Topic 10:

Gaussian & Laplacian Pyramids

- The Gaussian pyramid
- Constructing the Gaussian pyramid
  - The REDUCE() function
- Constructing the Laplacian pyramid
  - The EXPAND() function
- Applications

The Gaussian Pyramid

Gaussian Pyramid Representation of the Photo

Original photo

The Gaussian Pyramid

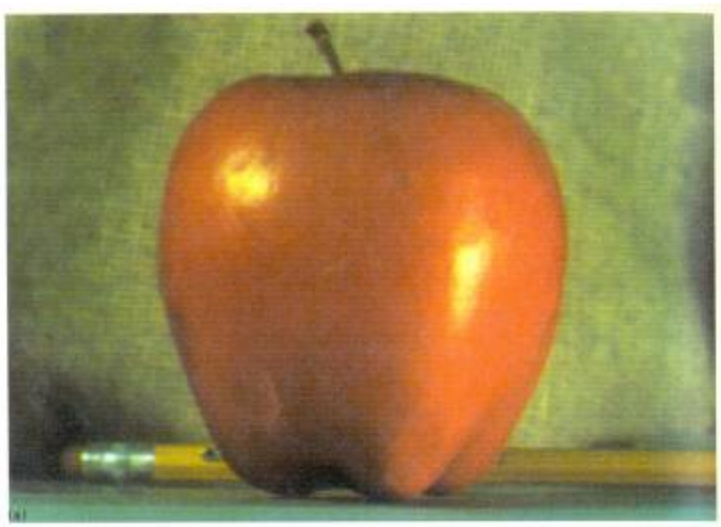
Goal: Develop representation to decompose images into information at multiple scales, to extract feature or structure of interest, to attenuate noise.

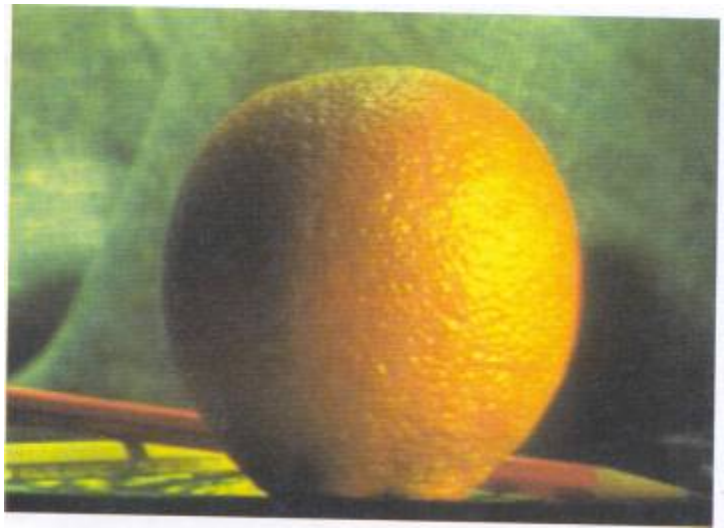
Applications:

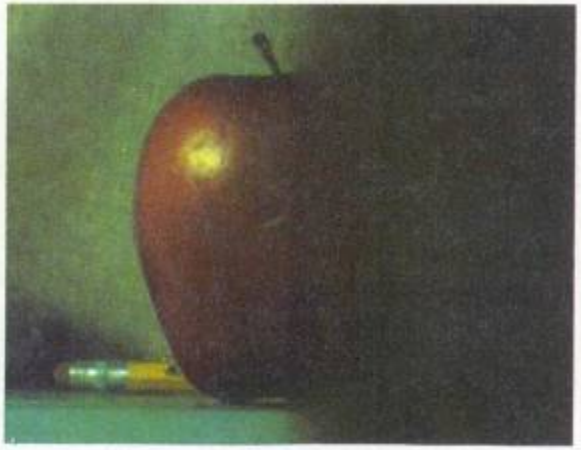
- Efficient image coding
- Progressive transmission
- Image blending
- Image enhancement
- Efficient Processing
- …too many to list here!

Application: Pyramid Image Blending

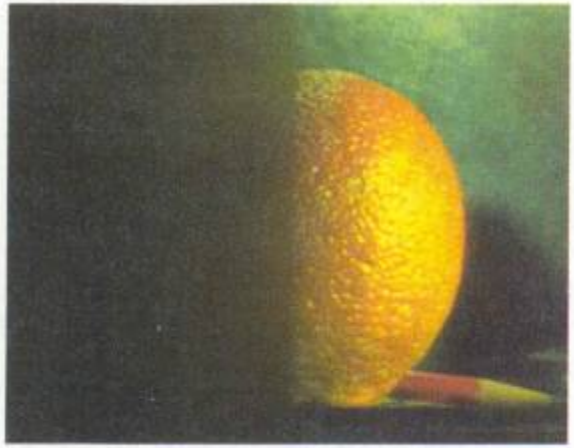
Goal: Merge two images without visible seams



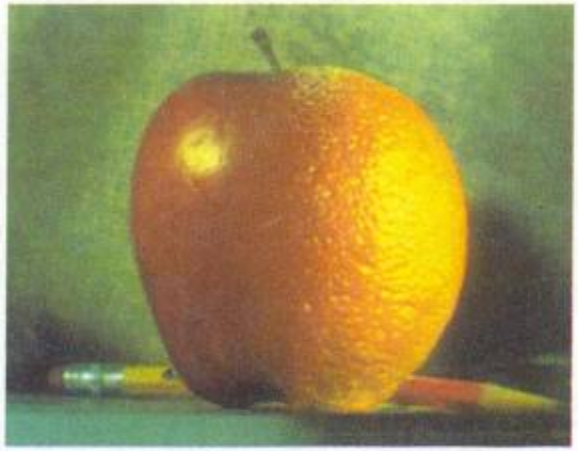




(d)

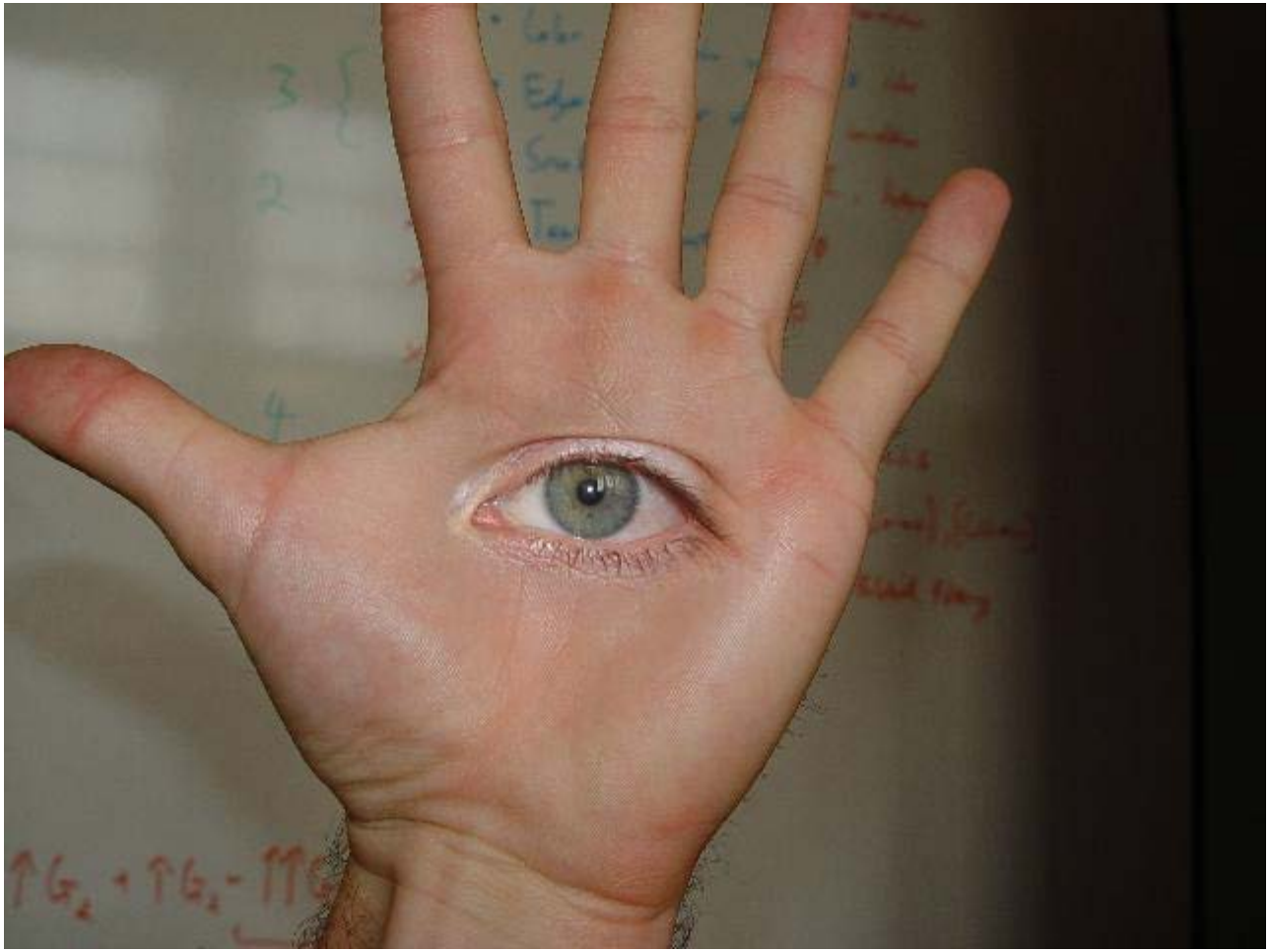


(h)



(l)

Horror Photo



© prof. dmartin

The Gaussian Pyramid

Goal: Develop representation to decompose images into information at multiple scales, to extract feature or structure of interest, to attenuate noise.

Input:

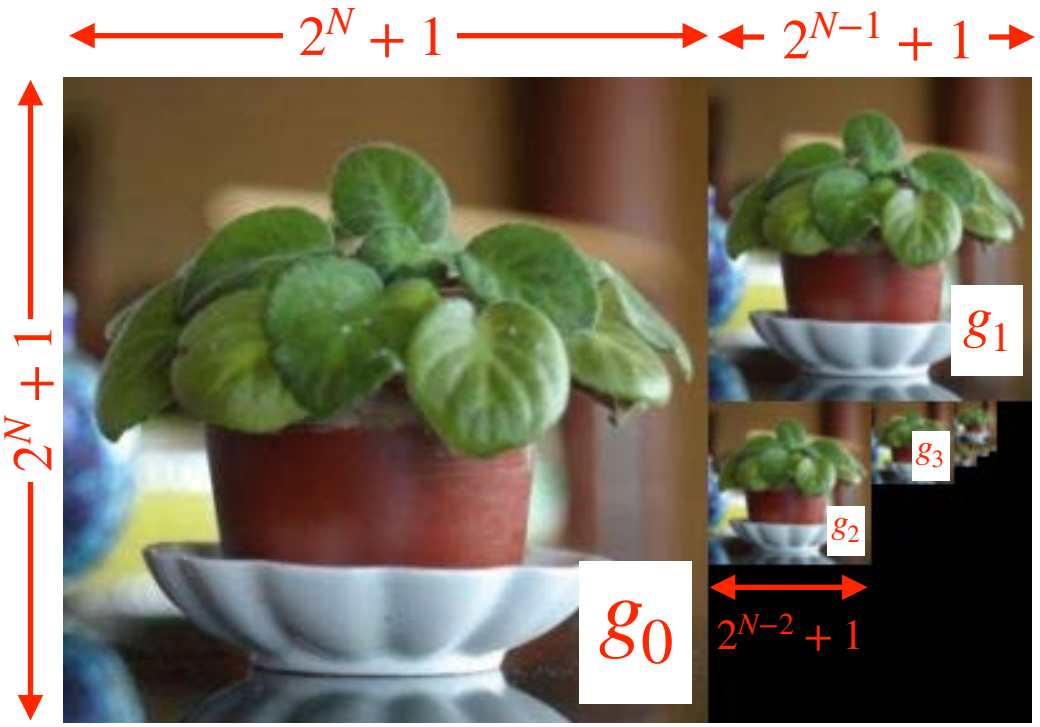
Image I of size

$(2^N + 1) \times (2^N + 1)$

Output:

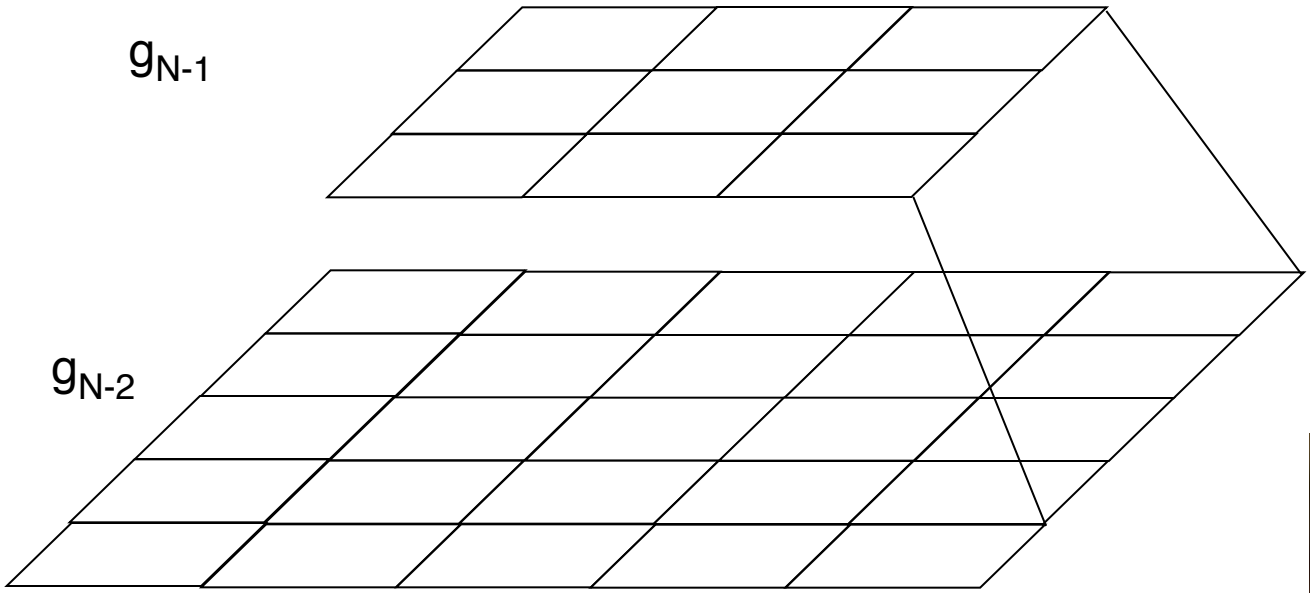
- N images $g_0, \ldots, g_{N-1}$
- g_l has size

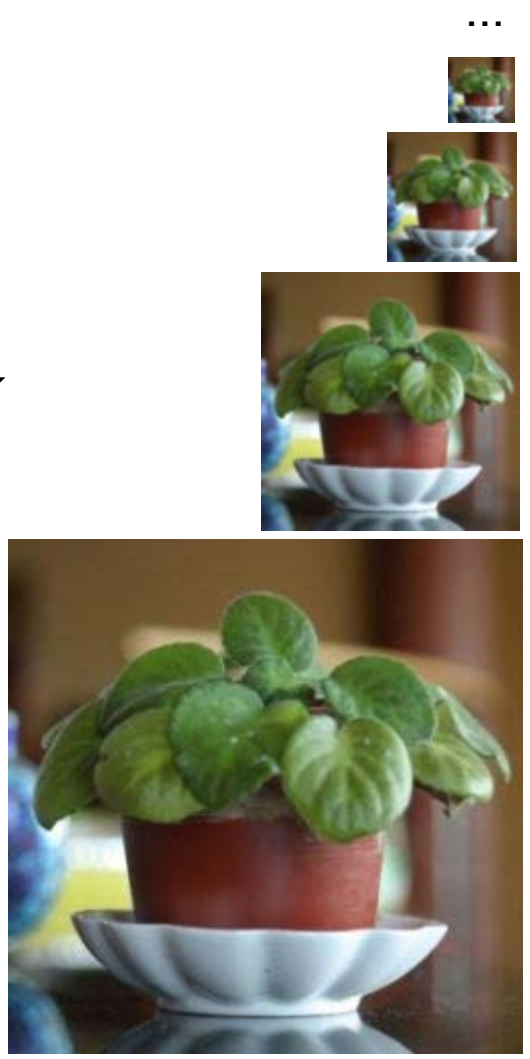
$(2^{N-l} + 1) \times (2^{N-l} + 1)$



Why is it Called a Pyramid?

Idea: Representation can be pictured as a "pyramid" of 3x3, 5x5, 9x9,…, $(2^N+1)x(2^N+1)$ images





g_0 (= original image)

The Gaussian Pyramid

The representation is based on 2 basic operations:

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 half after each smoothing.

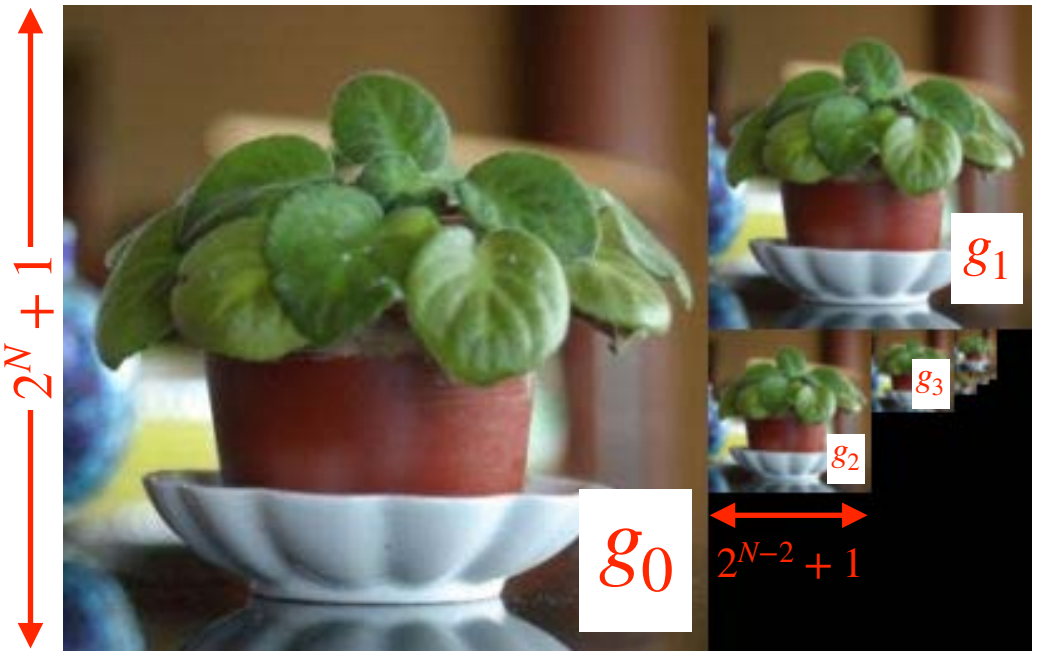
Aside:
Downsampling is any linear transformation of the form

$$\begin{bmatrix} \vdots \\ \vdots \end{bmatrix} = \begin{bmatrix} \cdot & \cdot \\ \text{matrix} \\ \cdot & \cdot \end{bmatrix} \begin{bmatrix} \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \end{bmatrix}$$

downsampled image

#rows < #columns

original image



Topic 10:

Gaussian & Laplacian Pyramids

- The Gaussian pyramid
- Constructing the Gaussian pyramid
  - The REDUCE() function
- Constructing the Laplacian pyramid
  - The EXPAND() function
- Applications

The Gaussian Pyramid

The representation is based on 2 basic operations:

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 half after each smoothing.

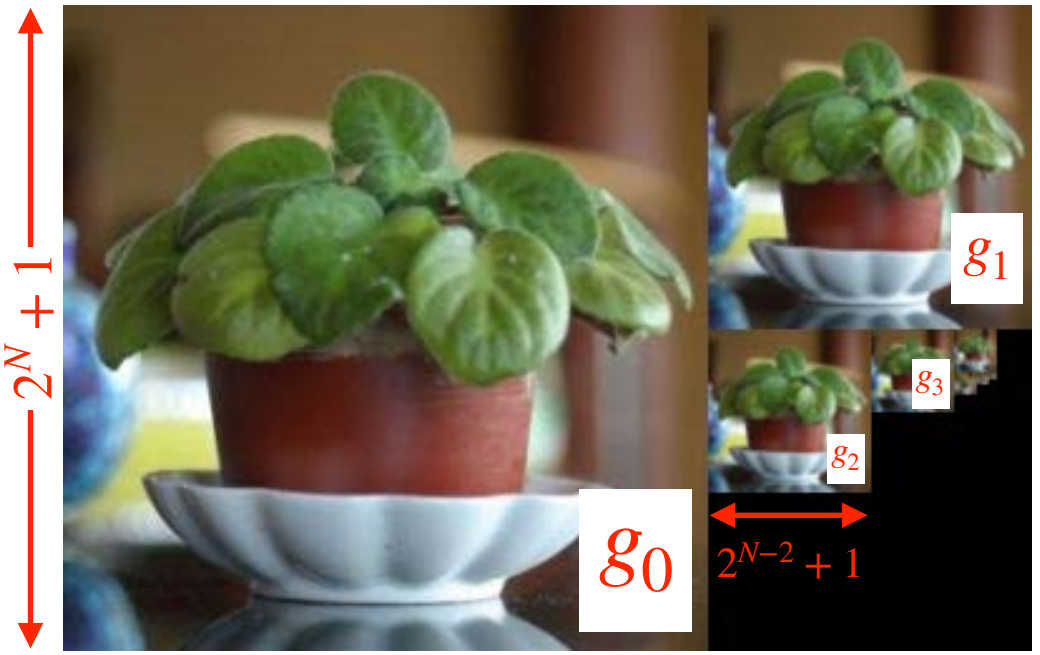
Aside:
Downsampling is any linear transformation of the form

$$\begin{bmatrix} \vdots \\ \vdots \end{bmatrix} = \begin{bmatrix} \cdot & \cdot \\ \text{matrix} \\ \cdot & \cdot \end{bmatrix} \begin{bmatrix} \vdots \\ \vdots \\ \vdots \\ \vdots \\ \vdots \end{bmatrix}$$

downsampled image

#rows < #columns

original image

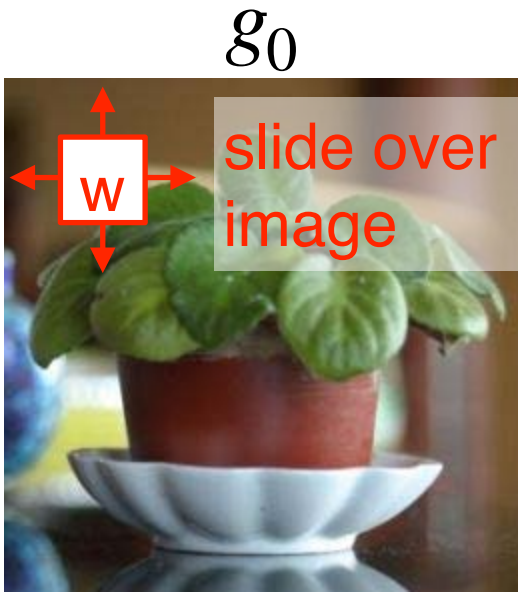


Operation #1: Smooth Image at N-1 Scales

Original photo g_0

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

a 5×5 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

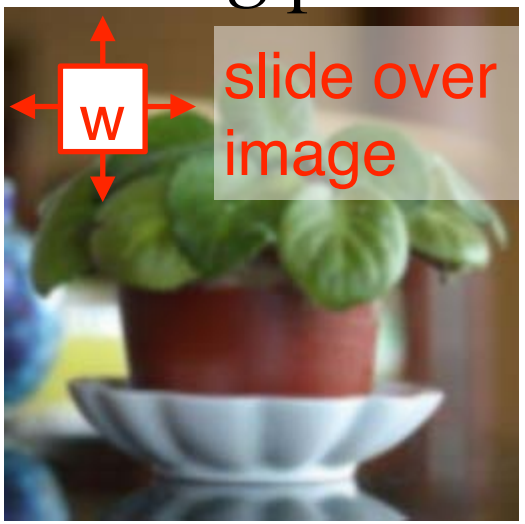
$$\hat{g}_2 = w * \hat{g}_1$$
$$= w * (w * g_0)$$
$$= (w * w) * g_0$$

can be thought of as a filter

h = w * w

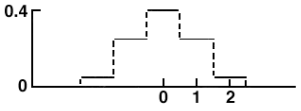
whose radius is **twice** that of w.

$\hat{g}_1$

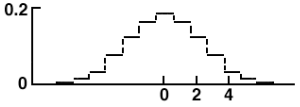


In 1D:

w :



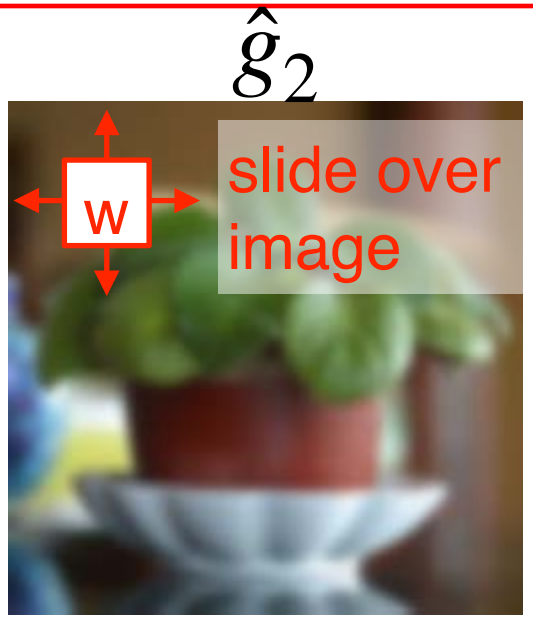
$w * w$:



Operation #1: Smooth Image at N-1 Scales

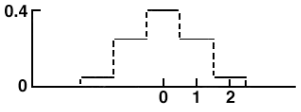
$$\hat{g}_3 = w * \hat{g}_2$$
$$= (\underbrace{w * w * w}) * g_0$$

radius is 4 times that of w



In 1D:

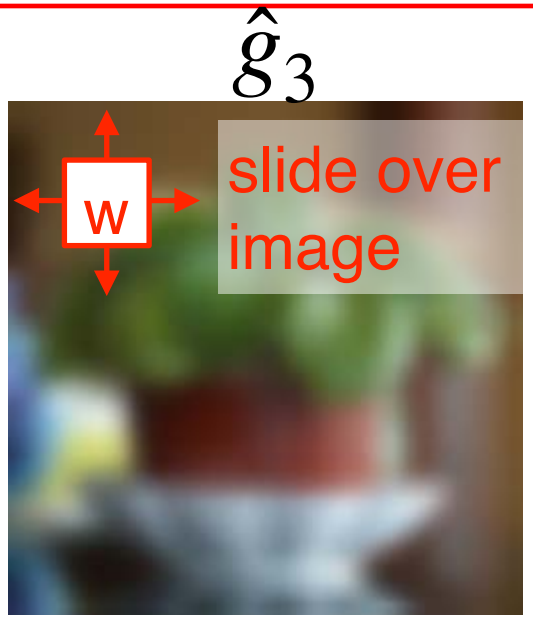
w :



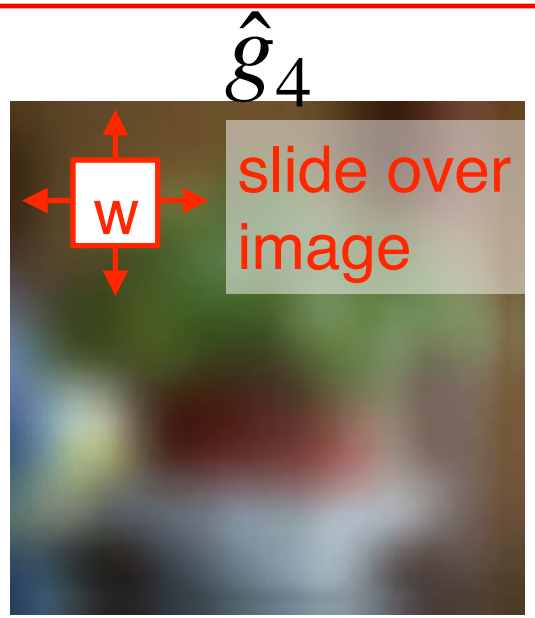
$w * w * w$:

Operation #1: Smooth Image at N-1 Scales

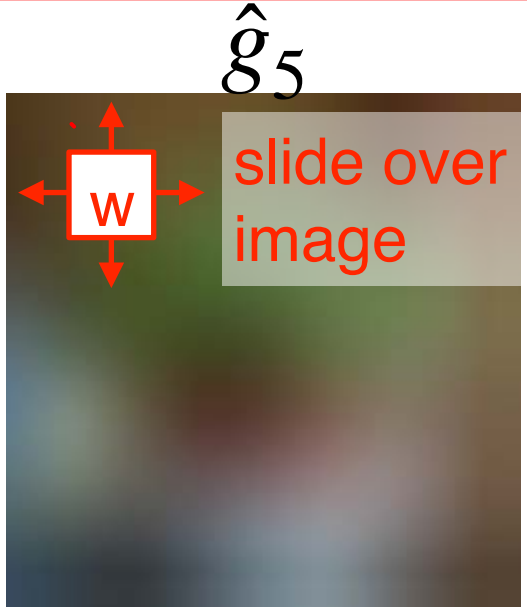
$$\hat{g}_4 = w * \hat{g}_3$$
$$= (w * w * w * w) * g_0$$



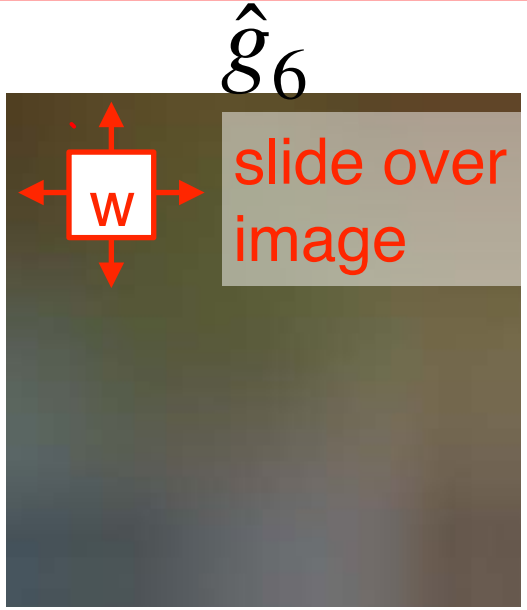
Operation #1: Smooth Image at N-1 Scales



Operation #1: Smooth Image at N-1 Scales



Operation #1: Smooth Image at N-1 Scales



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 (aka "5-tap" filter)

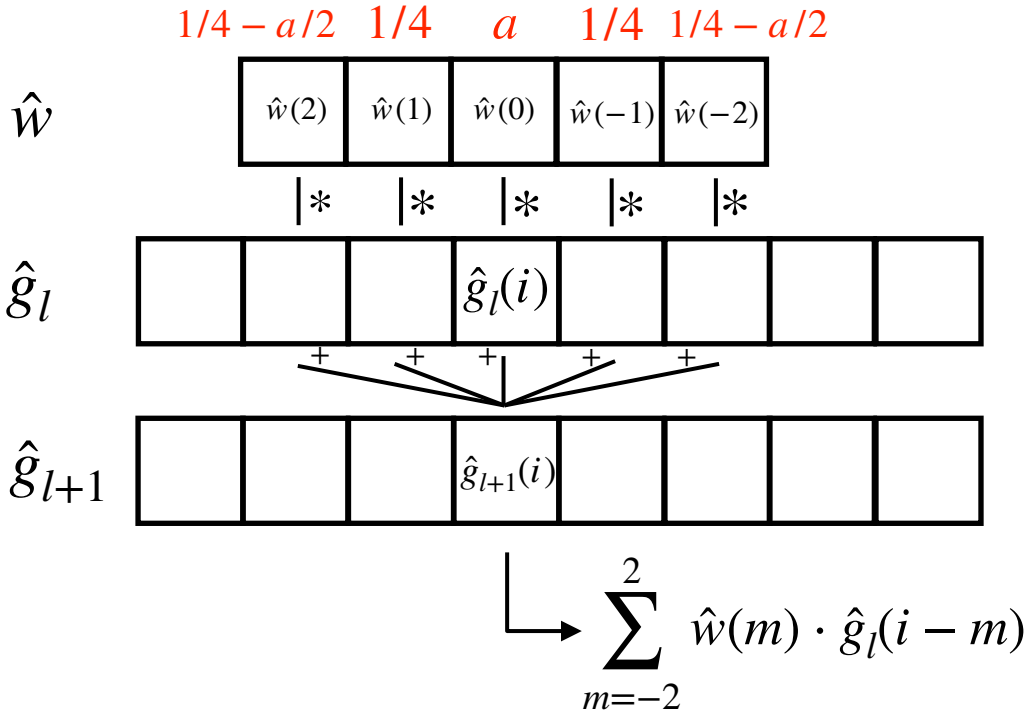
2. $\hat{w}$ symmetric about *o*:

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

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

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

$$\Longleftrightarrow \quad a + 2b + 2c = 1$$



4. Equal contribution

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

To satisfy Criteria 1-4 we have a 2 equations & 3 unknowns

⇒ *a* remains a free parameter

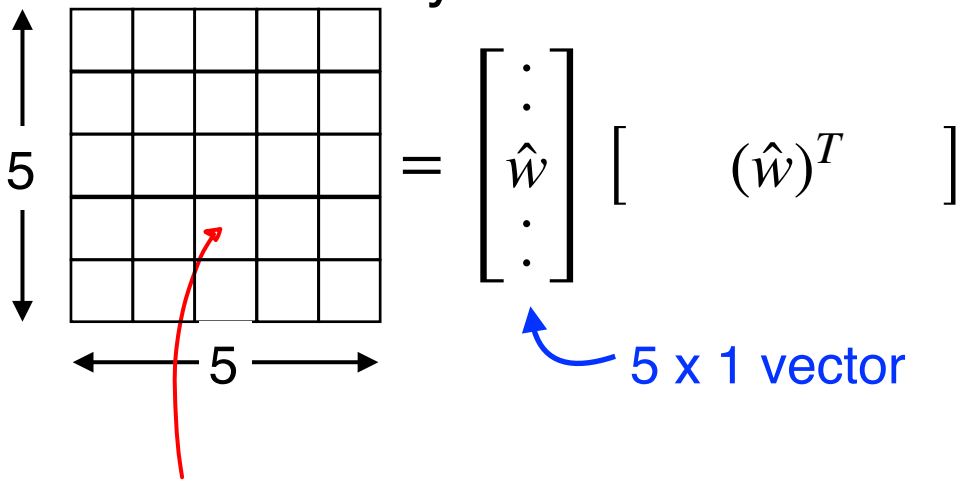
$$\hat{w}(2) = \hat{w}(-2) = \frac{1}{4} - \frac{a}{2},\ \hat{w}(-1) = \hat{w}(1) = \frac{1}{4}$$

usually $a \in [0.3, 0.6]$

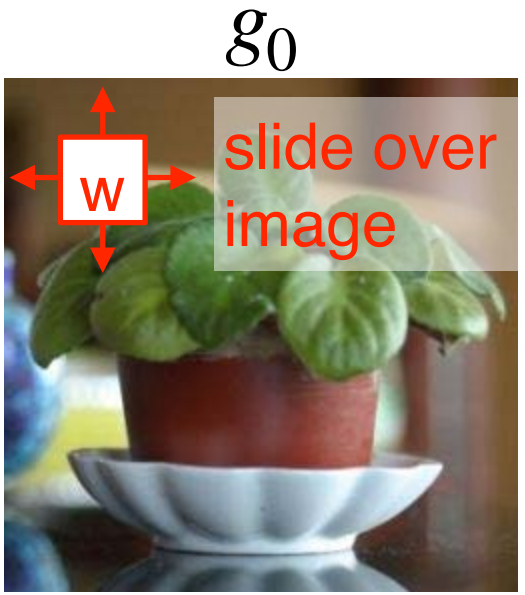
Defining the Smoothing Filter in 2D

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

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



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



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

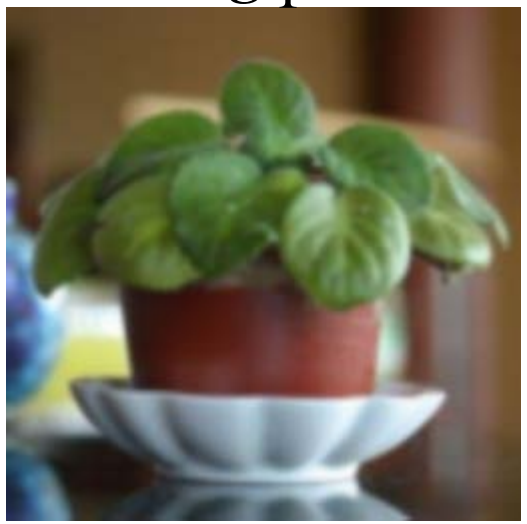
1. Convolve each row of g_0 with $\hat{w}$.
2. Convolve the columns fo the result with $\hat{w}$ again.

Operation #2: Downsample the Smoothed Image

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

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

$\hat{g}_1$



$2^N + 1$

g_1

$2^{N-1} + 1$

Topic 10:

Gaussian & Laplacian Pyramids

- The Gaussian pyramid
- Constructing the Gaussian pyramid
  - The REDUCE() function
- Constructing the Laplacian pyramid
  - The EXPAND() function
- Applications

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

Smoothing & downsampling are combing into a single REDUCE function.

g_l

$2^{N-l} + 1$

$w * g_l$

downsample $\times 2$

g_{l+1}

$2^{N-l-1} + 1$

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

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

The REDUCE() function

...

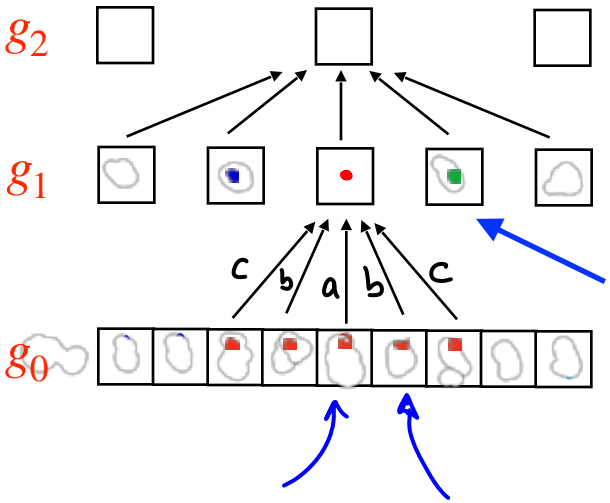
REDUCE(g_2)

REDUCE(g_1)

REDUCE(g_0)

The REDUCE() function

Visualizing in 1D



REDUCE(REDUCE(g_0))

REDUCE(g_0)

each pixel receives contribution from 5 pixels below

contributes to • weight a

contributes to • weight b

Equal contribution criterion:
when a + 2c = 2b, all pixels of level g_l contribute equally to level g_{l+1}

...

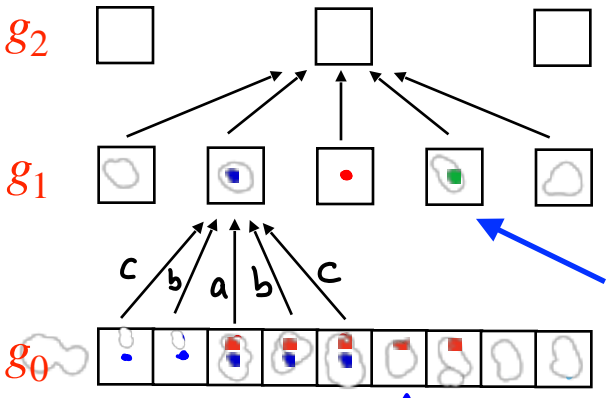
REDUCE(g_2)

REDUCE(g_1)

REDUCE(g_0)

The REDUCE() function

Visualizing in 1D



REDUCE(REDUCE(g_0))

REDUCE(g_0)

each pixel receives contribution from 5 pixels below

contributes
to • weight a
to • weight c

contributes
to • weight b

Equal contribution criterion:
when a + 2c = 2b, all pixels of level g_l contribute equally to level g_{l+1}

...

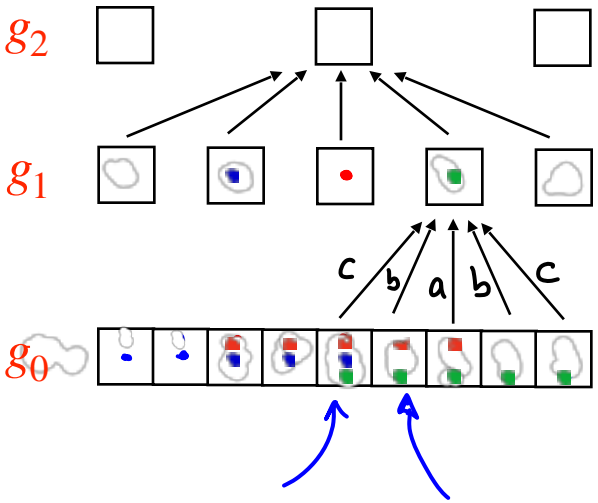
REDUCE(g_2)

REDUCE(g_1)

REDUCE(g_0)

The REDUCE() function

Visualizing in 1D



REDUCE(REDUCE(g_0))

REDUCE(g_0)

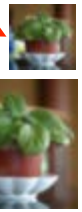
contributes
to • weight a
to • weight c
to • weight c

contributes
to • weight b
to • weight b

Equal contribution criterion:
when a + 2c = 2b, all pixels of level g_l contribute equally to level g_{l+1}

...

REDUCE(g_2)

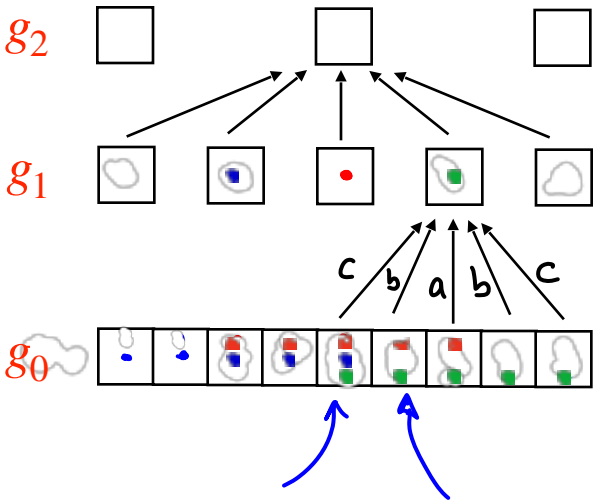


REDUCE(g_1)

REDUCE(g_0)

The REDUCE() function

Visualizing in 1D



REDUCE(REDUCE(g_0))

REDUCE(g_0)

contributes
to • weight a
to • weight c
to • weight c
total: a + 2c

contributes
to • weight b
to • weight b

total: 2b

Equal contribution criterion:
when a + 2c = 2b, all pixels of level g_l contribute equally to level g_{l+1}

...

REDUCE(g_2)

REDUCE(g_1)

REDUCE(g_0)

The 1D REDUCE() function in matrix notation

$$\begin{bmatrix} \vdots \\ g_1 \\ \vdots \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & \cdots & 0 \\ 0 & 0 & 1 & \cdots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \cdots & 1 \end{bmatrix} \begin{bmatrix} c & b & a & b & c & 0 & \cdots & 0 \\ 0 & c & b & a & b & c & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & 0 & 0 & \cdots & c \end{bmatrix} \begin{bmatrix} \vdots \\ g_0 \\ \vdots \end{bmatrix}$$

downsampling matrix D_0

convolution matrix C_0

$$g_1 = D_0 \cdot C_0 \cdot g_0$$

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 feature or structure of interest, to attenuate noise.

Input:

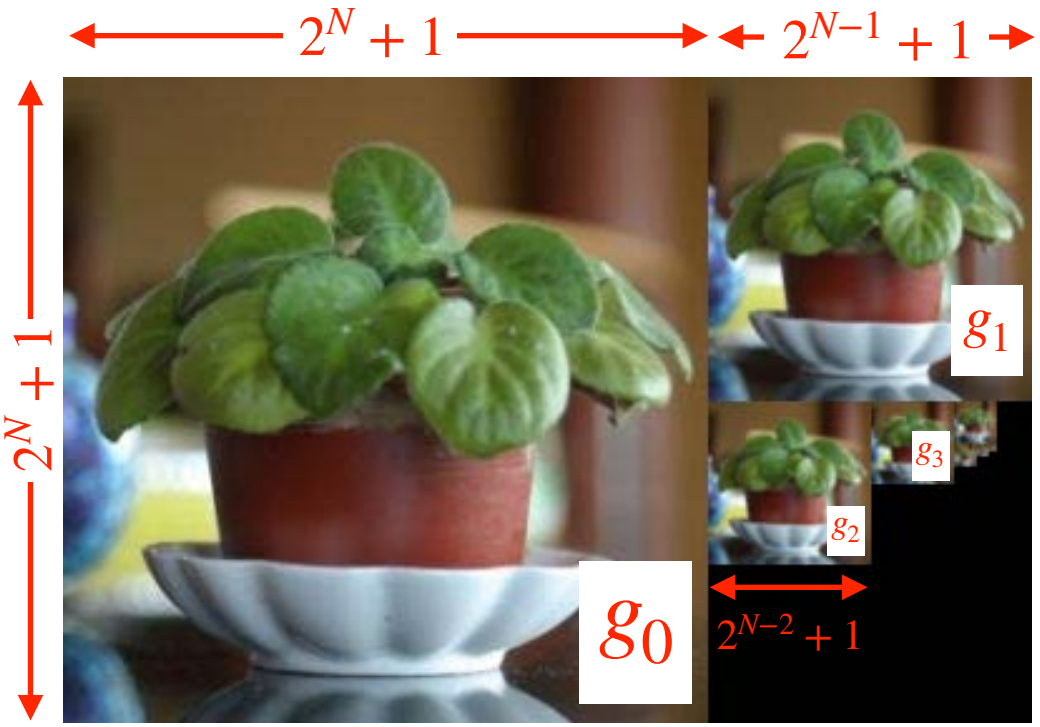
Image I of size

$(2^N + 1) \times (2^N + 1)$

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 combing into a single REDUCE function.

g_l

$2^{N-l} + 1$

$w * g_l$

downsample $\times 2$

g_{l+1}

$2^{N-l-1} + 1$

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

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

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 **not** represented in $\hat{g}_1$.

Topic 10:

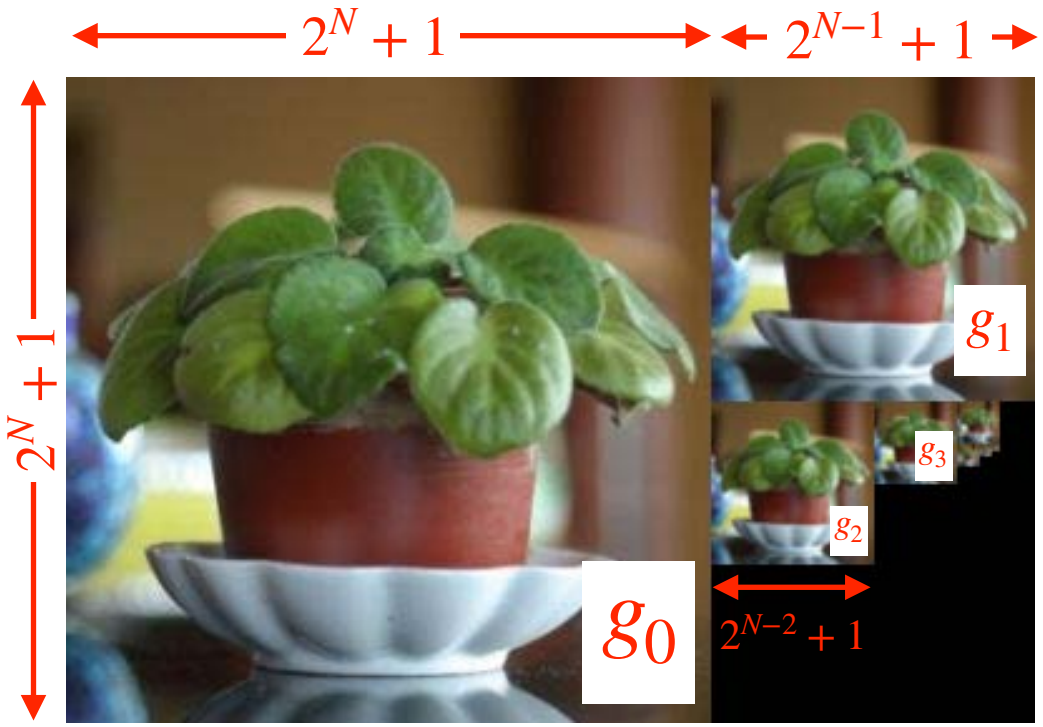
Gaussian & Laplacian Pyramids

- The Gaussian pyramid
- Constructing the Gaussian pyramid
  - The REDUCE() function
- Constructing the Laplacian pyramid
  - The EXPAND() function
- Applications

The Laplacian Pyramid

Idea:

Rather than store the smoothed images, store only **the difference** between levels g_l and g_{l+1}



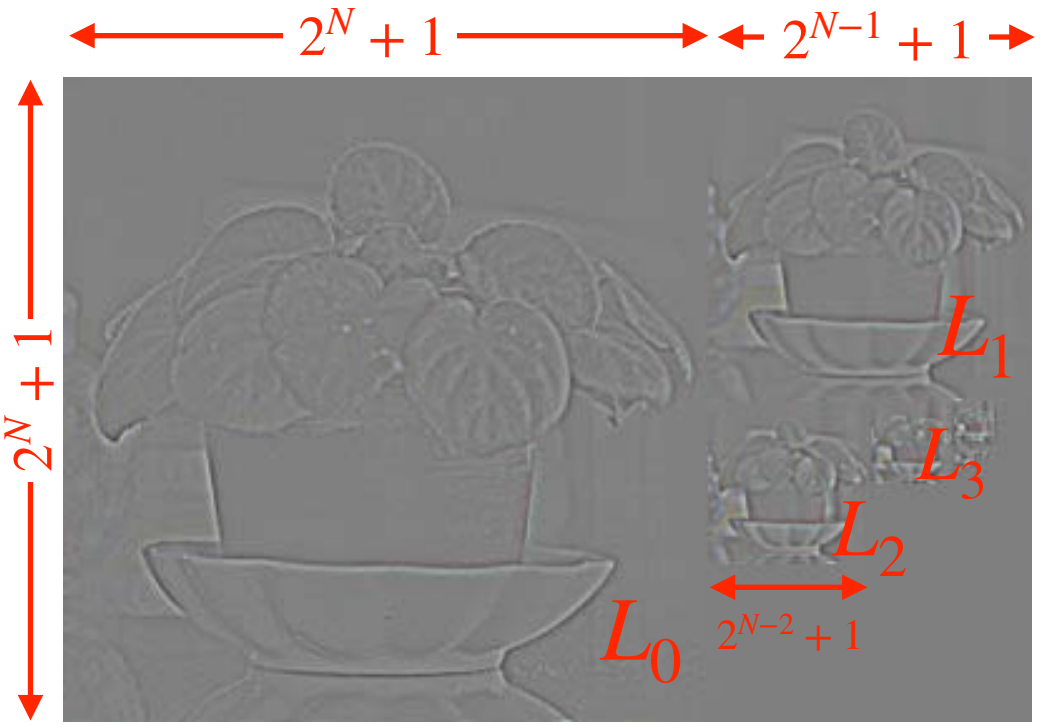
The Laplacian Pyramid

Idea:

Rather than store the smoothed images, store only **the difference** between levels g_l and g_{l+1}

most values are 0 (gray) ⇒ can be

quantized and represented with few bits



The Laplacian Pyramid

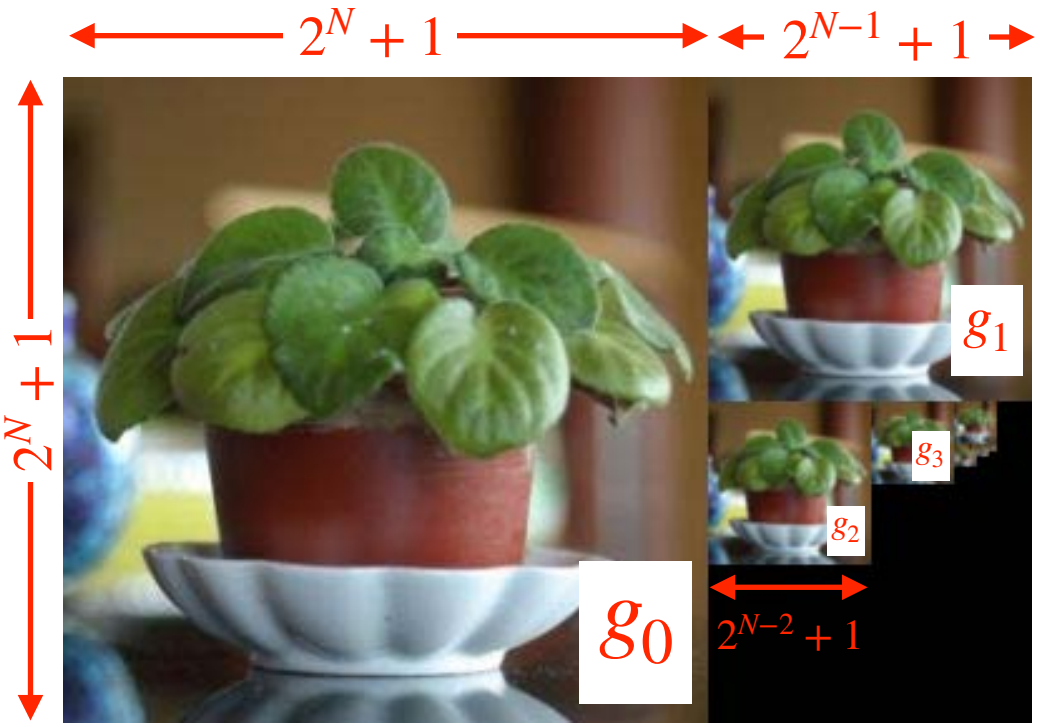
Idea:

Rather than store the smoothed images, store only **the difference** between levels g_l and g_{l+1}

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

  ⇒ **Expand** g_{l+1} to

  make it equal size to g_l



Operation #3: The EXPAND() Function

$2^{N-l}+1$

g_l

EXPAND( )

$\Rightarrow$

$2^{N-l+1}+1$

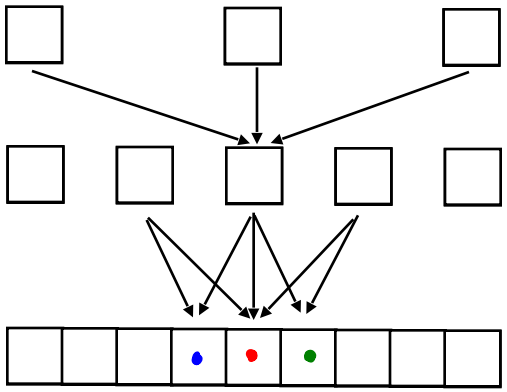
EXPAND(g_l)

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

The EXPAND() function

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

Visualizing in 1D



g_1

$\text{EXPAND}(g_1)$

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)

Generalizing expression 1D

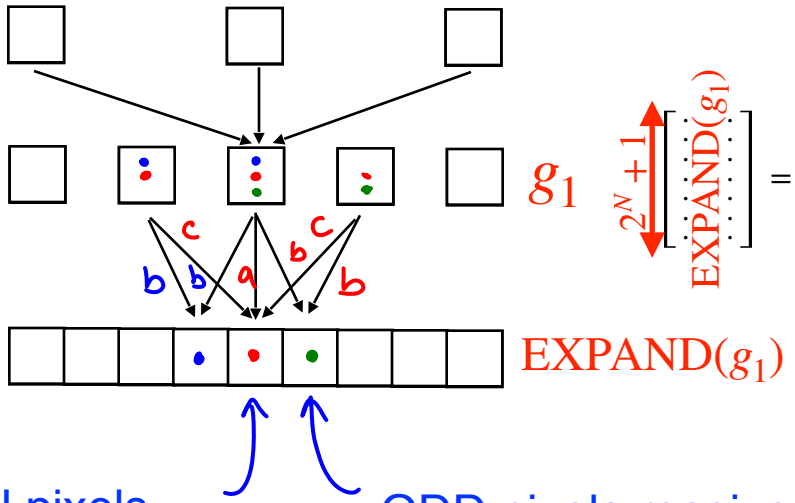
$$\text{EXPAND}(g_l)(i) = 2 \sum_{m=-2}^{2} \hat{w}(m) \cdot g_l\left(\frac{i-m}{2}\right)$$

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 $(2^{N-l}+1) \times (2^{N-l}+1)$ to $(2^{N-l+1}+1) \times (2^{N-l+1}+1)$

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)

EXPAND function in matrix notation (1D)

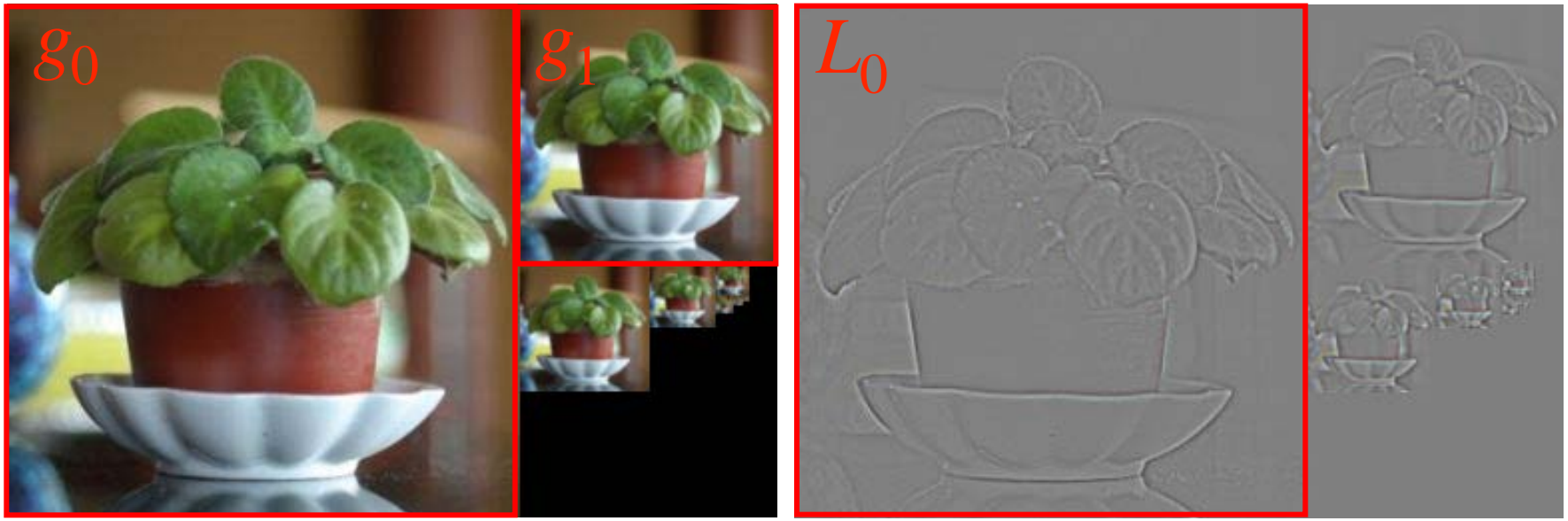
$$\underbrace{\begin{bmatrix} \cdot \\ \vdots \\ \cdot \end{bmatrix}}_{2^N+1} \text{EXPAND}(g_1) = \underbrace{\begin{bmatrix} c & b & a & b & c & \cdots & 0 \\ 0 & c & b & a & b & \cdots & 0 \\ \vdots & \ddots & \ddots & \ddots & \ddots & \ddots & \vdots \\ 0 & 0 & 0 & 0 & 0 & \cdots & c \end{bmatrix}}_{2^N+1}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

convolution matrix C_0

upsampling matrix $(D_0)^T$

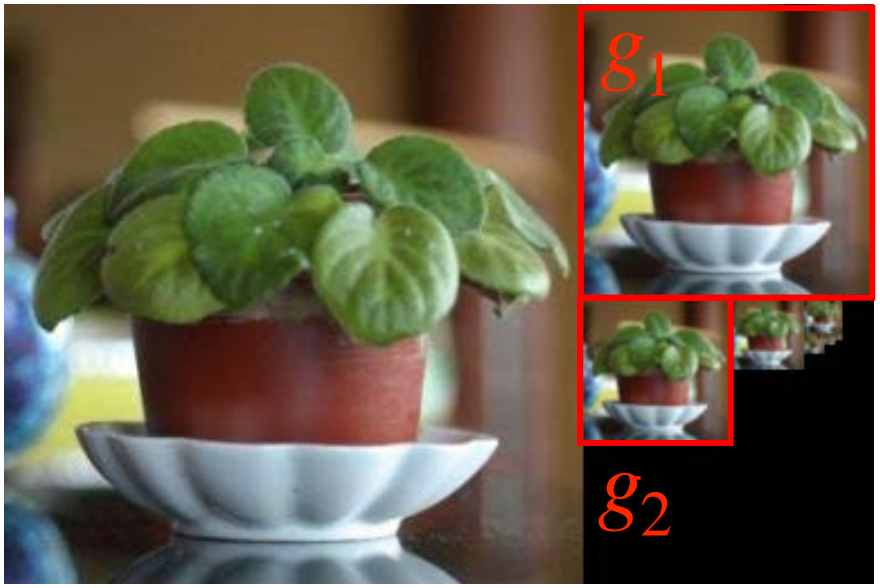
$$\text{EXPAND}(g_1) = C_0 \cdot (D_0)^T \cdot g_1$$

The Laplacian Pyramid



The Laplacian Pyramid

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



The Laplacian Pyramid

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

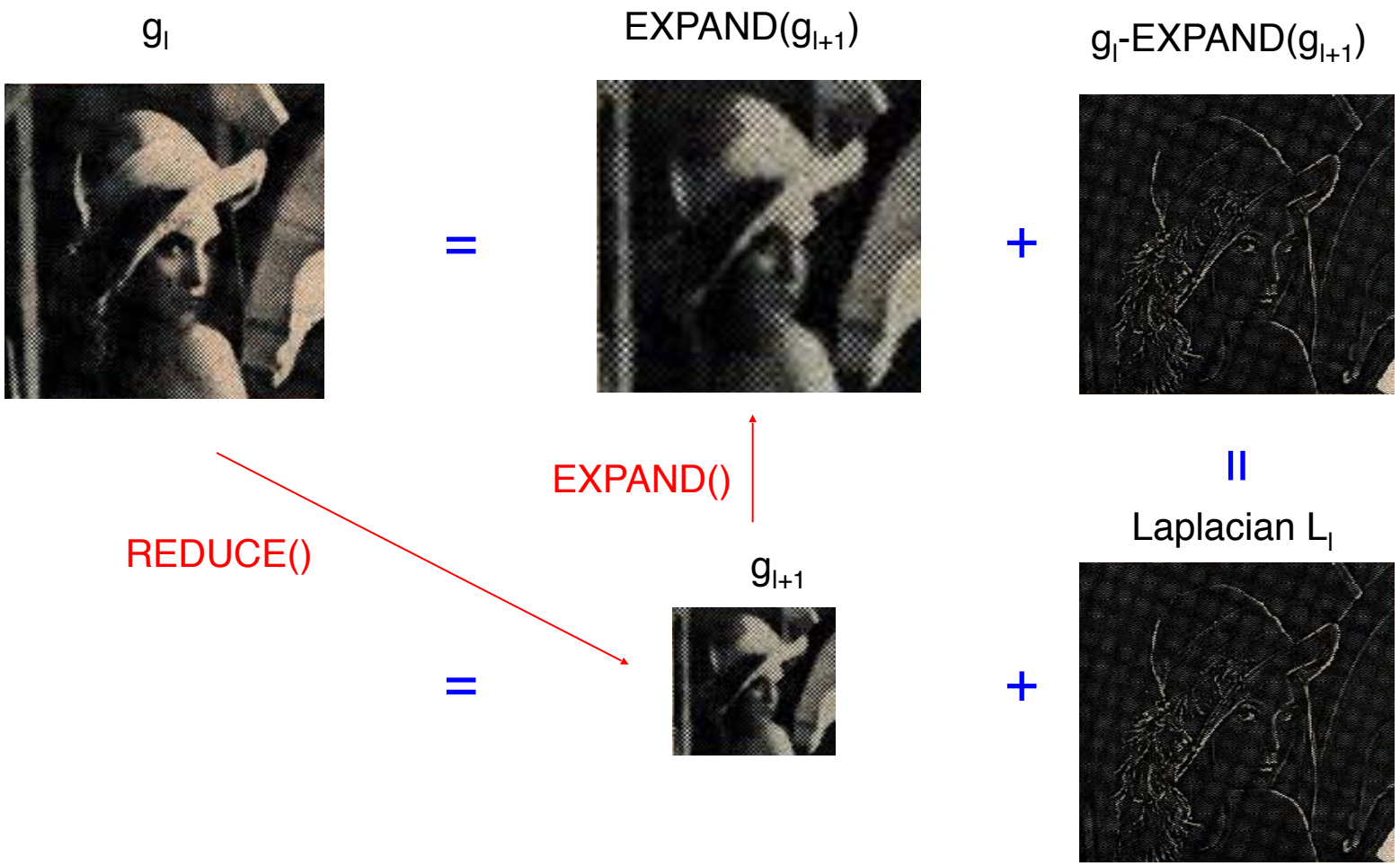
The Laplacian Pyramid

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!!

g_l

=

g_{l+1}

+

Laplacian L_l

The Laplacian Image Pyramid

Idea: This decomposition can be repeated several times!!

g_l

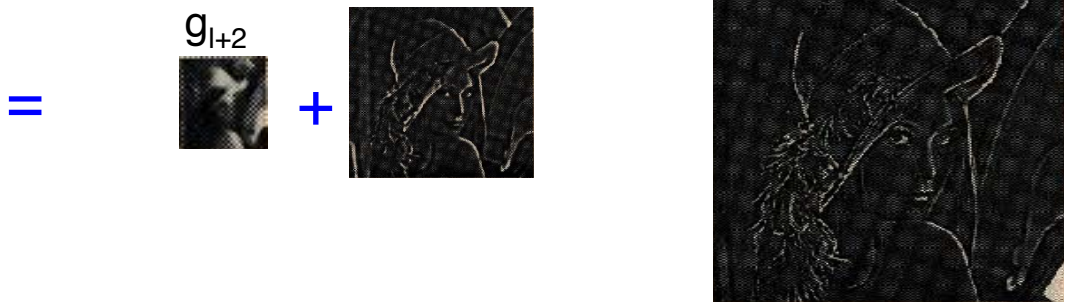
Given an input image of size $(2^N+1) \times (2^N+1)$, the Laplacian pyramid representation consists of

- $L_0, \ldots, L_{k-1}$
- g_k for some $k \leq N$

Laplacian L_{l+1}

Laplacian L_l

g_{l+2}



=

+

Transmission using EXPAND

RECEIVER

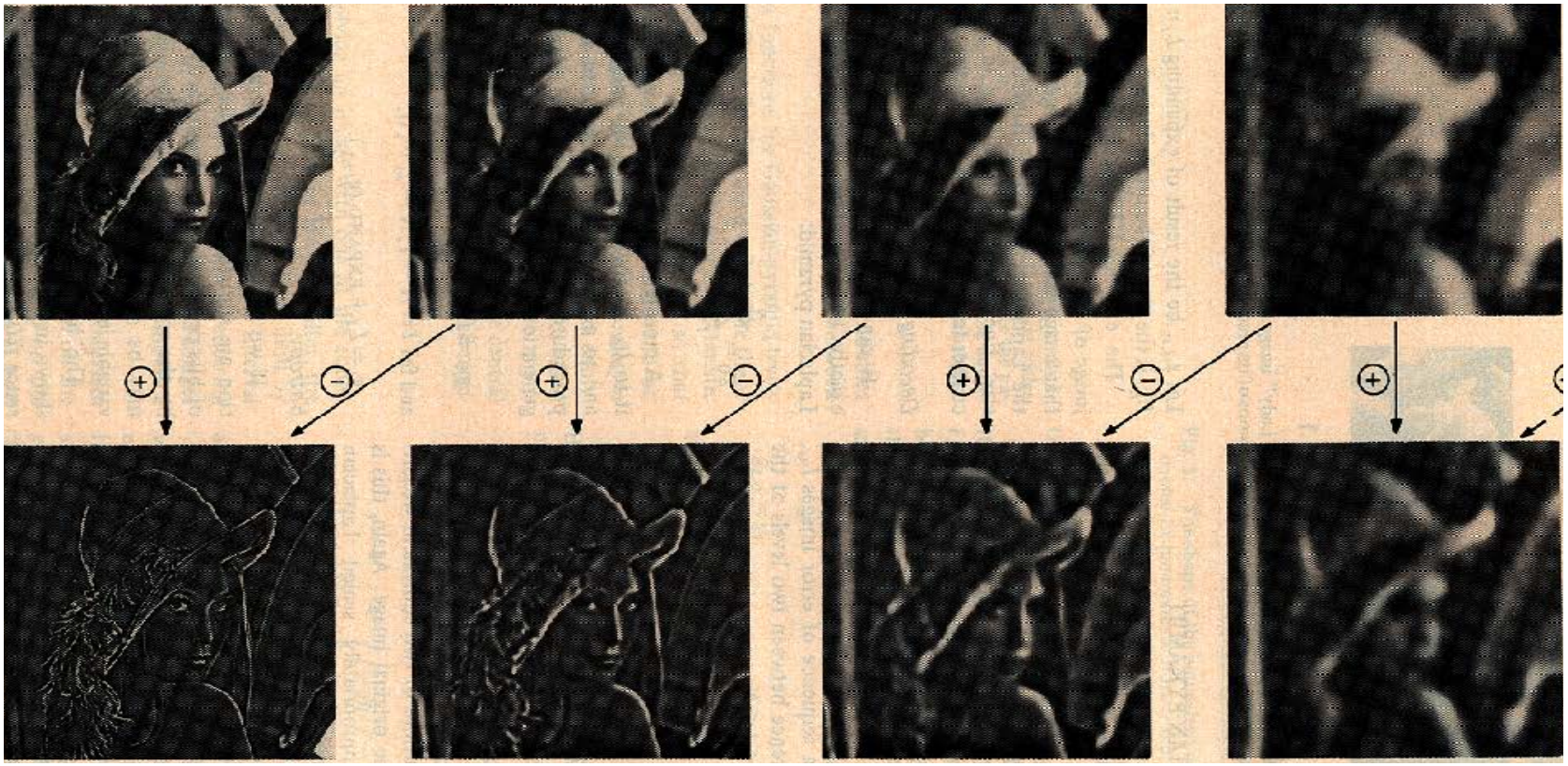
TRANSMITTER

3. Display $d_3 = d_2 + \text{EXPAND}^{k-2}(L_{k-2})$

2. Display $d_2 = d_1 + \text{EXPAND}^{k-1}(L_{k-1})$

1. Display $d_1 = \text{EXPAND}^{k}(g_k)$

1. Transmit g_k



5. Transmit L_{k-4}

4. Transmit L_{k-3}

3. Transmit L_{k-2}

2. Transmit L_{k-1}

Topic 10:

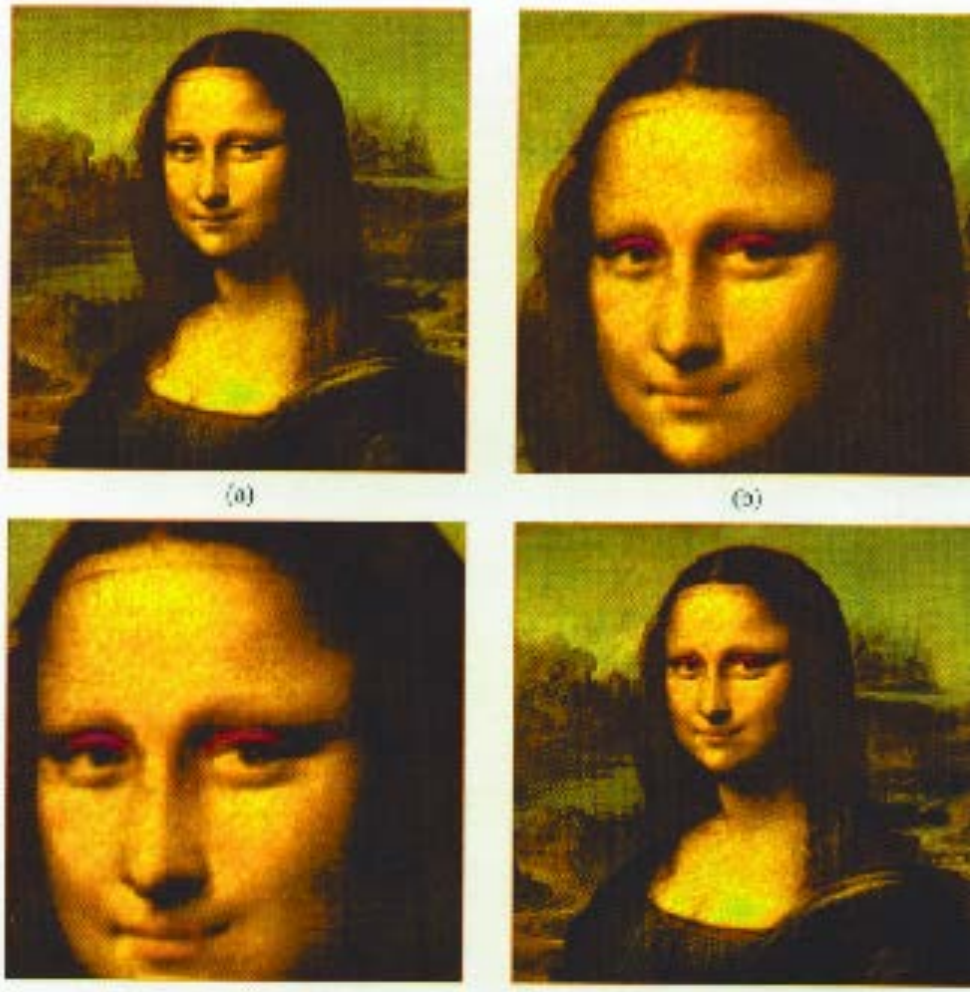
Gaussian & Laplacian Pyramids

- The Gaussian pyramid
- Constructing the Gaussian pyramid
  - The REDUCE() function
- Constructing the Laplacian pyramid
  - The EXPAND() function
- Applications
  - Multi-resolution image blending
  - Multi-resolution image editing

Application #2: Multi-Resolution Image Editing

How can we achieve the painting operations below?

Original photo



Adding makeup

Adding a glint

Final photo

(Berman, Bartell, Salesin, SIGGRAPH'94)

Image Editing

Approach #1

- Modify the image pixels directly, using a brush tool

Image Editing

Approach #1

- Modify the image pixels directly, using a brush tool

Difficulty: Manipulating pixel colors directly will distort/flatten fine scale detail

Image Editing

Approach #1

- Modify the image pixels directly, using a brush tool

Difficulty: Manipulating pixel colors directly will distort/flatten fine scale detail

Approach #2

- Edit a lower-resolution version of the photo (e.g. level g_k of the Gauss pyramid for $k > 0$)
- How do we create an edited full-res photo?

Pyramid-Based Image Editing

1. Edit g_1 to obtain $\tilde{g}_1$.

2. Add the details using L_0

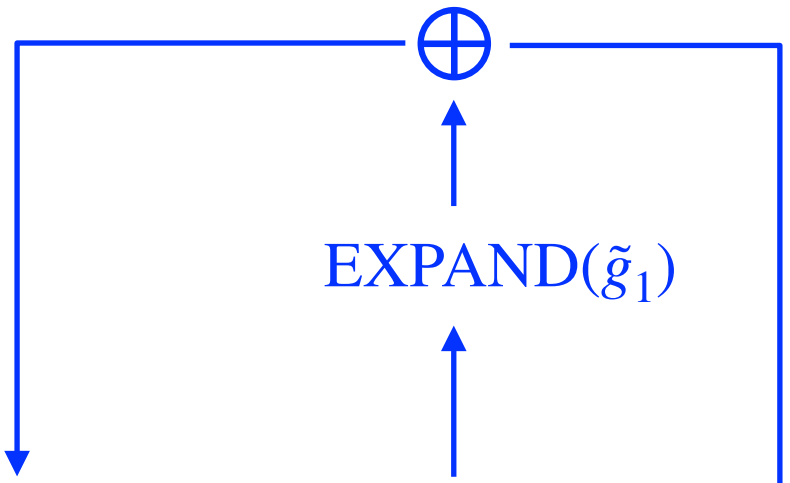
Definition of L_0 :

$$L_0 = g_0 - \text{EXPAND}(g_1)$$

Pyramid-Based Image Editing

1. Edit g_1 to obtain $\tilde{g}_1$.

2. Add the details using L_0

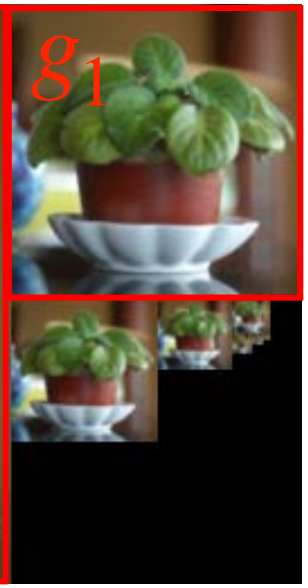


Definition of L_0 :

$$L_0 = g_0 - \text{EXPAND}(g_1)$$

Final image:

$$\tilde{g}_0 = \text{EXPAND}(\tilde{g}_1) + L_0$$



Multi-Resolution Image Editing

Original photo

Adding makeup

The procedure allows us to apply “paint” to an image while preserving image detail.

Multi-Resolution Image Editing

(from Berman et al)

2. 100 000 x zoom

(a)

(b)

1. Original

3. add “smog”

(c)

(d)

4. zoomed-in result

PLATE 4 Adding smog to an image with a range in scale of 100,000 to 1: (a) a sunny day in the park; (b) zooming out by a factor of 100,000; (c) adding smog over Chicago; (d) smog affects the park view. (See Chapter 4: Multiresolution Image Editing.)

Note: Authors used a wavelet pyramid rather than a Gaussian pyramid, but the principle is the same.

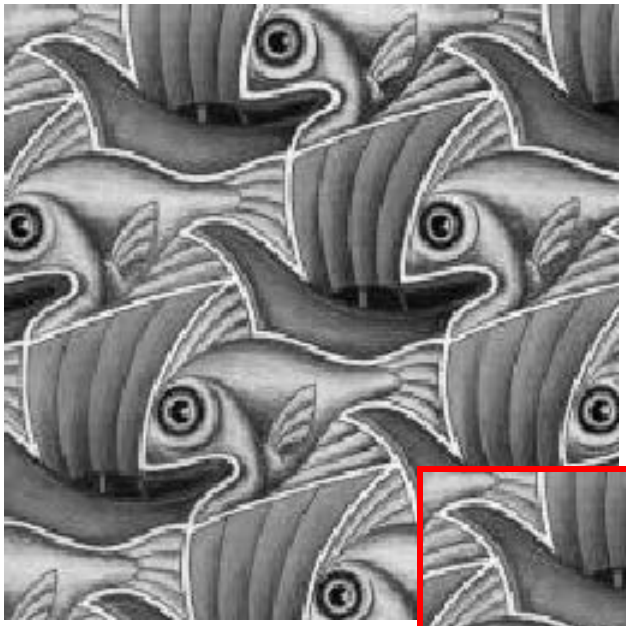
Topic 10:

Hierarchical image representations

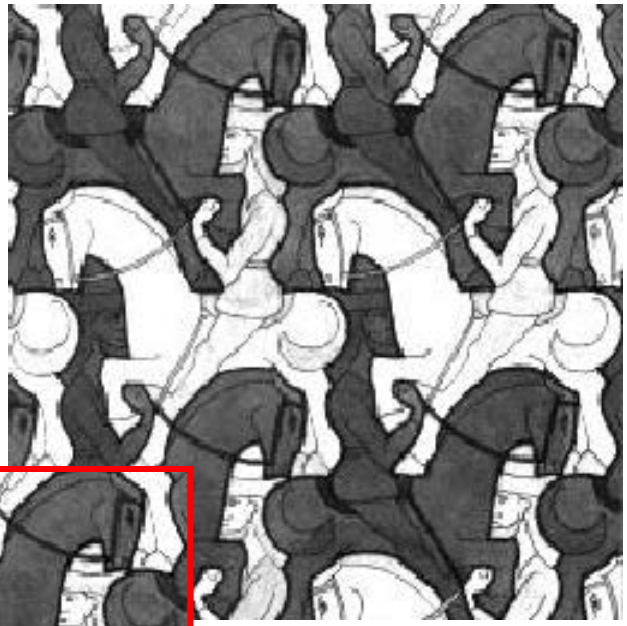
- The Gaussian pyramid
- Constructing the Gaussian pyramid
  - The REDUCE() function
- Constructing the Laplacian pyramid
  - The EXPAND() function
- Applications
  - Multi-resolution image blending
  - Multi-resolution image editing

Image Blending

Source A



Source B

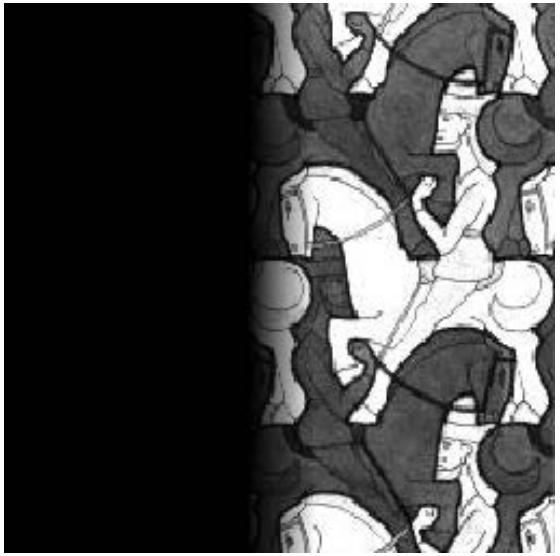


Blend

Slides adapted from A. Efros (CMU)

Feathering

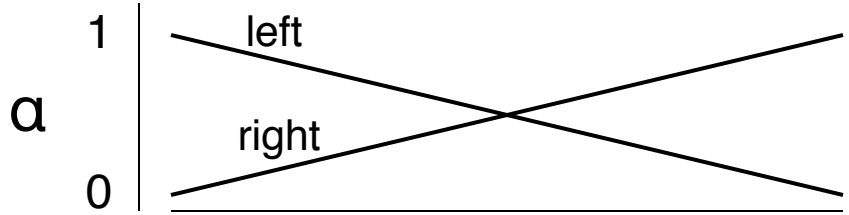
+

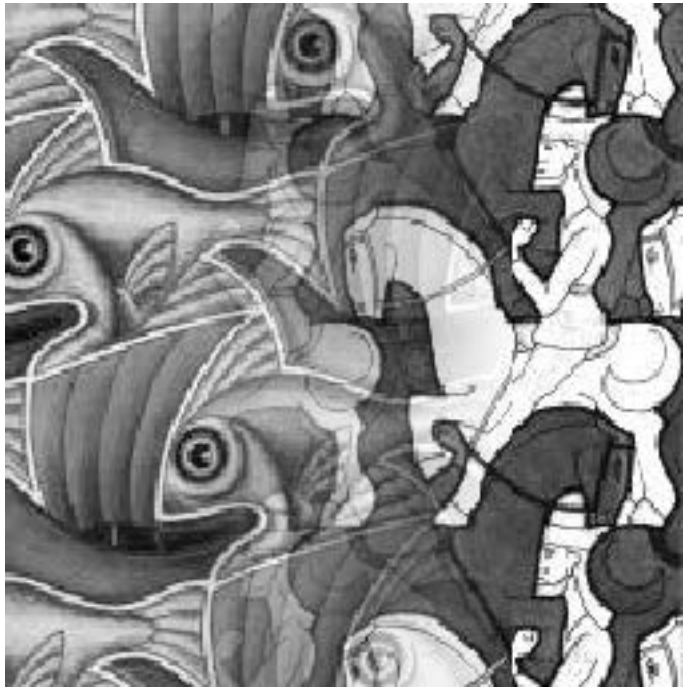


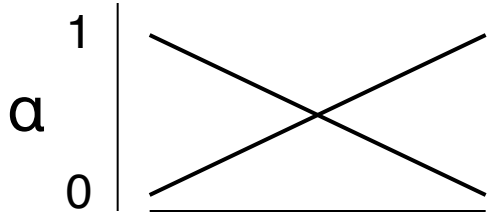
=

Assign an alpha value to each pixel near the seam to make it less visible.

Effect of Window Size

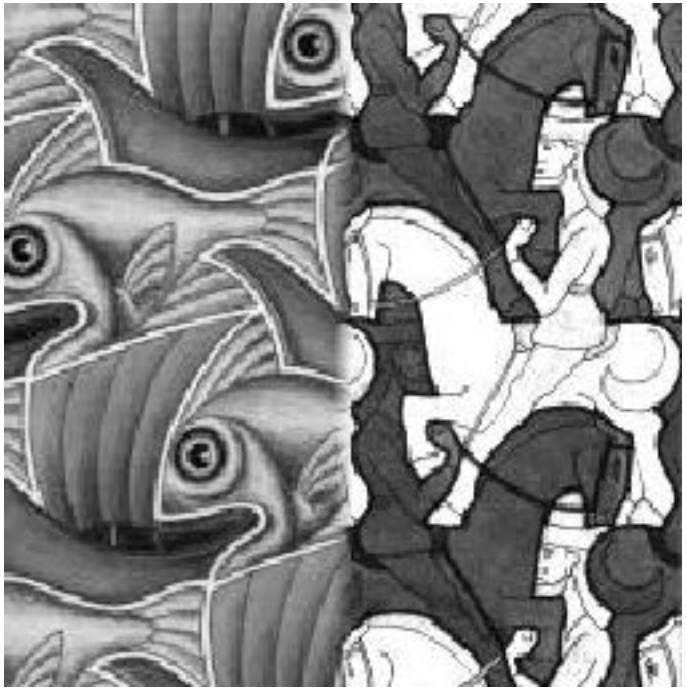


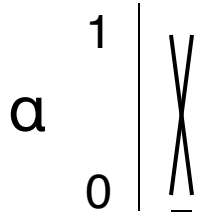


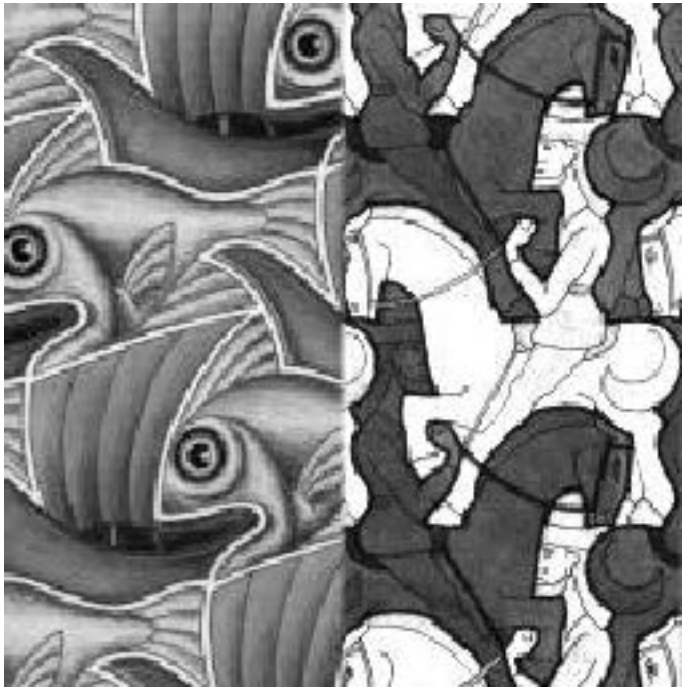


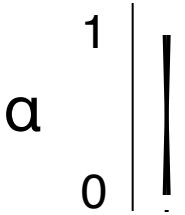
Ghosting is most apparent here

Effect of Window Size





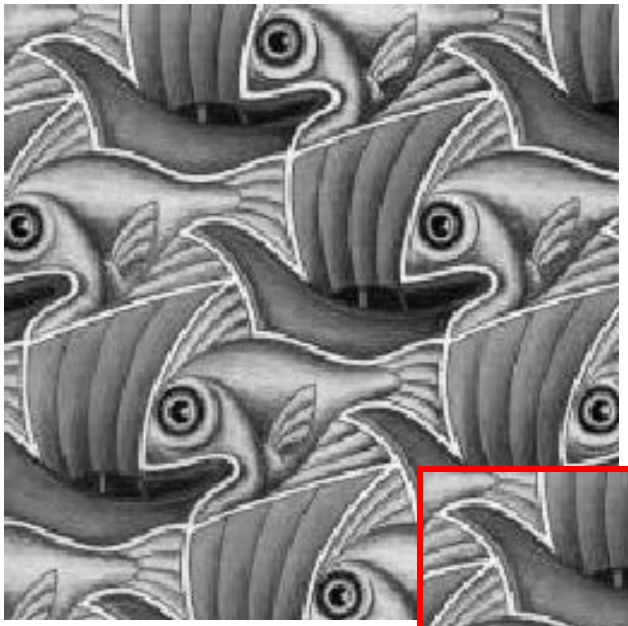




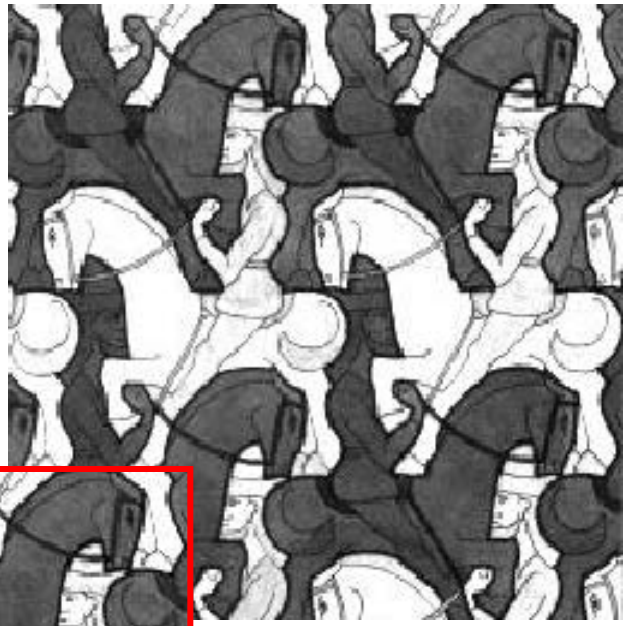
seam most visible here

Image Blending

Source A



Source B



Blend

Main challenge:

- Minimize ghosting without making visible seams

Slides adapted from A. Efros (CMU)

Application #1: Pyramid Blending Algorithm

Input: Source images A, B & binary matte M
Output: Blended image S

A

B

Pyramid Blending Algorithm

Input: Source images A, B & binary matte M
Output: Blended image S

A

Pad to make size
$(2^N + 1) \times (2^N + 1)$

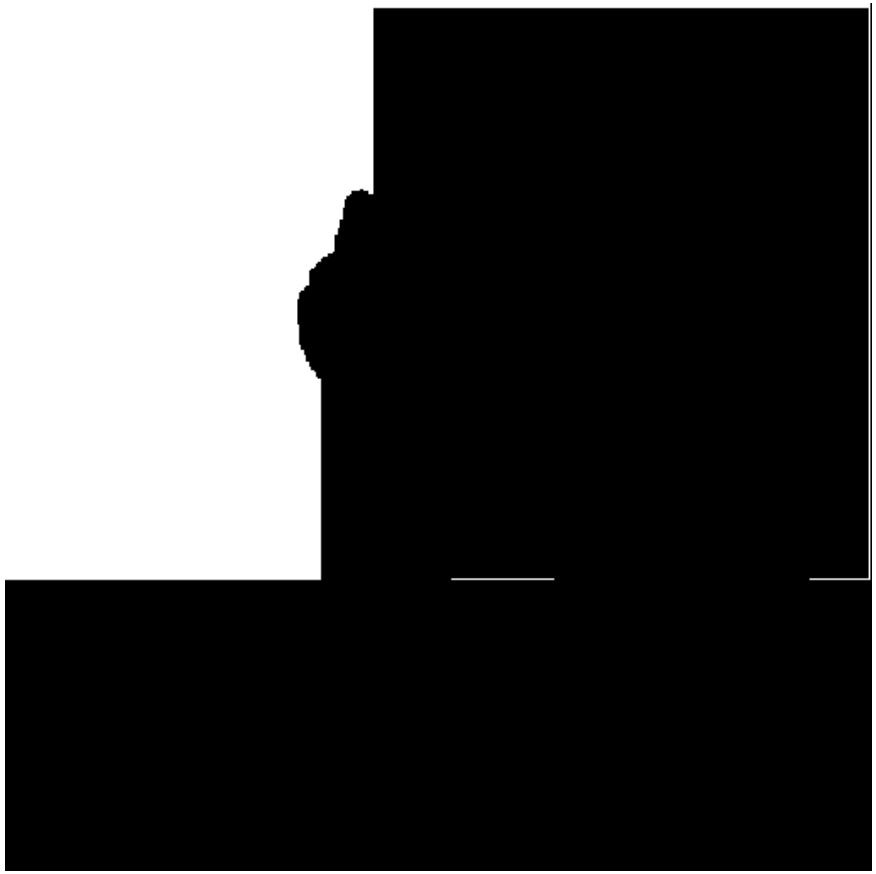
B

Pyramid Blending Algorithm

Input: Source images A, B & binary matte M
Output: Blended image S

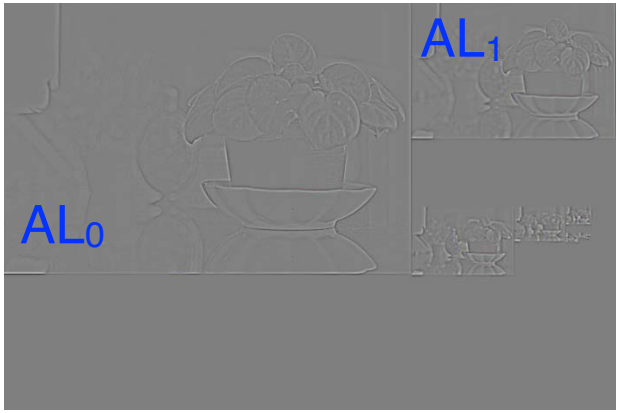
A

M



Pyramid Blending Algorithm

Input: Source images A, B & binary matte M
Output: Blended image S



1. Compute A's Laplacian pyramid: $AL_0, \ldots, AL_{N-1}, Ag_N$

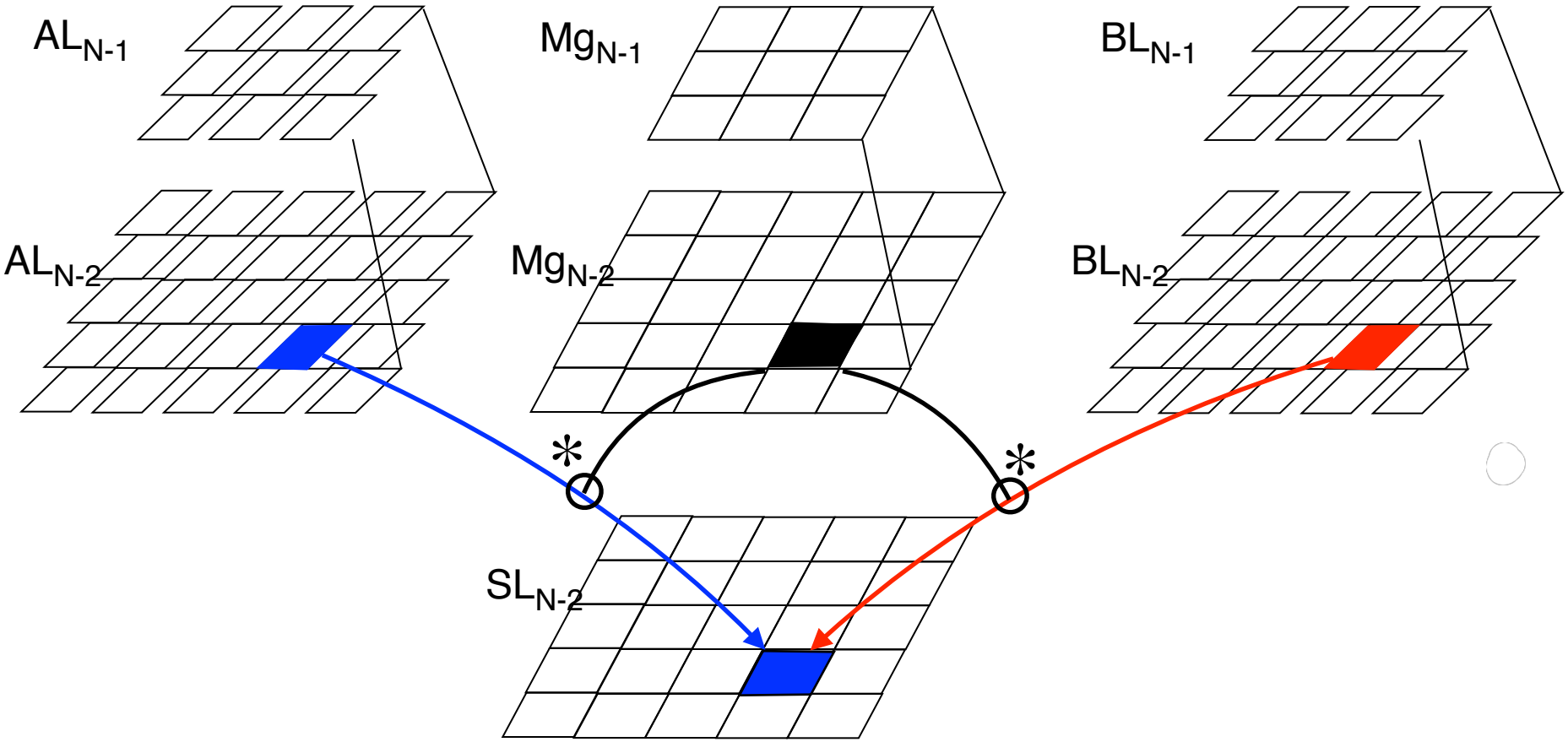
3. Compute M's Gaussian pyramid: $Mg_0, \ldots, Mg_N$

2. Compute B's Laplacian pyramid: $BL_0, \ldots, BL_{N-1}, Bg_N$

Pyramid Blending Algorithm

4. Compute the Laplacian`pyramid, SL_0, SL_1,… SL_{N-1}, Sg_N by applying the matting equation with matte Mg_l.

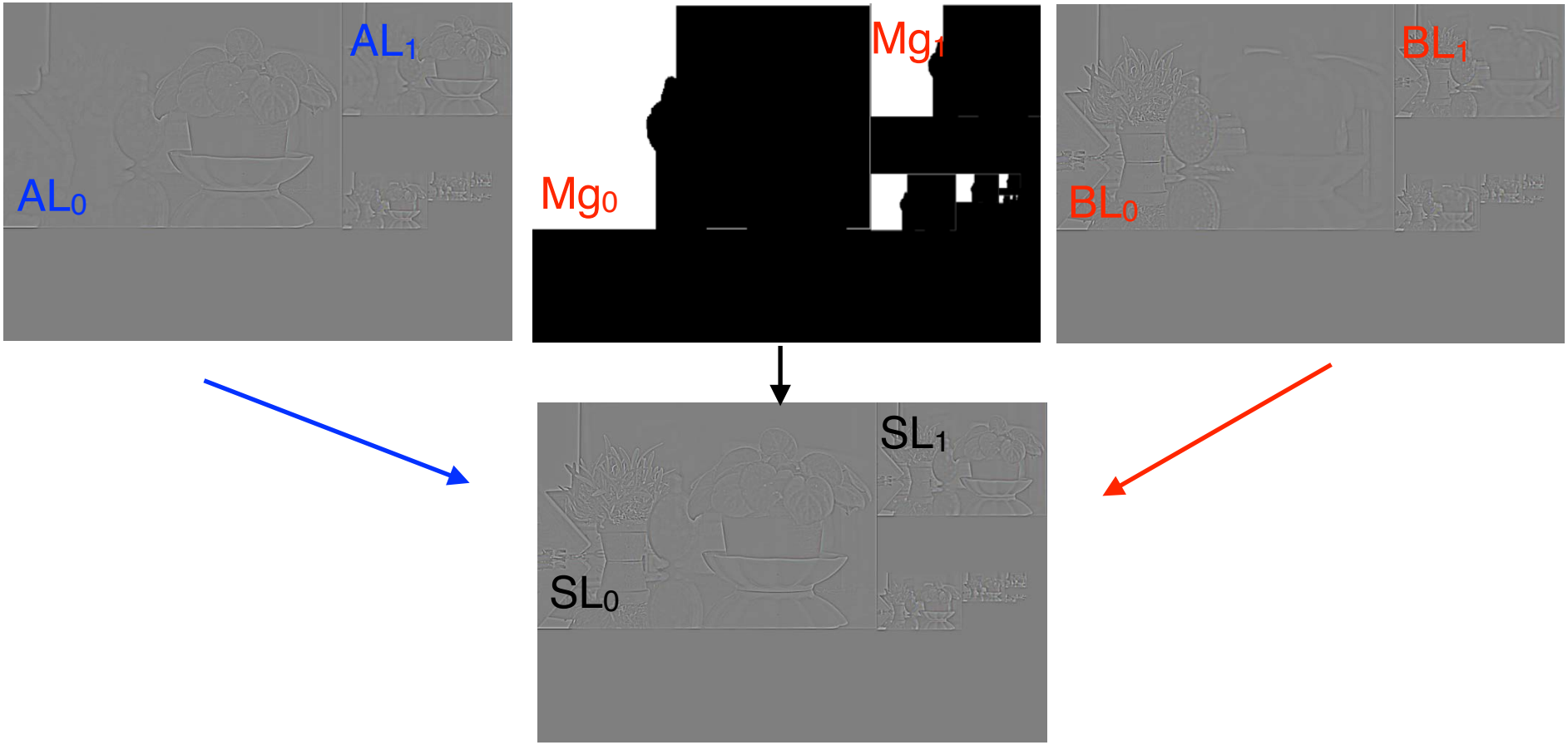
$$SL_l(i,j) = Mg_l(i,j)AL_l(i,j) + (1 - Mg_l(i,j))BL_l(i,j)$$



Pyramid Blending Algorithm

4. Compute the Laplacian`pyramid, SL_0, SL_1,… SL_{N-1}, Sg_N by applying the matting equation with matte Mg_l.

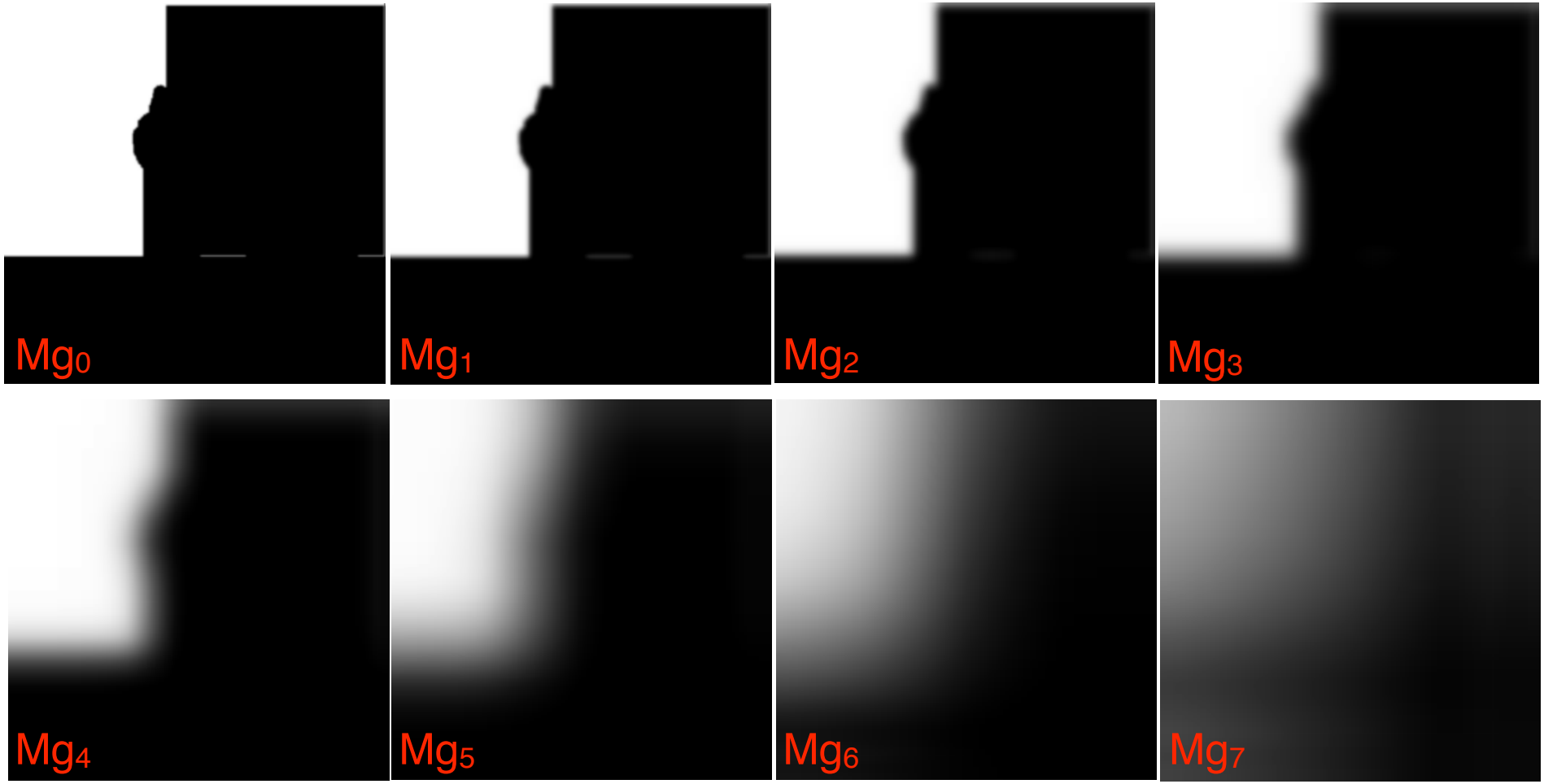
$$SL_l(i,j) = Mg_l(i,j)AL_l(i,j) + (1 - Mg_l(i,j))BL_l(i,j)$$



Pyramid Blending Algorithm

The algorithm effectively uses a different alpha matte for each level of detail.

↑ detail ⇒ ↓ feathering window ↓ detail ⇒ ↑ feathering window



Pyramid Blending Algorithm

5. Compute level 0 of the Gaussian pyramid of S from SL_0, SL_1,… SL_{N-1}, Sg_N.

Result S

⇑

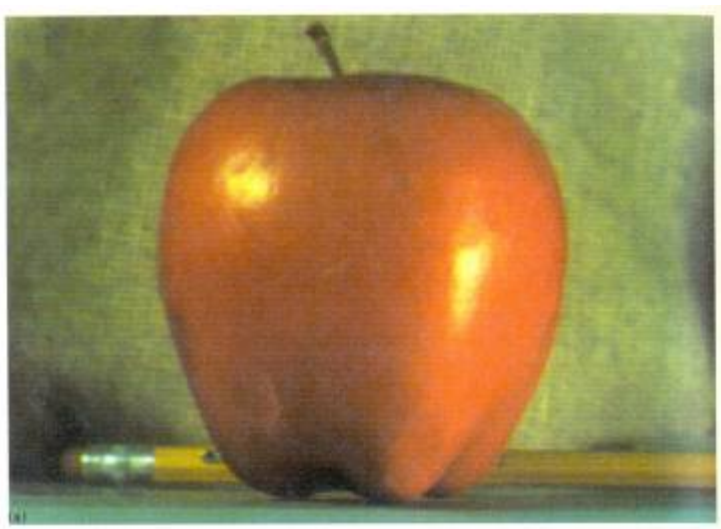
Pyramid Blending Algorithm

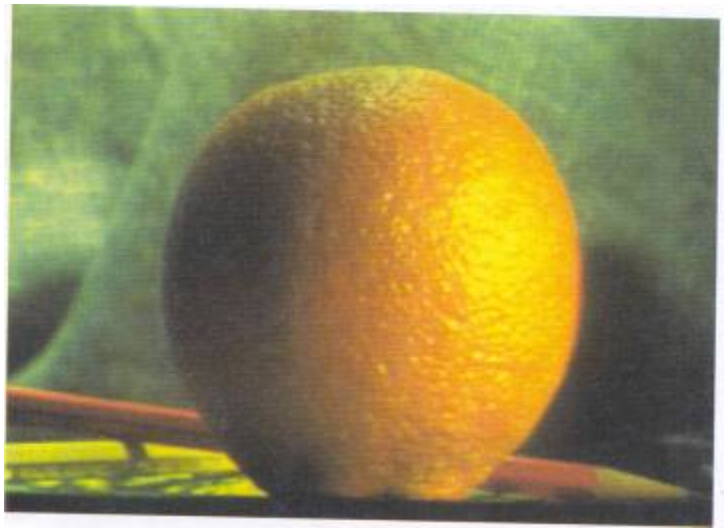
Blending Mis-Matched Photos (still looks OK)

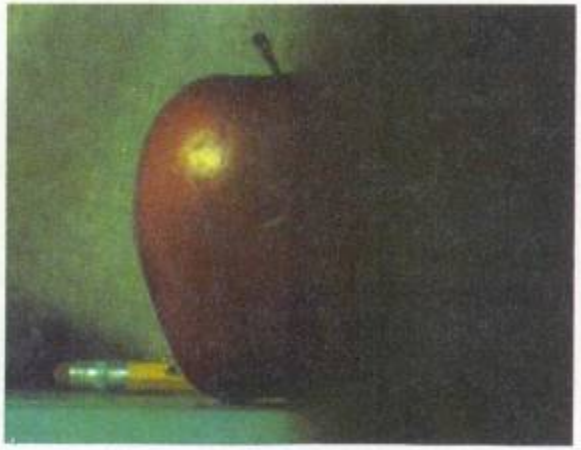
Merging Mis-Matched Photos (no blend)

Blending without Using Pyramid

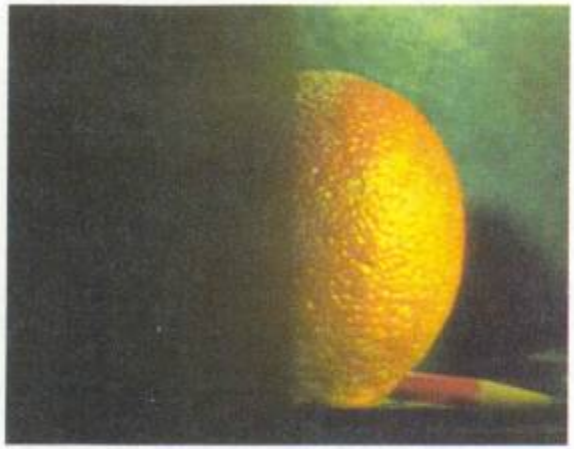
Pyramid Blending



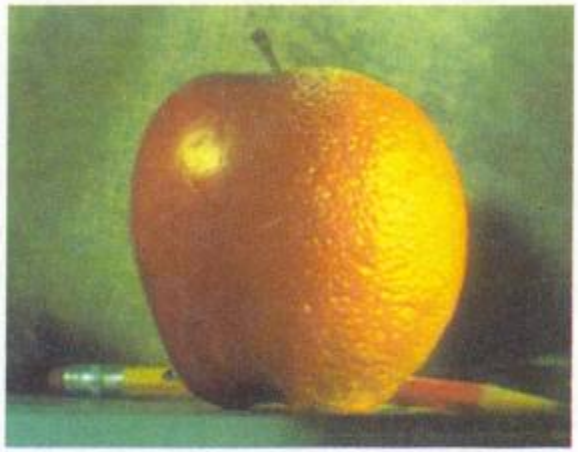




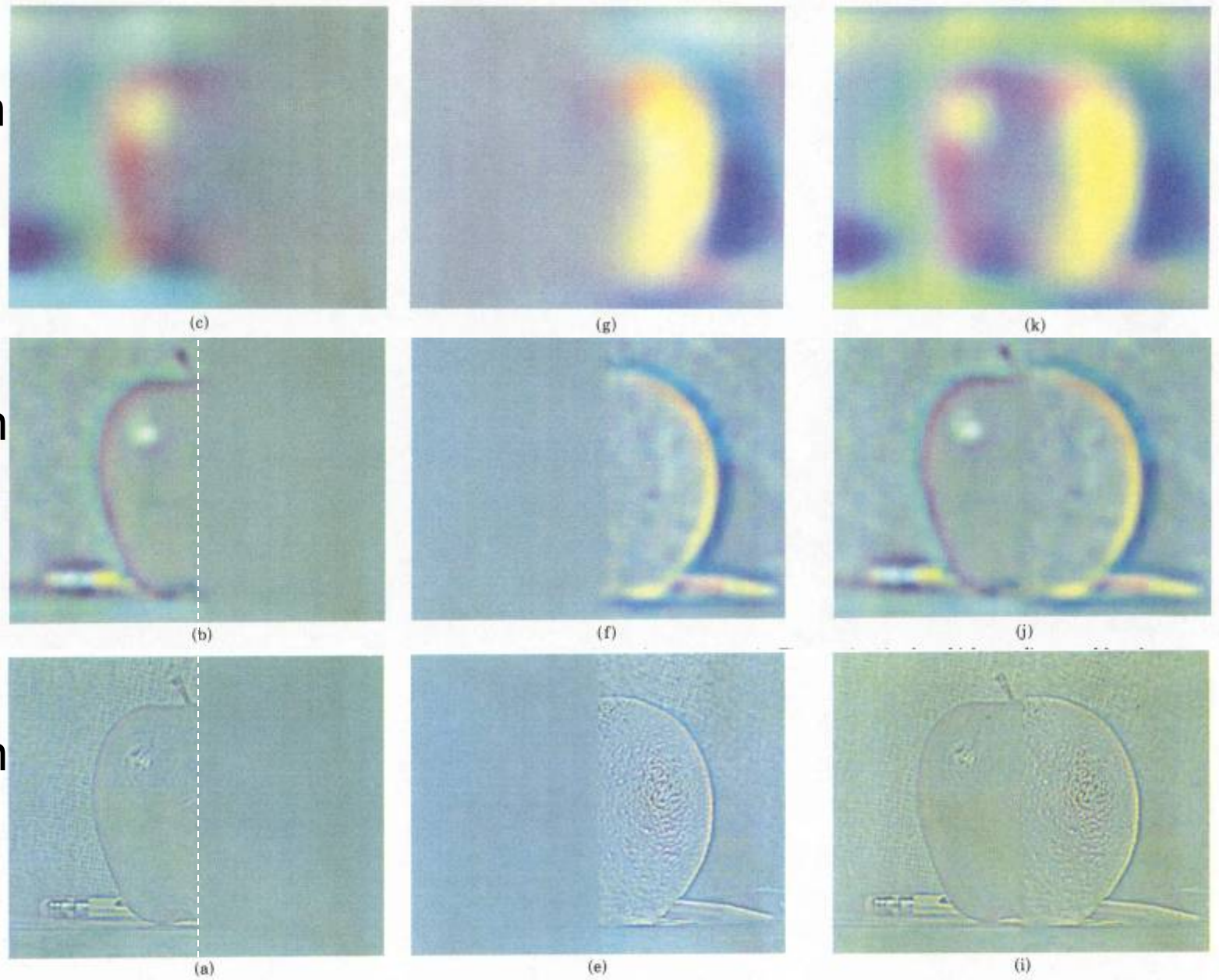
(d)



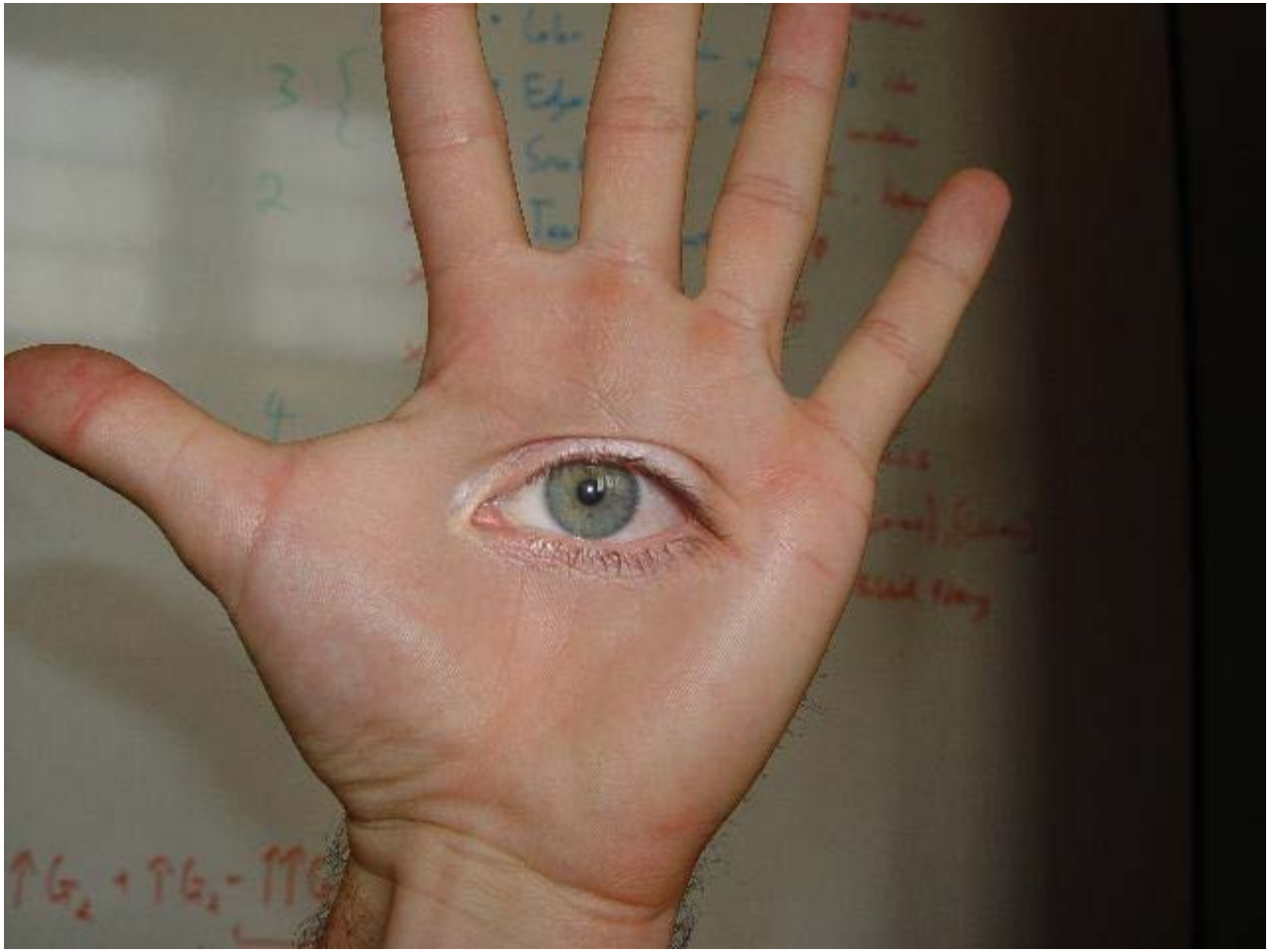
(h)



(l)


laplacian level 4
laplacian level 2
laplacian level 0
(c)
(g)
(k)
(b)
(f)
(j)
(a)
(e)
(i)
left pyramid
right pyramid
blended pyramid

Horror Photo



© prof. dmartin