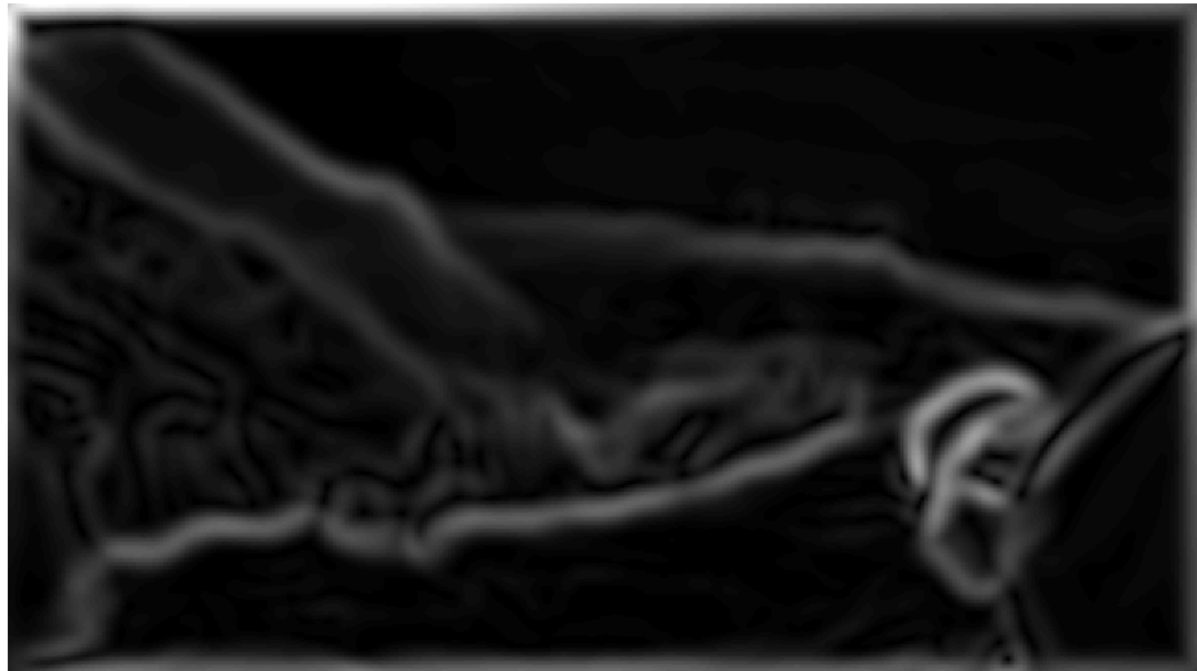
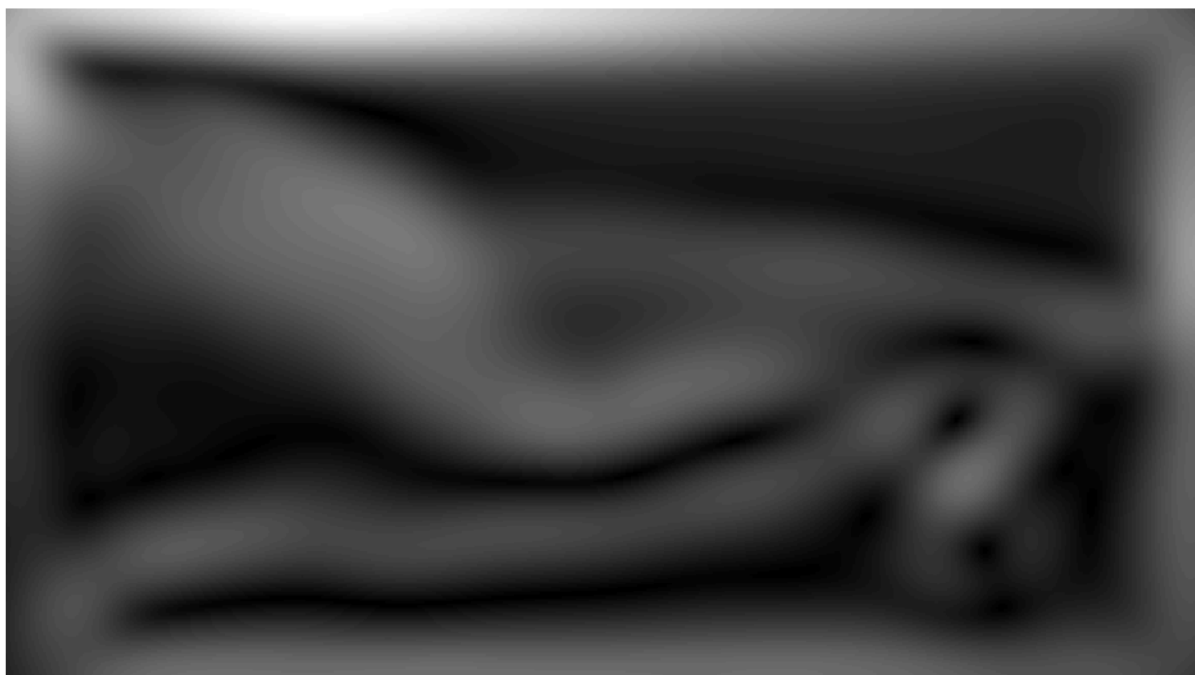


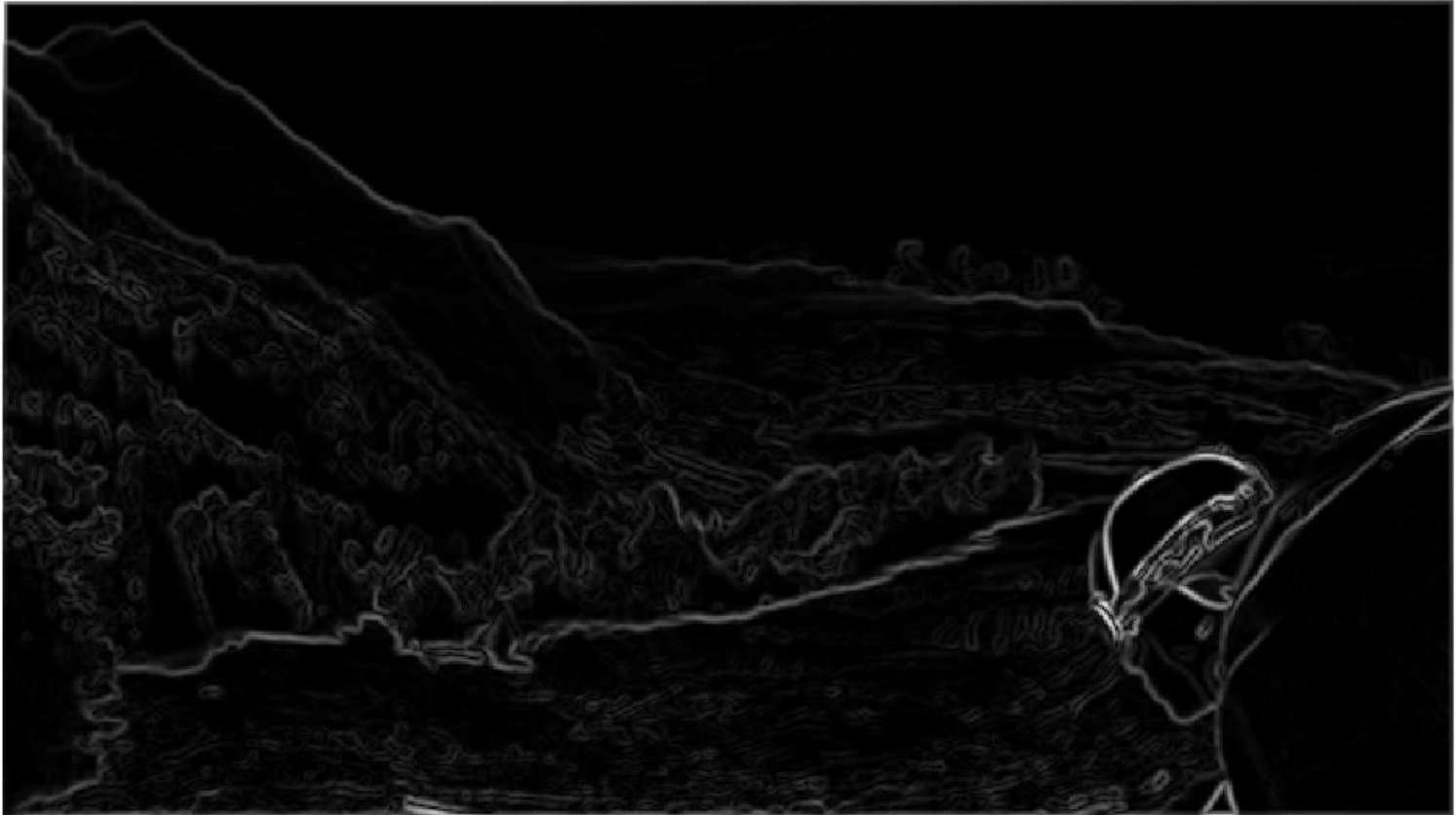


Image Scale





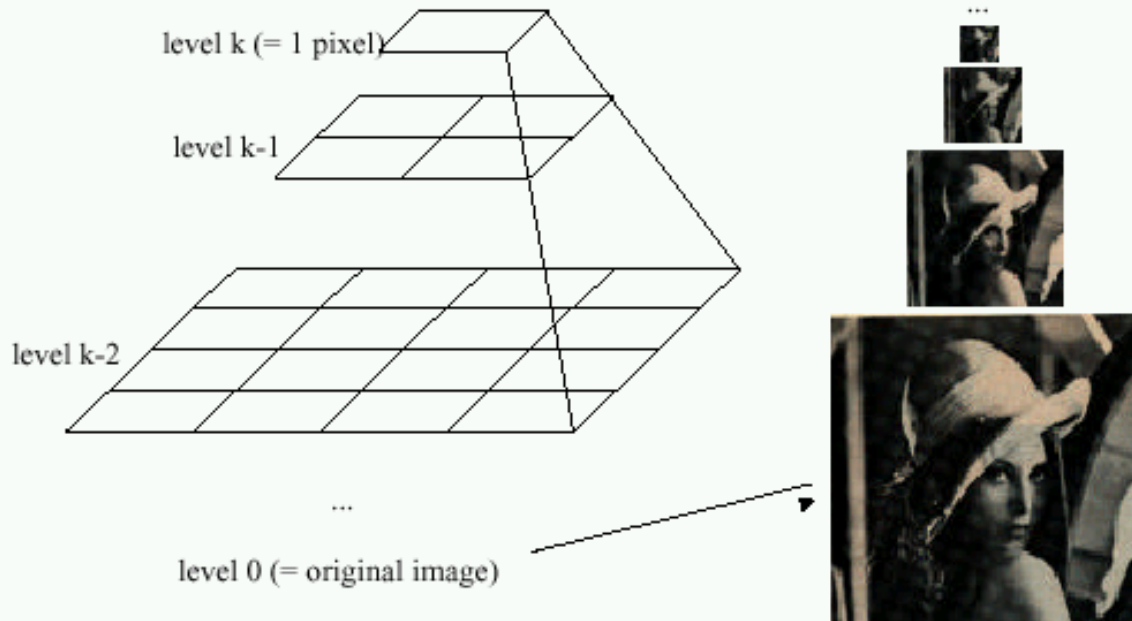




Different scale of image encodes different edge response.

Image Pyramids

Idea: Represent $N \times N$ image as a “pyramid” of $1 \times 1, 2 \times 2, 4 \times 4, \dots, 2^k \times 2^k$ images (assuming $N = 2^k$)



Known as a Gaussian Pyramid [Burt and Adelson, 1983]

- In computer graphics, a *mip map* [Williams, 1983]
- A precursor to *wavelet transform*



512

256

128

64

32

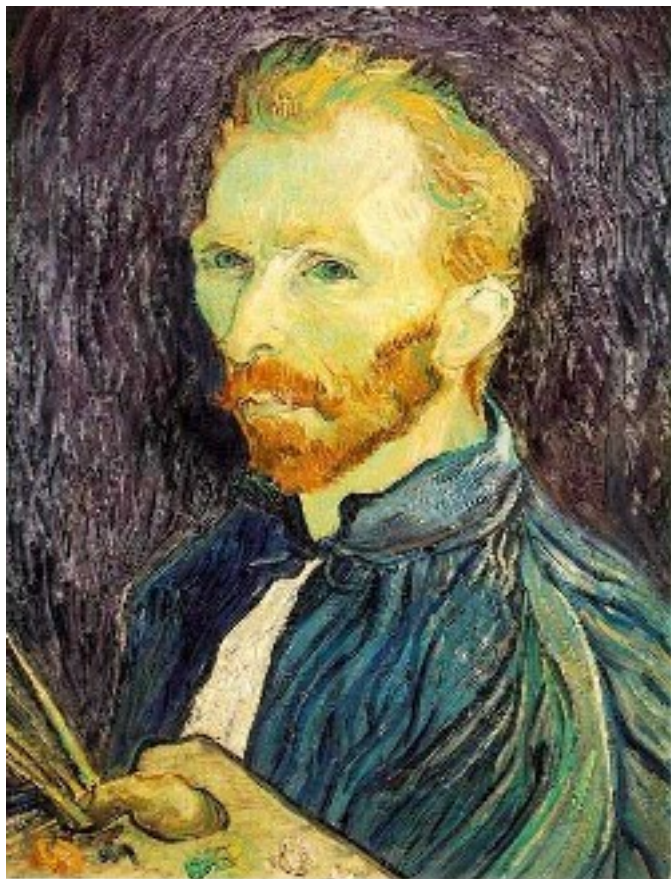
16

8



Figure from David Forsyth

Image sub-sampling



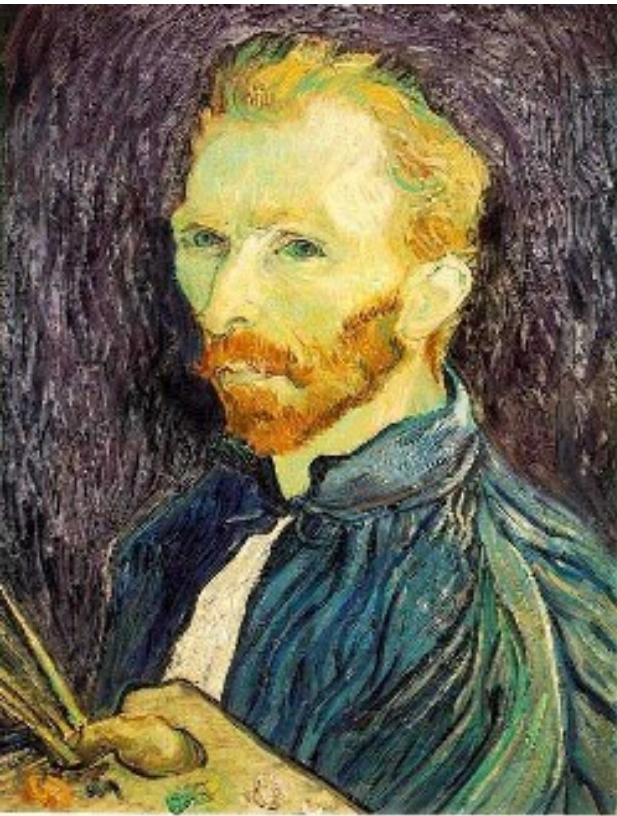
1/4



1/8

Throw away every other row and column to create a $1/2$ size image
- called *image sub-sampling*

Image sub-sampling



1/2



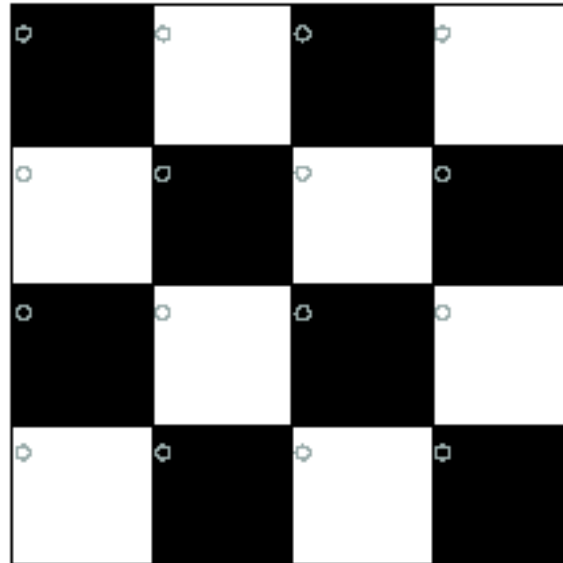
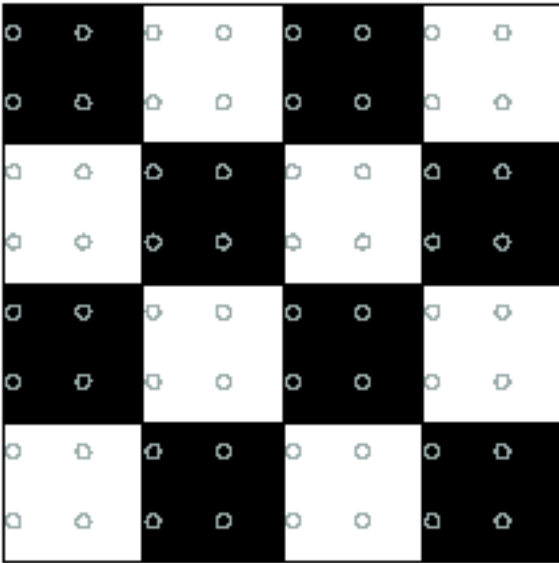
1/4 (2x zoom)



1/8 (4x zoom)

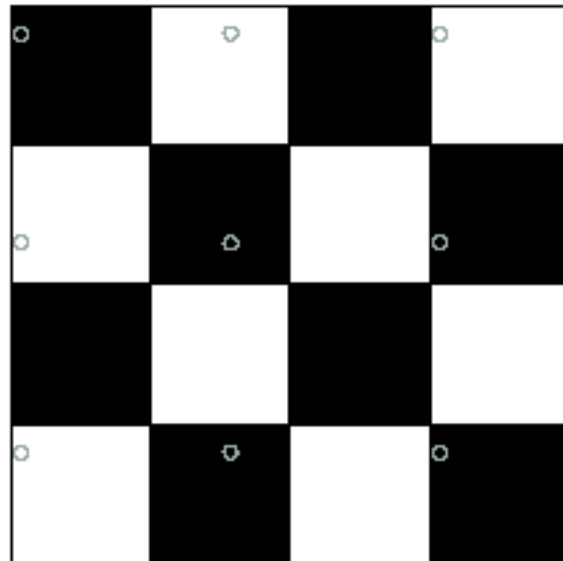
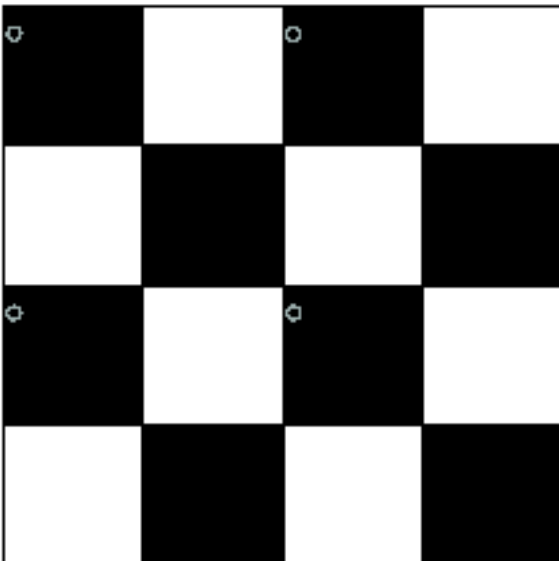
Why does this look so bad?

Sampling



Good sampling:

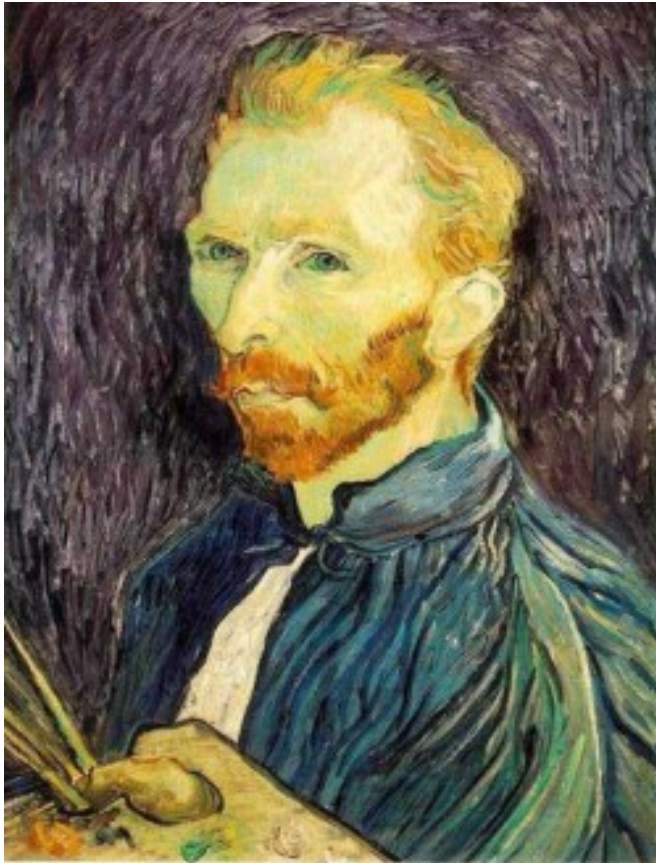
- Sample often or,
- Sample wisely



Bad sampling:

- see aliasing in action!

Gaussian pre-filtering



Gaussian 1/2



G 1/4



G 1/8

Solution: filter the image, *then* subsample

- Filter size should double for each $\frac{1}{2}$ size reduction. Why?

Subsampling with Gaussian pre-filtering



Gaussian 1/2



G 1/4

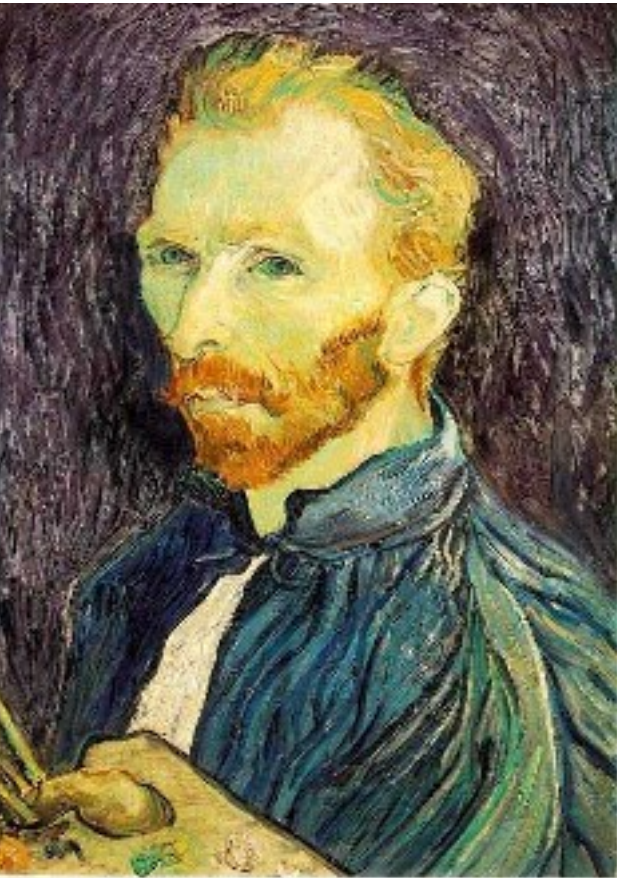


G 1/8

Solution: filter the image, *then* subsample

- Filter size should double for each $\frac{1}{2}$ size reduction. Why?
- How can we speed this up?

Comparison



1/2



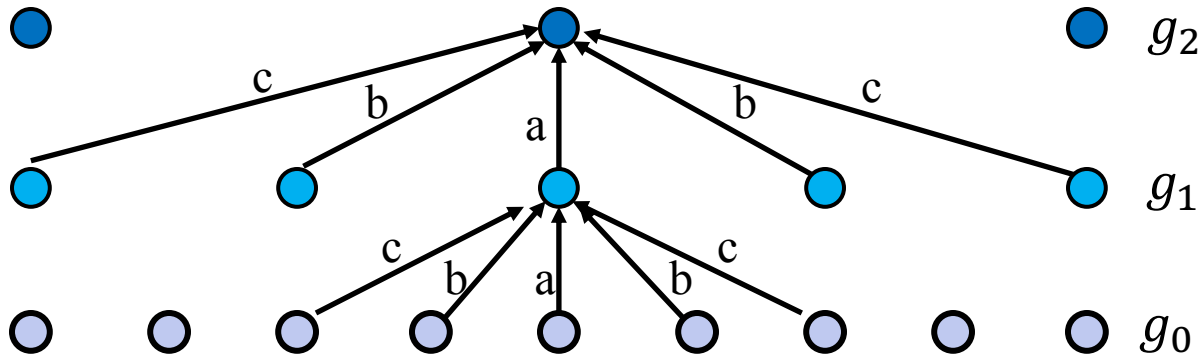
1/4 (2x zoom)



1/8 (4x zoom)

Image Reduce

[Burt & Adelson, 1983]

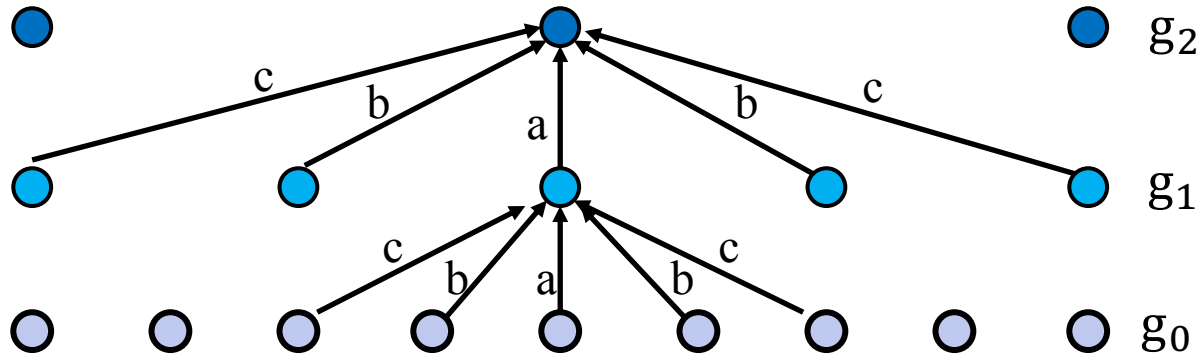


$g_0 = \text{Image}$

$g_1 = \text{REDUCE}[g_{l-1}]$

Image Reduce

[Burt & Adelson, 1983]



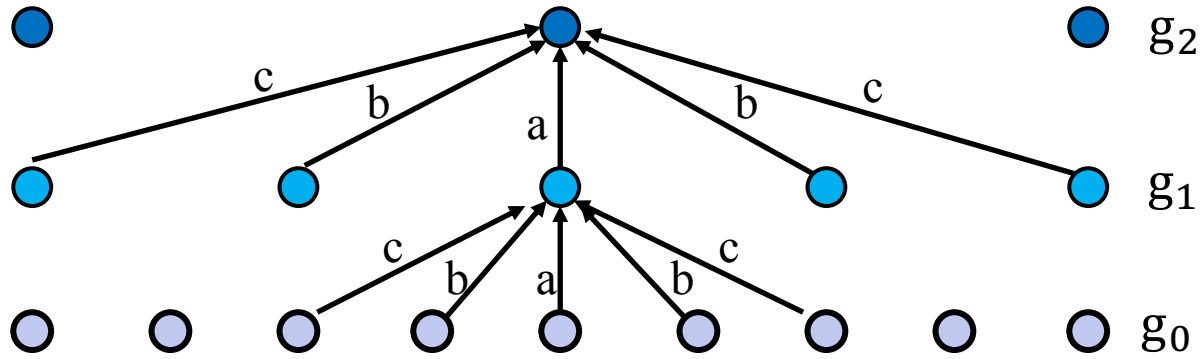
$g_0 = \text{Image}$

$g_l = \text{REDUCE}[g_{l-1}]$

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

$$w(m, n) = \hat{w}(m)\hat{w}(n) \quad \sum_m \hat{w}(m) = 1 \quad \hat{w}(m) = \hat{w}(-m)$$

Image Reduce



$g_0 = \text{IMAGE}$

$g_l = \text{REDUCE}[g_{l-1}]$

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

$$g_l = [g_{l-1} \otimes w] \downarrow 2$$

Choice in weighting function

$$\hat{w}(0) = a$$

$$\hat{w}(1) = \hat{w}(-1) = 1/4$$

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

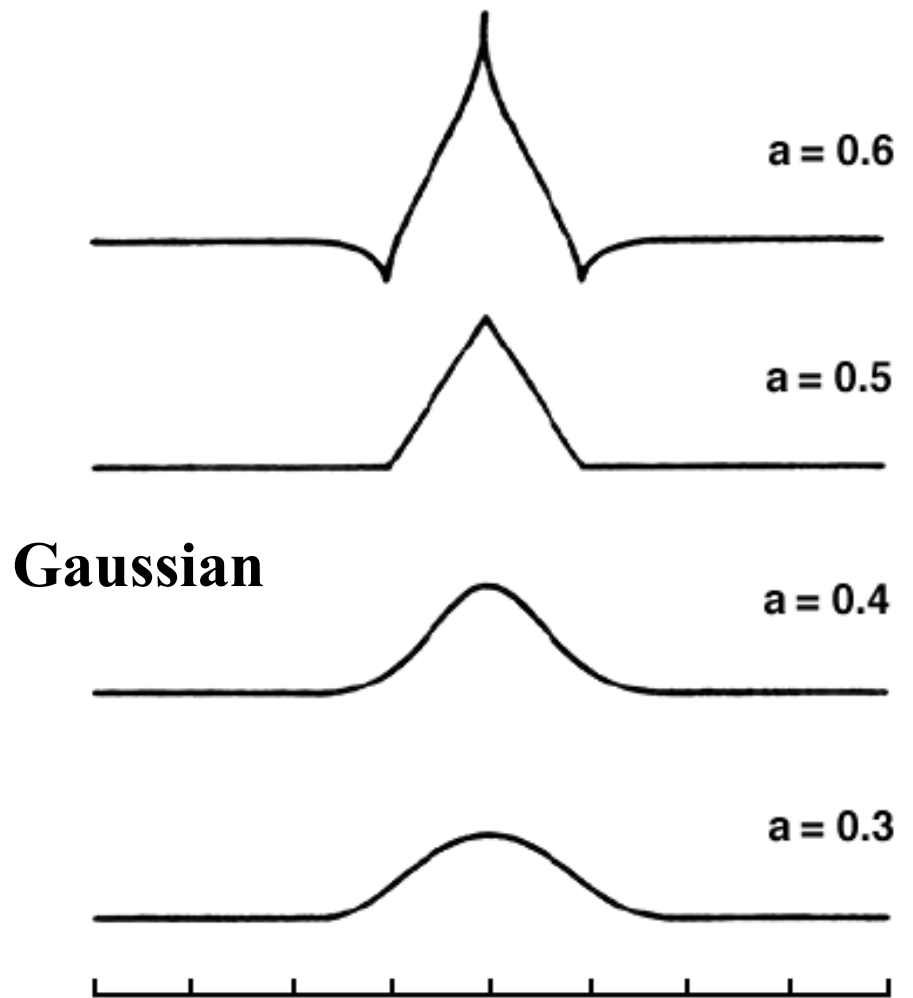
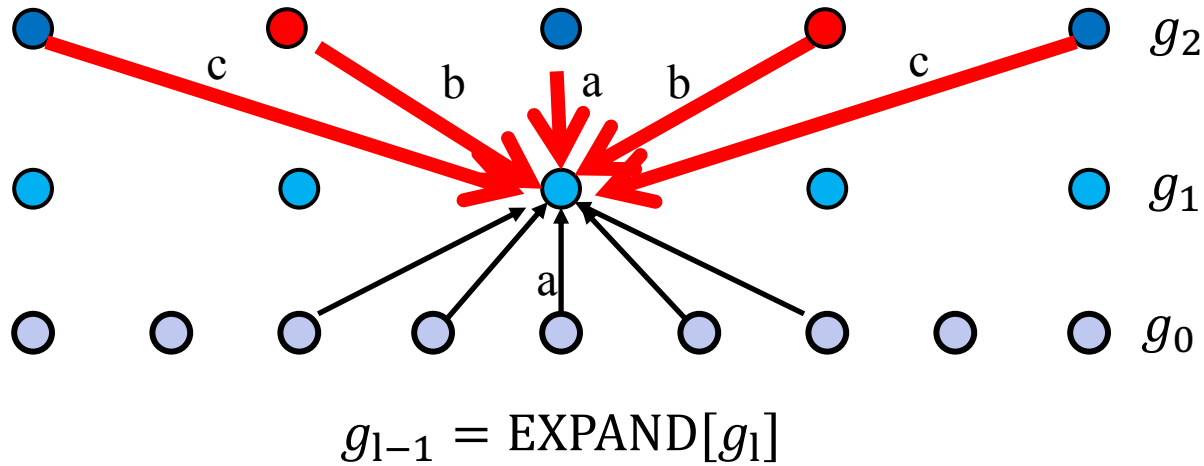


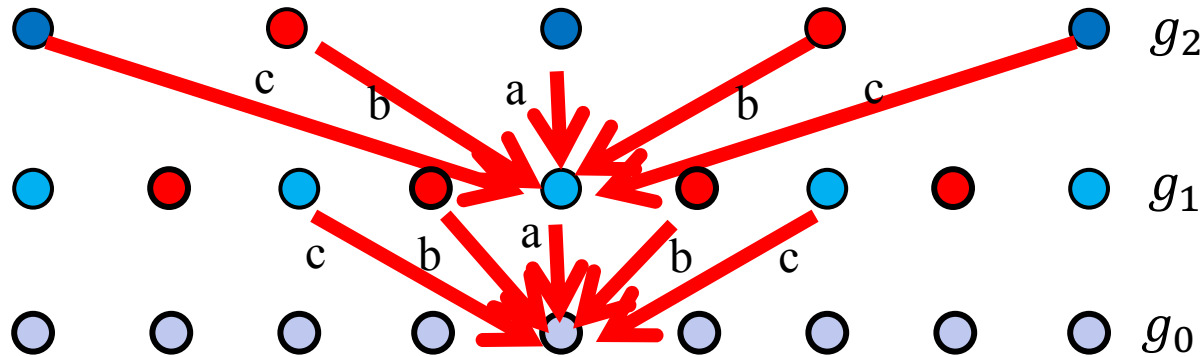
Image Expansion



$$g_l(i, j) = 4 \sum_{m=-2}^2 \sum_{n=-2}^2 w(m, n) \cdot g_{l-1} \left(\frac{i-m}{2}, \frac{j-n}{2} \right)$$

$(i-m)/2$ and $(j-n)/2$ are integers

Image Expansion

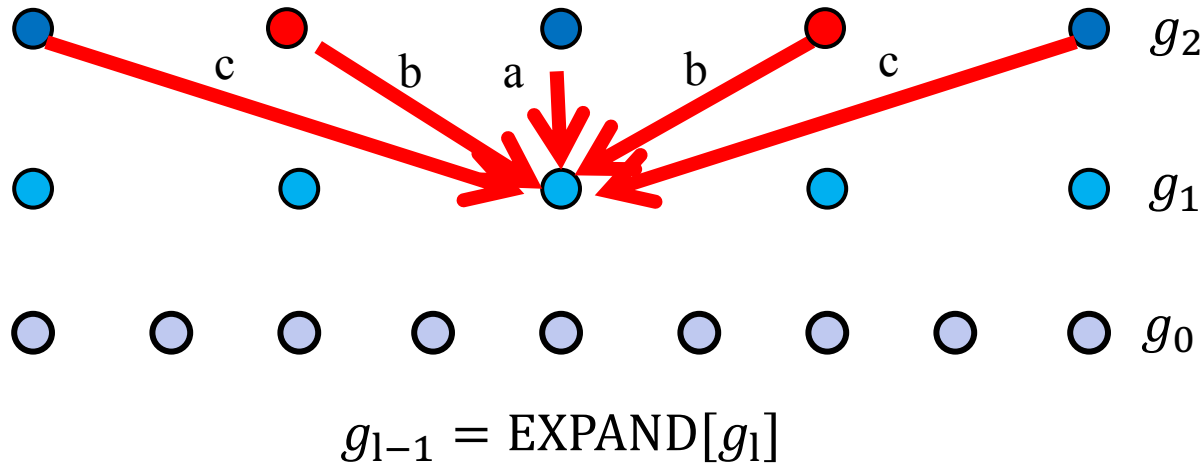


$$g_{l-1} = \text{EXPAND}[g_l]$$

$$g_l(i, j) = 4 \sum_{m=-2}^2 \sum_{n=-2}^2 w(m, n) \cdot g_{l-1} \left(\frac{i-m}{2}, \frac{j-n}{2} \right)$$

$(i-m)/2$ and $(j-n)/2$ are integers

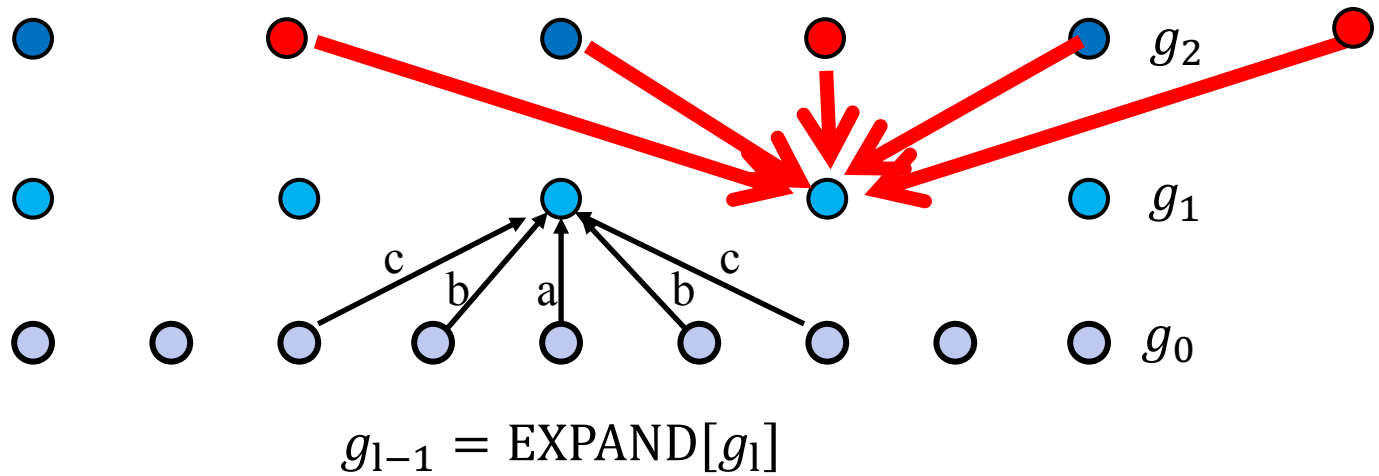
Image Expansion



$$g_l(i, j) = 4 \sum_{m=-2}^2 \sum_{n=-2}^2 w(m, n) \cdot g_{l-1} \left(\frac{i-m}{2}, \frac{j-n}{2} \right)$$

$(i-m)/2$ and $(j-n)/2$ are integers

Image Expansion

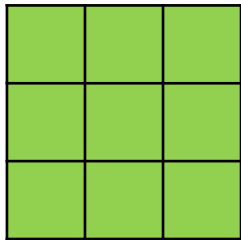


$$g_l(i, j) = 4 \sum_{m=-2}^2 \sum_{n=-2}^2 w(m, n) \cdot g_{l-1} \left(\frac{i-m}{2}, \frac{j-n}{2} \right)$$

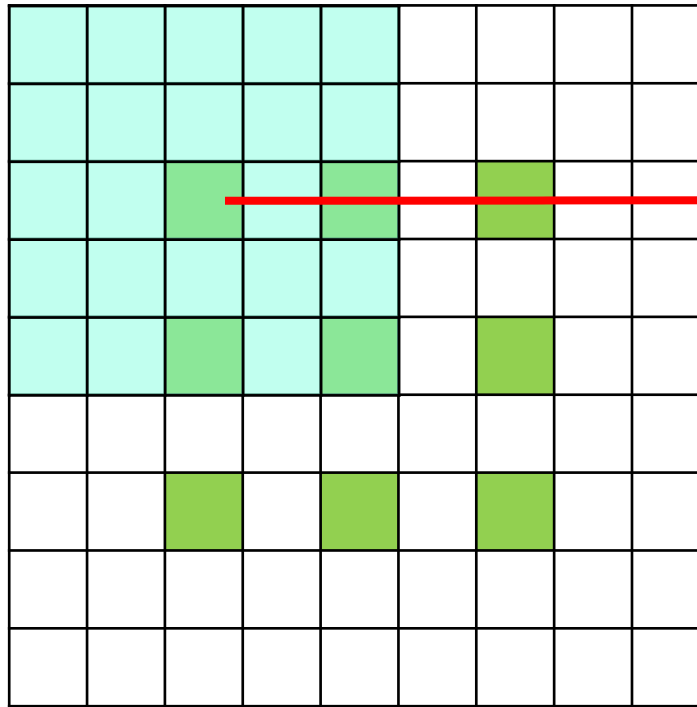
$(i-m)/2$ and $(j-n)/2$ are integers

2D Image Expansion (part1)

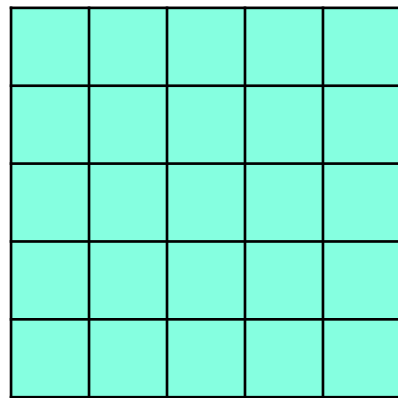
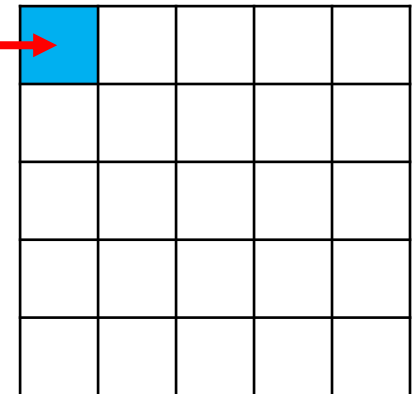
g_1



g_1 padded with 0s



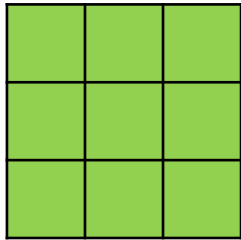
g_0



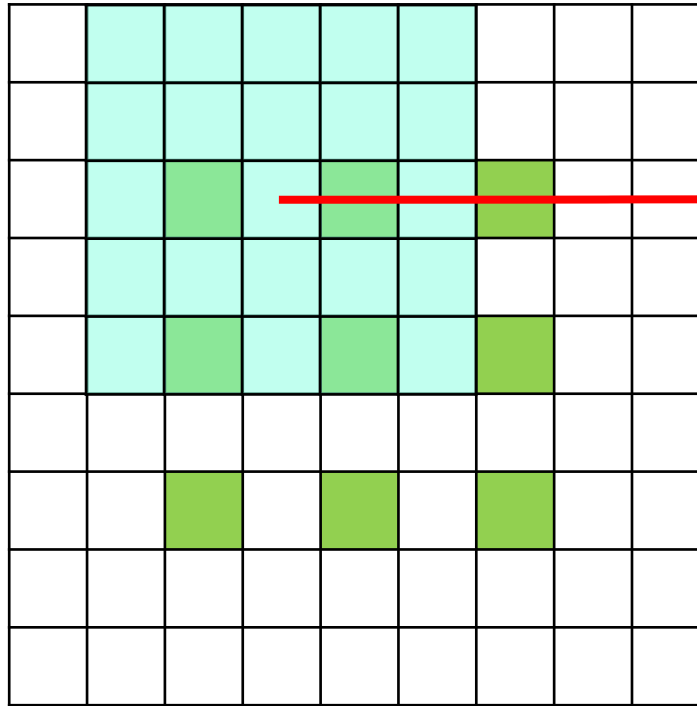
2D Gaussian kernel

2D Image Expansion (part2)

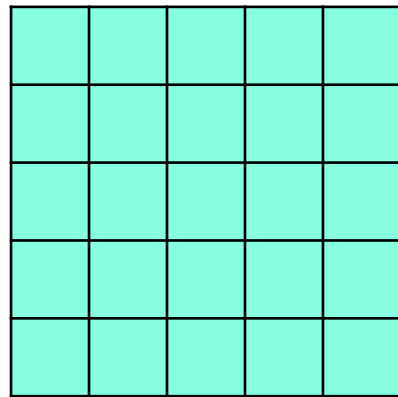
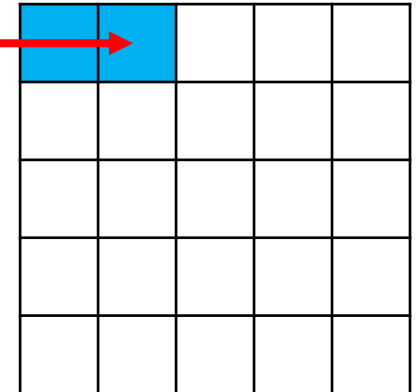
g_1



g_1 padded with 0s



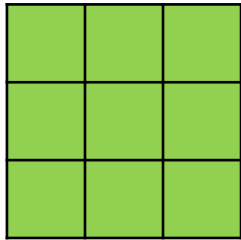
g_0



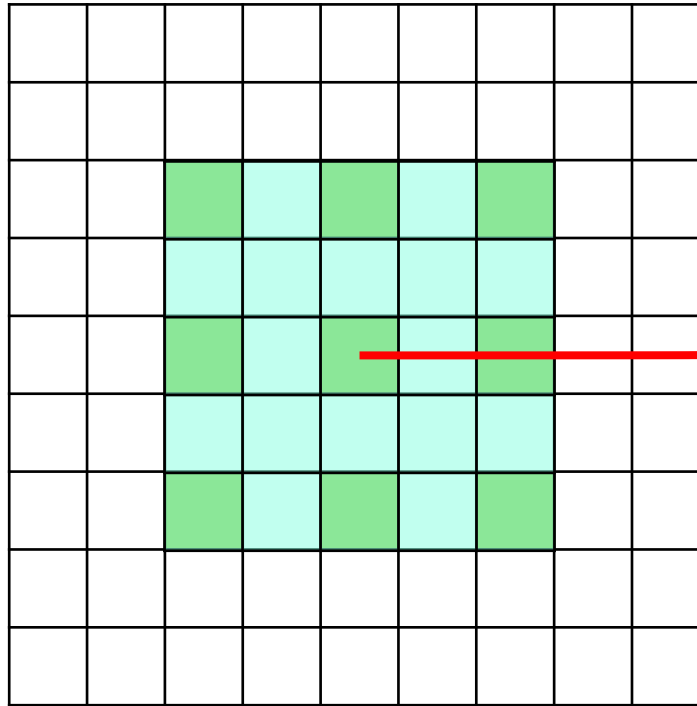
2D Gaussian kernel

2D Image Expansion (part3)

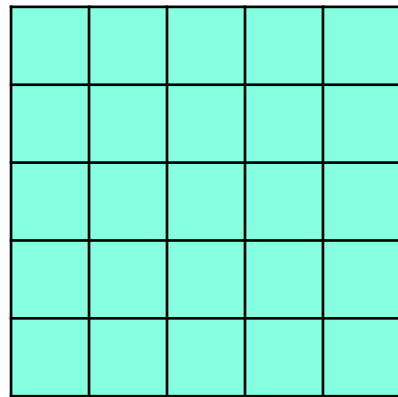
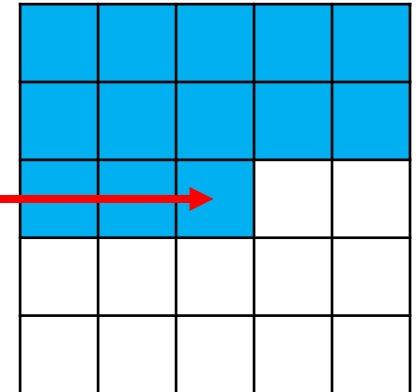
g_1



g_1 padded with 0s



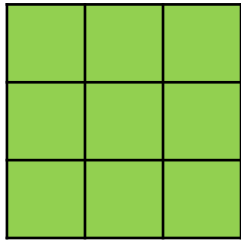
g_0



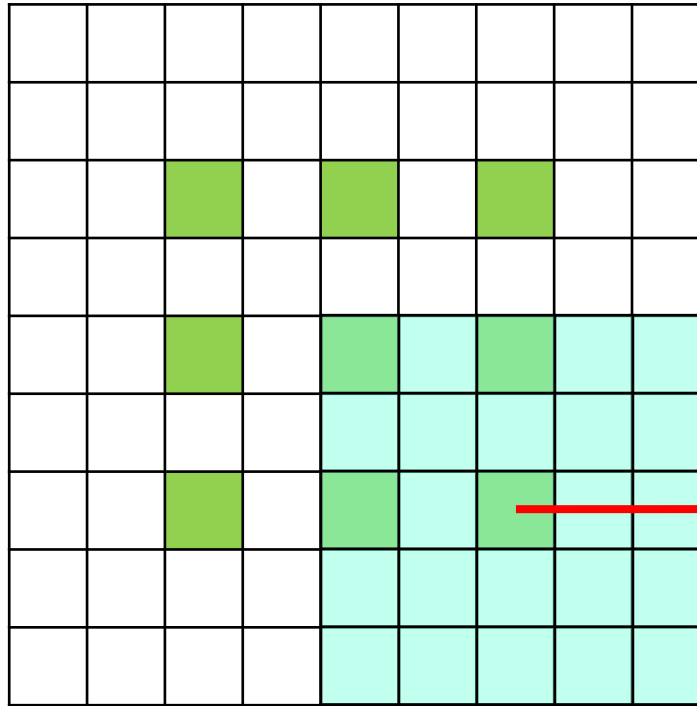
2D Gaussian kernel

2D Image Expansion (part4)

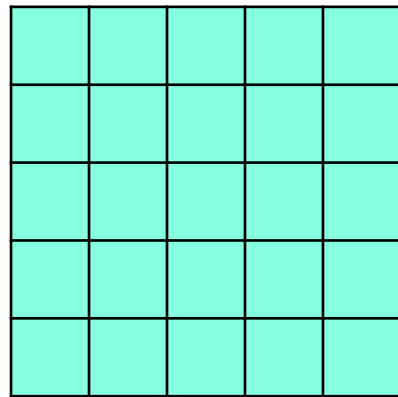
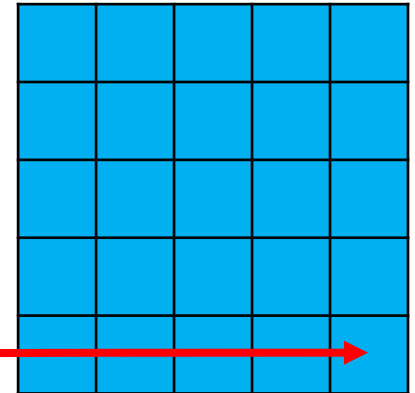
g_1



g_1 padded with 0s



g_0

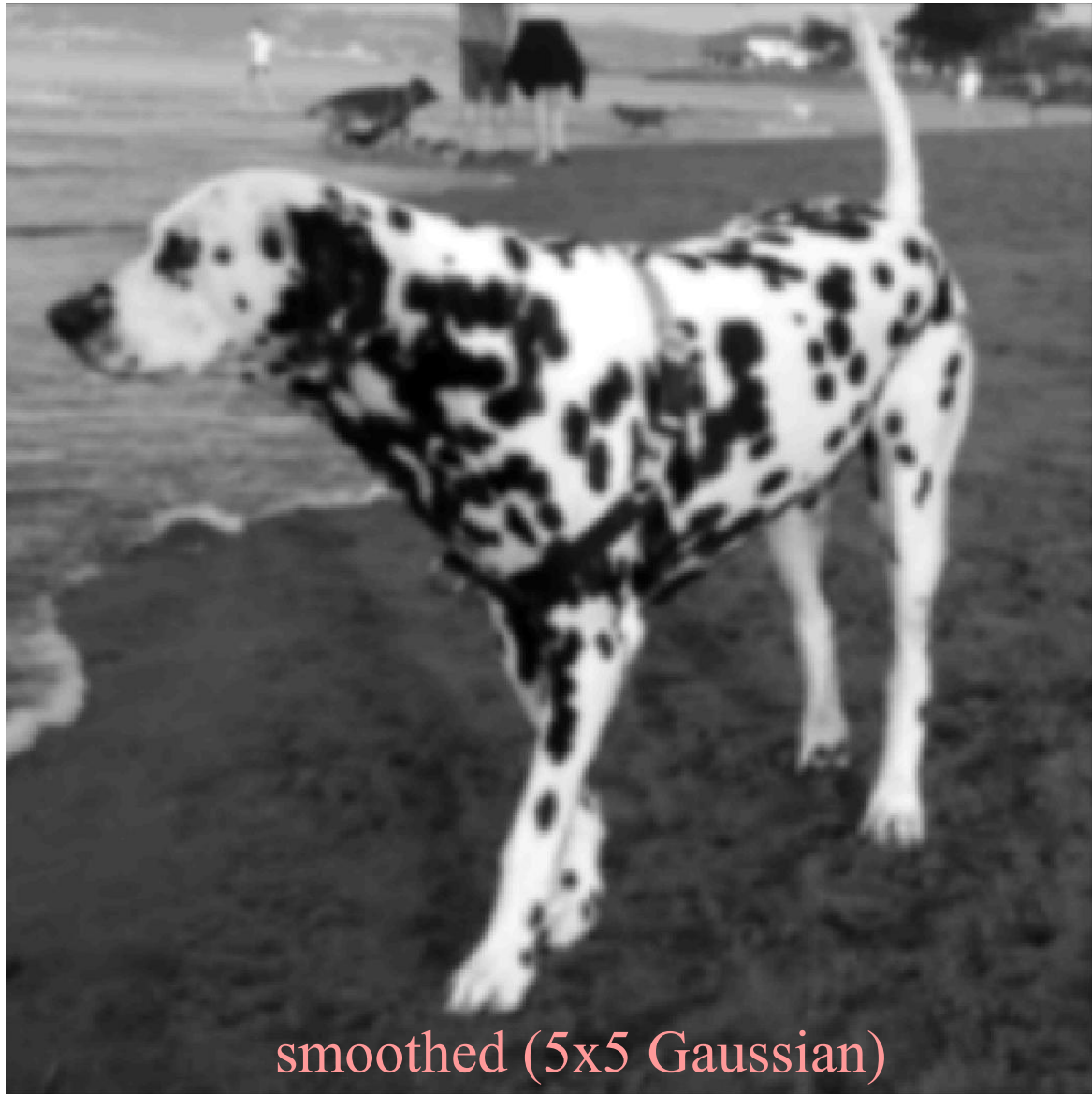


2D Gaussian kernel

What does blurring take away?

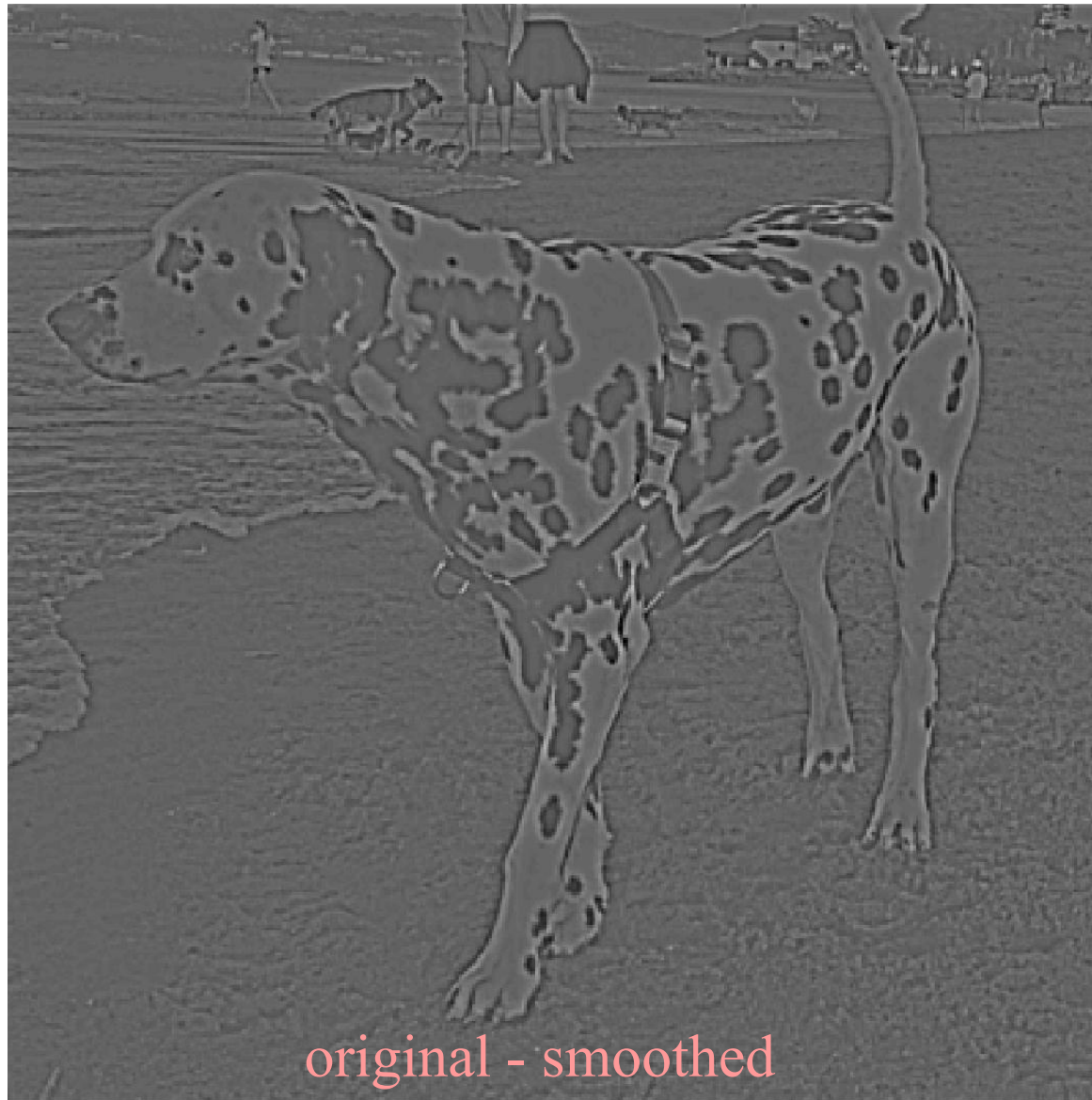


What does blurring take away?



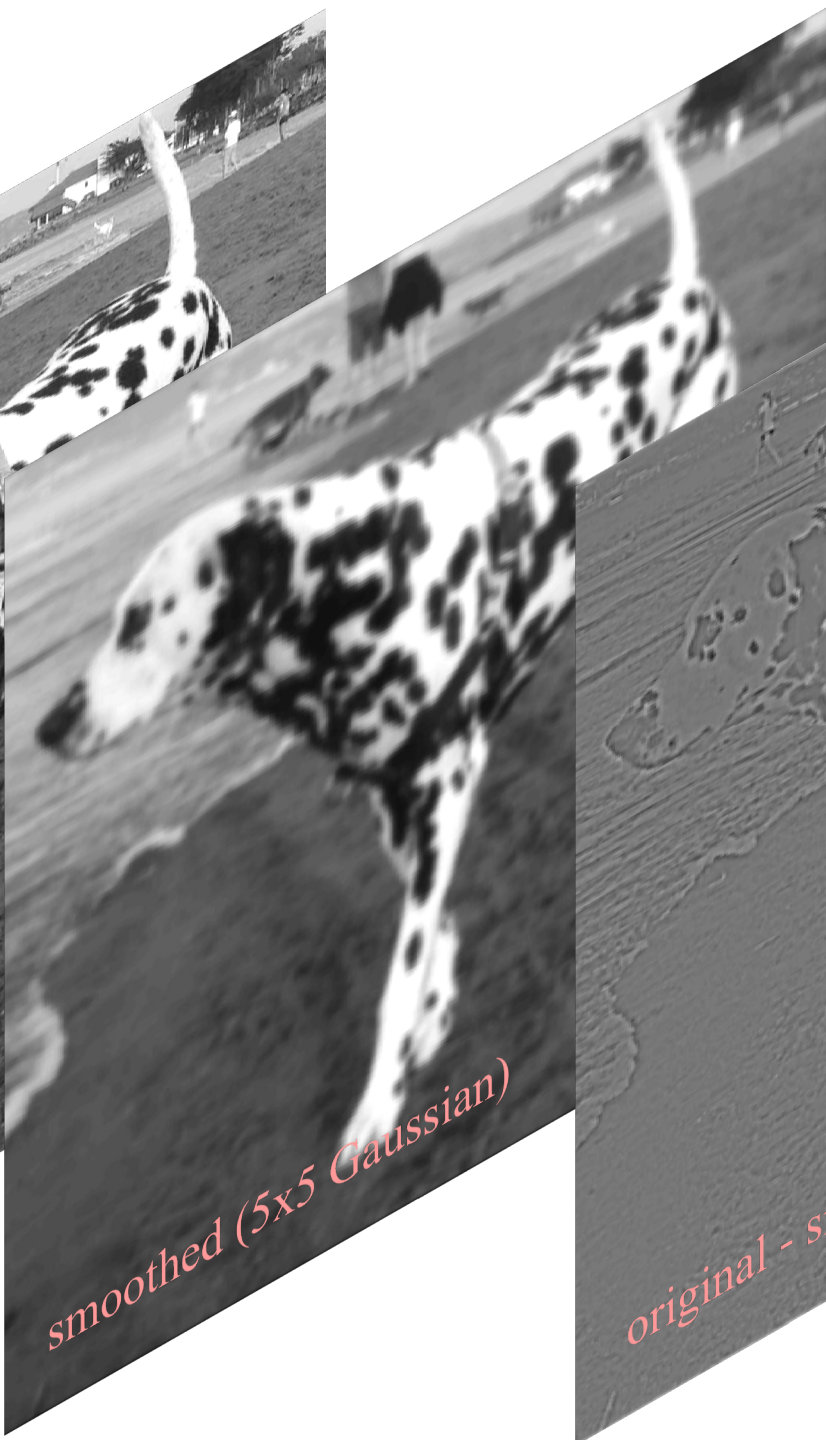
smoothed (5x5 Gaussian)

Difference as result of smoothing

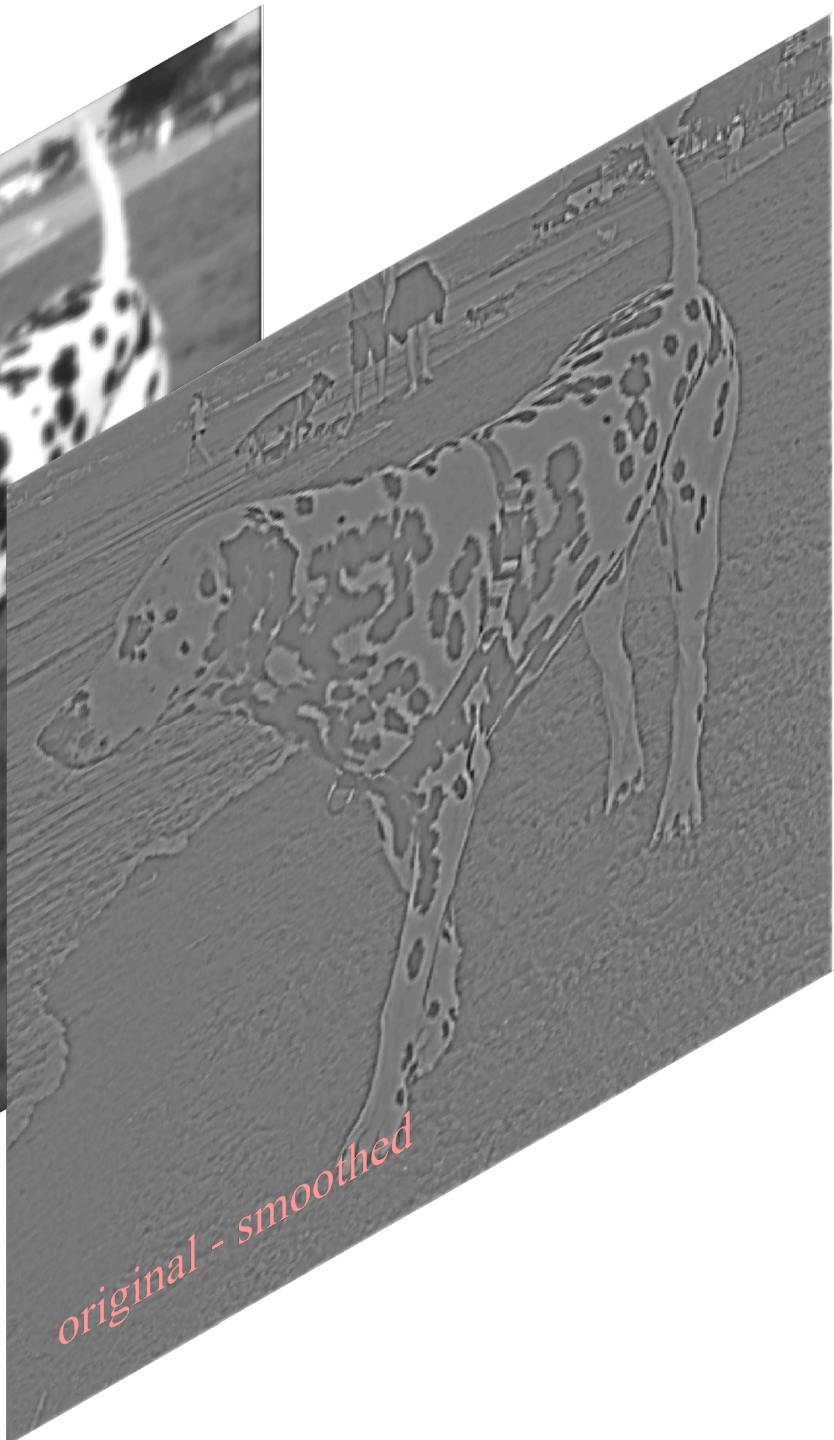




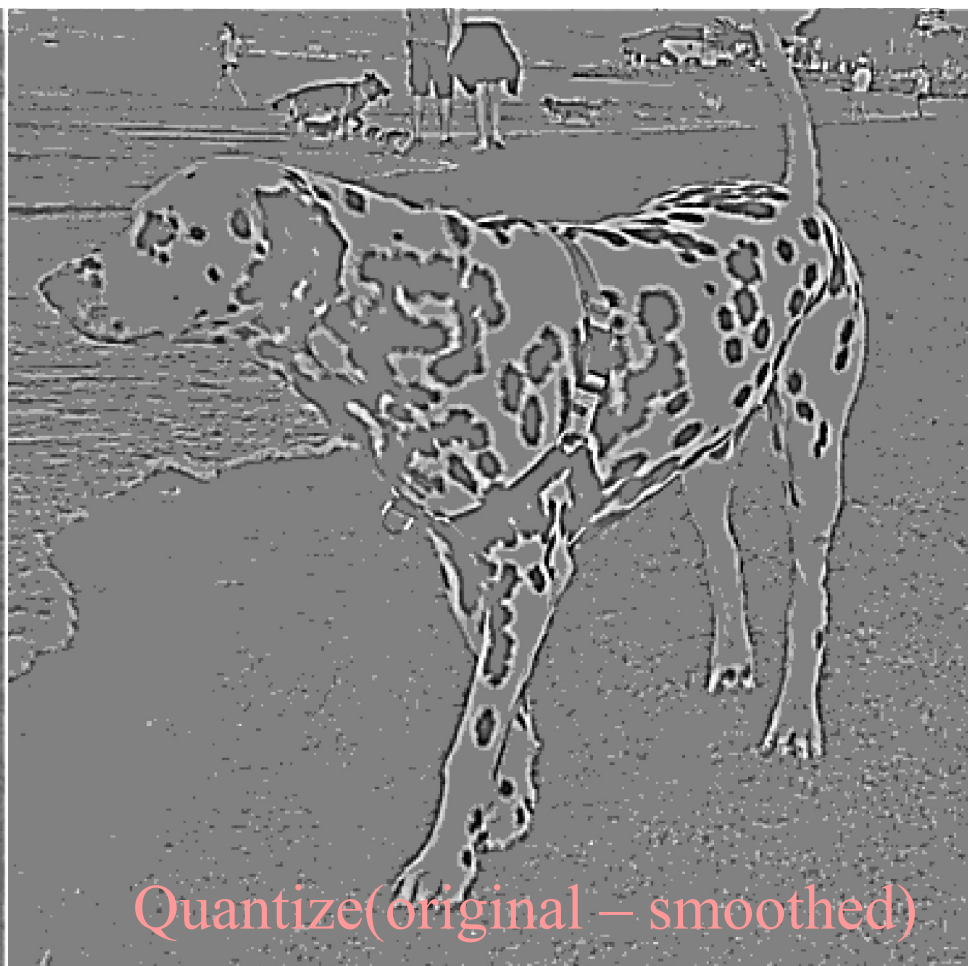
original



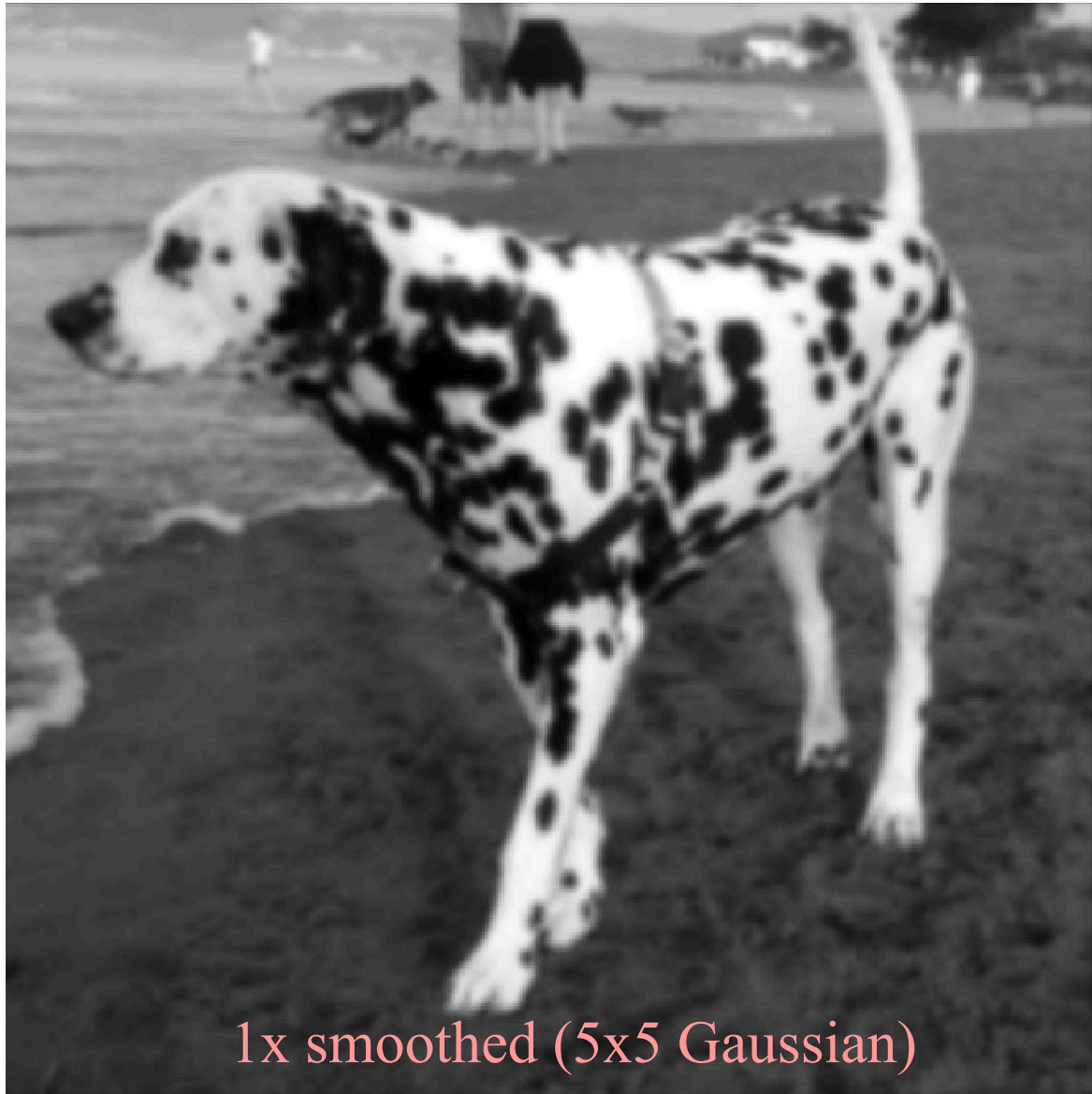
smoothed (5x5 Gaussian)



original - smoothed

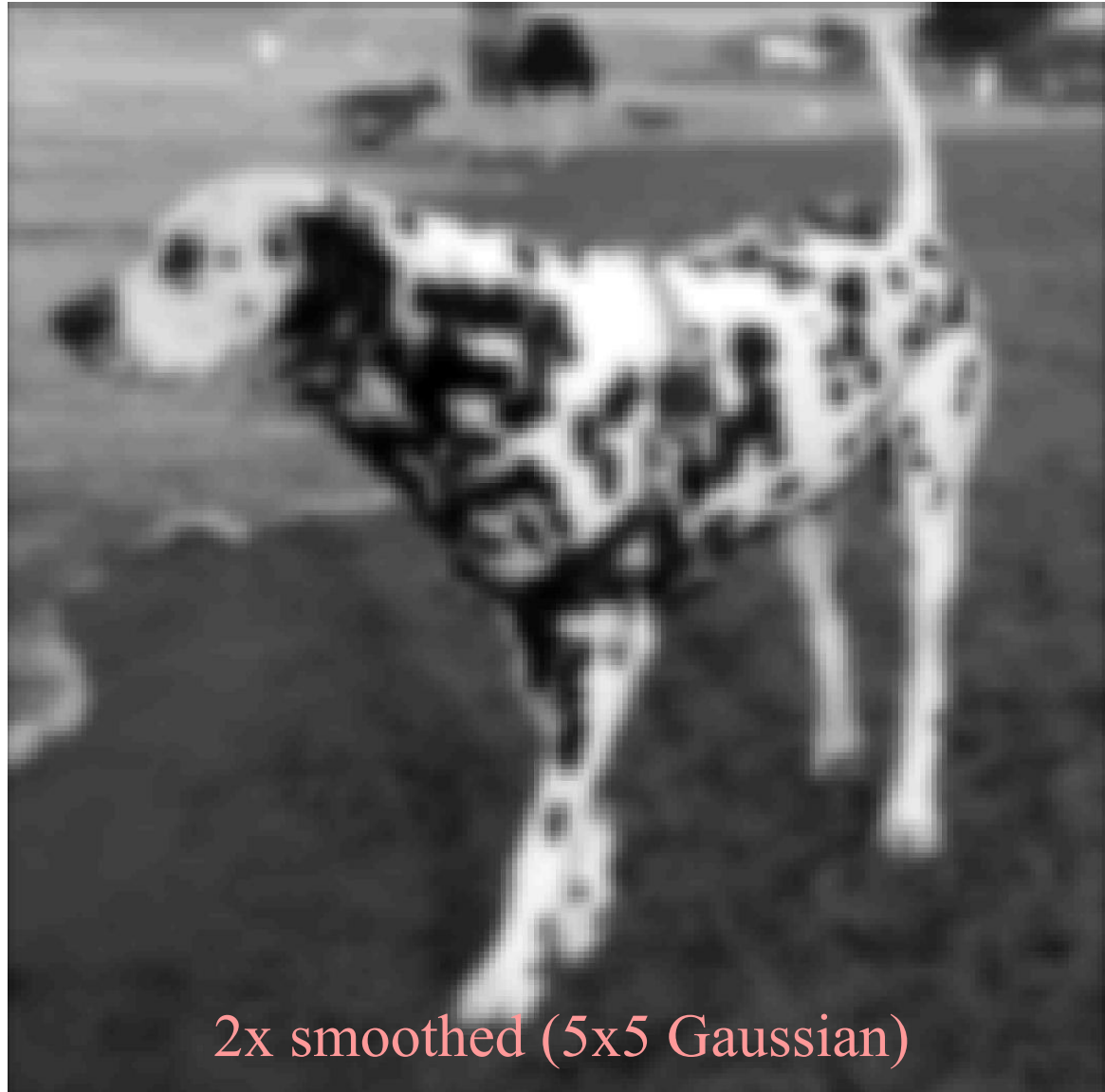


What does blurring take away?



1x smoothed (5x5 Gaussian)

What does blurring take away?

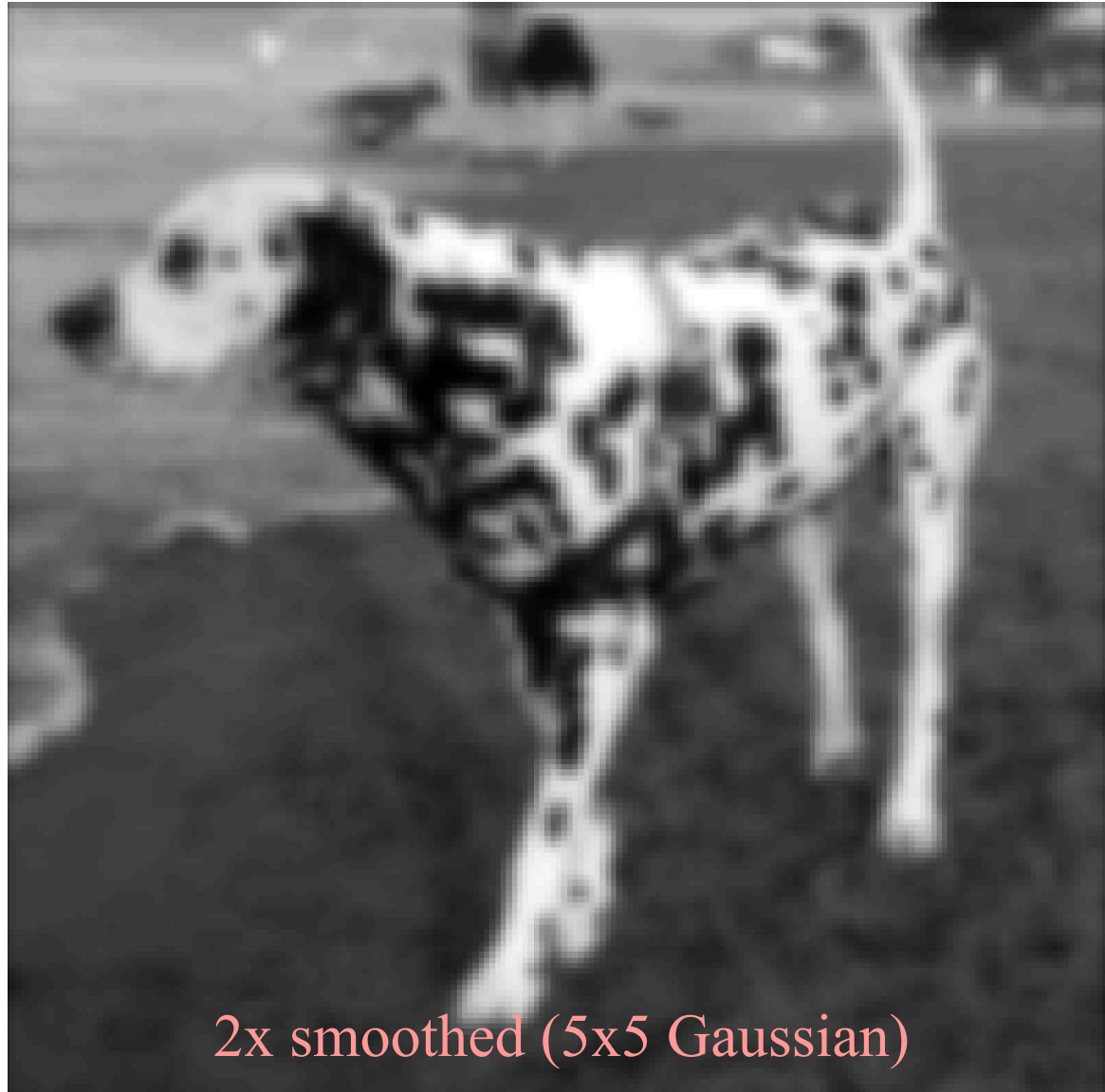


2x smoothed (5x5 Gaussian)

Difference of Gaussian



What does blurring take away?

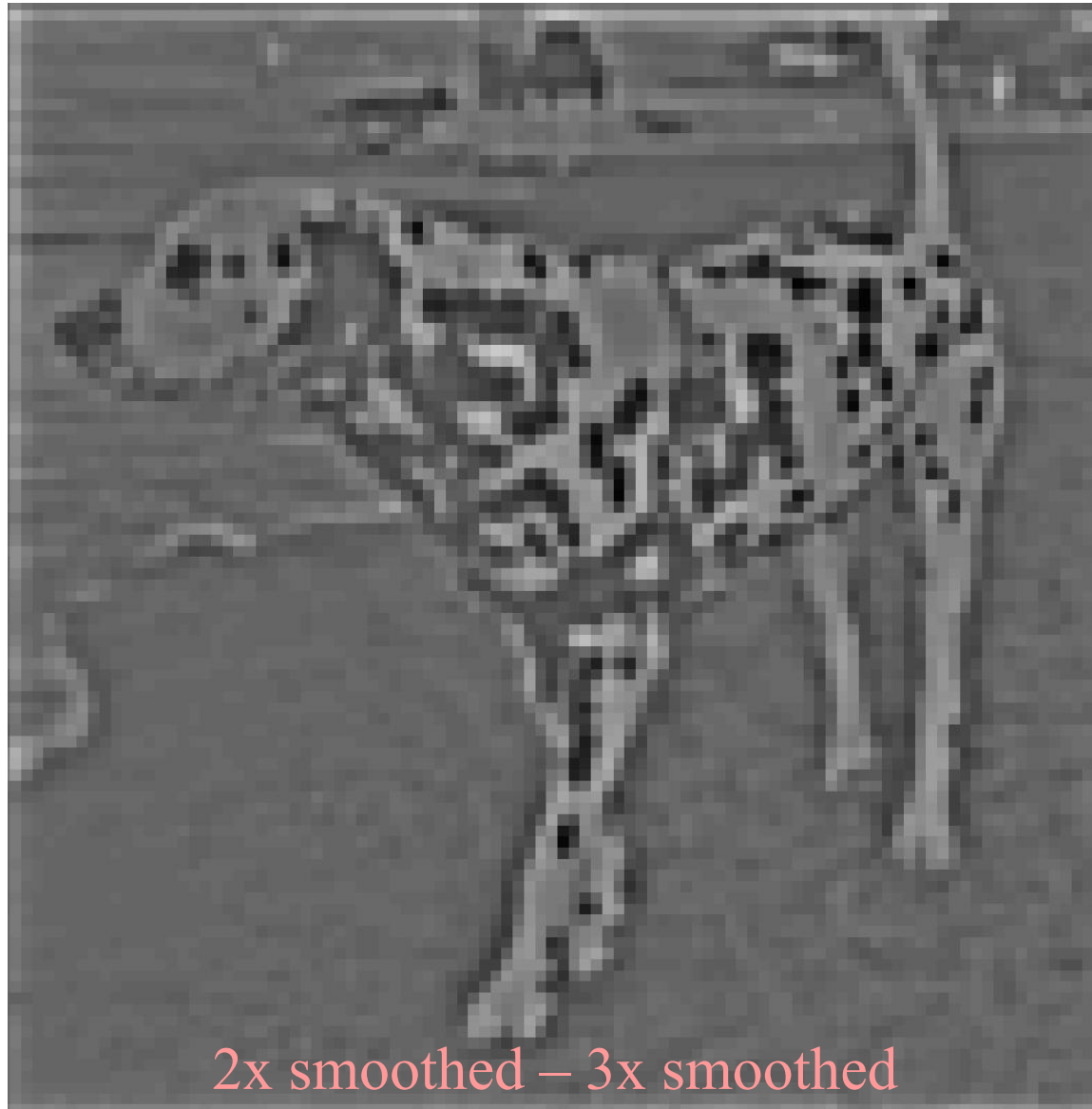


2x smoothed (5x5 Gaussian)



3x smoothed (5x5 Gaussian)

Difference of Gaussian



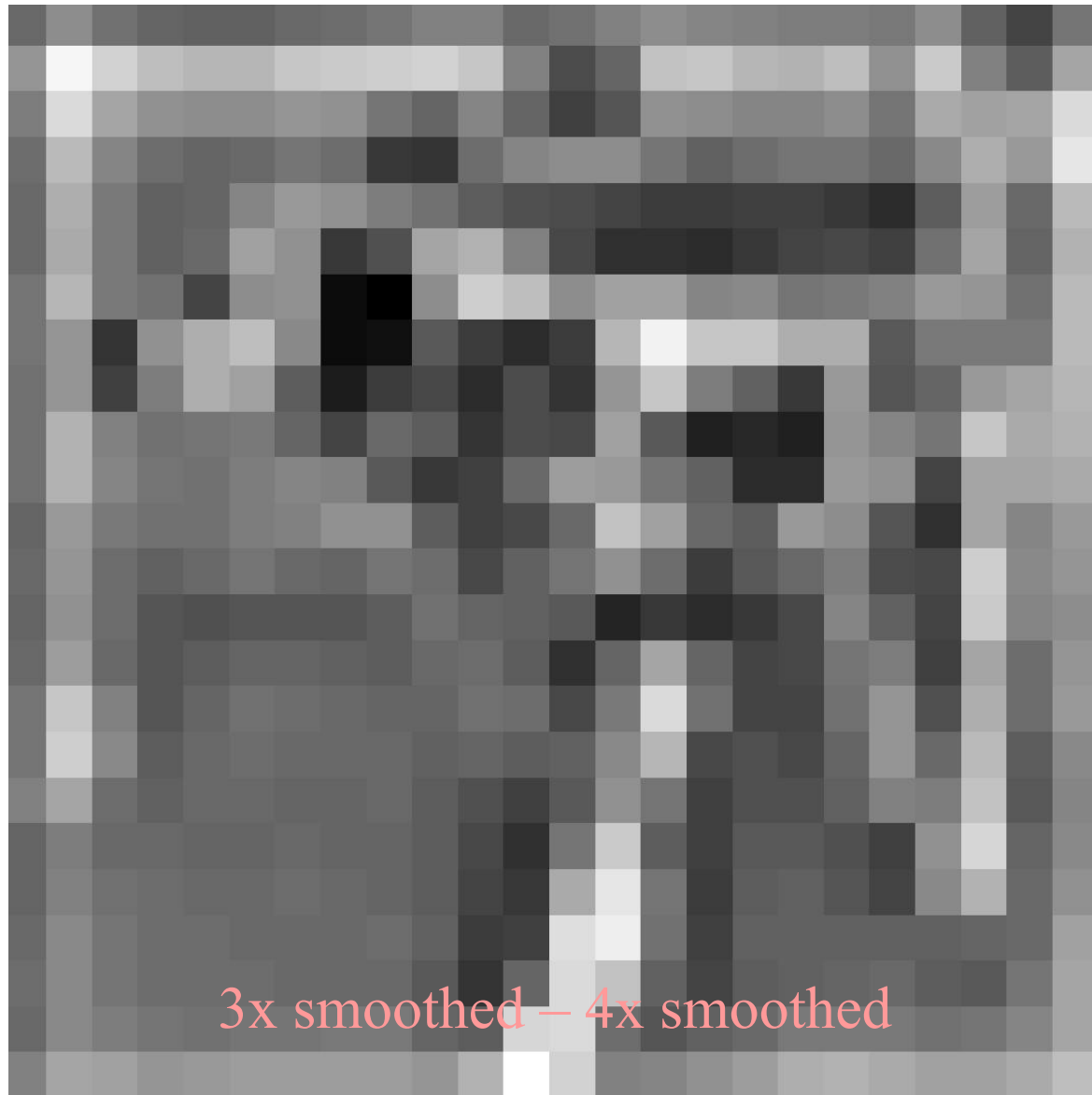
2x smoothed - 3x smoothed



3x smoothed (5x5 Gaussian)



4x smoothed (5x5 Gaussian)

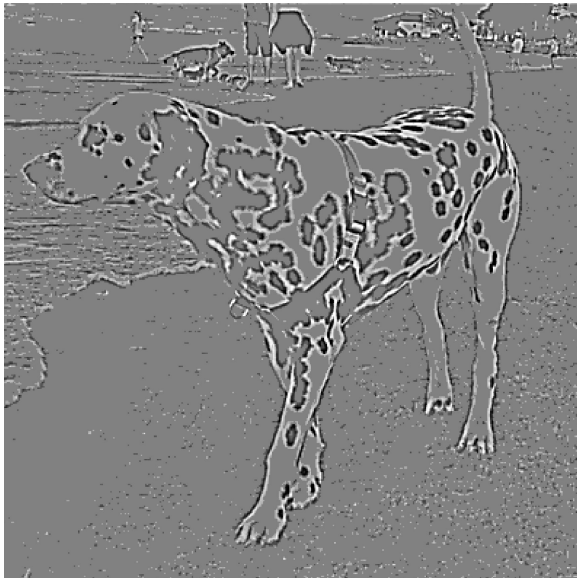


3x smoothed - 4x smoothed



+ ↓

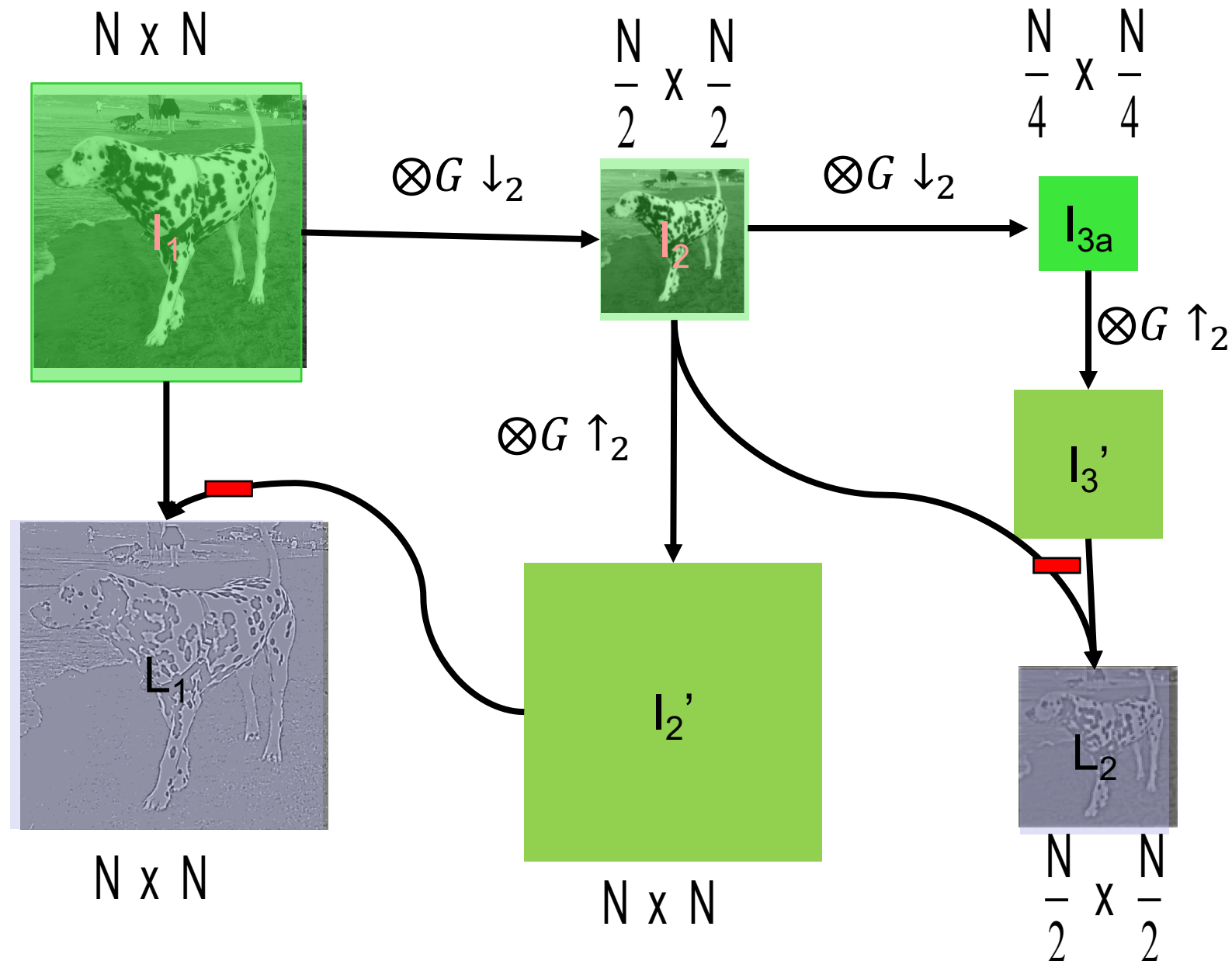
-

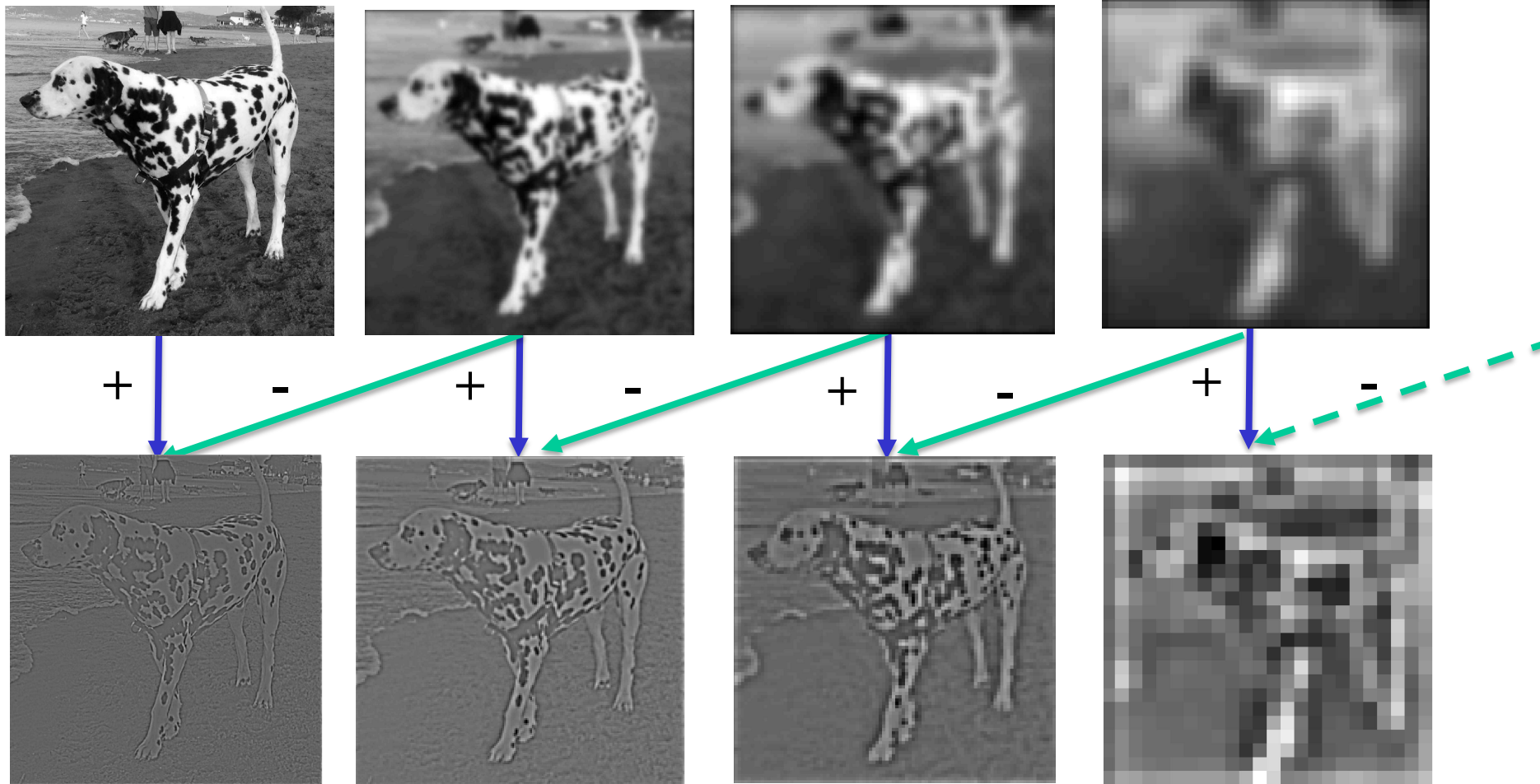


Laplacian Image

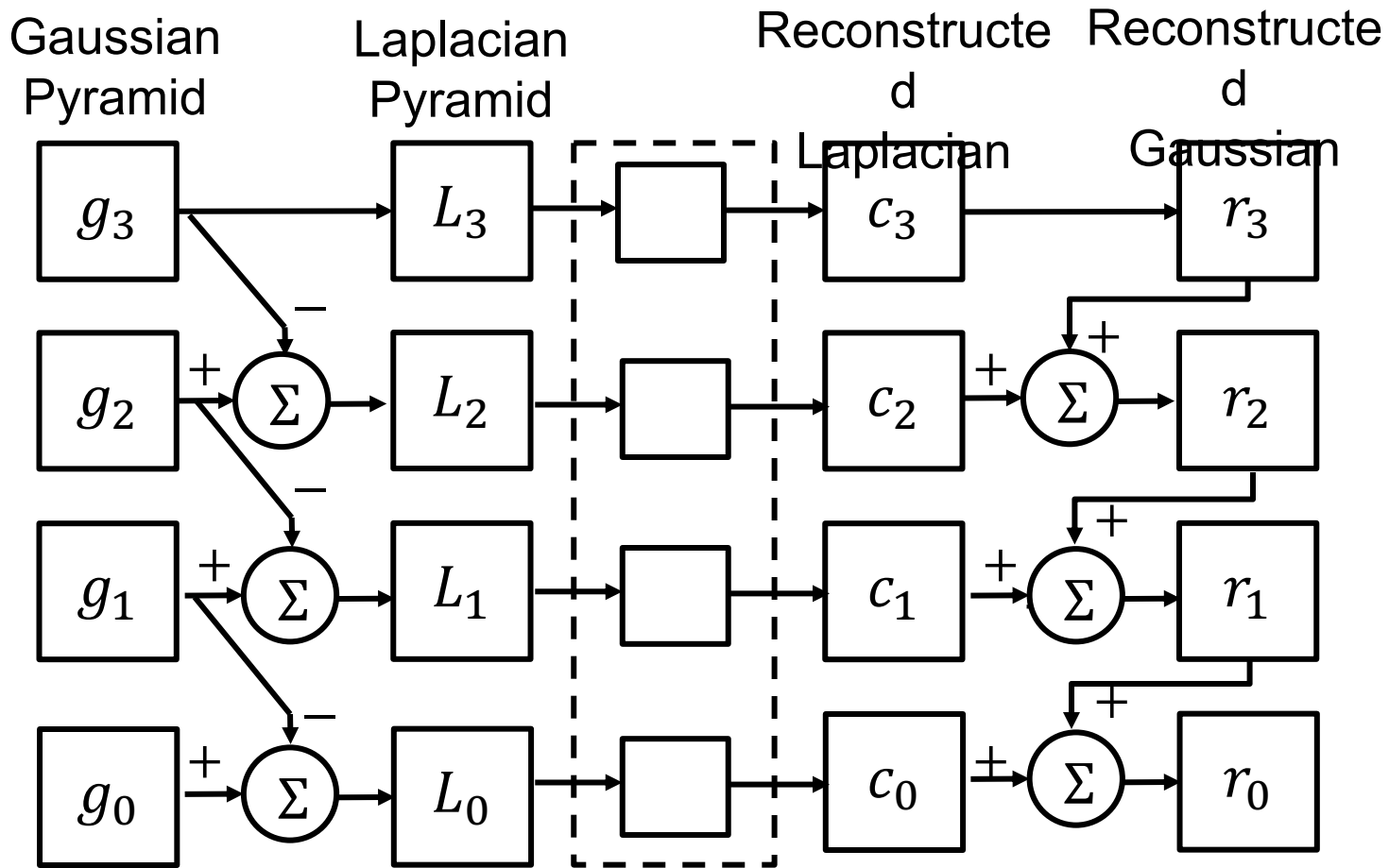
$$L_1 = g_l - \text{EXPAND}[g_{l+1}]$$

Extraction of Laplacian





Gaussian pyramid is smooth=> can be subsampled
Laplacian pyramid has narrow band of frequency=> compress



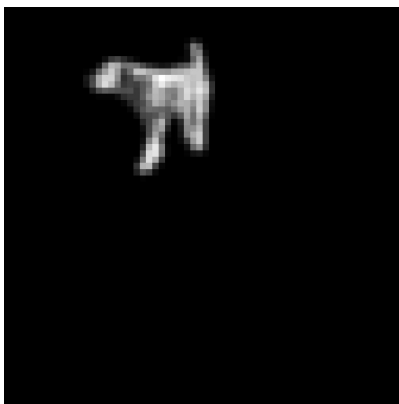
$$g_N = L_N$$

$$g_l = L_l + \text{EXPAND}[g_{l+1}]$$

Pyramid Blending



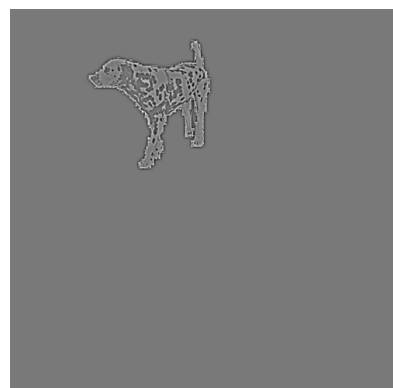
laplacian
level
4



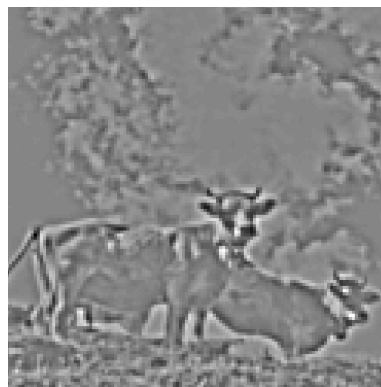
laplacian
level
2



laplacian
level
0

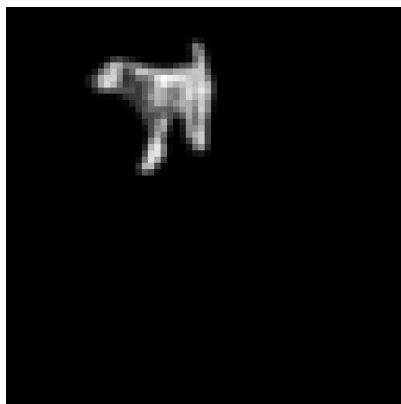


top pyramid



bottom pyramid

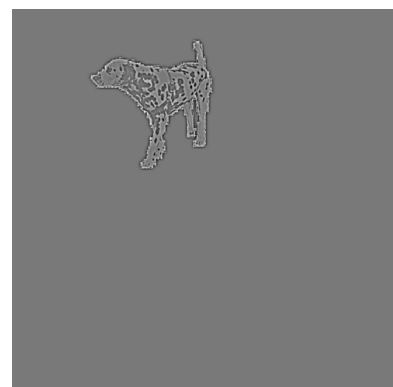
laplacian
level
4



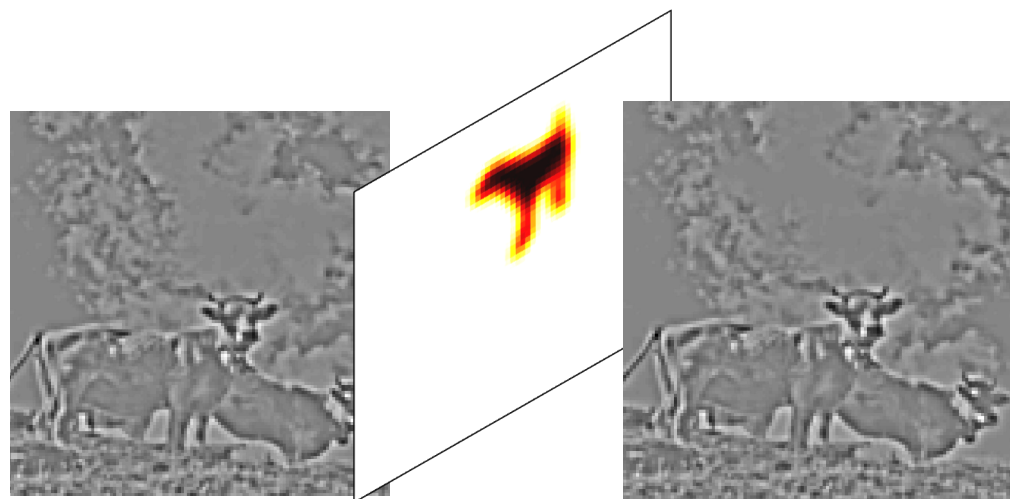
laplacian
level
2



laplacian
level
0

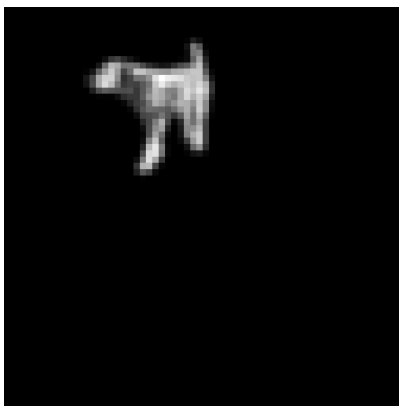


top pyramid



bottom pyramid

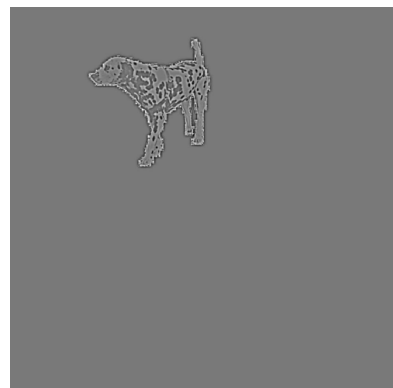
laplacian
level
4



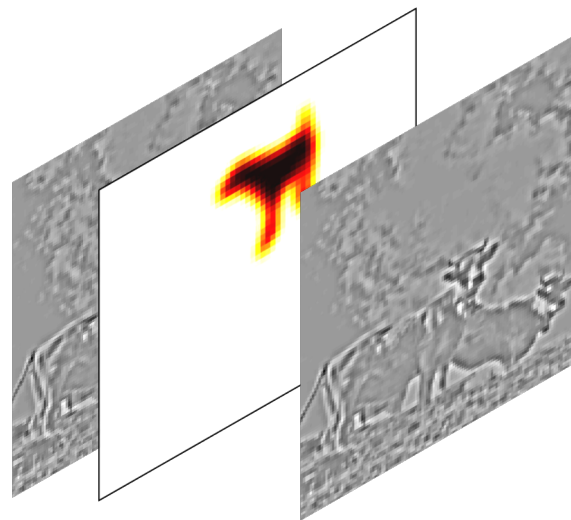
laplacian
level
2



laplacian
level
0

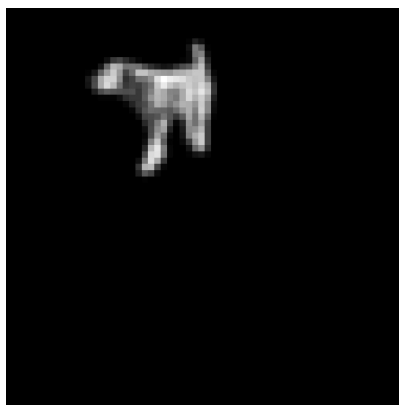


top pyramid



bottom pyramid

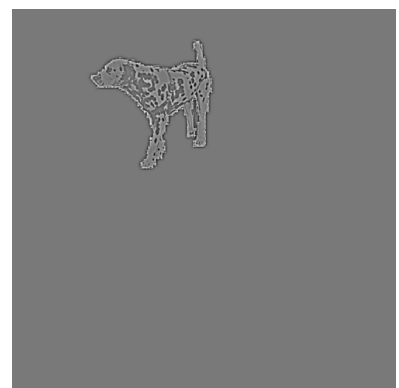
laplacian
level
4



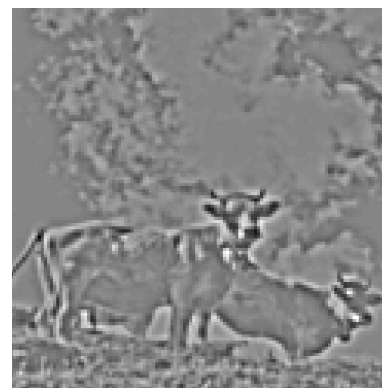
laplacian
level
2



laplacian
level
0



top pyramid

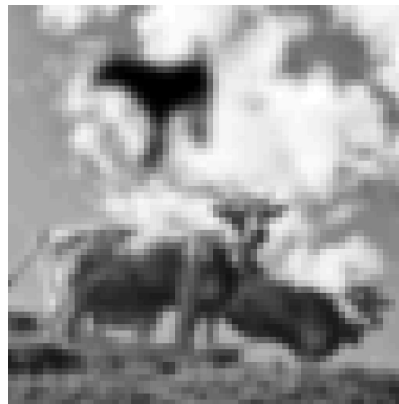
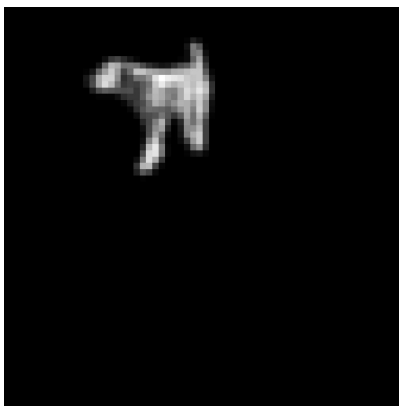


bottom pyramid

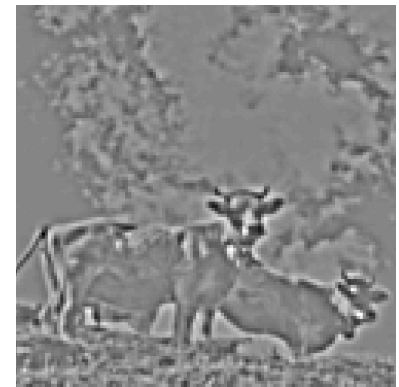


blended pyramid

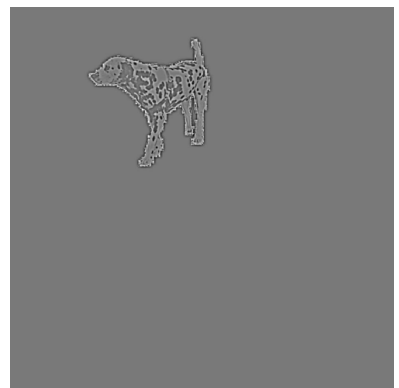
laplacian
level
4



laplacian
level
2



laplacian
level
0

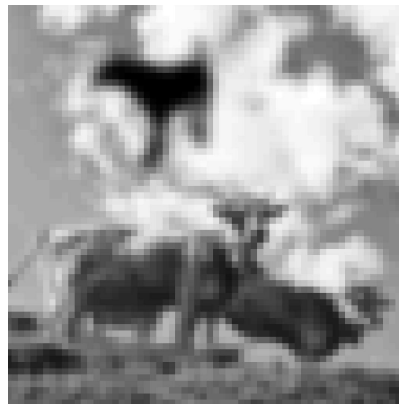
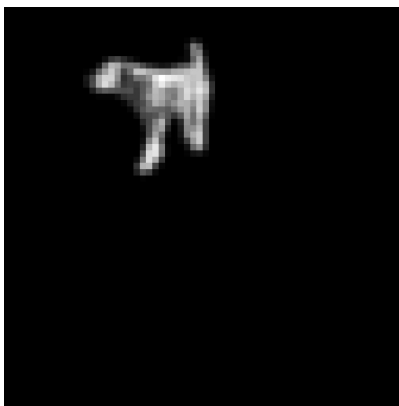


bottom pyramid

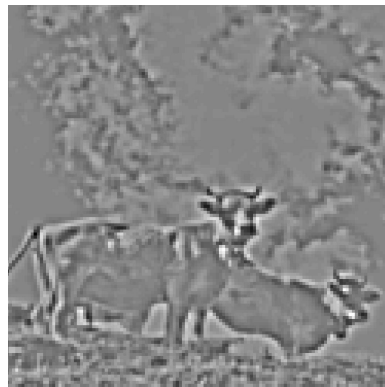
blended pyramid

top pyramid

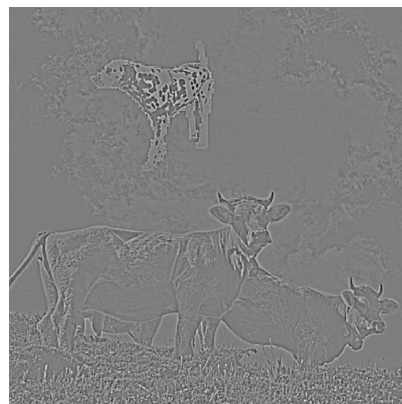
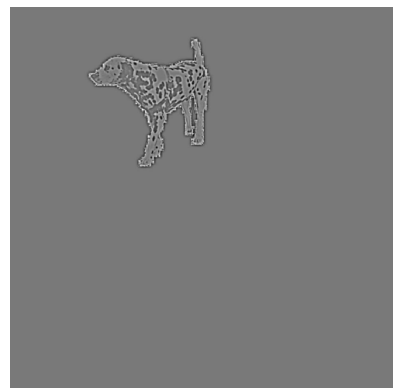
laplacian
level
4



laplacian
level
2



laplacian
level
0



top pyramid

bottom pyramid

blended pyramid

Laplacian Pyramid: Blending

General Approach:

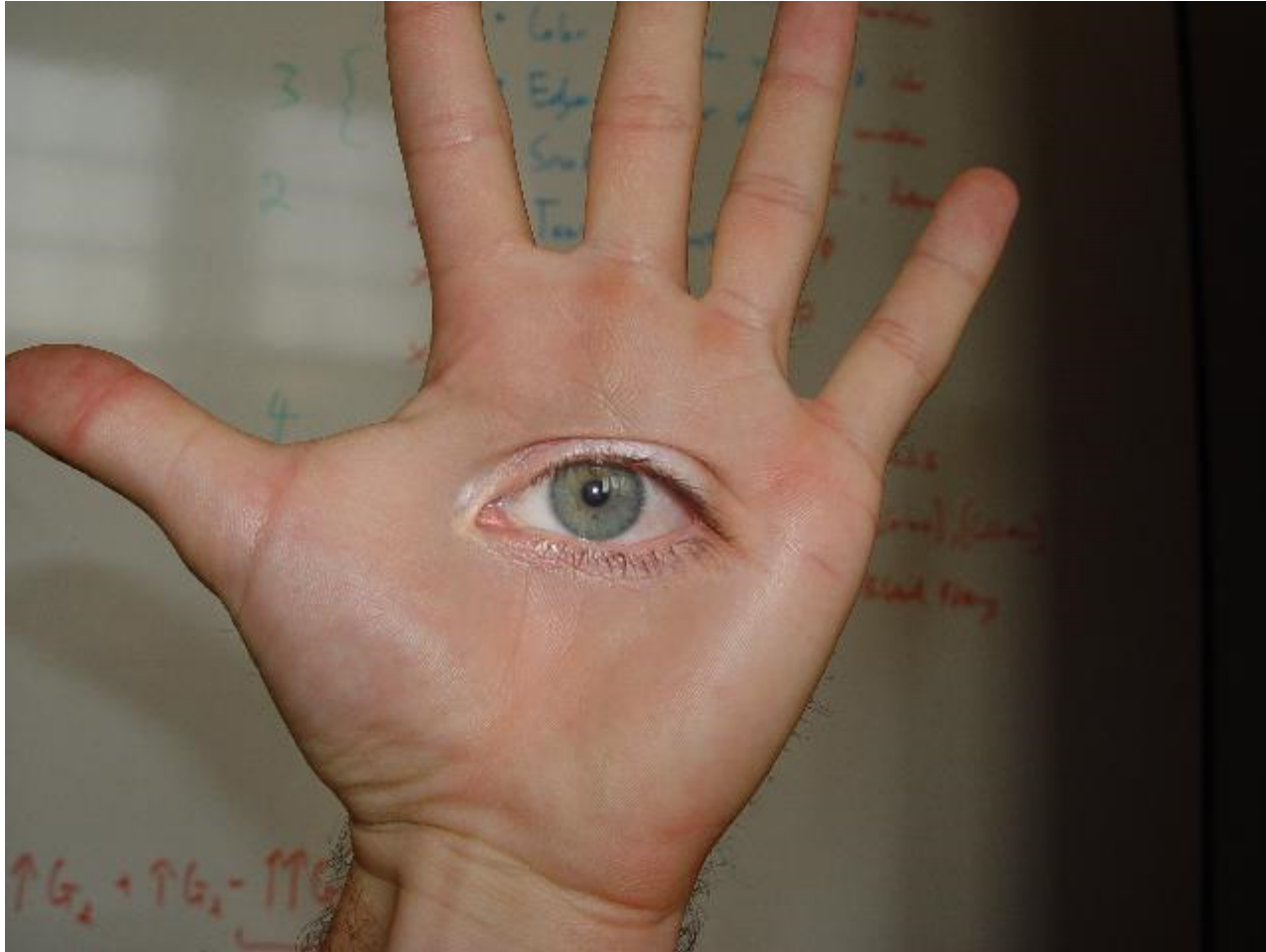
1. Build Laplacian pyramids LA and LB from images A and B
2. Build a Gaussian pyramid $MASK$ from selected region R
3. Form a combined pyramid LS from LA and LB using nodes of GR as weights:

$$LS(i, j) = MASK(i, j) * LA(i, j) + (1 - MASK(i, j)) * LB(i, j)$$

4. Collapse the LS pyramid to get the final blended image



Horror Photo



© prof. dmartin