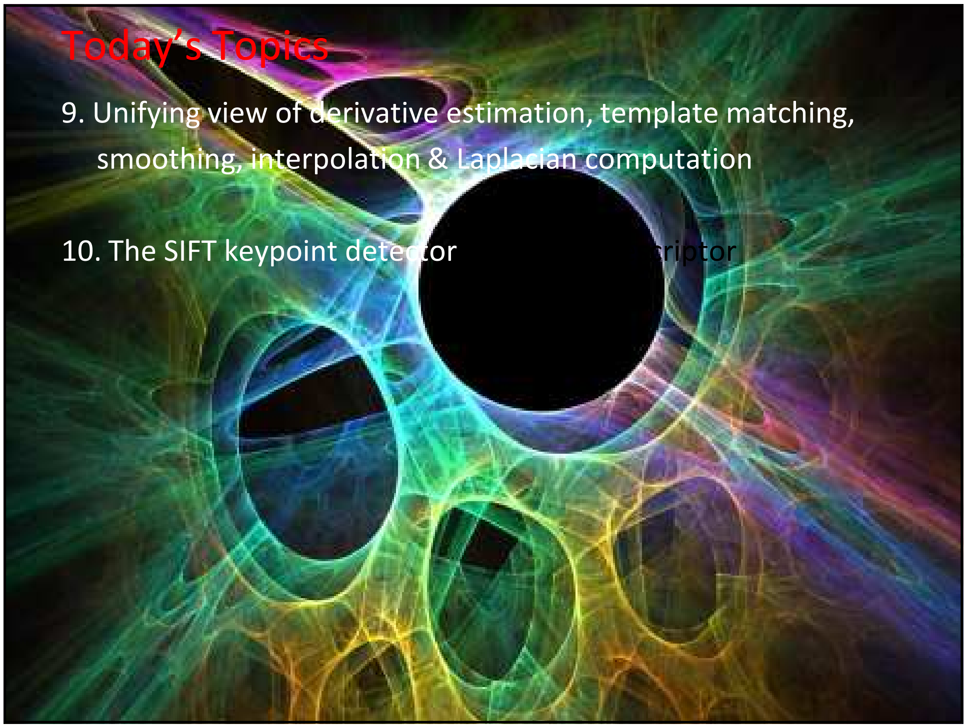


Today's Topics

9. Unifying view of derivative estimation, template matching, smoothing, interpolation & Laplacian computation

10. The SIFT keypoint detector … riptor

Announcements

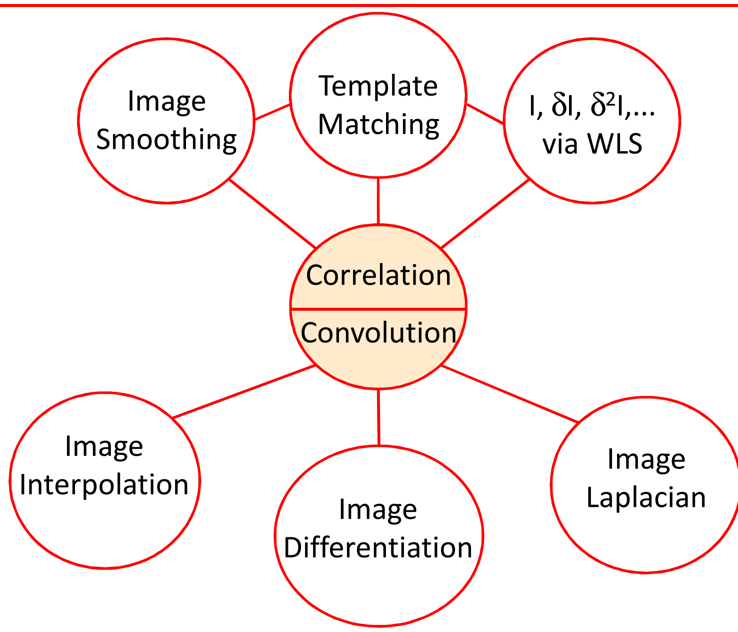
- A4 is due April 4
- Marks for A2 are starting to come. Will be available by Friday (apologies from our side, lateness due to a sick TA)
- Grace-days left will be posted on Blackboard tomorrow
- May use grace-days after April 4, keep an eye on the final exam (worth 4 times the value of the assignment).

Topic 9:

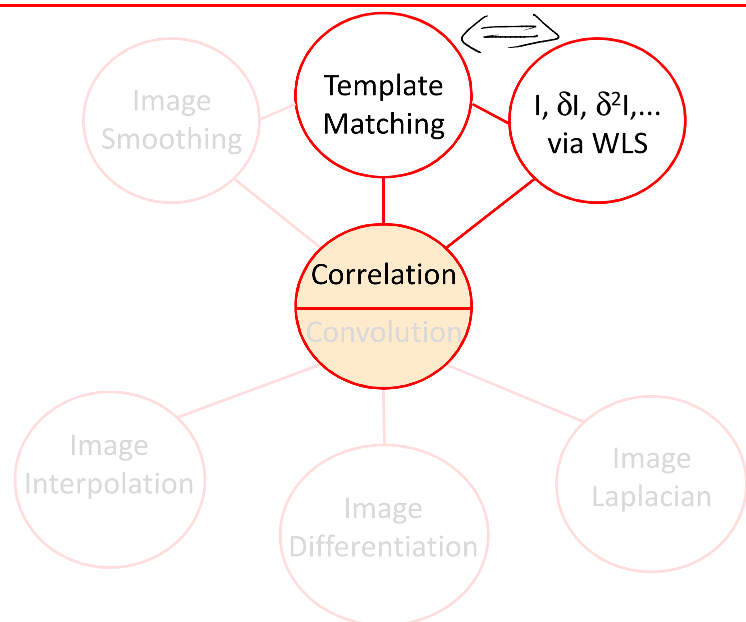
A Unifying View:

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

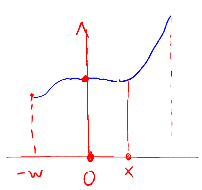
Unifying View

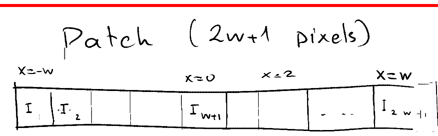


Unifying View

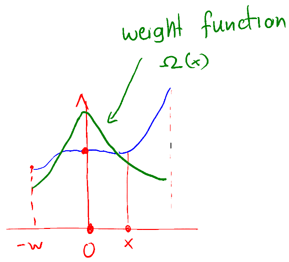


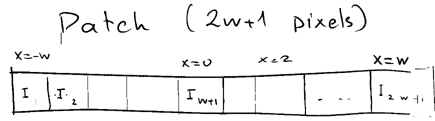
WLS Estimation ⇔ Cross Correlation





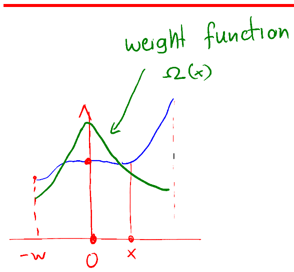
WLS Estimation ⇔ Cross Correlation

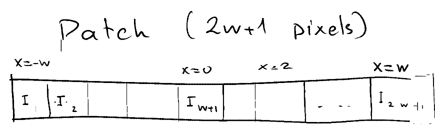




$$\begin{bmatrix}\Omega_1 & & 0\\ & \ddots & \\ 0 & & \Omega_{2w+1}\end{bmatrix} I = \begin{bmatrix}\Omega_1 & & \\ & \ddots & \\ & & \Omega_{2w+1}\end{bmatrix} X d$$

WLS Estimation ⇔ Cross Correlation



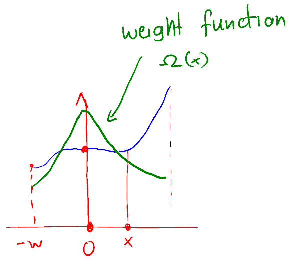


$$\underbrace{\begin{bmatrix}\Omega_1 & & 0\\ & \ddots & \\ 0 & & \Omega_{2w+1}\end{bmatrix}}_{W} I = \begin{bmatrix}\Omega_1 & & \\ & \ddots & \\ & & \Omega_{2w+1}\end{bmatrix} X d$$

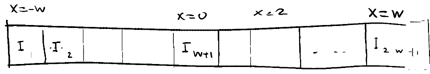
$$WXd = WI$$

W: weights
X: x, $\frac{1}{2}x^2$, $1/6x^3$, ...
d: derivatives
I: image contents

WLS Estimation ⇔ Cross Correlation



Patch (2w+1 pixels)

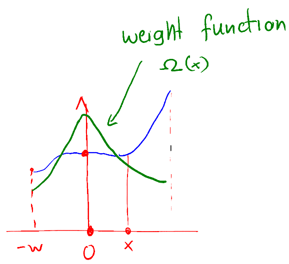


$$\underbrace{\begin{bmatrix} \Omega_1 & & 0 \\ 0 & \ddots & \\ & & \Omega_{2w+1} \end{bmatrix}}_{W} I = \begin{bmatrix} \Omega_1 & & \\ & \ddots & \\ & & \Omega_{2w+1} \end{bmatrix} X d$$

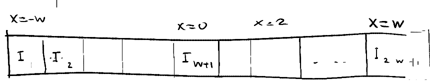
$$WXd = WI \Rightarrow (WX)^T (WX) d = (WX)^T W I$$

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

WLS Estimation ⇔ Cross Correlation



Patch (2w+1 pixels)



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

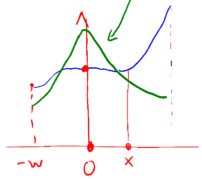
$$WXd = WI \Rightarrow (WX)^T (WX) d = (WX)^T W I$$

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

Solution: $d = \begin{bmatrix} I(0) \\ \frac{dI}{dx}(0) \\ \vdots \\ \frac{d^nI}{dx^n}(0) \end{bmatrix} = \begin{bmatrix} & & \\ & & \\ & & \end{bmatrix} \begin{bmatrix} I_1 \\ I_2 \\ \vdots \\ I_{2w+1} \end{bmatrix}$

WLS Estimation ⇔ Cross Correlation

weight function $\Omega(x)$



Patch (2w+1 pixels)

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

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

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

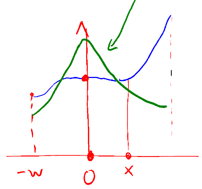
$$\Rightarrow d = \underbrace{\left[(WX)^T(WX)\right]^{-1}(WX)^T W}_{} I$$

one row of this matrix

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

WLS Estimation ⇔ Cross Correlation

weight function $\Omega(x)$



Patch (2w+1 pixels)

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

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

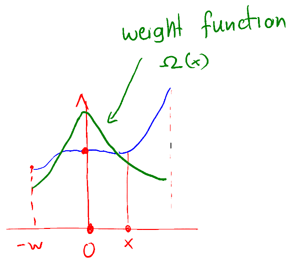
$$WXd = WI \Rightarrow (WX)^T(WX)d = (WX)^T W I$$

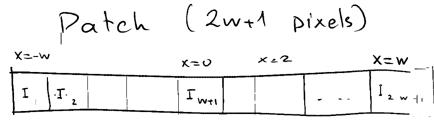
$$\Rightarrow d = \underbrace{\left[(WX)^T(WX)\right]^{-1}(WX)^T W}_{} I$$

one row of this matrix

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

WLS Estimation ⇔ Cross Correlation





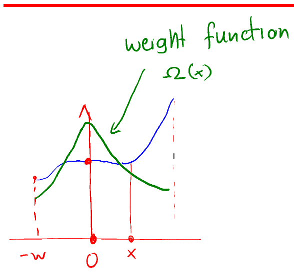
$$\begin{bmatrix} \Omega_1 & & 0 \\ 0 & \ddots & \Omega_{2w+1} \end{bmatrix} I = \begin{bmatrix} \Omega_1 & & \\ & \ddots & \\ & & \Omega_{2w+1} \end{bmatrix} X d$$

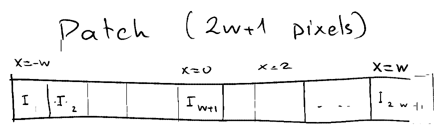
$$\boxed{I^T \cdot t^0 = CC(I, t^0)}$$

I(w=0) can be computed as the cross-correlation of the image patch (I) and the “template” t^0.

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

WLS Estimation ⇔ Cross Correlation





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

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

$\frac{dI}{dx}(0)$ can be computed as the cross-correlation of the image patch (I) and the “template” t^1.

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

WLS Estimation ⇔ Cross Correlation

One can think of the estimation of $\frac{dI}{dx}(0)$ as the "similarity" value between the image patch x_i and the 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

$$cc(x_i, t^1) = X_i^T \cdot t^1$$

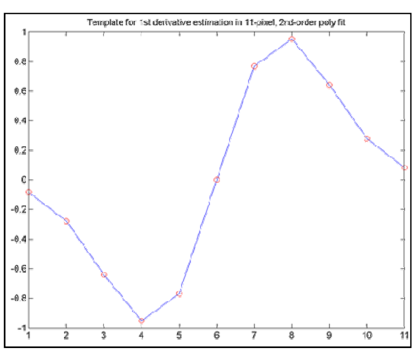
WLS Estimation ⇔ Cross Correlation

One can think of the estimation of $\frac{dI}{dx}(0)$ as the "similarity" value between the image patch x_i and the template t^1.

"Template" patch t^1 ↓

Image patch X_i

50	255	30	80	200	110	50	200	250	100	130

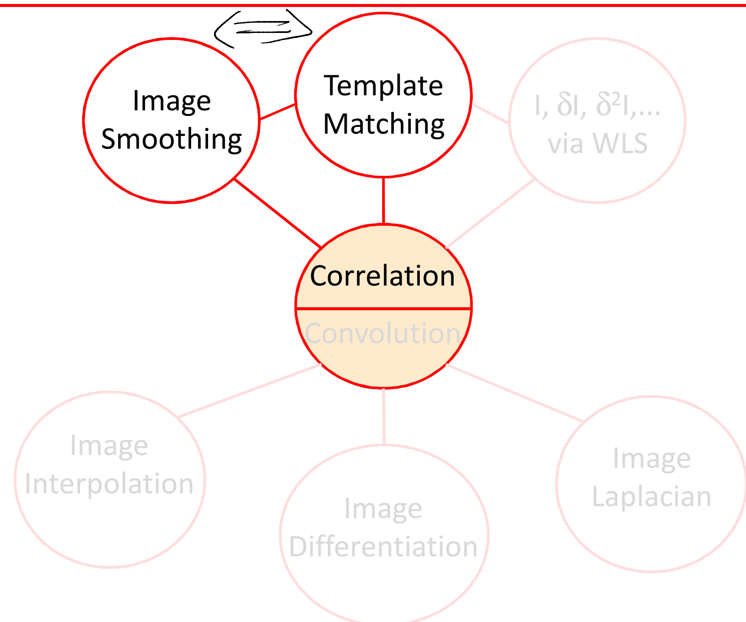


Similarity function

Cross-Correlation

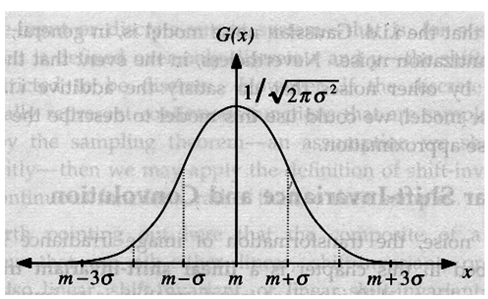
$$cc(x_i, t^1) = X_i^T \cdot t^1$$

Unifying View



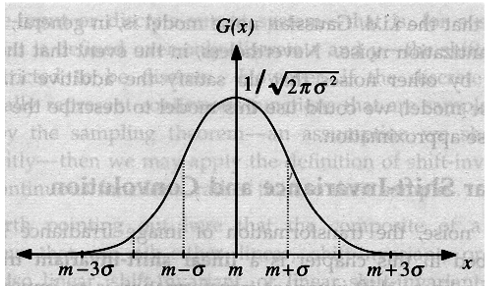
Gaussians in 1D and 2D

1D Gaussian



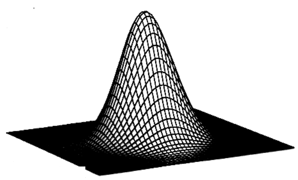
Gaussians in 1D and 2D

1D Gaussian



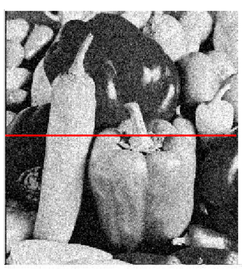
$$G_\sigma(x) = \frac{1}{\sqrt{2\pi\sigma^2}} e^{-\frac{(x-m)^2}{2\sigma^2}}$$

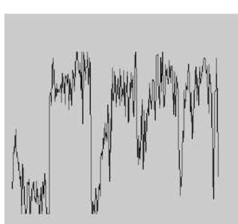
2D Gaussian



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

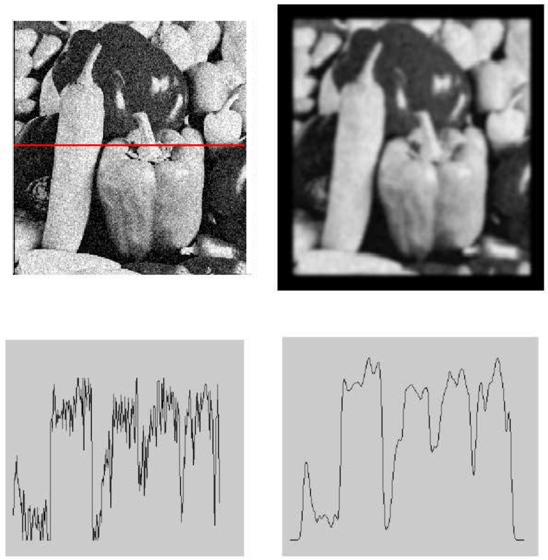
Example: Applying Gaussian Smoothing





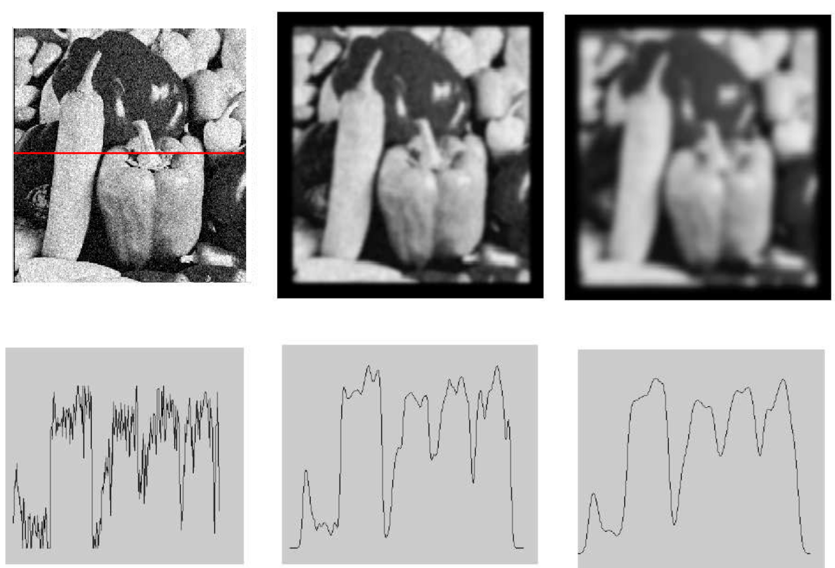
No smoothing

Example: Applying Gaussian Smoothing



No smoothing $\sigma = 2$

Example: Applying Gaussian Smoothing



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

Gaussian Smoothing ⇔ Cross-Correlation

Gaussian smoothing: estimate the Image intensity term as the similarity between the image patch and a Gaussian template

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:

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

Gaussian Smoothing ⇔ Cross-Correlation

Gaussian smoothing: estimate the Image intensity term as the similarity between the image patch and a Gaussian template

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:

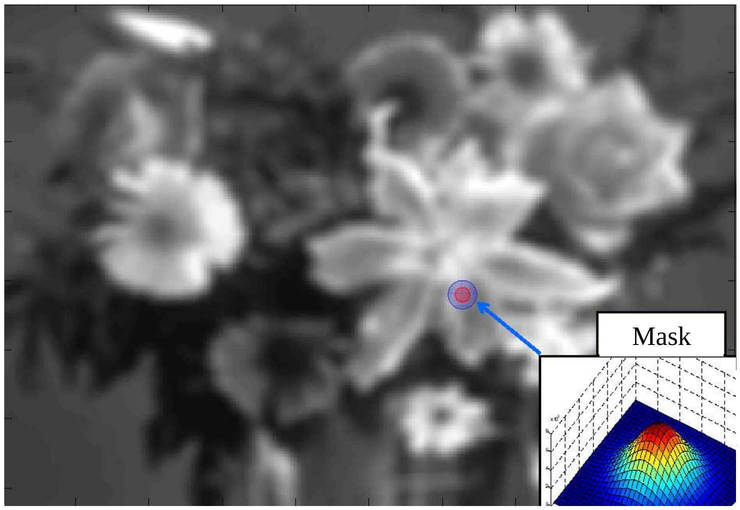
$CC(X_i, \bar{G}_\sigma) =$ 0th order weighted least squares fit using the weight function $\Omega(x) = \sqrt{\bar{G}_\sigma(x)}$

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

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

Averaging vs. Gaussian Smoothing

Result of Cross-Correlation with Gaussian Mask



Unifying View

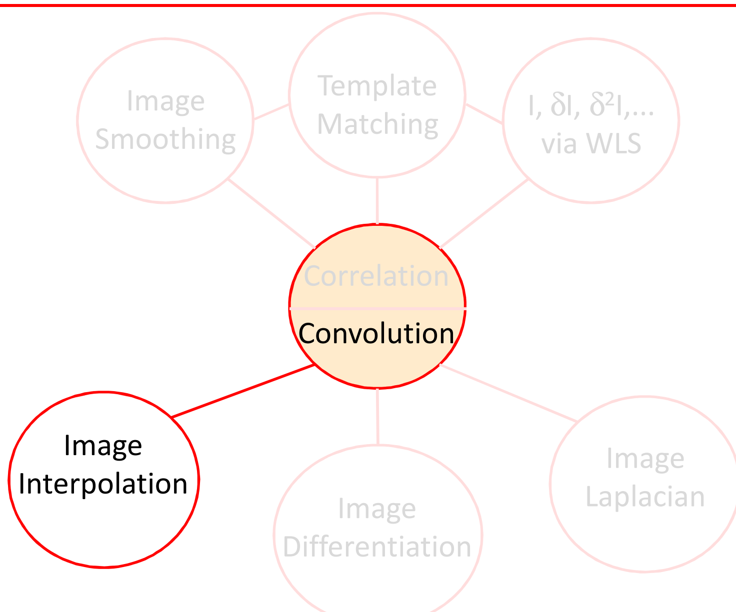


Image Interpolation: Definition

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 at any real $x \in [0, M-1]$

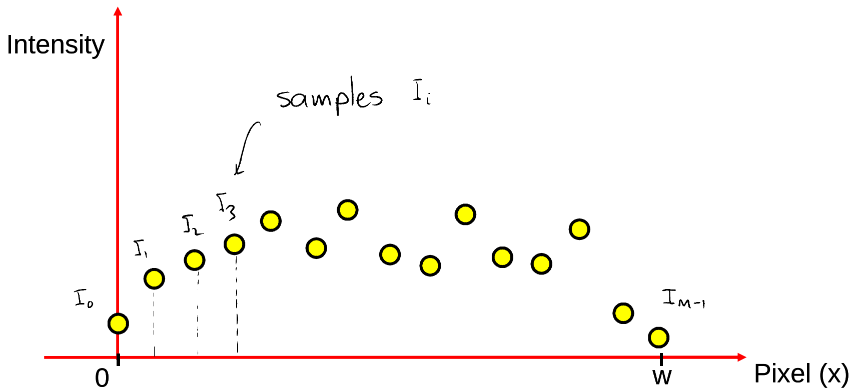


Image Interpolation: Definition

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 at any real $x \in [0, M-1]$

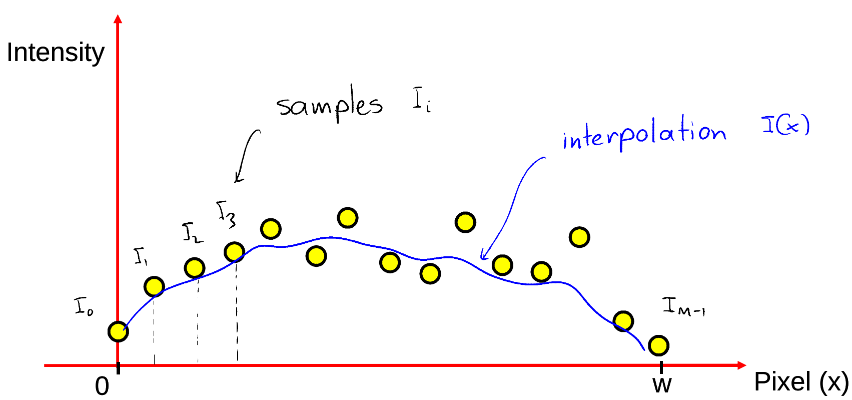


Image Interpolation: Applications

Interpolation applications:

Image warping

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

Image Interpolation: Applications

Interpolation applications:

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	?

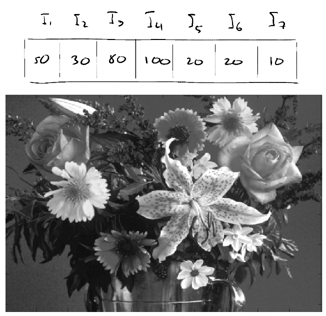
= x2 Stretch ⇒

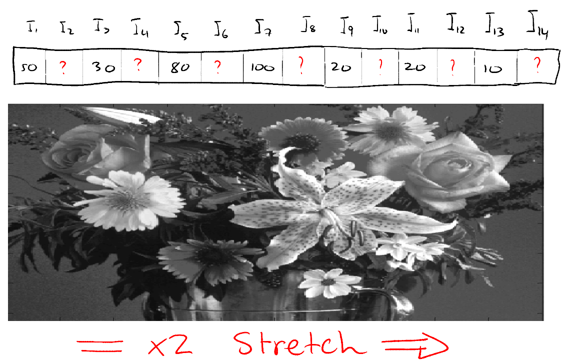
Image Interpolation: Applications

Interpolation applications:

Image warping

Design of differentiation templates



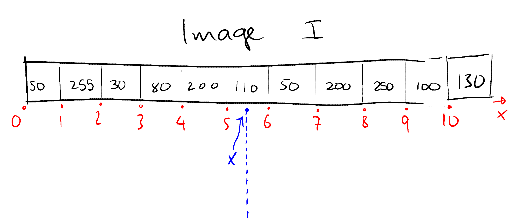


Interpolation: General Expression

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

I_k: intensity at k-th pixel
x-k : distance between x and k pixel
T(x-k): contribution of the k-th pixel

The result of interpolation is a function that can be evaluated at non-integer values of x.



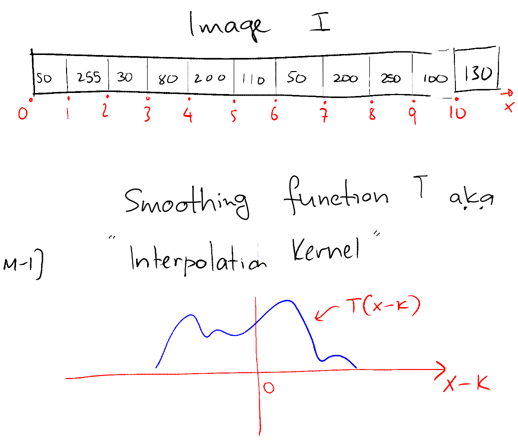
Interpolation: General Expression

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

I_k: intensity at k-th pixel
x-k : distance between x and k pixel
T(x-k): contribution of the k-th pixel

The result of interpolation is 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 $x-k \in [-M+1, M-1]$



Interpolation: General Expression

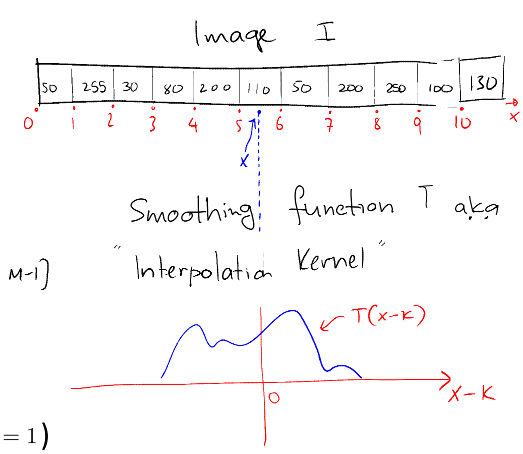
$$(I * T)(x) = \sum_{x=0}^{M-1} I_k T(x-k)$$

I_k: intensity at k-th pixel
x-k : distance between x and k pixel
T(x-k): contribution of the k-th pixel

The result of interpolation is 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 (eg. differentiability and $\int T(x)dx = 1$)

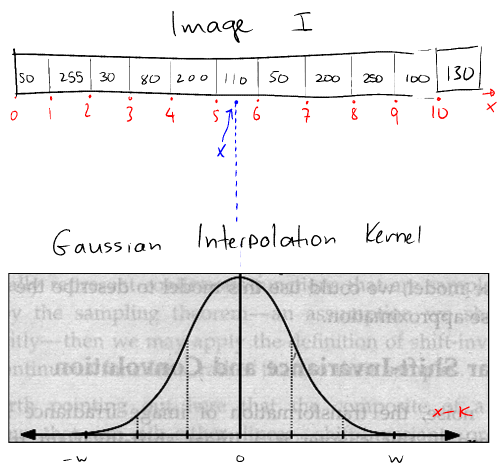


Example #1: Interpolation Using Gaussian Kernel

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

I_k: intensity at k-th pixel
x-k : distance between x and k pixel
G_σ(x-k): contribution of the k-th pixel

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



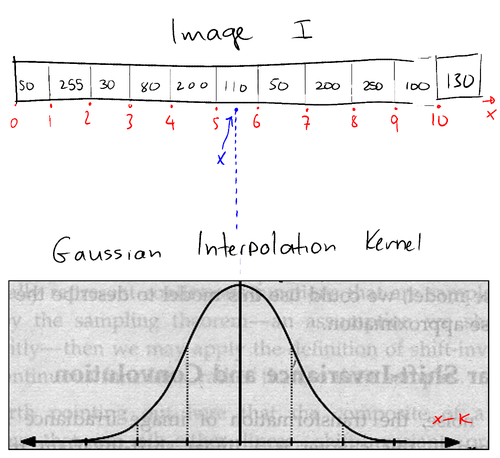
Example #1: Interpolation Using Gaussian Kernel

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

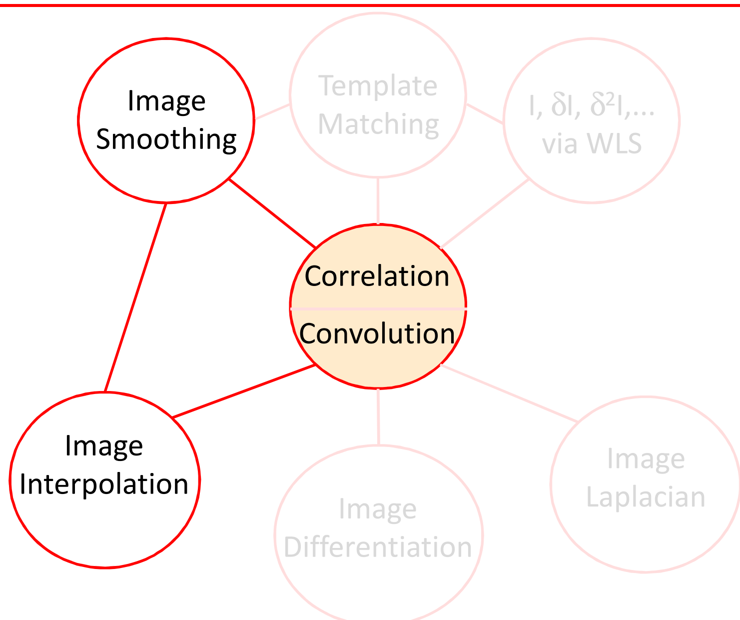
I_k: intensity at k-th pixel
x-k : distance between x and k pixel
G_σ(x-k): contribution of the k-th pixel

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

$$(I * G_\sigma)(x) =$$
$$I[5] \cdot G_\sigma(x-5) +$$
$$I[6] \cdot G_\sigma(x-6) +$$
$$I[4] \cdot G_\sigma(x-4) +$$
$$I[7] \cdot G_\sigma(\underbrace{x-7}_{\text{Distance between the desired location and the known pixel}}) + \ldots$$



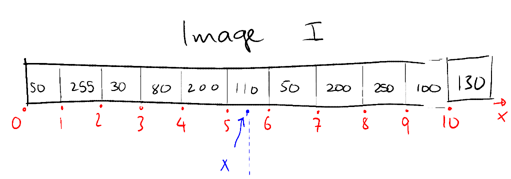
Familiar?



Example #2: Interpolation Using Linear Kernel

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

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

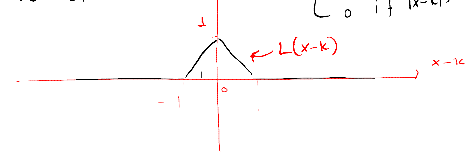


$(I * L)(x) =$

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

Linear Interpolation kernel L

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

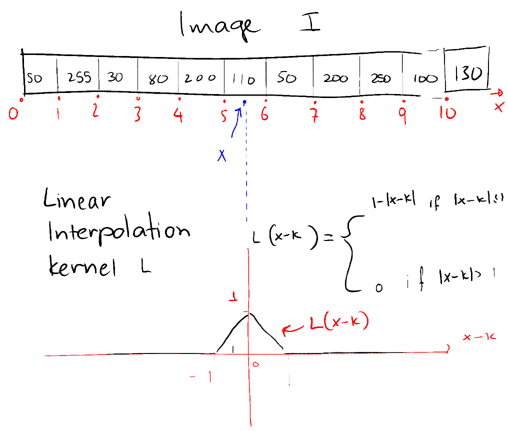


Example #2: Interpolation Using Linear Kernel

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

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

$$(I * L)(x) =$$
$$I[5] \cdot L(x-5) +$$
$$I[6] \cdot L(x-6)$$



Unifying View

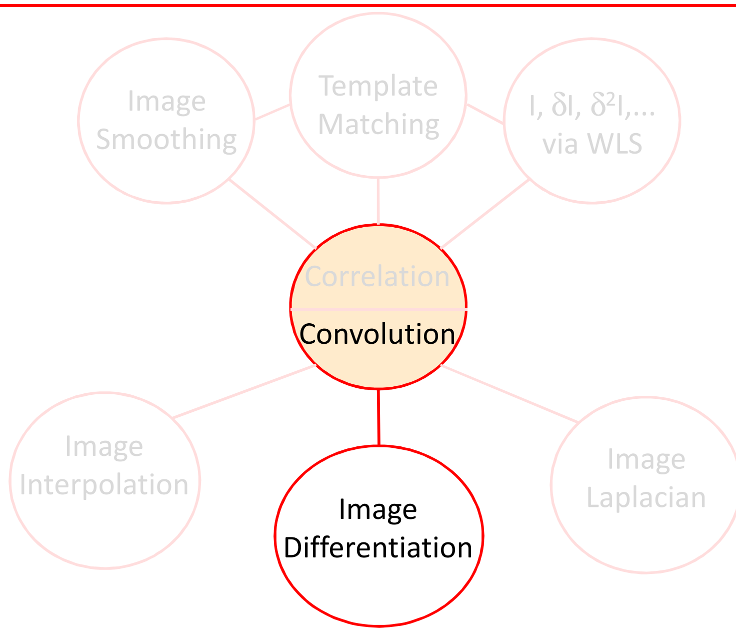


Image differentiation (derivatives computation)

Step 1: Interpolate to define a continuous function

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

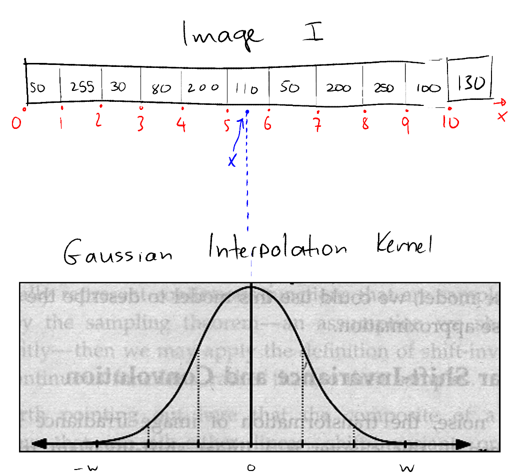
Step 2: Take the derivative of this continuous function

$$\frac{\delta}{\delta x}(I * G_\sigma)(x)$$

Step #1: Interpolate Using Gaussian Kernel

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

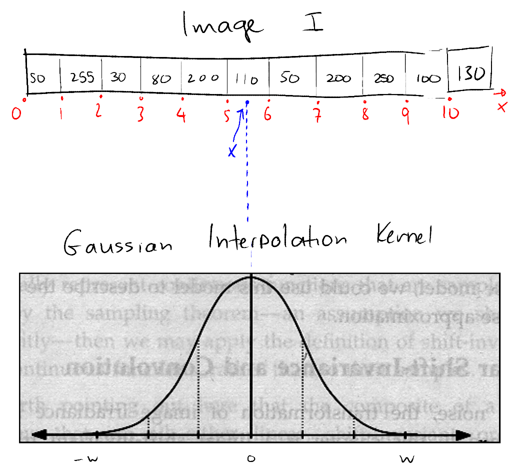
To interpolate, we evaluate the expression at continuous values x



Step #1: Interpolate Using Gaussian Kernel

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

To interpolate, we evaluate the expression at continuous values x



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

Step #2: Differentiate the Interpolated Image

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

continuous

discrete

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

Step #2: Differentiate the Interpolated Image

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

continuous

discrete

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

Step #2: Differentiate the Interpolated Image

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

continuous

discrete

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

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

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

Image Differentiation ⇔ Convolution w/ Gaussian Derivative

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

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

Gaussian

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

Convolution with the Derivative of a Gaussian

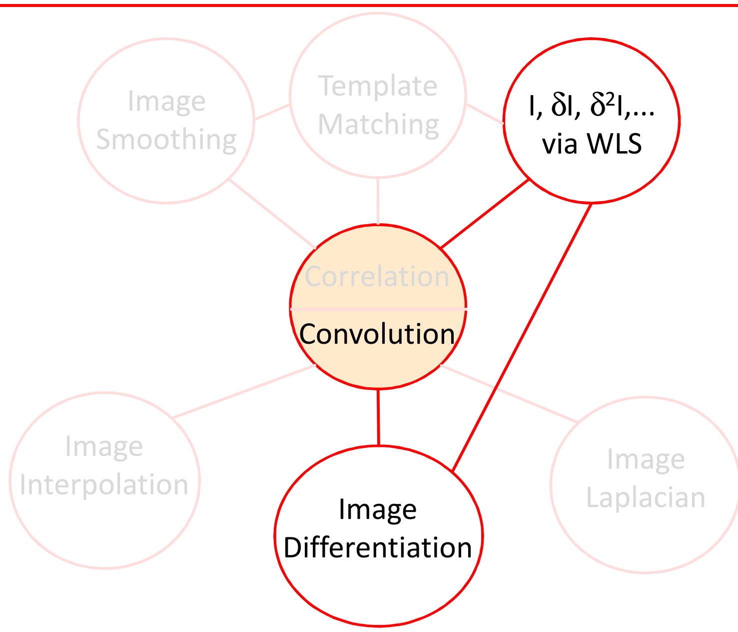
Gaussian

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

First derivative

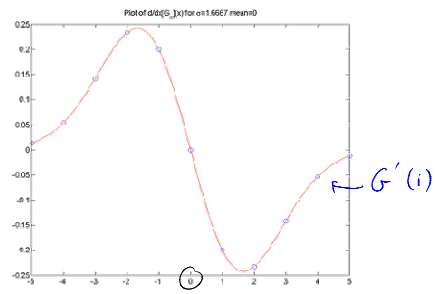
$$\begin{aligned} G'_\sigma(x) &= \frac{\delta}{\delta x}\left(\frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}}\right) \\ &= -\frac{2x}{2\sigma^2}\left(\frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}}\right) \\ &= -\frac{2x}{2\sigma^2} G_\sigma(x) \end{aligned}$$

Familiar?



Convolution with the Derivative of a Gaussian

This is the plot of $G'_\sigma(x)$

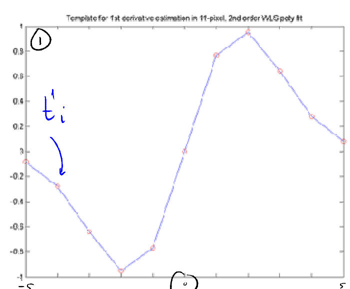


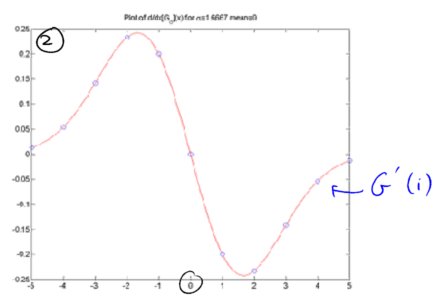
$$\frac{\delta I}{\delta x}(0) = \frac{1}{\sum_{i=-5}^{5} G'(i)} \cdot [G'(-5), \ldots, G'(5)] * [I_1, I_2, \ldots, I_{11}]$$

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 $\frac{dG}{dx}$





$$\frac{dI}{dx}(0) = [t'_{-5}\ t'_{-4} \cdots t'_{5}] \cdot \begin{bmatrix} I_1 \\ I_2 \\ I_3 \\ \vdots \\ I_{11} \end{bmatrix}$$

2nd row of polyfit matrix

$$\frac{\delta I}{\delta x}(0) = \frac{1}{\sum_{i=-5}^{5} G'(i)} \cdot [G'(-5), \ldots, G'(5)] * [I_1, I_2, \ldots, I_{11}]$$

Convolution with the Derivative of a Gaussian

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

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

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

Convolution with the Derivative of a Gaussian

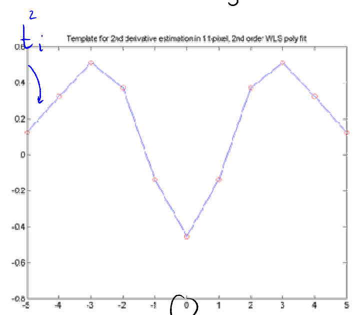
$$
\begin{aligned}
G''_\sigma(x) &= \frac{\delta^2}{\delta x^2}\left(\frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}}\right) \\
&= \left(\frac{x^2}{\sigma^2} - 1\right)\left(\frac{1}{\sigma^2}\right)\left(\frac{1}{\sqrt{2\pi}\sigma} e^{-\frac{x^2}{2\sigma^2}}\right) \\
&= \left(\frac{x^2}{\sigma^2} - 1\right)\left(\frac{1}{\sigma^2}\right) G_\sigma(x)
\end{aligned}
$$

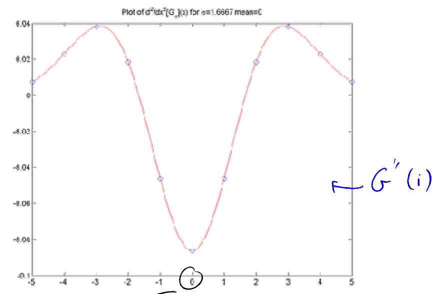
$$\boxed{\frac{d}{dx}(I * G_\sigma)(x) = \left[I * \left(\frac{d^2}{dx^2} G_\sigma \right) \right](x)}$$

Convolution with the Derivative of a Gaussian

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

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





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

3rd row of polyfit matrix

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

Gaussian 2nd derivative evaluated at i=1,..,11

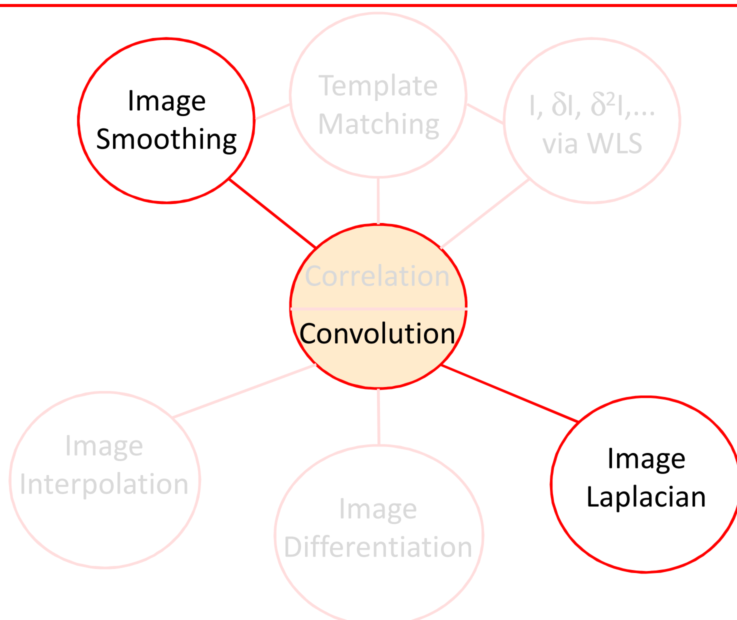
Convolution with the Derivative of a Gaussian

The advantages of estimating derivatives using a convolution with a Derivative of a Gaussian template are:

- The Derivative of G_σ can be computed analytically
- One can control the size of σ (the scale of the smoothing
- One can compute higher order derivatives by convolving with $\frac{d^n}{dx^n} G_\sigma(x)$
- Computations are always dot products (i.e. efficient)

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

Unifying View



Principle #5

Subtracting Gaussian-smoothed versions of an image I at nearby scales σ_1 and σ_2

$$\Longleftrightarrow$$

Computing the Laplacian of I

What does smoothing take away?

$I * G_{\sigma_1}$

What Does Smoothing Take Away?

$I * G_{\sigma_2}$

The Difference-Of-Gaussians (DOG) Filter

The Difference-Of-Gaussians (DOG) Filter

$$I * G_{\sigma_1} - I * G_{\sigma_2} = I * \underbrace{(G_{\sigma_1} - G_{\sigma_2})}$$

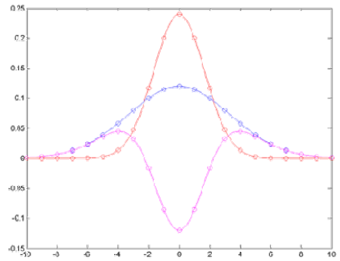
This is called a DOG filter
(Diferencie of Gaussians)

Equivalence of DOG and 2nd Derivative Filter

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

What is $G_{\sigma_1} - G_{\sigma_2}$?

Consider G to be a function of both x and σ



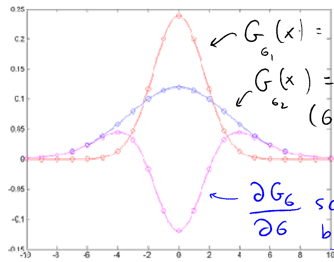
$$G_{\sigma_1} - G_{\sigma_2} = (\sigma_2 - \sigma_1)\frac{\delta G_\sigma}{\delta \sigma}(x, \sigma_1)$$

Equivalence of DOG and 2nd Derivative Filter

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

What is $G_{\sigma_1} - G_{\sigma_2}$?

Consider G to be a function of both x and σ



$G_{\sigma_1}(x) = G(x; \sigma_1)$

$G_{\sigma_2}(x) = G(x; \sigma_2)$

$(\sigma_2 > \sigma_1)$

$\frac{\partial G_\sigma}{\partial \sigma}$ scaled by $(\sigma_2 - \sigma_1)$

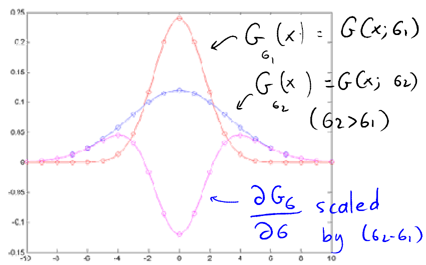
$$G_{\sigma_1} - G_{\sigma_2} = (\sigma_2 - \sigma_1)\frac{\delta G_\sigma}{\delta \sigma}(x, \sigma_1)$$

Equivalence of DOG and 2nd Derivative Filter

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

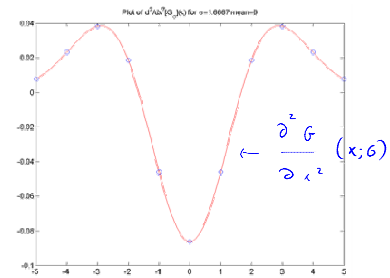
What is $G_{\sigma_1} - G_{\sigma_2}$?

Consider G to be a function of both x and σ



$$G_{\sigma_1} - G_{\sigma_2} =$$
$$(\sigma_2 - \sigma_1)\frac{\delta G_\sigma}{\delta \sigma}(x, \sigma_1)$$

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

Using approximate differences, the derivative can be computed as:

$$\frac{\delta G(x,\sigma)}{\delta \sigma} = \frac{G(x,\sigma_2) - G(x,\sigma_1)}{\sigma_2 - \sigma_1}$$

From where:

$$G_{\sigma_1} - G_{\sigma_2} =$$
$$(\sigma_2 - \sigma_1)\frac{\delta G_\sigma}{\delta \sigma}(x, \sigma_1)$$

$$\text{Compare } \frac{\delta G}{\delta \sigma} \text{ and } \frac{\delta^2 G}{\delta x^2}$$

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

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

$$\frac{\delta G}{\delta \sigma} = \sigma \frac{\delta^2 G}{\delta x^2}$$

$\Downarrow$

$\Longrightarrow$

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

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:

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

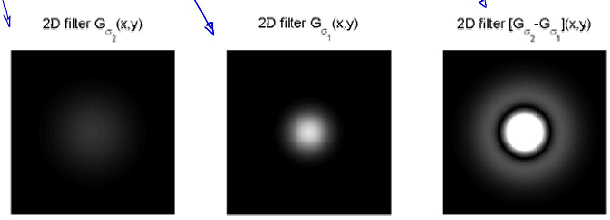
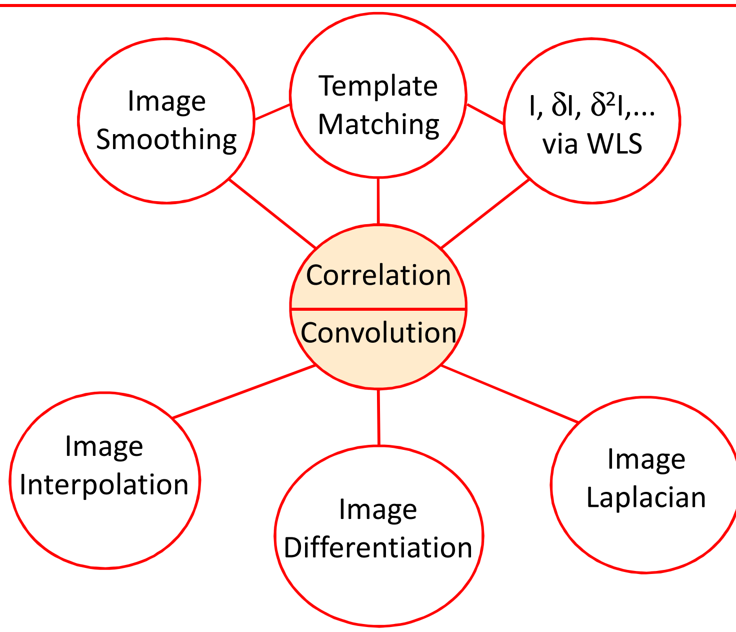


Image Laplacian from the difference of two Gaussians:

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

Unifying View



Topic 10:

Feature Detection & Image Matching

- Introduction to the image matching problem
- Image matching using SIFT features
- The SIFT feature detector
- The SIFT descriptor

SIFT

SIFT: Scale Invariant Feature Transform

Developed by David Lowe in 1999

One of the most powerful representations for feature detection and matching.

Widely used in applications that range from robotics to image retrieval and recognition, image stitching, camera calibration and video analysis.

The Image Matching Problem

Image I

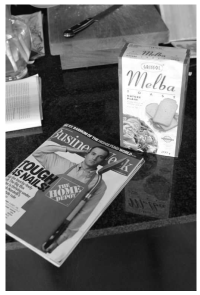


Image I'

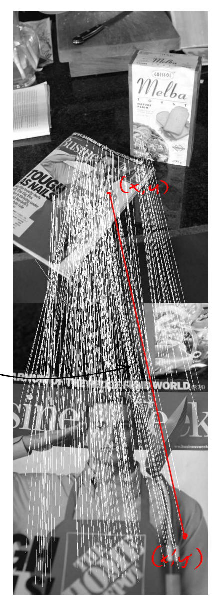
Goal: to identify "features" or patches in image I, that appear in another image I'

The Image Matching Problem

Image I

Image I'

Lines indicate a correspondence between location (x,y) in image I, and location (x', y') in image I'.



The Image Matching Problem

Is it possible to solve this problem by direct template matching between the two images?

The Image Matching Problem

Is it possible to solve this problem by direct template matching between the two images?

The Image Matching Problem

Is it possible to solve this problem by direct template matching between the two images?

Yes, but it would be impossibly inefficient.

(i.e. must search over all possible pairs of patches)

Feature-Based Image Matching

Feature detection & matching

Feature-Based Image Matching

Image I

Image I′

Detect features in I → Match features across the two images

Detect features in I′ → Match features across the two images

→

Errors in Feature-Based Image Matching

Image I

Image I′

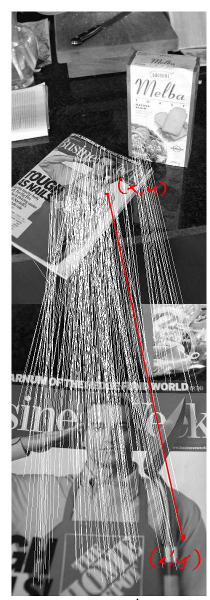
- In general some/many of these correspondences may be incorrect
- Two types of error:

① False positive matches

algorithm returns a correspondence between 2 locations where none exists

② False negative matches

algorithm fails to detect a correspondence between two instances of the same feature/patch



Errors in Feature-Based Image Matching

Image I

Image I'

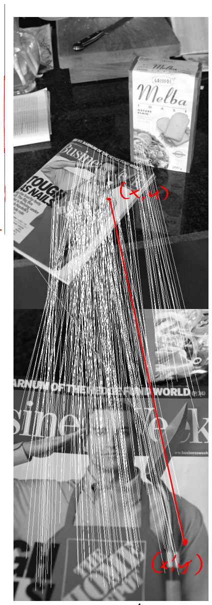
Goal: Minimize false positives
AND false negatives
across a wide range of imaging conditions

① False positive matches

algorithm returns a correspondence between 2 locations where none exists

② False negative matches

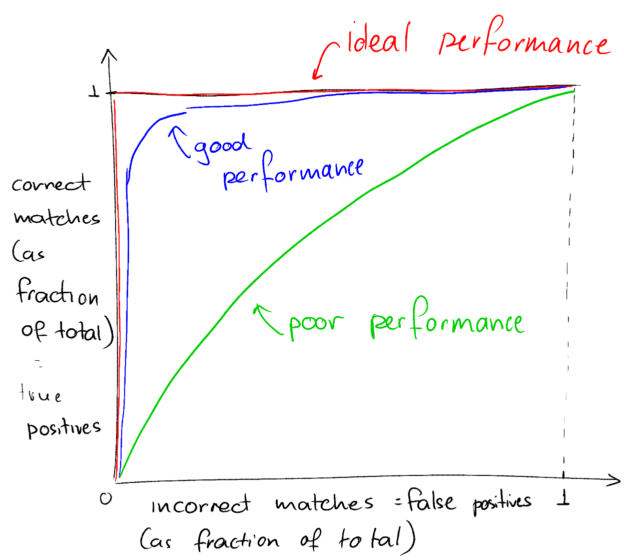
algorithm fails to detect a correspondence between two instances of the same feature/patch



Evaluating a Feature Detector's Performance

Image I

Image I'

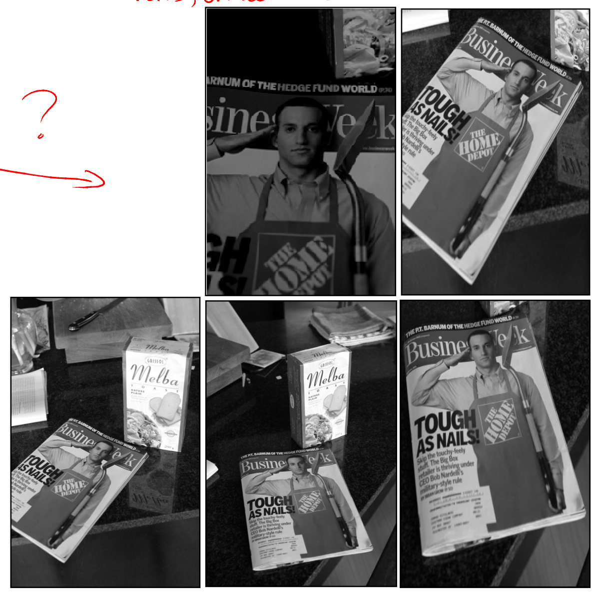


Feature Matching & Transformation Invariance

Source image I

?

"Transformed" source images



A useful feature detector and matching algorithm must be insensitive to a wide range of image transformations

Transformation-Invariant Feature Detectors

Source image I

?

"Transformed" source images

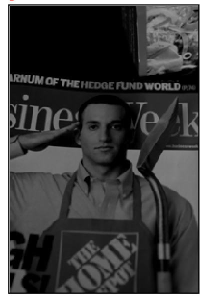
A feature detector is called invariant to a certain image transformation if it can reliably detect features in a source image.

Transformation-Invariant Feature Detectors

Source image I

?

"Transformed" source images



Brightness transformation

A feature detector is called invariant to a certain image transformation if it can reliably detect features in a transformed version of the source image.

Transformation-Invariant Feature Detectors

Source image I

?

"Transformed" source images

Distortion due to change in viewpoint

A feature detector is called invariant to a certain image transformation if it can reliably detect features in a transformed version of the source image.

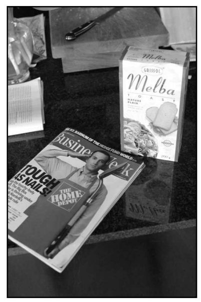
Transformation-Invariant Feature Detectors

Source image I

"Transformed" source images

?

A feature detector is called invariant to a certain image transformation if it can reliably detect features in a transformed version of the source image.



Distortion due to change in viewpoint & magnification (ie scale)

Topic 10:

Feature Detection & Image Matching

- Introduction to the image matching problem
- Image matching using SIFT features
- The SIFT feature detector
- The SIFT descriptor

We’ll define the SIFT descriptor next class.