Convolution Pyramids

Zeev Farbman, Raanan Fattal and Dani Lischinski
SIGGRAPH Asia Conference (2011)

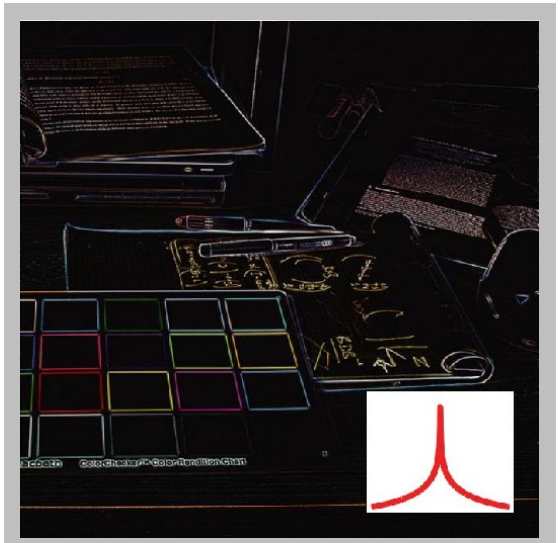


Fig. 1.1: Gradient integration example

presented by:
Julian Steil

supervisor:
Prof. Dr. Joachim Weickert

Seminar - Milestones and Advances in Image Analysis
Prof. Dr. Joachim Weickert, Oliver Demetz
Mathematical **I**mage **A**nalysis Group
Saarland University

13th of November, 2012

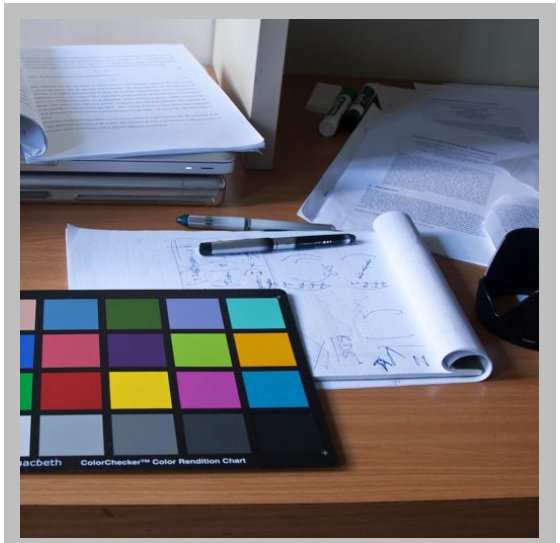


Fig. 1.2: Reconstruction result of Fig. 1.1

Overview

Motivation

Convolution Pyramids

Application 1 - Gaussian Kernels

Application 2 - Boundary Interpolation

Application 3 - Gradient Integration

Summary

Overview

Convolution

Two-Dimensional Convolution:

- discrete convolution of two images $g = (g_{i,j})_{i,j\in\mathbb{Z}}$ and $w = (w_{i,j})_{i,j\in\mathbb{Z}}$:

$$(g * w)_{i,j} := \sum_{k\in\mathbb{Z}} \sum_{\ell\in\mathbb{Z}} g_{i-k,j-\ell} w_{k,\ell} \tag{1}$$

- components of convolution kernel w can be regarded as mirrored weights for averaging the components of g
- the larger the kernel size the larger the runtime
- ordinary convolution implementation needs $O(n^2)$

Gaussian Pyramid

- sequence of images g_0, g_1, ..., g_n
- computed by a filtering procedure equivalent to convolution with a local, symmetric weighting function
  $\Longrightarrow$ e.g. a Gaussian kernel

Procedure:

- image initialised by array g_0 which contains C columns and R rows
- each pixel represents the light intensity I between 0 and 255
  $\Longrightarrow$ g_0 is the zero level of Gaussian Pyramid
- each pixel value in level i is computed as a weighting average of level $i-1$ pixel values

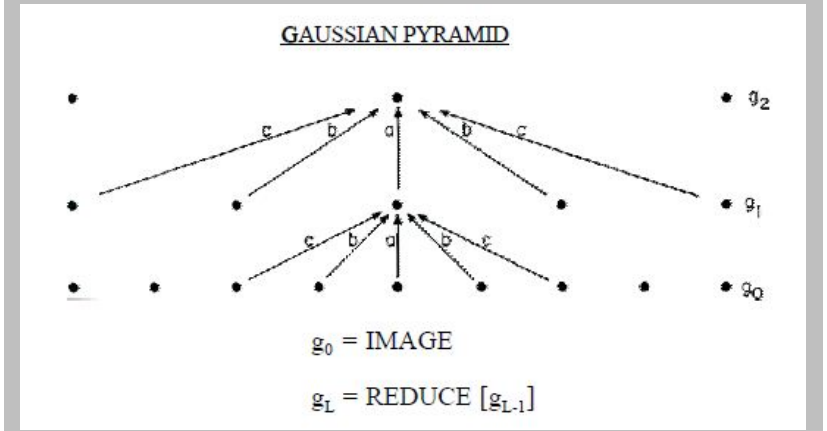


Fig. 2: One-dimensional graphic representation of the Gaussian pyramid

Gaussian Pyramid - Example

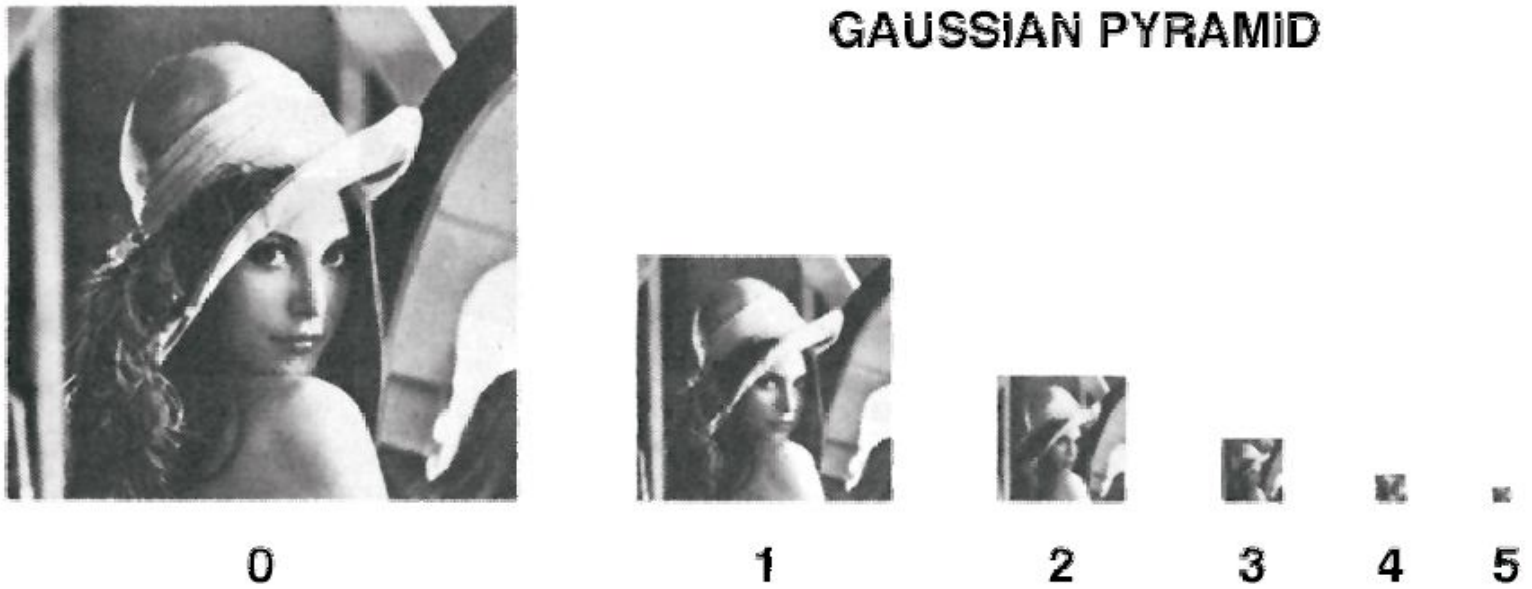


Fig. 3: First six levels of the Gaussian pyramid for the “Lena” image. The original image, level 0, measures 257x257 pixels $\Longrightarrow$ level 5 measures just 9x9 pixels

Remark:

density of pixels is reduced by half in one dimension and by fourth in two dimensions from level to level

From Gaussian to Laplacian Pyramid

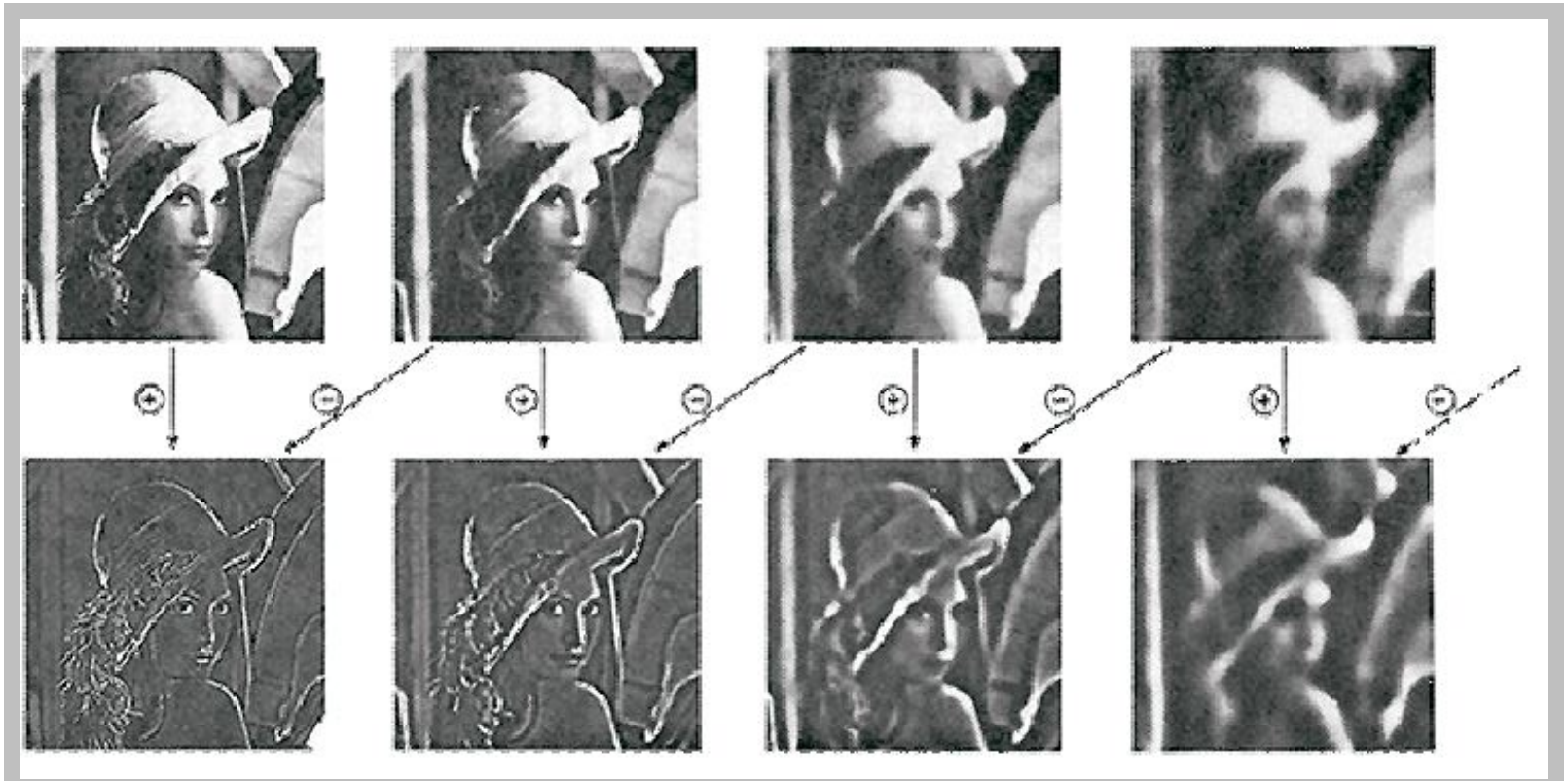


Fig. 4: First four levels of the Gaussian and Laplacian pyramid of Fig.3.

- each level of Laplacian pyramid is the difference between the corresponding and the next higher level of the Gaussian pyramid
- full expansion is used in Fig. 4 to help visualise the contents the pyramid images

Overview

1. Motivation

2. **Convolution Pyramids**
   - Approach
   - Forward and Backward Transform
   - Flow Chart and Pseudocode
   - Optimisation

3. Application 1 - Gaussian Kernels

4. Application 2 - Boundary Interpolation

5. Application 3 - Gradient Integration

6. Summary

Approach

Task:

- approximate effect of convolution with large kernels
  $\Longrightarrow$ higher spectral accuracy + translation-invariant operation
- Is it also possible in $O(n)$?

Idea:

- use of repeated convolution with small kernels on multiple scales
- disadvantage: not translation-invariant due to subsampling operation to reach $O(n)$ performance

Method:

- pyramids rely on a spectral “divide-and-conquer” strategy
- no subsampling of the decomposed signal increases the translation-invariance
- use finite impulse response filters to achieve some spacial localisation and runtime $O(n)$

Approach

Task:

- approximate effect of convolution with large kernels
  $\Longrightarrow$ higher spectral accuracy + translation-invariant operation
- Is it also possible in $O(n)$?

Idea:

- use of repeated convolution with small kernels on multiple scales
- disadvantage: not translation-invariant due to subsampling operation to reach $O(n)$ performance

Method:

- pyramids rely on a spectral “divide-and-conquer” strategy
- no subsampling of the decomposed signal increases the translation-invariance
- use finite impulse response filters to achieve some spacial localisation and runtime $O(n)$

Approach

Task:

- approximate effect of convolution with large kernels
  $\Longrightarrow$ higher spectral accuracy + translation-invariant operation
- Is it also possible in $O(n)$?

Idea:

- use of repeated convolution with small kernels on multiple scales
- disadvantage: not translation-invariant due to subsampling operation to reach $O(n)$ performance

Method:

- pyramids rely on a spectral “divide-and-conquer” strategy
- no subsampling of the decomposed signal increases the translation-invariance
- use finite impulse response filters to achieve some spacial localisation and runtime $O(n)$

Forward and Backward Transform

Forward Transform - *Analysis* Step:

- convolve a signal with a first filter h_1
- subsample the result by a factor of two
- process is repeated on the subsampled data
- an unfiltered and unsampled copy of the signal is kept at each level

$$a_0^l = a^l \tag{2}$$

$$a^{l+1} = \downarrow (h_1 * a^l) \tag{3}$$

Backward Transform - *Synthesis* Step:

- upsample by inserting a zero between every two samples
- convolve the result with a second filter h_2
- combine upsampled signal with the signal stored at each level after convolving with a third filter g

$$\hat{a}^l = h_2 * (\uparrow \hat{a}^{l+1}) + g * a_0^l \tag{4}$$

Forward and Backward Transform

Forward Transform - *Analysis* Step:

- convolve a signal with a first filter h_1
- subsample the result by a factor of two
- process is repeated on the subsampled data
- an unfiltered and unsampled copy of the signal is kept at each level

$$a_0^l = a^l \tag{2}$$

$$a^{l+1} = \downarrow (h_1 * a^l) \tag{3}$$

Backward Transform - *Synthesis* Step:

- upsample by inserting a zero between every two samples
- convolve the result with a second filter h_2
- combine upsampled signal with the signal stored at each level after convolving with a third filter g

$$\hat{a}^l = h_2 * (\uparrow \hat{a}^{l+1}) + g * a_0^l \tag{4}$$

Flow Chart and Pseudocode

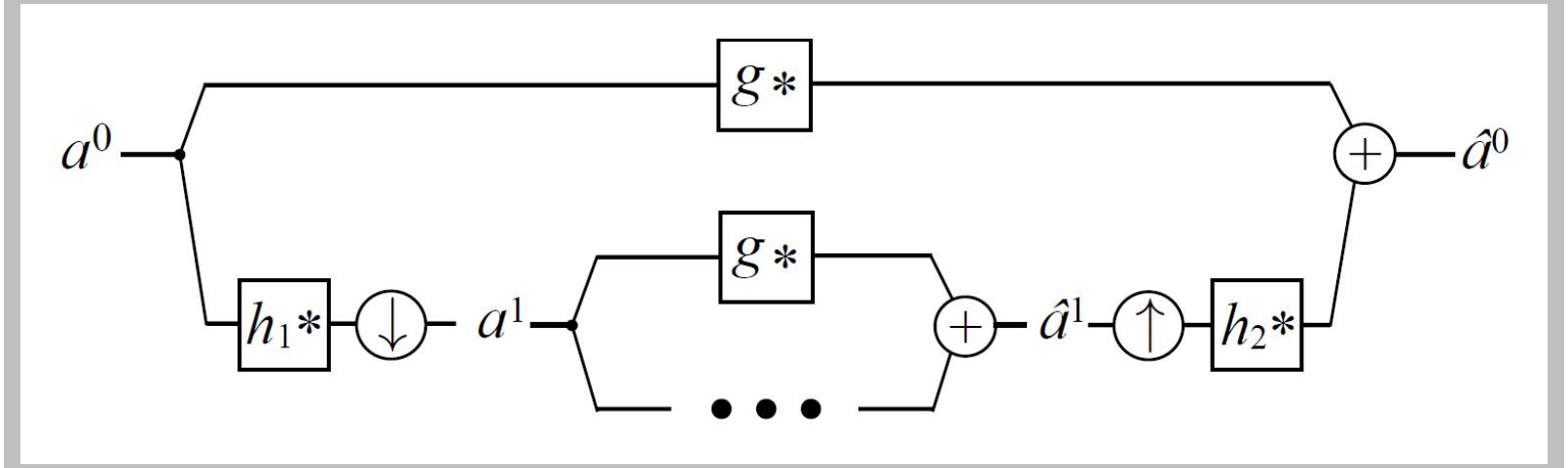


Fig. 5: Flow Chart to visualise pyramid structure, source taken from [1]

Algorithm 1 Multiscale Transform

Determine the number of levels L
{Forward transform (analysis)}
$a^0 = a$
for each level $l = 0 \ldots L-1$ **do**
    $a_0^l = a^l$
    $a^{l+1} = \downarrow (h_1 * a^l)$
end for
{Backward transform (synthesis)}
    $\hat{a}^L = g * a^L$
for each level $l = L-1 \ldots 0$ **do**
    $\hat{a}^l = h_2 * (\uparrow \hat{a}^{l+1}) + g * a_0^l$
end for

Optimisation

Kernel Determination:

- target kernel f is given
- seek a set of kernels $\mathscr{F} = \{h_1, h_2, g\}$ that minimise

$$\arg\min_{\mathscr{F}} \| \underbrace{\hat{a}^0_{\mathscr{F}}}_{\substack{\text{result of}\\ \text{multiscale}\\ \text{transform}}} - \underbrace{f}_{\substack{\text{target}\\ \text{kernel}}} * \underbrace{a}_{\substack{\text{input}\\ \text{signal}}} \| \tag{5}$$

- kernels in $\mathscr{F}$ should be small and separable
- use larger and/or non-separable filters increase accuracy
  $\Longrightarrow$ specific choice depends on application requirements
- remarkable results using separable kernels in $\mathscr{F}$ for non-separable target filters f
- target filters f with rotational and mirroring symmetries enforce symmetry on h_1, h_2, g

Overview

1. Motivation

2. Convolution Pyramids

3. **Application 1 - Gaussian Kernels**
   - Gaussian Kernel Convolution
   - Example - Gaussian Filter
   - Example - Scattered Data Interpolation

4. Application 2 - Boundary Interpolation

5. Application 3 - Gradient Integration

6. Summary

Gaussian Kernel Convolution

Task:

- approximate Gaussian kernels $e^{\frac{\|x\|^2}{2\sigma^2}}$ at the original fine grid in $O(n)$
- no truncated filter support

Determination of $\mathscr{F} = \{h_1, h_2, g\}$:

$$\arg\min_{\mathscr{F}} \| \underbrace{\hat{a}^0_{\mathscr{F}}}_{\text{result of multiscale transform}} - \underbrace{f}_{\text{target Gaussian kernel}} * \underbrace{a}_{\text{image to convolve}} \| \tag{5}$$

Problem:

- Gaussians are rather efficient low-pass filters
- pyramid contains high-frequent components coming from finer levels introduced by convolution with g

Gaussian Kernel Convolution

Task:

- approximate Gaussian kernels $e^{\frac{\|x\|^2}{2\sigma^2}}$ at the original fine grid in $O(n)$
- no truncated filter support

Determination of $\mathscr{F} = \{h_1, h_2, g\}$:

$$\arg\min_{\mathscr{F}} \| \underbrace{\hat{a}^0_{\mathscr{F}}}_{\substack{\text{result of} \\ \text{multiscale} \\ \text{transform}}} - \underbrace{f}_{\substack{\text{target} \\ \text{Gaussian} \\ \text{kernel}}} * \underbrace{a}_{\substack{\text{image} \\ \text{to} \\ \text{convolve}}} \| \tag{5}$$

Problem:

- Gaussians are rather efficient low-pass filters
- pyramid contains high-frequent components coming from finer levels introduced by convolution with g

Gaussian Kernel Convolution

Task:

- approximate Gaussian kernels $e^{\frac{\|x\|^2}{2\sigma^2}}$ at the original fine grid in $O(n)$
- no truncated filter support

Determination of $\mathscr{F} = \{h_1, h_2, g\}$:

$$\arg\min_{\mathscr{F}} \| \underbrace{\hat{a}^0_{\mathscr{F}}}_{\substack{\text{result of} \\ \text{multiscale} \\ \text{transform}}} - \underbrace{f}_{\substack{\text{target} \\ \text{Gaussian} \\ \text{kernel}}} * \underbrace{a}_{\substack{\text{image} \\ \text{to} \\ \text{convolve}}} \| \tag{5}$$

Problem:

- Gaussians are rather efficient low-pass filters
- pyramid contains high-frequent components coming from finer levels introduced by convolution with g

Example - Gaussian Filter

Solution:

- modulation of g at each level l
- higher w^l at the levels closest to the target size
- for different σ different sets of kernels $\mathscr{F}$ are necessary

▸ used kernels

Fig. 6.1: Original image, source: taken from [1]

Fig. 6.2: Exact convolution with a Gaussian filter ($\sigma = 4$), source: taken from [1]

Fig. 6.3: Convolution using optimization approach for $\sigma = 4$, source: taken from [1]

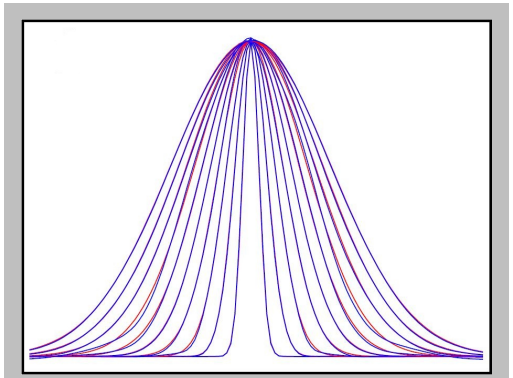


Fig. 7.1: Exact kernels (in red) with approximated kernels (in blue), source: taken from [1]

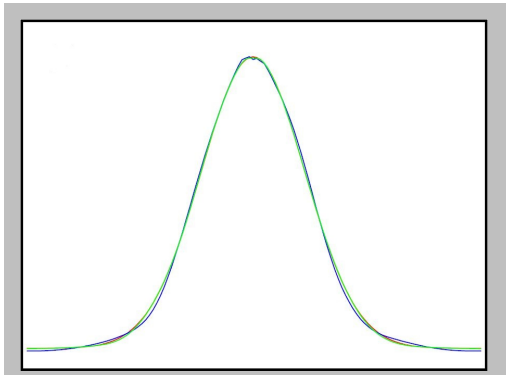


Fig. 7.2: Exact Gaussian (red), approximation using $5x5$ kernels (blue) and $7x7$ kernel (green) , source: taken from [1]

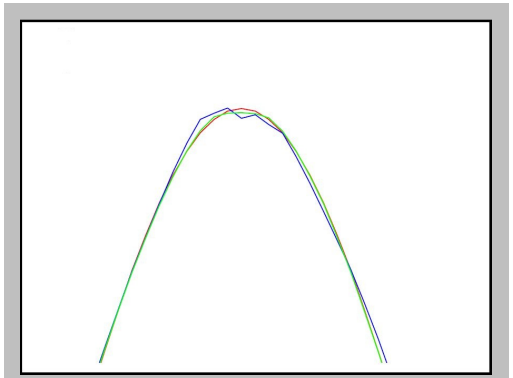


Fig. 7.3: Magnification of Fig. 7.2 shows better accuracy of larger kernels, source: taken from [1]

Example - Scattered Data Interpolation

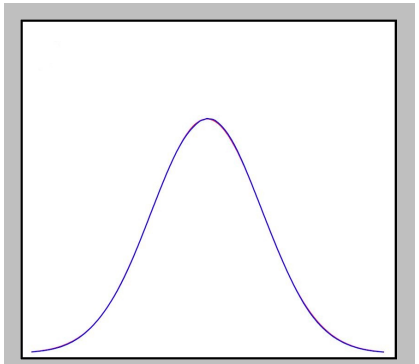


Fig. 8.1: Horizontal slice through exact wider Gaussian (red) and approximation (blue), source: taken from [1]

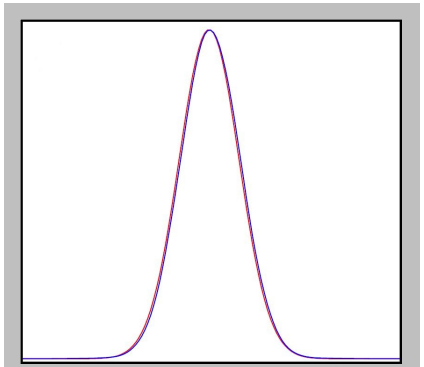


Fig. 8.2: Horizontal slice through exact narrower Gaussian (red) and approximation (blue), source: taken from [1]

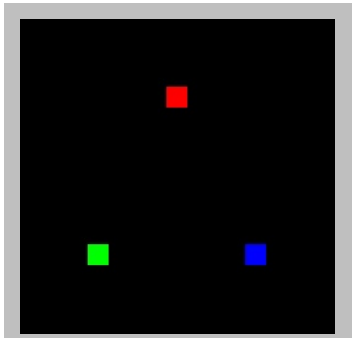


Fig. 8.3: Scattered data interpolation input , source: taken from [1]

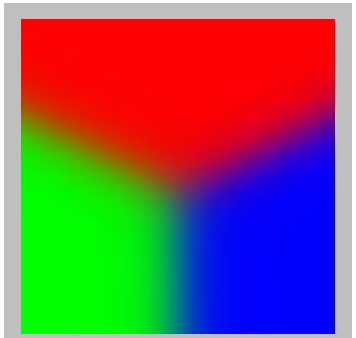


Fig. 8.4: Approximation with wider Gaussian, source: taken from [1]

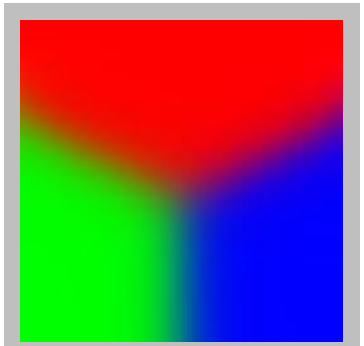


Fig. 8.5: Approximation with narrower Gaussian, source: taken from [1]

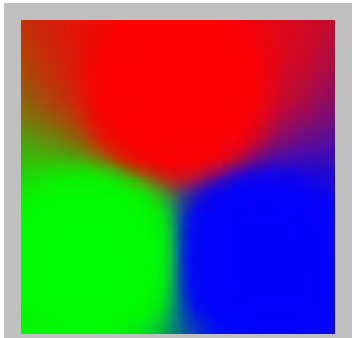


Fig. 8.6: Exact results corresponding to red wider Gaussian , source: taken from [1]

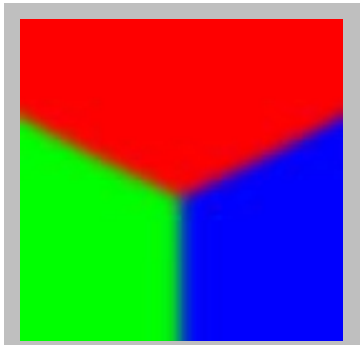


Fig. 8.7: Exact results corresponding to red narrower Gaussian, source: taken from [1]

Overview

1. Motivation
2. Convolution Pyramids
3. Application 1 - Gaussian Kernels
4. **Application 2 - Boundary Interpolation**
   - How to use boundary interpolation?
   - Example - Seamless Cloning
5. Application 3 - Gradient Integration
6. Summary

How to use boundary interpolation?

Seamless Image Cloning:

- formulation as boundary value problem
- effectively solved by constructing a smooth membrane
- interpolation of differences along a seam between two images

Shepard's Method:

- Ω is region of interest and boundary values are given by $b(x)$
- smoothly interpolates boundary values to all grid points inside Ω
- defines interpolant r at x as weighted average of boundary values:

$$r(x) = \frac{\sum_k w_k(x) b(x_k)}{\sum_k w_k(x)} \Longrightarrow r(x_i) = \frac{\sum_{j=0}^{n} w(x_i, x_j)\hat{r}(x_j)}{\sum_{j=0}^{n} w(x_i, x_j)\chi_{\hat{r}}(x_j)} = \frac{w * \hat{r}}{w * \chi_{\hat{r}}} \quad (6)$$

- x_k = boundary points, $b(x_k)$ = boundary values
- weight function $w_k(x)$ is given by

$$w_k(x) = w(x_k, x) = \frac{1}{d(x_k, x)^3} \quad (7)$$

- strong spike at x_k and decays rapidly away from it
- computational cost $O(Kn)$, K boundary values and n points in Ω

How to use boundary interpolation?

Seamless Image Cloning:

- formulation as boundary value problem
- effectively solved by constructing a smooth membrane
- interpolation of differences along a seam between two images

Shepard's Method:

- Ω is region of interest and boundary values are given by $b(x)$
- smoothly interpolates boundary values to all grid points inside Ω
- defines interpolant r at x as weighted average of boundary values:

$$r(x) = \frac{\sum_k w_k(x) b(x_k)}{\sum_k w_k(x)} \Longrightarrow r(x_i) = \frac{\sum_{j=0}^{n} w(x_i, x_j)\hat{r}(x_j)}{\sum_{j=0}^{n} w(x_i, x_j)\chi_{\hat{r}}(x_j)} = \frac{w * \hat{r}}{w * \chi_{\hat{r}}} \quad (6)$$

- x_k = boundary points, $b(x_k)$ = boundary values
- weight function $w_k(x)$ is given by

$$w_k(x) = w(x_k, x) = \frac{1}{d(x_k, x)^3} \quad (7)$$

- strong spike at x_k and decays rapidly away from it
- computational cost $O(Kn)$, K boundary values and n points in Ω

Example - Seamless Cloning

Determination of $\mathscr{F} = \{h_1, h_2, g\}$:

$$\arg\min_{\mathscr{F}} \| \underbrace{\hat{a}^0_{\mathscr{F}}}_{\substack{\text{result of} \\ \text{multiscale} \\ \text{transform}}} - \underbrace{f * a}_{\substack{\text{exact} \\ \text{membrane} \\ r(x)}} \| \tag{5}$$

Fig. 9.1: Source image, source: taken from [2]

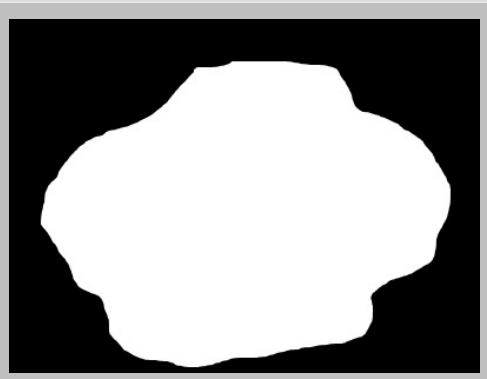


Fig. 9.2: Membrane mask, source: taken from [2]

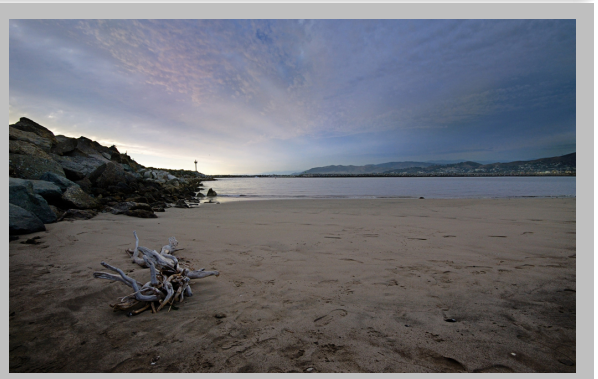


Fig. 9.3: Target image, source: taken from [2]

▸ Used Kernels

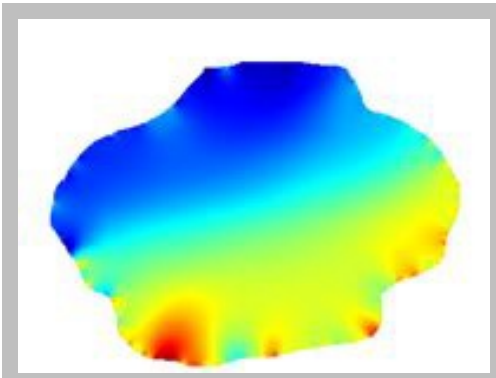


Fig. 9.4: Approximated membrane source: taken from [1]

Fig. 9.5: Superimposed image with a cloned patch, source: taken from [1]

Fig. 9.6: Result of applying Fig. 9.4 to Fig. 9.5, source: taken from [1]

Overview

1. Motivation

2. Convolution Pyramids

3. Application 1 - Gaussian Kernels

4. Application 2 - Boundary Interpolation

5. **Application 3 - Gradient Integration**
   - Kernel Detection
   - Example - Gradient Integration
   - How does the target filter look like?
   - Reconstruction of Target Filter

6. Summary

Kernel Detection

Determination of $\mathscr{F} = \{h_1, h_2, g\}$:

- choose a natural image I
- a is the divergence of its gradient field:

$$a = \text{div}\, \nabla I \tag{8}$$

$$I = f * a \tag{9}$$

$$\arg\min_{\mathscr{F}} \| \underbrace{\hat{a}^0_{\mathscr{F}}}_{\substack{\text{result of} \\ \text{multiscale} \\ \text{transform}}} - \underbrace{f * a}_{\substack{\text{natural} \\ \text{image} \\ I}} \| \tag{5}$$

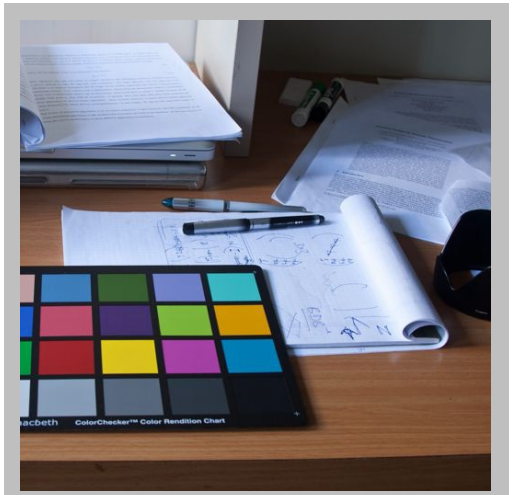


Fig. 10.1: Natural image I, source: taken from [1]

Fig. 10.2: Corresponding gradient image a of Fig. 10.1, source: taken from [1]

Example - Gradient Integration

Fig. 11.1: Gradient image of Fig 11.4, source: taken from [1]

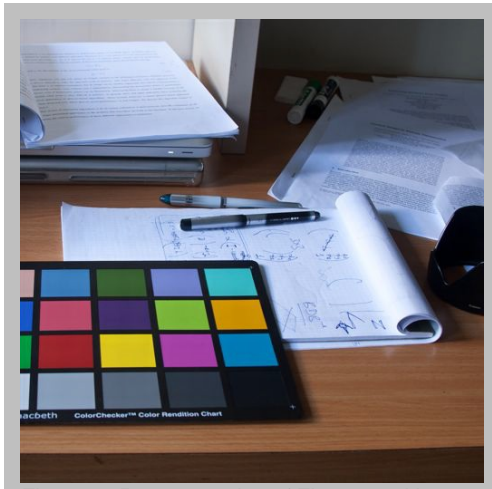


Fig. 11.2: Reconstruction of Fig. 11.1 with $\mathcal{F}_{5,3}$, source: taken from [1]

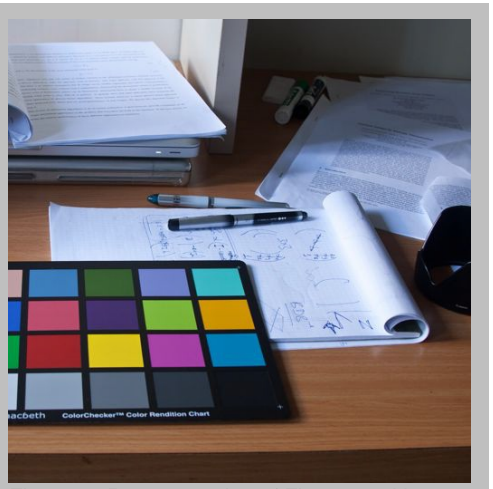


Fig. 11.3: Reconstruction of Fig. 11.1 with $\mathcal{F}_{7,5}$, source: taken from [1]

▸ Used Kernels

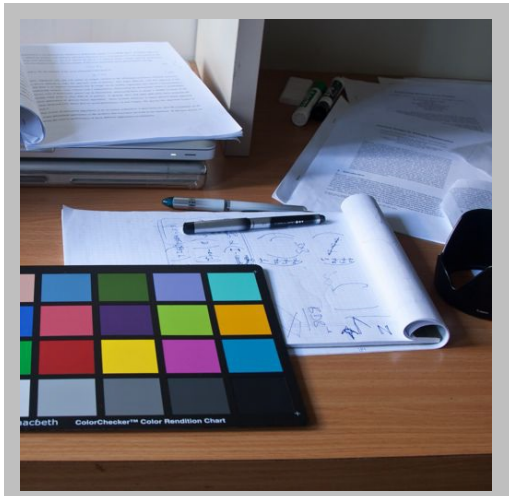


Fig. 11.4: Original image (512x512), source: taken from [1]

Fig. 11.5: Absolute errors of Fig. 11.2 (magnified by x50), source: taken from [1]

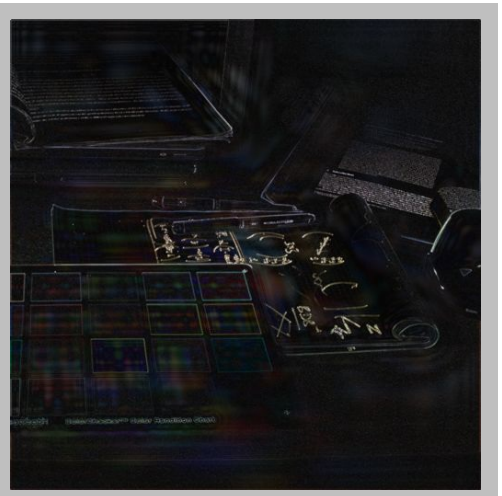


Fig. 11.6: Absolute errors of Fig. 11.3 (magnified by x50), source: taken from [1]

How does the target filter look like?

Task:

- recover image u (here: $u = \hat{a}^0_{\mathscr{F}}$) by solving the Poisson equation

$$\triangle u = \text{div } v \tag{10}$$

- v = gradient field

Solution:

- Green's functions

$$G(x, x') = G(\|x - x'\|) = \frac{1}{2\pi} \log \frac{1}{\|x - x'\|} \tag{11}$$

define fundamental solutions to the Poisson equation

$$\triangle G(x, x') = \delta(x, x') \tag{12}$$

- δ = discrete delta function
- (10) is defined over an infinite domain with no boundary constraints
  $\Longrightarrow$ Laplace operator becomes spatially invariant
  $\Longrightarrow$ Green's function becomes translation invariant
- solution of (10) is given by the convolution

$$u = G * \text{div } v \tag{13}$$

How does the target filter look like?

Task:

- recover image u (here: $u = \hat{a}^0_{\mathscr{F}}$) by solving the Poisson equation

$$\triangle u = \text{div } v \tag{10}$$

- $v =$ gradient field

Solution:

- Green's functions

$$G(x, x') = G(\|x - x'\|) = \frac{1}{2\pi} \log \frac{1}{\|x - x'\|} \tag{11}$$

define fundamental solutions to the Poisson equation

$$\triangle G(x, x') = \delta(x, x') \tag{12}$$

- $\delta =$ discrete delta function
- (10) is defined over an infinite domain with no boundary constraints
  $\Longrightarrow$ Laplace operator becomes spatially invariant
  $\Longrightarrow$ Green's function becomes translation invariant
- solution of (10) is given by the convolution

$$u = G * \text{div } v \tag{13}$$

Reconstruction of Target Filter

Target Filter Determination:

- using results of previous $\mathscr{F} = \{h_1, h_2, g\}$
- a is a centered delta function

$$a = \text{div}\, \nabla I \tag{8}$$

$$I = f * a \tag{9}$$

- Green's function provides a suitable result for f

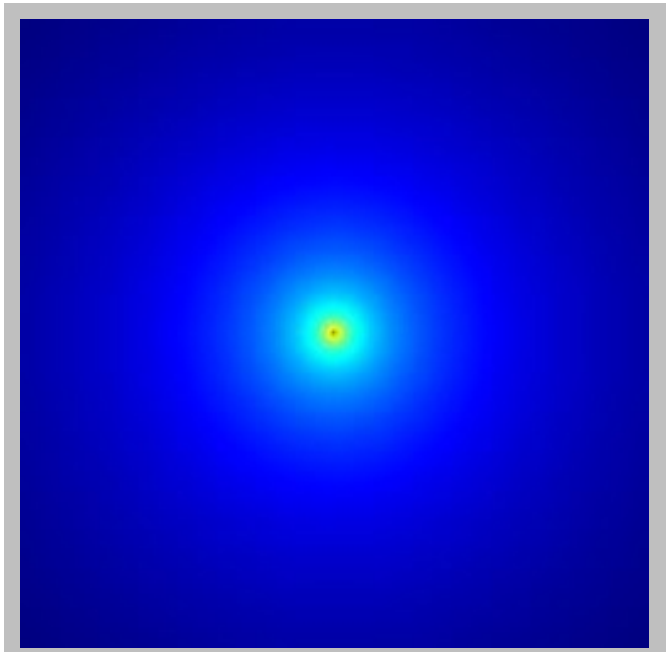


Fig. 12.1: Reconstruction of the Green's function, source: taken from [1]

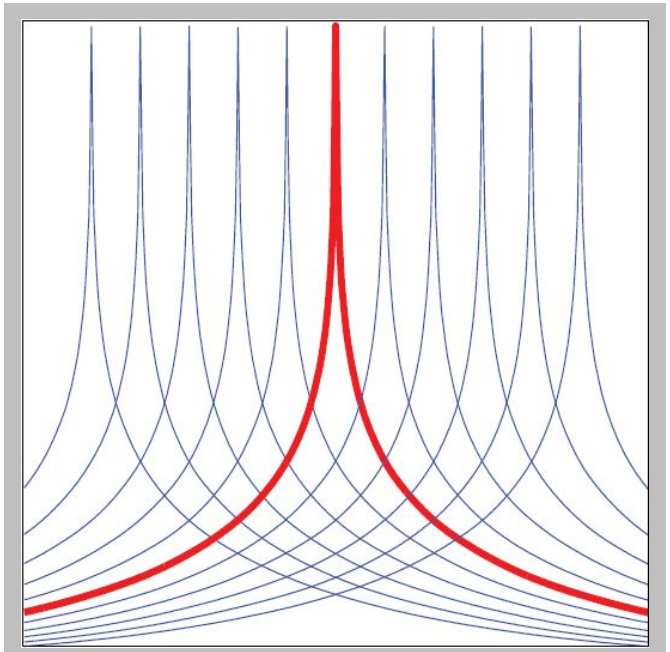


Fig. 12.2: space invariant corresponding kernel of Fig. 12.1, source: taken from [1]

Overview

1. Motivation
2. Convolution Pyramids
3. Application 1 - Gaussian Kernels
4. Application 2 - Boundary Interpolation
5. Application 3 - Gradient Integration
6. **Summary**
   - Summary

Summary

- approximation of large convolution filters in $O(n)$
  $\Longrightarrow$ using kernels of small support $\mathscr{F} = \{h_1, h_2, g\}$
  + multiscale pyramid scheme
- kernel determination by optimization:

$$\arg\min_{\mathscr{F}} \| \underbrace{\hat{a}^0_{\mathscr{F}}}_{\substack{\text{result of}\\ \text{multiscale}\\ \text{transform}}} - \underbrace{f}_{\substack{\text{target}\\ \text{kernel}}} * \underbrace{a}_{\substack{\text{input}\\ \text{signal}}} \|$$

- suitable for different applications like...
  - gradient integration
  - seamless cloning
  - scattered data interpolation

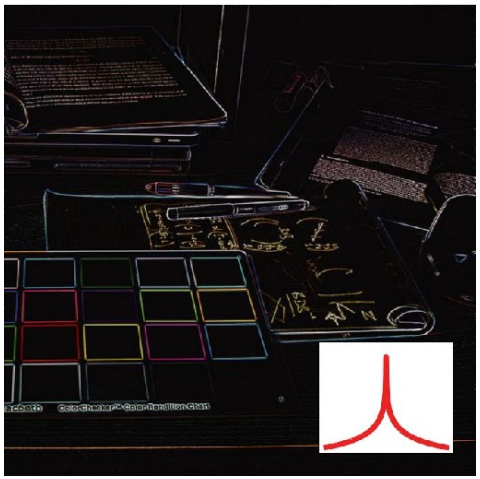
References

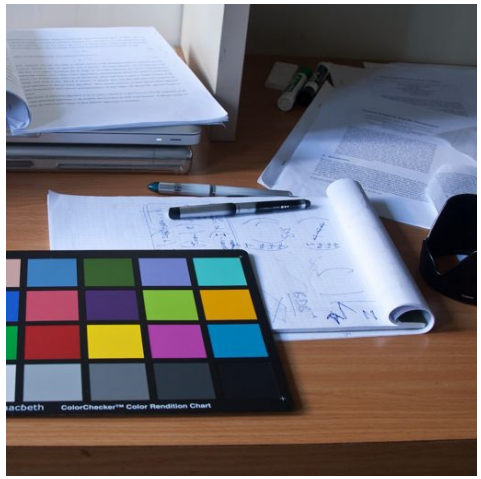
[1] Zeev Farbman, Raanan Fattal, Dani Lischinski
Convolution pyramids
Proc. 2011 SIGGRAPH Asia Conference, Article No. 175
The Hebrew University (2011)

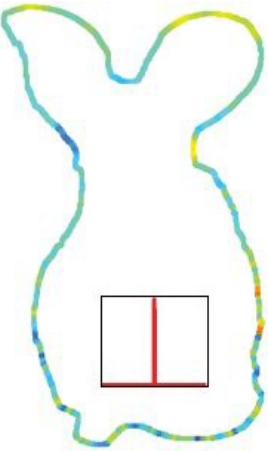
[2] Computer Graphics & Computational Photography Lab
Supplementary Materials of the paper "Convolution pyramids"
The Hebrew University (2011)
http://www.cs.huji.ac.il/labs/cglab/projects/convpyr/

[3] Mathematical Image Analysis Group
Lecture notes of the "Image Processing and Computer Vision" lecture
Saarland University. Winter term (2011)
http://www.mia.uni-saarland.de/Teaching/ipcv06.shtml

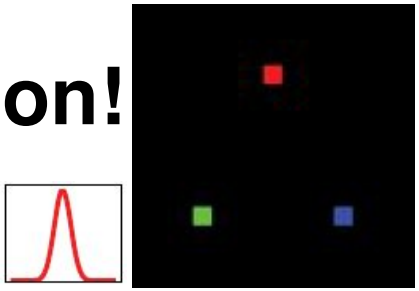
[4] Peter J. Burt, Edward H. Adelson
The Laplacian Pyramid as a Compact Image Code
IEEE Transcriptions on Communications
Vol. COM-31, No. 4, (April 1983)

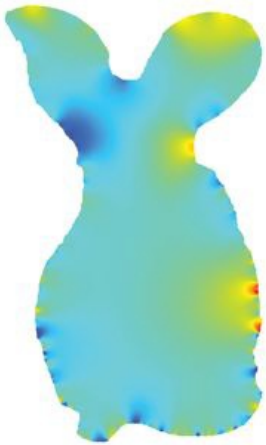


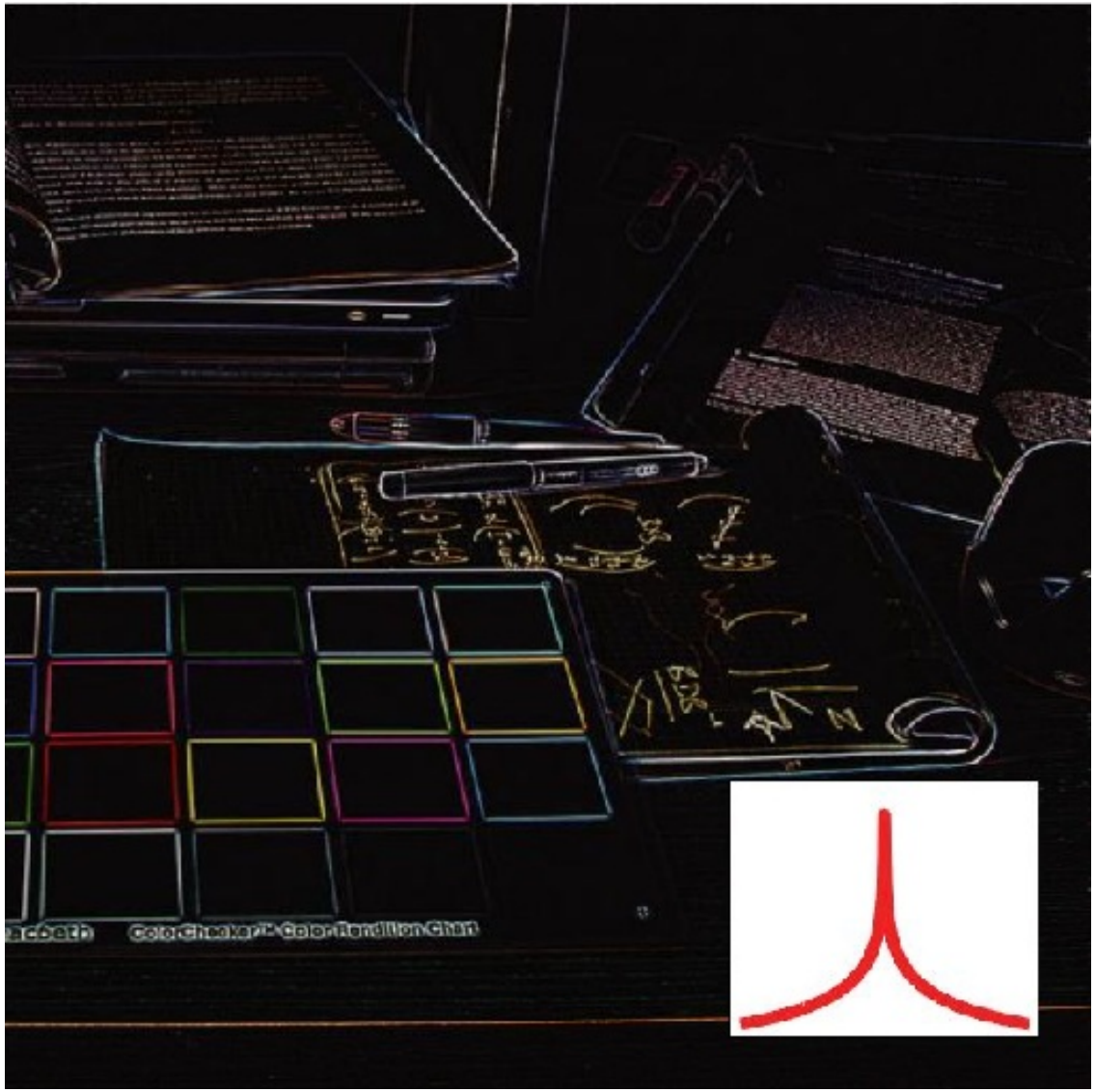




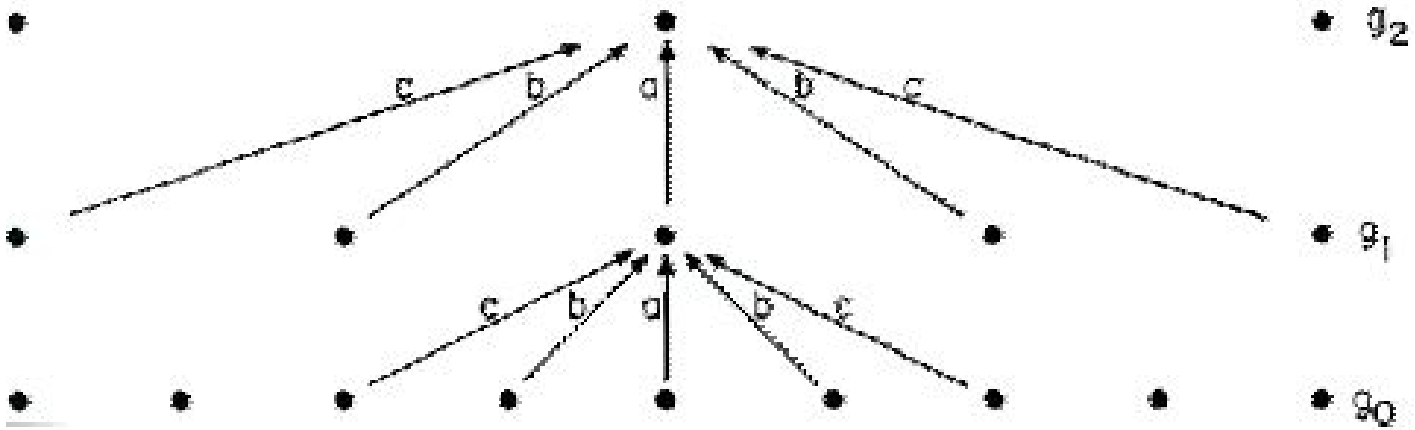
Thank you for your attention!







GAUSSIAN PYRAMID



g_0 = IMAGE

g_L = REDUCE $[g_{L-1}]$