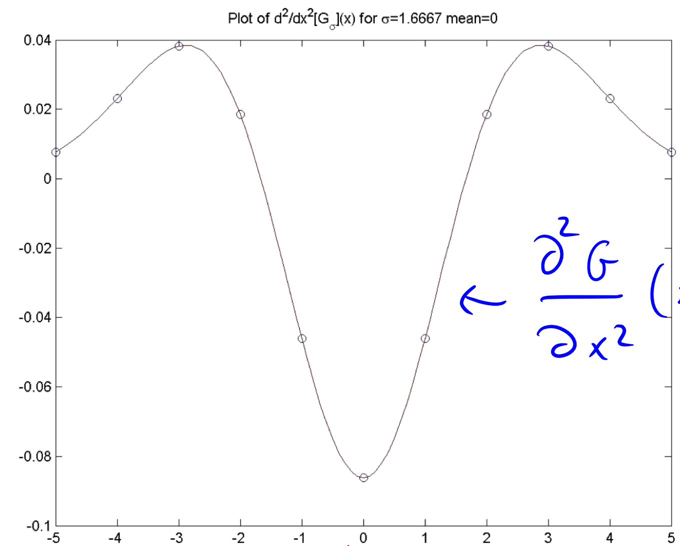


$\Longleftrightarrow$

$$\boxed{G_{\sigma_2}(x) - G_{\sigma_1}(x) = (\sigma_2 - \sigma_1) \cdot \frac{\partial G_\sigma}{\partial \sigma}(x;\sigma_1)}$$

(proof on next slide)

② Compare to the 2nd derivative of G with respect to x:



$\Downarrow$

$\Longrightarrow$ The DOG filter is just a scaled version of the Gaussian 2nd derivative filter

Equivalence of DOG and 2nd Derivative Filter

- What is $I * (G_{\sigma_1} - G_{\sigma_2})$?
- What is $G_{\sigma_1} - G_{\sigma_2}$?

To answer, consider G to be a function of both x and σ

1. Approximate difference by the derivative at scale σ_1

$$\frac{\partial G(x;\sigma_1)}{\partial \sigma} \simeq \frac{G(x;\sigma_2) - G(x;\sigma_1)}{\sigma_2 - \sigma_1}$$

$\iff$

$$\boxed{G_{\sigma_2}(x) - G_{\sigma_1}(x) = (\sigma_2 - \sigma_1) \cdot \frac{\partial G_\sigma}{\partial \sigma}(x;\sigma_1)}$$

2. Compare to the 2nd derivative of G with respect to x:

Differentiating the formula for $G(x;\sigma)$

$$\frac{\partial^2 G_\sigma}{\partial x^2} = \left(\frac{x^2}{\sigma^2} - 1\right) \frac{1}{\sigma^2} G_\sigma(x)$$

$$\frac{\partial G_\sigma}{\partial \sigma} = \left(\frac{x^2}{\sigma^2} - 1\right) \frac{1}{\sigma} G_\sigma(x)$$

i.e. $\boxed{\frac{\partial G_\sigma}{\partial \sigma} = \sigma \frac{\partial^2 G_\sigma}{\partial x^2}}$

$\Downarrow$

$\Rightarrow$

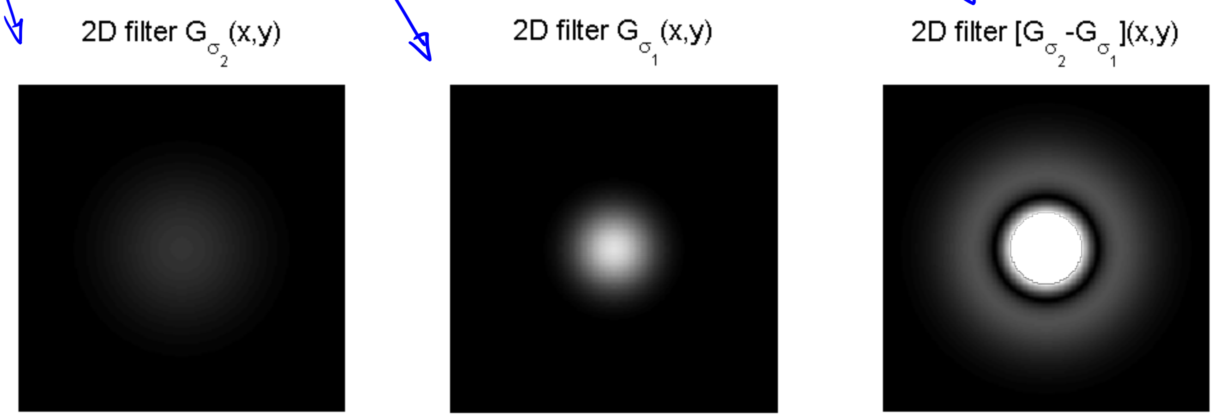
$$\boxed{G_{\sigma_2}(x) - G_{\sigma_1}(x) = (\sigma_2 - \sigma_1) \cdot \sigma_1 \cdot \frac{\partial^2 G_\sigma}{\partial x^2}}$$

The Difference-Of-Gaussians (DOG) Filter

- What is $I * (G_{\sigma_1} - G_{\sigma_2})$?
- What is $G_{\sigma_1} - G_{\sigma_2}$?
- In two dimensions:

Exercise: Prove this

$$G_{\sigma_2}(x,y) - G_{\sigma_1}(x,y) = \sigma_1(\sigma_1 - \sigma_2)\nabla^2 G_{\sigma_1}(x,y)$$



- Image Laplacian from the difference of two Gaussians:

$$\nabla^2(I * G_{\sigma_1}) = I * \nabla^2 G_{\sigma_1} = (I * G_{\sigma_2} - I * G_{\sigma_1})\frac{1}{\sigma_1 \cdot (\sigma_2 - \sigma_1)}$$