Salvador Dali, Abraham Lincoln
Salvador Dali, Abraham Lincoln
RECALL: OBJECT RECOGNITION WITH HOG
Recall: Object Recognition with HOG
RECALL: OBJECT RECOGNITION WITH HOG
RECALL: OBJECT RECOGNITION WITH HOG
**Fourier Transform**

- **Time signal**: $x(t)$
- **Frequency response**: $X(f)$

The Fourier Transform (FT) converts a time-domain signal into its frequency-domain representation. The Inverse Fourier Transform (Inverse FT) performs the reverse operation, converting the frequency-domain representation back into the time domain.
**Fourier Transform**

Time signal \( x(t) \) * Guassian filter \( g(t) \)

\[ X(f) \ast G(f) \]

FT → Inverse FT

\( |G(f)| \)

\( |X(f)| \)

\(-f \quad f\)

Frequency response
**Fourier Transform**

Time signal \( x(t) \)

Guassian filter \( g(t) \)

\[ X(f) G(f) \]
**Fourier Transform**

Time signal | Guassian filter

\[ x(t) \ast g(t) \]

\[ X(f) G(f) \]

Forward Fourier Transform (FT) → Frequency response

Inverse Fourier Transform (Inverse FT) ← Frequency response

\[ |X(f)G(f)| \]
GAUSSIAN FILTERING ~ LOW-PASS FILTERING
Gaussian Filtering \sim Low-pass Filtering

\[ g(\sigma = 2) \]
Gaussian Filtering ~ Low-pass Filtering

\[ g(\sigma = 8) \]
GAUSSIAN FILTERING ~ LOW-PASS FILTERING

\[ g(\sigma = 32) \]
GAUSSIAN FILTERING ~ LOW-PASS FILTERING ~ IMAGE BLURRING
MULTI-DIMENSIONAL IMAGE REPRESENTATION
SUBSAMPLING

\[ x(t) \]
Subsampling

\( x(t) \) 

Subsampling 

\( \downarrow x(t) \)
SUBSAMPLING WITH G. FILTERING

\[ x(t) \ast g(t) \]

Subsampling

\[ \downarrow x(t) \]
SUBSAMPLING WITH G. FILTERING

\[ x(t) \ast g(t) \]
Subsampling with G. Filtering

$\downarrow x(t)$
Naïve Subsampling

\[ x(t) \]
### Aliasing

<table>
<thead>
<tr>
<th>Resolution</th>
<th>Image</th>
</tr>
</thead>
<tbody>
<tr>
<td>256x256</td>
<td><img src="image1.png" alt="Image" /></td>
</tr>
<tr>
<td>128x128</td>
<td><img src="image2.png" alt="Image" /></td>
</tr>
<tr>
<td>64x64</td>
<td><img src="image3.png" alt="Image" /></td>
</tr>
<tr>
<td>32x32</td>
<td><img src="image4.png" alt="Image" /></td>
</tr>
<tr>
<td>16x16</td>
<td><img src="image5.png" alt="Image" /></td>
</tr>
</tbody>
</table>

Naïve subsampling

<table>
<thead>
<tr>
<th>Resolution</th>
<th>Image</th>
</tr>
</thead>
<tbody>
<tr>
<td>256x256</td>
<td><img src="image1.png" alt="Image" /></td>
</tr>
<tr>
<td>128x128</td>
<td><img src="image2.png" alt="Image" /></td>
</tr>
<tr>
<td>64x64</td>
<td><img src="image3.png" alt="Image" /></td>
</tr>
<tr>
<td>32x32</td>
<td><img src="image4.png" alt="Image" /></td>
</tr>
<tr>
<td>16x16</td>
<td><img src="image5.png" alt="Image" /></td>
</tr>
</tbody>
</table>

Smoothing and subsampling: eliminating aliasing effects.
GAUSSIAN FILTERING AND THEN SUBSAMPLING
Image Reconstruction: Upsampling and Gaussian Blurring
Cf) Naïve Image Subsampling and Upsampling
Multi-dimensional Image Representation
\[ |I| \left(1 + \frac{1}{4} + \frac{1}{16} + \cdots \right) = \frac{4}{3} |I| \]
COMPOSITION OF GAUSSIAN FILTERS

Time signal

\[ x(t) \]

Guassian filter

\[ g(t; \sigma_1) \quad g(t; \sigma_2) \]

\[ X(f) \quad G(f) \quad G(f) \]
**Composition of Gaussian Filters**

- Time signal: $x(t)$
- Gaussian filter: $g(t; \sqrt{\sigma_1^2 + \sigma_2^2})$

The process involves the Fourier transform (FT) of the time signal and the frequency response of the Gaussian filter. The convolution in the time domain is equivalent to multiplication in the frequency domain, resulting in:

$$X(f) G(f) G(f)$$
**Composition of Gaussian Filters**

\[ \mathbf{X}(f) G(f) G(f) \]

\[ x(t) \ast g(t; \sqrt{\sigma_1^2 + \sigma_2^2}) \]

Time signal \quad Guassian filter

FT \quad \text{Inverse FT}

\[ f \quad \text{Frequency response} \quad -f \]
$1 + \frac{1}{4} + \frac{1}{16} + \cdots = \frac{4}{3} |I|$
$\frac{1}{2}$ subsampling

$\frac{1}{2}$ subsampling

$\frac{1}{2}$ subsampling

Memory consumption

$|I| (1 + \frac{1}{4} + \frac{1}{16} + \cdots) = \frac{4}{3} |I|$
function im_pyramid = BuildGaussianPyramid(im, level)

im_pyramid{1} = im;

for i = 1 : level
    filter = fspecial('gaussian',10, sqrt(2));
    im_f = conv2(im_pyramid{i}, filter, 'same');
    im_pyramid{i+1} = imresize(im_f, 0.5, 'nearest');
end
REDUNDANT REPRESENTATION OF GAUSSIAN PYRAMID
**Fourier Transform**

- $x(t)$
- $g(t; \sigma_1)$
- $g(t; \sigma_2)$
- $X(f)$
- $G(f)$
- $G(f; \sigma_1)$
- $G(f; \sigma_2)$
- $\sigma_1 < \sigma_2$

Diagram showing the Fourier transform process:
- Convolution in time domain
- Fourier transform
- Inverse Fourier transform

Redundant frequency band:
- $f$
- $-f$

Scale dependent frequency band:
- $\leq \sigma_2$
- $\geq \sigma_1$
DIFFERENCE OF GAUSSIAN (DOG) ~ BAND-PASS FILTER

\[
x(t) \ast g(t) \rightarrow \text{FT} \rightarrow \text{Inverse FT} \rightarrow -f \rightarrow f
\]

\[
|X(f)G(f;\sigma_2)| - |X(f)G(f;\sigma_1)|
\]

\[
x(t) \ast g(t) = X(f)G(f)
\]
**Difference of Gaussian (DoG) ~ Band-pass Filter**

\[ x(t) \ast (g(t;\sigma_1) - g(t;\sigma_2)) \]

\[ X(f)(G(f;\sigma_1) - G(f;\sigma_2)) \]
Difference of Gaussian (DoG) ~ Band-pass Filter

$x(t)$ $*$ $g(t;\sigma_1) - g(t;\sigma_2)$

$X(f)(G(f;\sigma_1) - G(f;\sigma_2))$
**Difference of Gaussian (DoG) ~ Band-pass Filter**

\[ x(t) * (g(t;\sigma_1) - g(t;\sigma_2)) \]

\[ X(f)(G(f;\sigma_1) - G(f;\sigma_2)) \]
Laplacian of Gaussian (LOG) ~ DoG

\[ x(t) \ast (g(t;\sigma_1) - g(t;\sigma_2)) \approx \nabla \cdot \nabla g \]

\[ \text{Laplacian of Gaussian} \]

\[ X(f) \left( G(f;\sigma_1) - G(f;\sigma_2) \right) \]
**Laplacian of Gaussian (LoG) ~ DoG**
**Image Laplacian**

$I$

$I - I \ast G$

$I \ast G$
Image Laplacian

$I \ast G$

$I \ast G - I \ast G \ast G$

$I \ast G \ast G$
function [g_pyramid, l_pyramid] = BuildLaplacianPyramid(im, level)

g_pyramid{1} = im;

for i = 1 : level
    filter = fspecial('gaussian',10, sqrt(2));
    im_f = conv2(g_pyramid{i}, filter, 'same');
    l_pyramid{i} = double(g_pyramid{i})-im_f;
    g_pyramid{i+1} = imresize(im_f, 0.5, 'nearest');
end
**Difference of Gaussian (DOG) ~ Band-pass Filter**

\[ x(t) \ast (g(t;\sigma_1) - g(t;\sigma_2)) \]

\[ X(f)(G(f;\sigma_1) - G(f;\sigma_2)) \]
**Signal Reconstruction ~ Log + G. Filtering**

\[ l(t) = x(t) * (g(t; \sigma_1) - g(t; \sigma_2)) \]

\[ x(t) * g(t; \sigma_1) = l(t) + x(t) * g(t; \sigma_2) \]

Signal reconstruction with laplacian

\[ L(f) = X(f)(G(f; \sigma_1) - G(f; \sigma_2)) \]

\[ X(f)G(f; \sigma_1) = L(f) + X(f)G(f; \sigma_2) \]

LoG Smoother signal
IMAGE LAPLACIAN