The Gaussian Pyramid

The representation is based on 2 basic operations

Aside: Downsampling = any linear transformation

[ ] = [matrix] [ ]

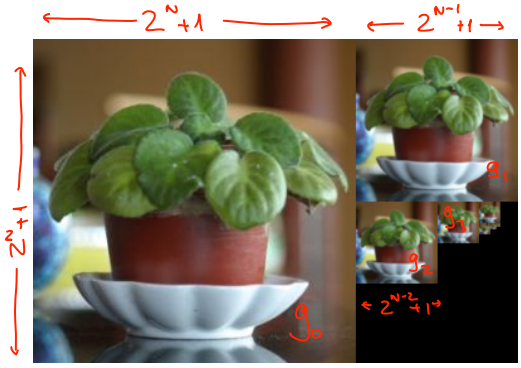
downsampled image; # rows < # columns; original image

① Smoothing

Smooth the image with a sequence of smoothing filters each of which has twice the radius of the previous one

② Downsampling

Reduce image size by 1/2 after each smoothing



Topic 6.1:

Gaussian & Laplacian Pyramids

- The gaussian pyramid (intro)
- The convolution operation
- Constructing the gaussian pyramid
  - The REDUCE() function
- Constructing the Laplacian pyramid
  - The EXPAND() function

Aside ...

Matrix Multiplication $\Longleftrightarrow$ The Convolution Operation

$\Updownarrow$

Template Matching Using Sliding Window Algorithm

Image Smoothing Using Averaging Masks

Original Image

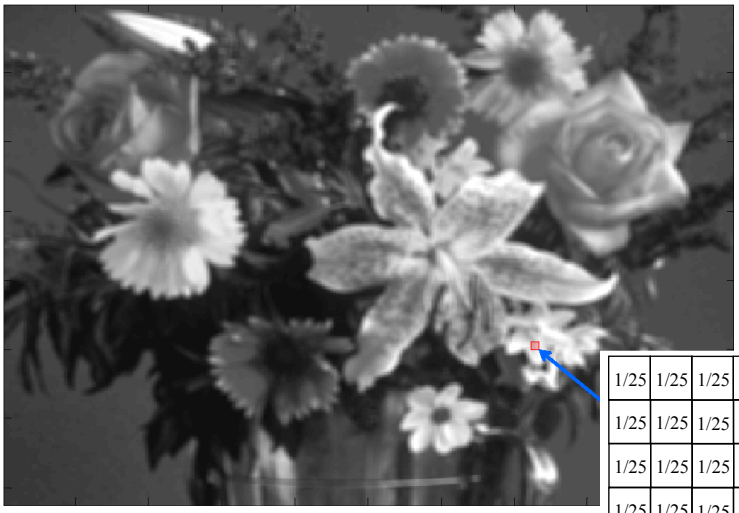
Image Smoothing Using Averaging Masks

Result of Cross-Correlation with 3x3 Mask

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

Image Smoothing Using Averaging Masks

Result of Cross-Correlation 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

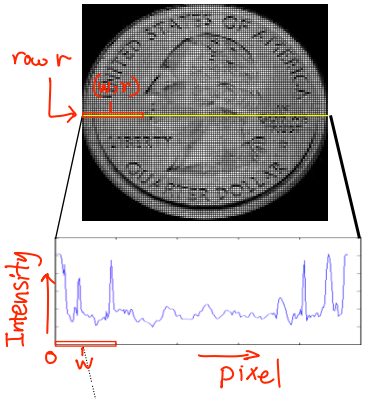
Image Smoothing Using Averaging Masks

Result of Cross-Correlation with 15x15 Mask

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



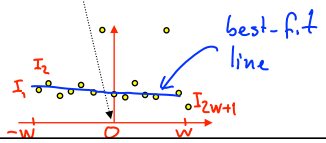


Image Cross Correlation ⇔ Matrix Multiplication

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

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

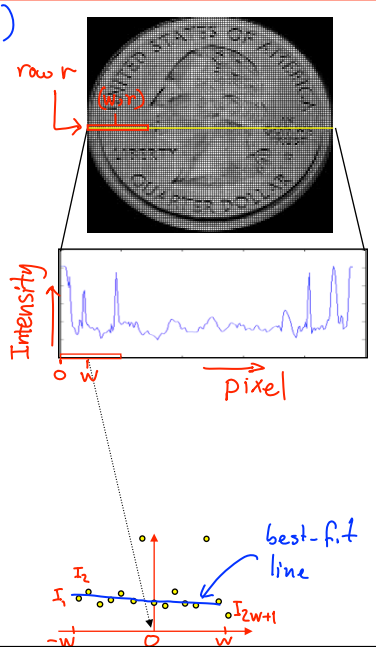


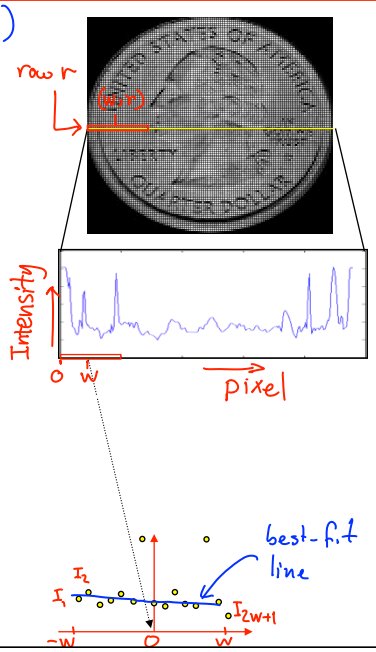
Image Cross Correlation ⇔ Matrix Multiplication

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

cross-correlation at pixel w+1

pixel w
$[T_{-w}\ T_{-w+1} \cdots T_0 \cdots T_w\ 0 \cdots 0]$
$[0\ T_{-w}\ T_{-w+1} \cdots T_{w-1}\ T_w \cdots 0]$
pixel w+1

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



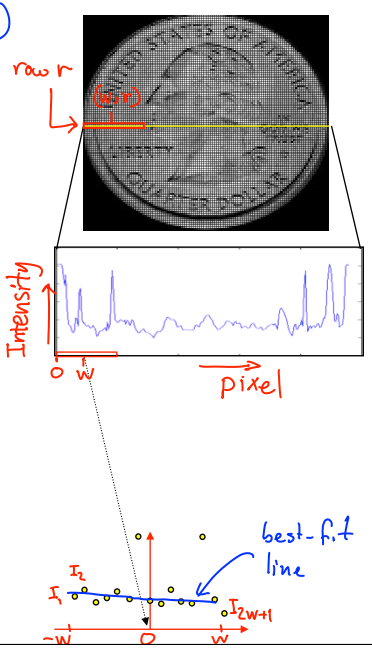
The Toeplitz Matrix of a Template

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

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

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

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



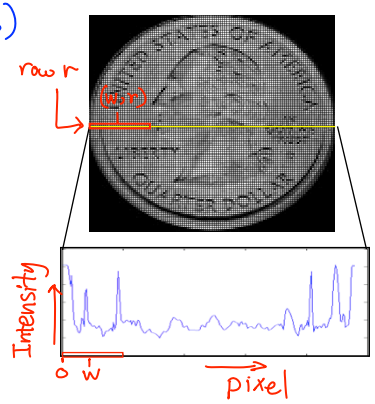
The Toeplitz Matrix of a Template

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

what about pixels 0, ..., w-1 & M-w, ..., M-1?

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

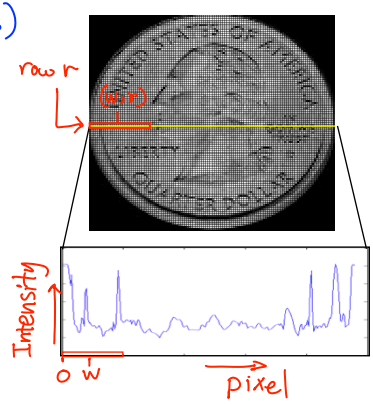
each row is a 1-pixel right shift of the row above it
(called a TOEPLITZ matrix)



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)

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

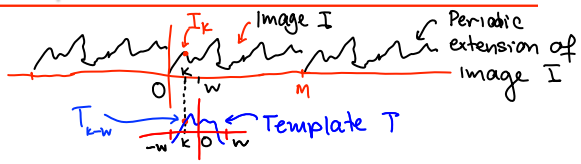
each row is a 1-pixel right shift of the row above it
(called a TOEPLITZ matrix)

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:

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



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

Examples

$$J_w = \sum_{k=0}^{M-1} I_k \cdot T_{k-w}$$

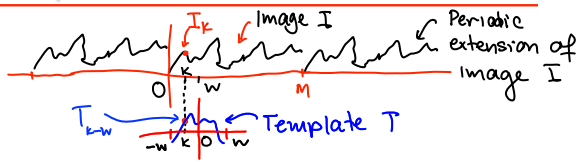
$$J_{w+1} = \sum_{k=0}^{M-1} I_k \cdot T_{k-w-1} = \sum_{k=0}^{M-1} I_k \cdot T_{k-(w+1)}$$

Cross-Correlation Expressed as a Sum

General sum notation:

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

$0 \le i \le M-1$



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

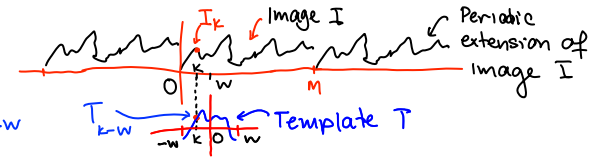
Alternative sum notation:

$$J_i = \sum_{l=0}^{M-1} I_{l+i} T_l$$

$0 \le i \le M-1$

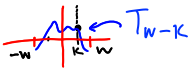
* obtained by substituting $l = k-i$ in General Sum Notation formula

The Convolution Operation



$$J_w = \sum_{k=0}^{M-1} I_k T_{k-w}$$

$$(I * T)_w = \sum_{k=0}^{M-1} I_k \cdot T_{w-k}$$



Cross-correlation:

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

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

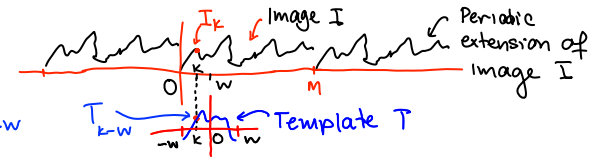
Convolution:

$$(I * T)_i = \sum_{k=0}^{M-1} I_k \cdot T_{i-k}$$

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

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

The Convolution Operation



$$J_w = \sum_{k=0}^{M-1} I_k T_{k-w}$$

$$(I*T)_w = \sum_{k=0}^{M-1} I_k \cdot T_{w-k}$$

Cross-correlation:

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

$$[\,T_{(-w)}\ T_{-w+1} \cdots T_0 \cdots T_{(w)}\ 0 \cdots 0\,]$$

Convolution:

$$(I*T)_i = \sum_{k=0}^{M-1} I_k \cdot T_{i-k}$$

$$[\,T_{(w)}\ T_{w-1} \cdots T_0 \cdots T_{(-w)}\ 0 \cdots 0\,]$$

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

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

Convolution:

$$(I*T)_i = \sum_{k=0}^{M-1} I_k \cdot T_{i-k}$$

Properties (exercise: prove them)

1. For symmetric masks T convolution is equal to cross-correlation:
   $CC(I,T) = I*T$ when $T_i = T_{-i}$
2. Commutativity:
   $I*T = T*I$
3. Linearity:
   $(aI+bJ)*T = a(I*T)+b(J*T)$ for any constants a, b

The Convolution Operation

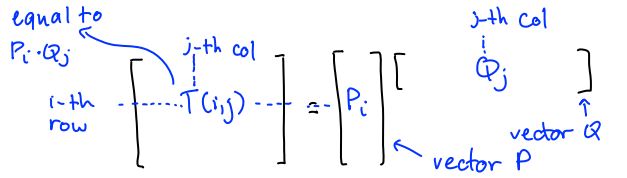
Convolution in 2D (MxN image)

$$(I*T)(i,j) = \sum_{k=0}^{M-1} \sum_{l=0}^{N-1} I(k,l) \cdot T(i-k, j-l)$$

Very important special case:

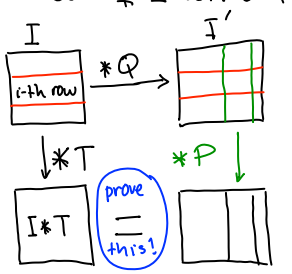
Definition (Separable 2D mask)

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

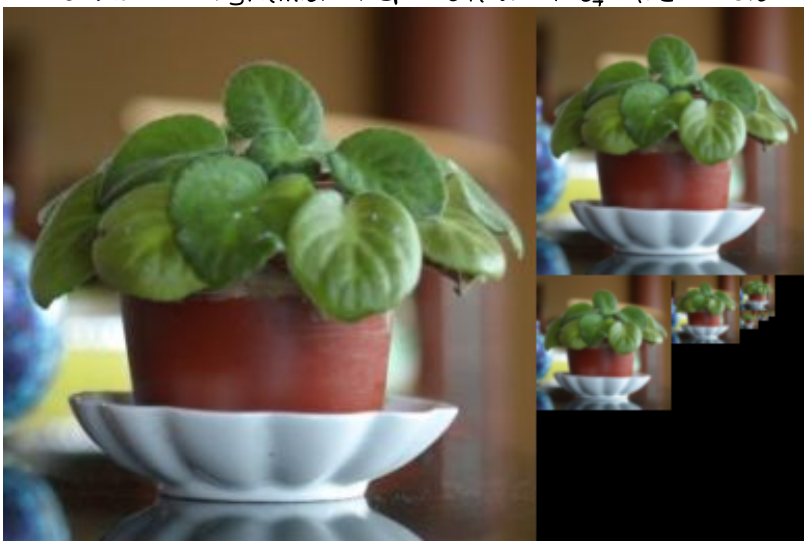
Topic 6.1:

Gaussian & Laplacian Pyramids

- The gaussian pyramid (intro)
- The convolution operation
- Constructing the gaussian pyramid
  - The REDUCE() function
- Constructing the Laplacian pyramid
  - The EXPAND() function

The Gaussian Pyramid

Gaussian Pyramid Representation of the Photo



Original photo

The Gaussian Pyramid

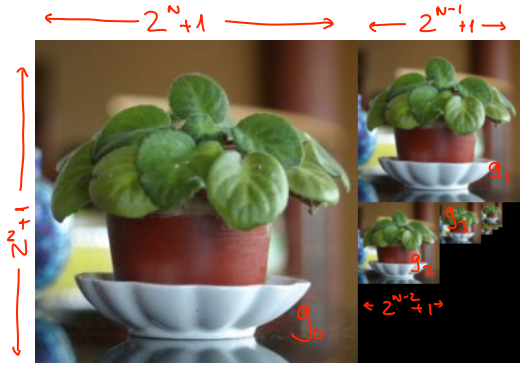
Goal: Develop representation to decompose images into information at multiple scales, to extract features or structures of interest, to attenuate noise

Input:

Image I of size $(2^N+1) \times (2^N+1)$

Output:

- N images $g_0, \ldots, g_{N-1}$
- g_ℓ has size $(2^{N-\ell}+1) \times (2^{N-\ell}+1)$



The Gaussian Pyramid

The representation is based on 2 basic operations

Aside: Downsampling = any linear transformation

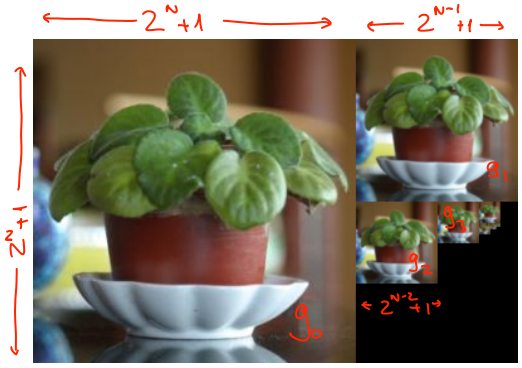
$$\begin{bmatrix} \text{downsampled image} \end{bmatrix} = \begin{bmatrix} \text{matrix} \\ \#\text{rows} < \#\text{columns} \end{bmatrix} \begin{bmatrix} \text{original image} \end{bmatrix}$$

① **Smoothing**

Smooth the image with a sequence of smoothing filters each of which has twice the radius of the previous one

② **Downsampling**

Reduce image size by ½ after each smoothing



Operation #1: Smooth Image at N-1 Scales

Original photo g_0

$$\hat{g}_1 = w * g_0$$

a 5x5 filter

$$\hat{g}_1(i,j) = \sum_{m=-2}^{2} \sum_{n=-2}^{2} w(m,n) \cdot g_0(i-m, j-n)$$

Operation #1: Smooth Image at N-1 Scales

$\hat{g}_1$

$$\hat{g}_2 = w * \hat{g}_1$$

$$= w * (w * g_0)$$

$$= (w * w) * g_0$$

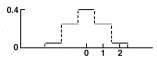
can be thought of as a filter

$h = w * w$

whose radius is twice that of w

In 1D:

w:



w*w:

Operation #1: Smooth Image at N-1 Scales

$\hat{g}_2$

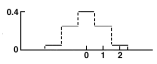
$$\hat{g}_3 = w * \hat{g}_2$$

$$= (w * w * w) * g_0$$

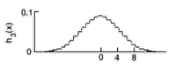
radius is 4 times that of w

In 1D:

w:

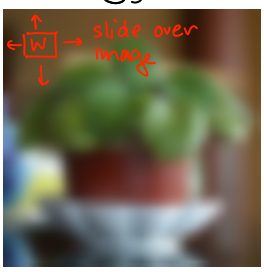


w*w*w:



Operation #1: Smooth Image at N-1 Scales

$\hat{g}_3$

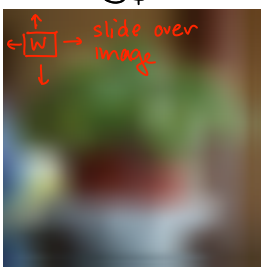


$$\hat{g}_4 = w * \hat{g}_3$$

$$= (w * w * w * w) * g_0$$

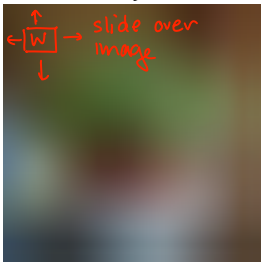
Operation #1: Smooth Image at N-1 Scales

$\hat{g}_4$



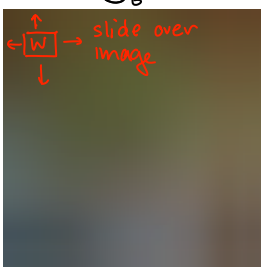
Operation #1: Smooth Image at N-1 Scales

$\hat{g}_5$



Operation #1: Smooth Image at N-1 Scales

$\hat{g}_6$



Operation #1: Smooth Image at N-1 Scales

$\hat{g}_7$

Smoothing Filter in 1D: Derivation from 4 Criteria

① $\hat{w}$ always has 5 elements (a.k.a "5-tap" filter)

② $\hat{w}$ symmetric about 0:

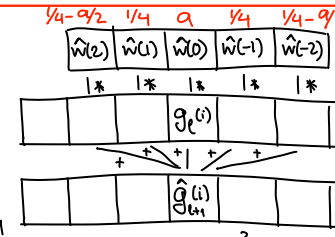
$$\hat{w} = \begin{bmatrix} c \\ b \\ a \\ b \\ c \end{bmatrix} = \begin{bmatrix} 1/4 - a/2 \\ 1/4 \\ a \\ 1/4 \\ 1/4 - a/2 \end{bmatrix}$$

③ applying $\hat{w}$ to a constant image does not change it

$$\Leftrightarrow \sum_{m=-2}^{2} \hat{w}(m) = 1$$

$$\Leftrightarrow a + 2b + 2c = 1$$

$\hat{w}$



$$\hookrightarrow \sum_{m=-2}^{2} \hat{w}(m) \cdot \hat{g}_\ell(i-m)$$

④ Equal contribution.

$$a + 2c = 2b = 1/2$$

To satisfy Criteria 1-4 we have 2 eqs & 3 unknowns
$\Rightarrow$ a remains free parameter

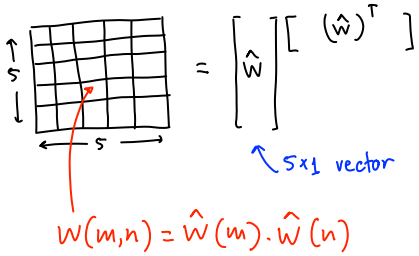
$\hat{w}(2) = \hat{w}(-2) = \frac{1}{4} - \frac{a}{2}$, $\hat{w}(-1) = \hat{w}(1) = \frac{1}{4}$

usually $a \in [0.3, 0.6]$

Defining the Smoothing Filter in 2D

$\hat{g}_1 = w * g_0$

w is a separable smoothing filter defined by $\hat{w}$



$w(m,n) = \hat{w}(m) \cdot \hat{w}(n)$

g_0

Exploiting separability to compute $\hat{g}_1$:

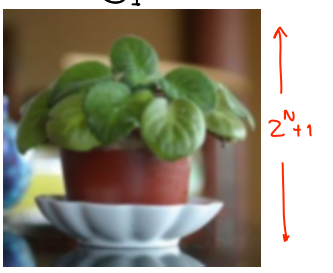
1. Convolve each row of g_0 with $\hat{w}$
2. Convolve the columns of the result with $\hat{w}$ again

Operation #2: Downsample the Smoothed Image

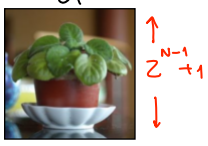
Since $\hat{g}_1$ contains less image detail, we downsample it by 2 (i.e. store every other pixel)

$$g_1(i,j) = \hat{g}_1(2i, 2j)$$

$\hat{g}_1$



g_1



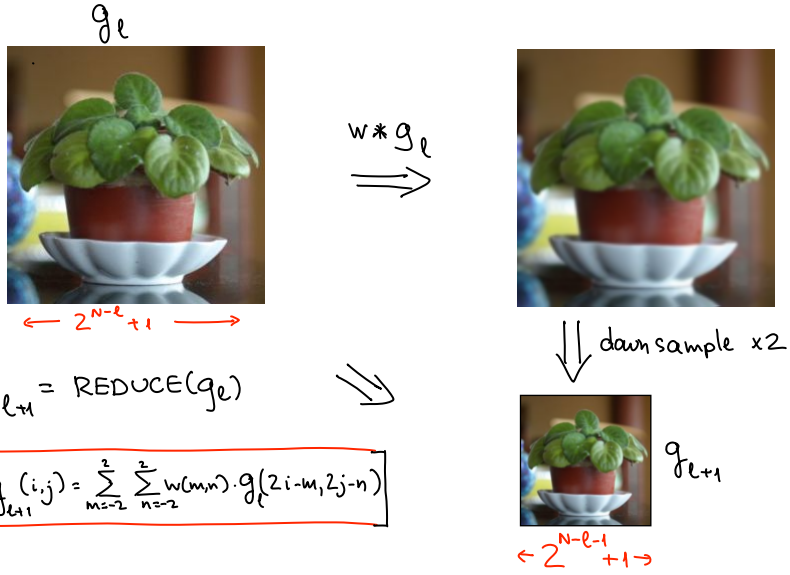
Topic 6.1:

Gaussian & Laplacian Pyramids

- The gaussian pyramid (intro)
- The convolution operation
- Constructing the gaussian pyramid
  - The REDUCE() function
- Constructing the Laplacian pyramid
  - The EXPAND() function

Operations #1 & #2: The REDUCE() Function

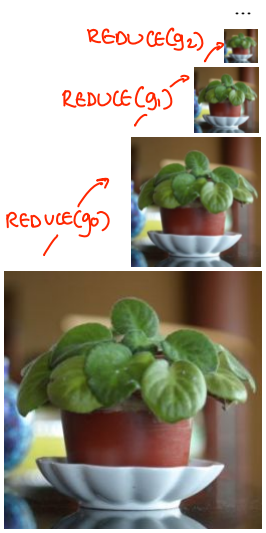
Smoothing & downsampling are combined into a single REDUCE function



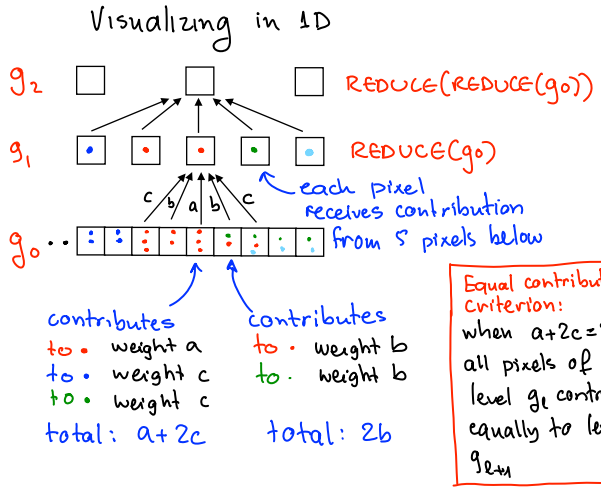
$g_{l+1} = \text{REDUCE}(g_l)$

$$g_{l+1}(i,j) = \sum_{m=-2}^{2} \sum_{n=-2}^{2} w(m,n) \cdot g_l(2i-m, 2j-n)$$

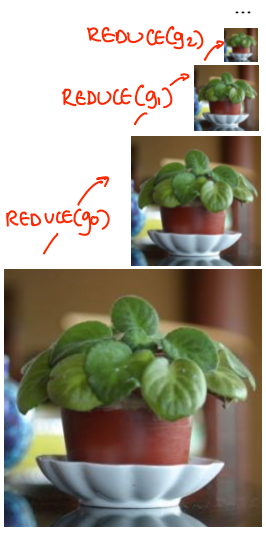
The REDUCE() function



The REDUCE() function

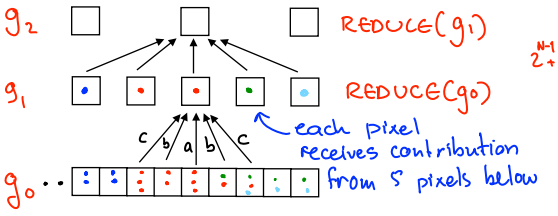


Equal contribution criterion:
when $a+2c=2b$
all pixels of level g_i contribute equally to level g_{i+1}



The REDUCE() function

Visualizing in 1D



REDUCE function in matrix notation (1D)

$$\underset{2^{N-1}+1}{\downarrow}\left[g_1\right] = \underset{\leftarrow 2^N+1 \rightarrow}{\begin{bmatrix} 100\cdots & 0 \\ 001\cdots & \cdots 0 \\ 00001\cdots & 0 \end{bmatrix}} \underset{\leftarrow 2^N+1 \rightarrow}{\begin{bmatrix} c\,b\,a\,b\,c & 0 \\ & c\,b\,a\,b\,c \\ & \ddots \\ 0 & \end{bmatrix}} \left[g_0\right]$$

downsampling matrix D_0

convolution matrix C_0

$$g_1 = D_0 \cdot C_0 g_0$$

General expression:

$$g_{l+1} = D_l \cdot C_l \cdot g_l$$

The Gaussian Pyramid

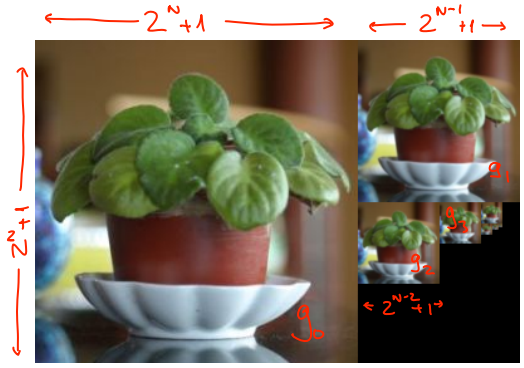
Goal: Develop representation to decompose images into information at multiple scales, to extract features or structures of interest, to attenuate noise

Input:

Image I of size $(2^N+1) \times (2^N+1)$

Output:

- N images $g_0, \ldots, g_{N-1}$
- g_l has size $(2^{N-l}+1) \times (2^{N-l}+1)$



Operations #1 & #2: The REDUCE() Function

Smoothing & downsampling are combined into a single REDUCE function

g_l

$\leftarrow 2^{N-l}+1 \rightarrow$

$w * g_l$

downsample x2

$g_{l+1} = \text{REDUCE}(g_l)$

$$g_{l+1}(i,j) = \sum_{m=-2}^{2} \sum_{n=-2}^{2} w(m,n) \cdot g_l(2i-m, 2j-n)$$

g_{l+1}

$\leftarrow 2^{N-l-1}+1 \rightarrow$

What Does Smoothing Take Away?

$g_0 = I$

original photo

What Does Smoothing Take Away?

Smoothed photo

$\hat{g}_1 = w * g_0$

What Does Smoothing Take Away?

$g_0 - \hat{g}_1$

Details in g_0 that were <u>not</u> represented in $\hat{g}_1$

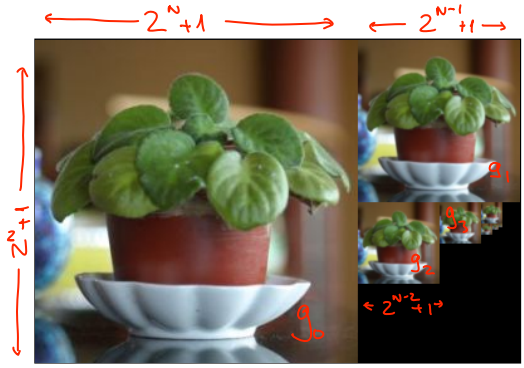
Topic 6.1:

Gaussian & Laplacian Pyramids

- The gaussian pyramid (intro)
- The convolution operation
- Constructing the gaussian pyramid
  - The REDUCE() function
- Constructing the Laplacian pyramid
  - The EXPAND() function

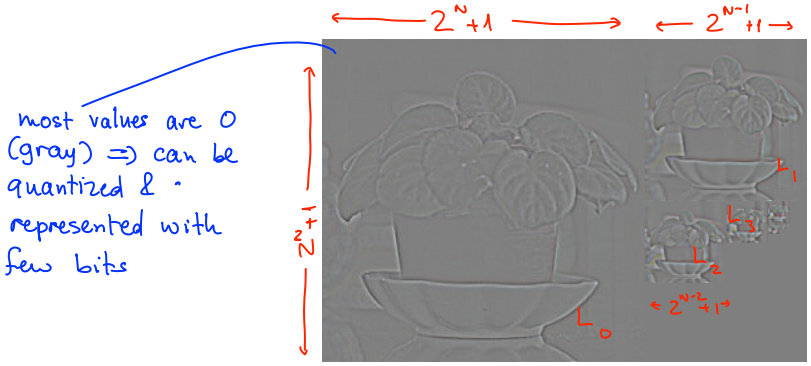
The Laplacian Pyramid

Idea: Rather than store the smoothed images, store only the difference between levels g_ℓ and $g_{\ell+1}$



The Laplacian Pyramid

Idea: Rather than store the smoothed images, store only the difference between levels g_ℓ and $g_{\ell+1}$

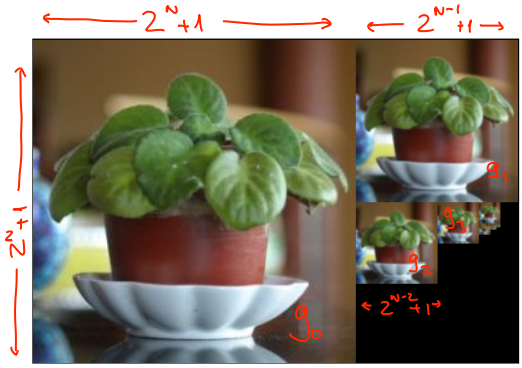


The Laplacian Pyramid

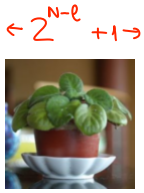
Idea: Rather than store the smoothed images, store only the difference between levels g_ℓ and $g_{\ell+1}$

- To do this, we must compare adjacent levels g_ℓ and $g_{\ell+1}$
- But $g_{\ell+1}$, g_ℓ are not the same size!

  ⇒ Expand $g_{\ell+1}$ to make it equal size to g_ℓ



Operation #3: The EXPAND() Function



g_ℓ

EXPAND( )

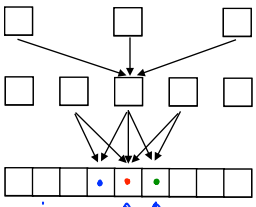
EXPAND(g_ℓ)

$$\text{EXPAND}(g_\ell) = 4\sum_{m=-2}^{2}\sum_{n=-2}^{2} w(m,n)\, g_\ell\left(\frac{i-m}{2}, \frac{j-n}{2}\right)$$

The EXPAND() function

- The function upsamples level g_ℓ by doubling its size from $(2^{N-\ell}+1)\times(2^{N-\ell}+1)$ to $(2^{N-\ell+1}+1)\times(2^{N-\ell+1}+1)$

Visualizing in 1D



EVEN pixels receive contribution from 3 pixels above them (total weight = $2c+a=1/2$)

ODD pixels receive contribution from 2 pixels above them (total weight = $2b=1/2$)

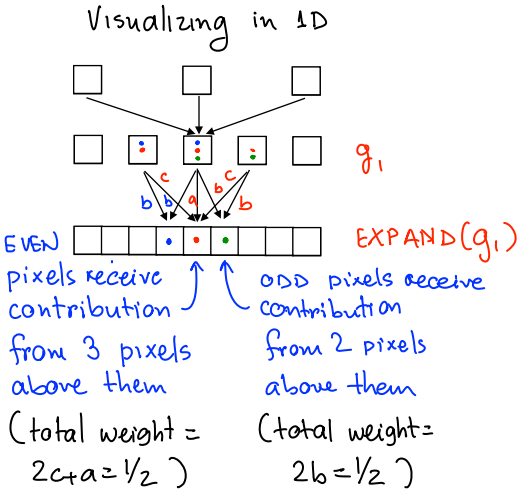
General expression in 1D

$$\text{EXPAND}(g_\ell)(i) = 2\sum_{m=-2}^{2} \hat{w}(m)\cdot g_\ell\left(\underbrace{\frac{i-m}{2}}\right)$$

evaluated only for i, m where $\frac{i-m}{2}$ is an integer

The EXPAND() function

- The function upsamples level g_ℓ by doubling its size from $(2^{N-\ell}+1)\times(2^{N-\ell}+1)$ to $(2^{N-\ell+1}+1)\times(2^{N-\ell+1}+1)$



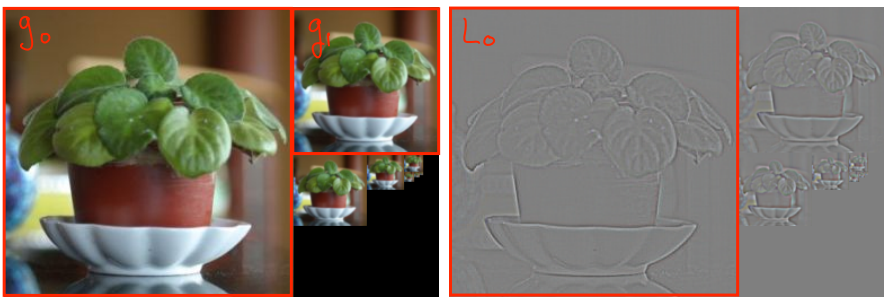
EXPAND function in matrix notation (1D)

$$\underbrace{\begin{bmatrix} \\ EXPAND(g_1) \\ \\ \end{bmatrix}}_{2^N+1} = \underbrace{\begin{bmatrix} c\ b\ a\ b\ c & 0 \\ & c\ b\ a\ b\ c \\ 0 & \ddots \end{bmatrix}}_{2^N+1 \times 2^{N-1}+1} \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & 0 \\ 0 & 1 & 0 & \cdots & 0 \\ 0 & 0 & 0 & \cdots & 0 \\ & & \cdots & & \end{bmatrix} \begin{bmatrix} \\ g_1 \\ \\ \end{bmatrix}$$

Convolution matrix C_0 — upsampling matrix $(D_0)^T$

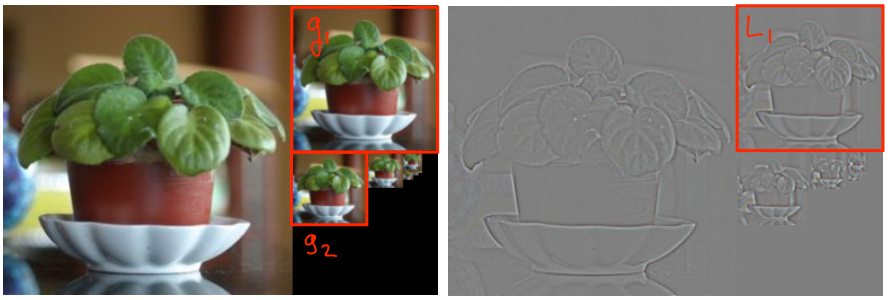
$$EXPAND(g_1) = C_0 \cdot (D_0)^T g_1$$

The Laplacian Pyramid



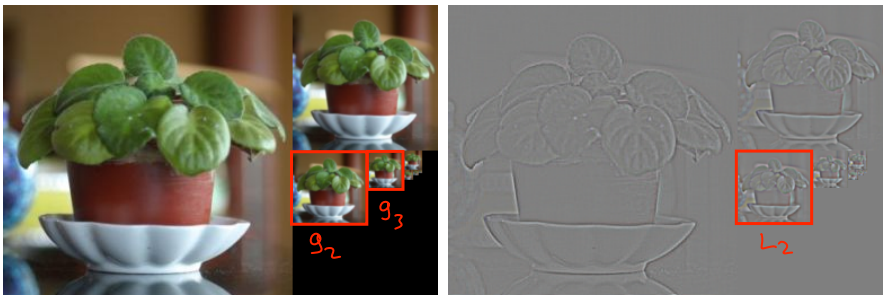
The Laplacian Pyramid

$L_1 = g_1 - \text{EXPAND}(g_2)$



The Laplacian Pyramid

$L_2 = g_2 - \text{EXPAND}(g_3)$



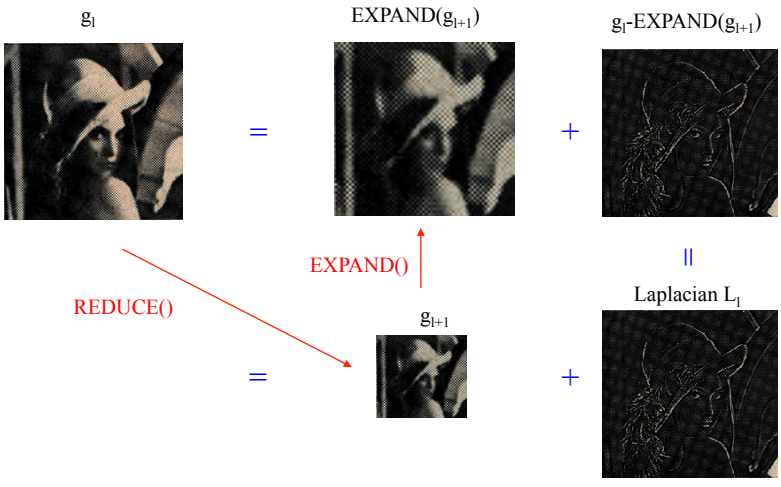
The Laplacian Pyramid

- Often we use a truncated Laplacian pyramid by storing images $L_0, L_1, \ldots, L_K, g_{K+1}$ for $K+1 < N$

Base case: store g_N

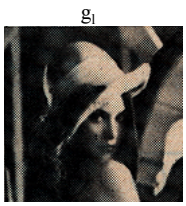
The Laplacian Image Pyramid

Idea: Represent g_l image by a g_{l+1} image and a detail image (called the Laplacian L_l image) whose size is equal to the g_l image



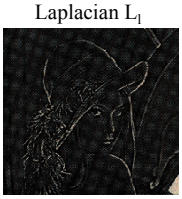
The Laplacian Image Pyramid

Idea: This decomposition can be repeated several times!!



=

+



The Laplacian Image Pyramid

Idea: This decomposition can be repeated several times!!

Given an input image of size $(2^N+1) \times (2^N+1)$, the Laplacian pyramid representation consists of

- $L_0, \ldots, L_{K-1}$
- g_K for some $K \leq N$

=

+

Transmission using EXPAND