Lecture 4

Motion filters
Spatial pyramids
Recap
Box filter
Gaussian filter

$\sigma = 2$

$\sigma = 4$

$\sigma = 8$
### Binomial filter

<table>
<thead>
<tr>
<th>$b_1$</th>
<th>$b_2$</th>
<th>$b_3$</th>
<th>$b_4$</th>
<th>$b_5$</th>
<th>$b_6$</th>
<th>$b_7$</th>
<th>$b_8$</th>
</tr>
</thead>
<tbody>
<tr>
<td></td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>2</td>
<td>1</td>
<td>1</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>3</td>
<td>3</td>
<td>3</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>4</td>
<td>6</td>
<td>4</td>
<td>1</td>
<td></td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>5</td>
<td>10</td>
<td>10</td>
<td>5</td>
<td>1</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>6</td>
<td>15</td>
<td>20</td>
<td>15</td>
<td>6</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>7</td>
<td>21</td>
<td>35</td>
<td>35</td>
<td>21</td>
</tr>
<tr>
<td></td>
<td></td>
<td>1</td>
<td>8</td>
<td>28</td>
<td>56</td>
<td>70</td>
<td>56</td>
</tr>
</tbody>
</table>

- $\sigma^2_1 = 1/4$
- $\sigma^2_2 = 1/2$
- $\sigma^2_3 = 3/4$
- $\sigma^2_4 = 1$
- $\sigma^2_5 = 5/4$
- $\sigma^2_6 = 3/2$
- $\sigma^2_7 = 7/4$
- $\sigma^2_8 = 2$
\[
\frac{\partial I}{\partial x} \approx I(x, y) - I(x - 1, y)
\]

\[
g[m,n] \times [-1, 1] = h[m,n]
\]

\[
f[m,n]
\]
Derivatives of Gaussians: Scale

$\sigma = 2$

$\sigma = 4$

$\sigma = 8$
Laplacian filter

Made popular by Marr and Hildreth in 1980 in the search for operators that locate the boundaries between objects.

The Laplacian operator is defined as the sum of the second order partial derivatives of a function:

$$\nabla^2 I = \frac{\partial^2 I}{\partial x^2} + \frac{\partial^2 I}{\partial y^2}$$

To reduce noise and undefined derivatives, we use the same trick:

$$\nabla^2 I \circ g = \nabla^2 g \circ I$$

Where:

$$\nabla^2 g = \frac{x^2 + y^2 - 2\sigma^2}{\sigma^4} g(x, y)$$
Comparison derivative and laplacian

Zero crossings
Contrast Sensitivity Function

Blackmore & Campbell (1969)

Maximum sensitivity

\[ \sim 6 \text{ cycles/degree of visual angle} \]

Things that are very close and large are hard to see

Things far away are hard to see
Vasarely visual illusion
Image sharpening filter

Subtract away the blurred components of the image:

\[
\text{sharpening filter} = \begin{bmatrix}
0 & 0 & 0 \\
0 & 2 & 0 \\
0 & 0 & 0 \\
\end{bmatrix} - \frac{1}{16} \begin{bmatrix}
1 & 2 & 1 \\
2 & 4 & 2 \\
1 & 2 & 1 \\
\end{bmatrix}
\]

This filter has an overall DC component of 1. It de-emphasizes the blur component of the image (low spatial frequencies).

The DC component is the mean value of the image.
Input image
Other “naturally” occurring filters
Artistic effects
Sequences
Sequences
Sequences

Cube size = 128x128x90
Sequences

Cube size = 128x128x90
Global constant motion

Let's work on the continuous space-time domain…
A global motion can be written as:

\[ f(x, y, t) = f_0(x - v_xt, y - v_yt) \]

Where:

\[ f_0(x, y) = f(x, y, 0) \]
\[ f(x, y, t) = f_0(x - v_xt, y - v_yt) \]
\[ F(w_x, w_y, w_t) = F_0(w_x, w_y) \delta(w_t - v_x w_x - v_y w_y) \]
Temporal Gaussian

\[ g(x, y, t; \sigma_x, \sigma_t) = \frac{1}{(2\pi)^{3/2}\sigma_x^2\sigma_t} \exp\left(-\frac{x^2 + y^2}{2\sigma_x^2}\right) \exp\left(-\frac{t^2}{2\sigma_t^2}\right) \]
Temporal Gaussian

How could we create a filter that keeps sharp objects that move at some velocity \((vx, vy)\) while blurring the rest?

(Note: although some of the analysis is done on continuous variables, the processing is done on the discrete domain)
Temporal Gaussian

How could we create a filter that keeps sharp objects that move at some velocity \((v_x, v_y)\) while blurring the rest?

\[ g_{v_x,v_y}(x,y,t) = g(x - v_xt, y - v_yt, t) \]
Space-time Gaussian derivatives

\[ \frac{\partial g}{\partial t} = \frac{-t}{\sigma_t^2} g(x, y, t) \]

\[ \nabla g = (g_x(x, y, t), g_y(x, y, t), g_t(x, y, t)) = \]

\[ = (\frac{-x}{\sigma^2}, \frac{-y}{\sigma^2}, \frac{-t}{\sigma_t^2}) g(x, y, t) \]

**Note**: we can discretize time derivatives in the same way we discretized spatial derivatives. For instance:

\[ f[m, n, t] - f[m, n, t - 1] \]
Cancelling moving objects

Can we create a filter that removes objects that move at some velocity \((v_x, v_y)\) while keeping the rest?
Space-time Gaussian derivatives

For a global translation, we can write:

\[ f(x, y, t) = f_0(x - v_x t, y - v_y t) \]

Therefore, we can write the temporal derivative of \( f \) as a function of the spatial derivatives of \( f_0 \):

\[
\frac{\partial f}{\partial t} = \frac{\partial f_0}{\partial t} = -v_x \frac{\partial f_0}{\partial x} - v_y \frac{\partial f_0}{\partial y}
\]

And from here (using derivatives of \( f \)):

\[
\frac{\partial f}{\partial t} + v_x \frac{\partial f}{\partial x} + v_y \frac{\partial f}{\partial y} = 0
\]

This relation is known as the “Brightness change constraint equation”, introduced by Horn & Schunck in 1981.
Can could we create a filter that removes objects that move at some velocity \((v_x, v_y)\) while keeping the rest?

Yes, we could create a filter that implements this constraint:

\[
\frac{\partial f}{\partial t} + v_x \frac{\partial f}{\partial x} + v_y \frac{\partial f}{\partial y} = 0
\]

We can create this filter as a combination of Gaussian derivatives:

\[
h(x, y, t; v_x, v_y) = g_t + v_x g_x + v_y g_y
\]

\[
= \nabla g \left(1, v_x, v_y\right)^T
\]
Space-time Gaussian derivatives

\[ h(x, y, t; v_x, v_y) = g_t + v_x g_x + v_y g_y \]
Gabor wavelets and quadrature filters
What is a good representation for image analysis?

- Fourier transform domain tells you “what” (textural properties), but not “where”.
- Pixel domain representation tells you “where” (pixel location), but not “what”.
- Want an image representation that gives you a local description of image events—what is happening where.
Analysis of local frequency

Fourier basis:

\[ e^{j 2\pi u_0 x} \]

Gabor wavelet:

\[ \psi(x,y) = e^{\frac{-x^2 + y^2}{2\sigma^2}} e^{j 2\pi u_0 x} \]

We can look at the real and imaginary parts:

\[ \psi_c(x,y) = e^{\frac{-x^2 + y^2}{2\sigma^2}} \cos(2\pi u_0 x) \]

\[ \psi_s(x,y) = e^{\frac{-x^2 + y^2}{2\sigma^2}} \sin(2\pi u_0 x) \]
Gabor wavelets

\[ \psi_c(x, y) = e^{-\frac{x^2 + y^2}{2\sigma^2}} \cos(2\pi u_0 x) \]

\[ \psi_s(x, y) = e^{-\frac{x^2 + y^2}{2\sigma^2}} \sin(2\pi u_0 x) \]
Gabor filters at different scales and spatial frequencies

Top row shows anti-symmetric (or odd) filters; these are good for detecting odd-phase structures like edges.

Bottom row shows the symmetric (or even) filters, good for detecting line phase contours.
Fourier transform of a Gabor wavelet

\[ \psi_c(x, y) = e^{-\frac{x^2+y^2}{2\sigma^2}} \cos(2\pi u_0 x) \]
Comparing Human and Machine Perception

**Structure**

- World
- Optics
- Photoreceptor Array
- LGN Cells
- Primary Visual Cortical Neurons: Simple & Complex

**Operations**

- \( I(x, y, t, \lambda) \)
- Low-pass spatial filtering
- Sampling, more low-pass filtering, temporal low/bandpass filtering, \( \lambda \) filtering, gain control, response compression
- Spatiotemporal bandpass filtering, \( \lambda \) filtering, multiple parallel representations
- Simple cells: orientation, phase, motion, binocular disparity, & \( \lambda \) filtering
- **Complex cells: no phase filtering (contrast energy detection)**

**2D Fourier Plane**

- 2D Fourier representations of the world, retinal image, and sensitivities typical of a ganglion and cortical cell.

**FIGURE 1** Schematic overview of the processing done by the early visual system. On the left, are some of the major structures to be discussed; in the middle, are some of the major operations done at the associated structure; in the right, are the 2-D Fourier representations of the world, retinal image, and sensitivities typical of a ganglion and cortical cell.
Fig. 5. Top row: illustrations of empirical 2-D receptive field profiles measured by J. P. Jones and L. A. Palmer (personal communication) in simple cells of the cat visual cortex. Middle row: best-fitting 2-D Gabor elementary function for each neuron, described by (10). Bottom row: residual error of the fit, indistinguishable from random error in the Chi-squared sense for 97 percent of the cells studied.
Quadrature filter pairs

A quadrature filter is a complex filter whose real part is related to its imaginary part via a Hilbert transform along a particular axis through origin of the frequency domain.
Contrast invariance! (same energy response for white dot on black background as for a black dot on a white background).
edge

ergy response to an edge
energy response to a line
A contour detector
Iris codes are compared using Hamming distance

John Daugman

Images from http://cnx.org/content/m12493/latest/
Setting the Bits in an IrisCode

\[ h_{Re} = 1 \text{ if } \text{Re} \int \int e^{-i\omega(\theta_0-\phi)}e^{-(\rho-\rho_0)^2/\alpha^2} e^{-(\theta_0-\theta)^2/\beta^2} I(\rho,\phi) \rho d\rho d\phi \geq 0 \]

\[ h_{Re} = 0 \text{ if } \text{Re} \int \int e^{-i\omega(\theta_0-\phi)}e^{-(\rho-\rho_0)^2/\alpha^2} e^{-(\theta_0-\theta)^2/\beta^2} I(\rho,\phi) \rho d\rho d\phi < 0 \]

\[ h_{Im} = 1 \text{ if } \text{Im} \int \int e^{-i\omega(\theta_0-\phi)}e^{-(\rho-\rho_0)^2/\alpha^2} e^{-(\theta_0-\theta)^2/\beta^2} I(\rho,\phi) \rho d\rho d\phi \geq 0 \]

\[ h_{Im} = 0 \text{ if } \text{Im} \int \int e^{-i\omega(\theta_0-\phi)}e^{-(\rho-\rho_0)^2/\alpha^2} e^{-(\theta_0-\theta)^2/\beta^2} I(\rho,\phi) \rho d\rho d\phi < 0 \]
Gabor filter measurements for iris recognition code

John Daugman, http://www.cl.cam.ac.uk/~jgd1000/
Gabor wavelet:

$$\psi(x, y) = e^{-\frac{x^2 + y^2}{2\sigma^2}} e^{j2\pi u_0 x}$$

Tuning filter orientation:

$$x' = \cos(\alpha) x + \sin(\alpha) y$$

$$y' = -\sin(\alpha) x + \cos(\alpha) y$$
Second directional derivative of a Gaussian and its quadrature pair
High resolution in orientation requires many oriented filters as basis (high order gaussian derivatives or fine-tuned Gabor wavelets).
Orientation analysis
Fig. 10. Measures of orientation derived from $G_4$ and $H_4$ steerable filter outputs: (a) Input image for orientation analysis; (b) angular average of oriented energy as measured by $G_4$, $H_4$ quadrature pair. This is an oriented features detector; (c) conventional measure of orientation: dominant orientation plotted at each point. No dominant orientation is found at the line intersection or corners; (d) oriented energy as a function of angle, shown as a polar plot for a sampling of points in the image (a). Note the multiple orientations found at intersection points of lines or edges and at corners, shown by the florets there.
Image pyramids
Image information occurs at all spatial scales
Gaussian filter

\[ g(x, y; \sigma) = \frac{1}{2\pi \sigma^2} \exp \left( -\frac{x^2 + y^2}{2\sigma^2} \right) \]
The Gaussian pyramid

For each level
  Blur input image with a Gaussian filter
  Downsample by a factor of 2
  Output downsampled image
The Gaussian pyramid

512×512

256×256

128×128 64×64 32×32

(original image)
The Gaussian pyramid

For each level

1. Blur input image with a Gaussian filter

\[ [1, 4, 6, 4, 1] \]
The Gaussian pyramid

For each level
1. Blur input image with a Gaussian filter
2. Downsample image
Downsampling

Blur → Δ2

(no frequency content is lost)
In 1D, one step of the Gaussian pyramid is:

\[ g_0 = \text{IMAGE} \]

\[ g_L = \text{REDUCE} [g_{L-1}] \]

Fig 1. A one-dimensional graphic representation of the process which generates a Gaussian pyramid. Each row of dots represents nodes within a level of the pyramid. The value of each node in the zero level is just the gray level of a corresponding image pixel. The value of each node in a high level is the weighted average of node values in the next lower level. Note that node spacing doubles from level to level, while the same weighting pattern or “generating kernel” is used to generate all levels.

Convolution and subsampling as a matrix multiply (1D case)

\[ x_2 = G_1 x_1 \]

\[ G_1 = \]

\[
\begin{bmatrix}
1 & 4 & 6 & 4 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 4 & 6 & 4 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 1 & 4 & 6 & 4 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 4 & 6 & 4 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 1 & 4 & 6 & 4 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 1 & 4 & 6 & 4 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 4 & 6 & 4 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 4 & 6 & 4 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 4 & 6 & 4 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 4 & 6 & 4 & 1 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 4 & 6 & 4 & 1 & 0 & 0 & 0 & 0 \\
\end{bmatrix}

(Normalization constant of 1/16 omitted for visual clarity.)
Next pyramid level

\[ x_3 = G_2 x_2 \]

\[ G_2 = \]

\[
\begin{array}{cccccccccc}
1 & 4 & 6 & 4 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 1 & 4 & 6 & 4 & 1 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 4 & 6 & 4 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 4 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 \\
\end{array}
\]
The combined effect of the two pyramid levels

\[ x_3 = G_2 G_1 x_1 \]

\[ G_2 G_1 = \]

\[
\begin{array}{cccccccccccccccccc}
1 & 4 & 10 & 20 & 31 & 40 & 44 & 40 & 31 & 20 & 10 & 4 & 1 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 1 & 4 & 10 & 20 & 31 & 40 & 44 & 40 & 31 & 20 & 10 & 4 & 1 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 4 & 10 & 20 & 31 & 40 & 44 & 40 & 30 & 16 & 4 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 1 & 4 & 10 & 20 & 25 & 16 & 4 & 0 & 0 & 0 & 0 \\
\end{array}
\]
1D Gaussian pyramid matrix, for \([1 \ 4 \ 6 \ 4 \ 1]\) low-pass filter

full-band image, highest resolution

lower-resolution image

lowest resolution image
Gaussian pyramids used for

• up- or down-sampling images.
• Multi-resolution image analysis
  – Look for an object over various spatial scales
  – Coarse-to-fine image processing: form blur estimate or the motion analysis on very low-resolution image, upsample and repeat. Often a successful strategy for avoiding local minima in complicated estimation tasks.
Image down-sampling

Blur

Image up-sampling

$\Delta^2$

$\otimes^2$
Image up-sampling

Start by inserting zeros

\[
1 2 1 \\
\circ 2 4 2 = \\
1 2 1
\]

64\times 64 \quad 128\times 128
Image up-sampling

\[
\begin{bmatrix}
1 & 2 & 1 \\
2 & 4 & 2 \\
1 & 2 & 1
\end{bmatrix}
\]
Convolution and up-sampling as a matrix multiply (1D case)

\[ y_2 = F_3 x_3 \]

Insert zeros between pixels, then apply a low-pass filter, \([1 \ 4 \ 6 \ 4 \ 1]\)

\[ F_3 = \begin{bmatrix} 6 & 1 & 0 & 0 \\ 4 & 4 & 0 & 0 \\ 1 & 6 & 1 & 0 \\ 0 & 4 & 4 & 0 \\ 0 & 1 & 6 & 1 \\ 0 & 0 & 4 & 4 \\ 0 & 0 & 1 & 6 \\ 0 & 0 & 0 & 4 \end{bmatrix} \]
The Laplacian Pyramid

- **Synthesis**
  - Compute the difference between upsampled Gaussian pyramid level and Gaussian pyramid level.
  - band pass filter - each level represents spatial frequencies (largely) unrepresented at other level.
Laplacian pyramid algorithm

\[ x_1 \xrightarrow{G_1 x_1 = x_2} x_2 \xrightarrow{(I - F_2 G_2)x_2} x_3 \xrightarrow{(I - F_3 G_3)x_3} \]

\[ (I - F_1 G_1)x_1 \]
Showing, at full resolution, the information captured at each level of a Gaussian (top) and Laplacian (bottom) pyramid.

Fig 5. First four levels of the Gaussian and Laplacian pyramid. Gaussian images, upper row, were obtained by expanding pyramid arrays (Fig. 4) through Gaussian interpolation. Each level of the Laplacian pyramid is the difference between the corresponding and next higher levels of the Gaussian pyramid.

Laplacian pyramid reconstruction algorithm:
recover $x_1$ from $L_1$, $L_2$, $L_3$ and $x_4$

$G#$ is the blur-and-downsample operator at pyramid level #
$F#$ is the blur-and-upsample operator at pyramid level #

Laplacian pyramid elements:
$L_1 = (I - F_1 G_1) x_1$
$L_2 = (I - F_2 G_2) x_2$
$L_3 = (I - F_3 G_3) x_3$
$x_2 = G_1 x_1$
$x_3 = G_2 x_2$
$x_4 = G_3 x_3$

Reconstruction of original image ($x_1$) from Laplacian pyramid elements:
$x_3 = L_3 + F_3 x_4$
$x_2 = L_2 + F_2 x_3$
$x_1 = L_1 + F_1 x_2$
Laplacian pyramid reconstruction algorithm: recover $x_1$ from $L_1$, $L_2$, $L_3$ and $g_3$
| 512 | 256 | 128 | 64  | 32  | 16  | 8   | (Low-pass residual) |

Laplacian pyramid
1-d Laplacian pyramid matrix, for \([1 \ 4 \ 6 \ 4 \ 1]\) low-pass filter
Laplacian pyramid applications

- Texture synthesis
- Image compression
- Noise removal

- Also related to SIFT
Image blending

(a) 

(b)
Figure 3.42  Laplacian pyramid blending details (Burt and Adelson 1983b) © 1983 ACM. The first three rows show the high, medium, and low frequency parts of the Laplacian pyramid.
Image blending

- Build Laplacian pyramid for both images: LA, LB
- Build Gaussian pyramid for mask: G
- Build a combined Laplacian pyramid: \( L(j) = G(j) \, LA(j) + (1-G(j)) \, LB(j) \)
- Collapse L to obtain the blended image
Sampling
Sampling

Continuous world

Pixels
Sampling
Sampling
Sampling
What will be the best sampling pattern in 2D?

Images from: http://www.cns.nyu.edu/~david/courses/perception/lecturenotes/retina/retina.html
Sampling

Continuous image $f(x, y)$

We can sample it using a rectangular grid as

$$f [n, m] = f (nT_x, mT_y)$$
Let’s start with this continuous image (it is not really continuous…)

Aliasing
Aliasing
Modeling the sampling process

Continuous image $f(x, y)$

We can sample it using a rectangular grid as

$$f[n, m] = f(nT_x, mT_y)$$

Or a more general sampling pattern

$$f[n, m] = f(an + bm, cn + dm)$$

If $a = T$, $b = 0$, $c = 0$, $d = T$ then we will have a rectangular sampling
Modeling the sampling process

\[ f[n] = f(nT_s) \]

A convenient writing:

\[ \hat{f}(t) = f(t) \sum_{n=-\infty}^{\infty} \delta(t - nT_s) \]

Delta train

Sampled signal
Modeling the sampling process

\[ \hat{f}(t) = f(t) \sum_{n=-\infty}^{\infty} \delta(t - nT_s) = f(t) \delta_{T_s}(t) \]

The Fourier transform is a convolution…

Interesting property of the delta train: the Fourier transform of a delta train of period T is another delta train with period \(2\pi/T\)
Interesting property of the delta train: the Fourier transform of a delta train of period $T$ is another delta train with period $2\pi/T$. Demo in the class notes.
Modeling the sampling process

What happens when the repetitions overlap?
Both waves fit the same samples. Aliasing consists in “perceiving” the red wave when the actual input was the blue wave.
Sampling theorem

The sampling theorem (also known as Nyquist theorem) states that for a signal to be perfectly reconstructed from it samples, the sampling period $T_s$ has to be $T_s > T_{\text{min}}/2$ where $T_{\text{min}}$ is the period of the highest frequency present in the input signal.

$$F(w)$$

$$w$$

$$2\pi/T_{\text{min}}$$
Antialiasing filtering

Before sampling, apply a low pass-filter to remove all the frequencies that will produce aliasing.

Without antialiasing filter.

With antialiasing filter.
Modeling the 2D sampling process

\[ f[n,m] = f(an + bm, cn + dm) \]

\[ \hat{f}(x, y) = f(x, y) \sum_{n=-\infty}^{\infty} \sum_{m=-\infty}^{\infty} \delta(x - an - bm, y - cn - dm) \]

Rectangular sampling

Hexagonal sampling
2D sampling

Images from: http://www.cns.nyu.edu/~david/courses/perception/lecturenotes/retina/retina.html