ELEG 404/604 - Imaging and Audio Signal Processing

Gonzalo R. Arce

Department of Electrical and Computer Engineering
University of Delaware
Newark, DE 19711
Spring 2016
IMAGE SAMPLING

Discrete-time image processing requires the representation of images by a sampled array on a 2-D Lattice. There are several practical methods of sampling. Modern devices, such as charged-coupled devices, contain an array of photodetectors, and a set of electronic switches:

where charge couple devices (CCDs) consist on an array of detectors

Common resolution range from \((256)^2 \leftrightarrow (2000)^2\)
Although images are not generally band limited, we can approximately represent them by bandlimited signals. Let $g(x,y)$ be a 2-D continuous image. Rectangular sampling is modelled as:

$$g_s(x,y) = \sum_m \sum_n g(mX, nY) \delta(x - mX, y - nY)$$

Assuming $g(x,y)$ is band limited, $G(u, v) = 0$ for $u > B_x$ and $v > B_y$
The Fourier transform of the sampled image is

\[
F\{g_s(x, y)\} = F \left\{ g(x, y) \sum_m \sum_n \delta(x - mX, y - uY) \right\} = G(u, v) \ast \frac{1}{X} \frac{1}{Y} \sum_m \sum_n \delta \left( u - \frac{m}{X}, v - \frac{n}{Y} \right)
\]
The Fourier transform of the sampled image is

\[ F\{g_s(x, y)\} = F \left\{ g(x, y) \sum_m \sum_n \delta(x - mX, y - uY) \right\} \]

\[ = G(u, v) * \frac{1}{X} \frac{1}{Y} \sum_m \sum_n \delta \left( u - \frac{m}{X}, v - \frac{n}{Y} \right) \]

\[ = \frac{1}{XY} \sum_m \sum_n G \left( u - \frac{m}{X}, v - \frac{n}{Y} \right) \]

\[ = \frac{1}{XY} \left( \text{rep} \frac{1}{X} \frac{1}{Y} (G(u, v)) \right) \]
Representing $G(u, v)$ as

then, the spectra of $g_s(x, y)$ is seen as the replication of $G(u, v)$:
To prevent aliasing, we require \( B_x < \frac{1}{2X} \) and \( B_y < \frac{1}{2Y} \).

In order to reconstruct the continuous signal, we can filter \( G_s(u, v) \) by the LPF (ideal):

\[
H(u, v) = \begin{cases} 
    XY & |v| < \frac{1}{2Y}, |u| < \frac{1}{2X} \\
    0 & \text{else}
\end{cases}
\]

The space domain filter is obtained as:

\[ h(x, y) = \text{sinc}[\frac{x}{X}, \frac{y}{Y}]. \]
Hence,

\[
\hat{g}(x, y) = g_s(x, y) * \text{sinc}[\frac{x}{X}, \frac{y}{Y}]
\]

\[
= \left[ \sum_m \sum_n g(mX, nY) \delta(x - mX, y - uY) \right] * \text{sinc}[\frac{x}{X}, \frac{y}{Y}]
\]

\[
\hat{g}(x, y) = \sum_m \sum_n g(mX, nY) \text{sinc} \left( \frac{x - mX}{X}, \frac{y - nY}{Y} \right)
\]
Hence,

\[
\hat{g}(x, y) = g_s(x, y) \ast \text{sinc}\left[\frac{x}{X}, \frac{y}{Y}\right]
\]

\[
= \left[ \sum_m \sum_n g(mX, nY) \delta(x - mX, y - uY) \right] \ast \text{sinc}\left[\frac{x}{X}, \frac{y}{Y}\right]
\]

\[
\hat{g}(x, y) = \sum_m \sum_n g(mX, nY) \text{sinc}\left(\frac{x - mX}{X}, \frac{y - nY}{Y}\right)
\]

The ideal LPF is difficult to obtain, hence, other filters are generally designed. For instance, if we are to obtain a continuous image by projecting into a CRT display, we are effectively replacing the 2-D function by a Gaussian function

\[
p(x, y) = \frac{1}{2\pi \sigma^2} \exp\left[\frac{-(x^2 + y^2)}{2\sigma^2}\right]
\]

Hence the reconstructed image is:

\[
\hat{g}(x, y) = \sum_m \sum_n g(mX, nY) \left(\frac{1}{2\pi \sigma^2}\right) \exp\left[\frac{-(x - mX)^2 - (y - nY)^2}{2\sigma^2}\right]
\]
The effect is the introduction of aliasing. To illustrate, consider a slice of $G_s(u, v)$ and $\hat{G}(u, v)$.
The effect is the introduction of aliasing. To illustrate, consider a slice of $G_s(u, v)$ and $\hat{G}(u, v)$.

These concepts are further discussed, in the interpolation and decimation of images, where the physical size of the images is varied by keeping the same spatial resolution. These are very important issue in commercial applications.
Aliasing

FIGURE 4.15
Two-dimensional Fourier transforms of (a) an over-sampled, and (b) under-sampled band-limited function.
Example of Aliasing

- Example of aliasing error in a sampled image
- Spurious spatial frequency components
- It creates low-spatial-frequency components in the reconstruction
- Known as moiré patterns
Aliasing

**Figure 4.16** Aliasing in images. In (a) and (b), the lengths of the sides of the squares are 16 and 6 pixels, respectively, and aliasing is visually negligible. In (c) and (d), the sides of the squares are 0.9174 and 0.4798 pixels, respectively, and the results show significant aliasing. Note that (d) masquerades as a “normal” image.
Aliasing

**FIGURE 4.17** Illustration of aliasing on resampled images. (a) A digital image with negligible visual aliasing. (b) Result of resizing the image to 50% of its original size by pixel deletion. Aliasing is clearly visible. (c) Result of blurring the image in (a) with a $3 \times 3$ averaging filter prior to resizing. The image is slightly more blurred than (b), but aliasing is not longer objectionable. (Original image courtesy of the Signal Compression Laboratory, University of California, Santa Barbara.)
Aliasing

Courtesy of Scientific Volume Imaging - http://www.svi.nl/antialiasing
Aliasing

**Figure 4.20**
Examples of the moiré effect. These are ink drawings, not digitized patterns. Superimposing one pattern on the other is equivalent mathematically to multiplying the patterns.
Aliasing

FIGURE 4.21
A newspaper image of size 246 × 168 pixels sampled at 75 dpi showing a moiré pattern. The moiré pattern in this image is the interference pattern created between the ±45° orientation of the halftone dots and the north–south orientation of the sampling grid used to digitize the image.
Aliasing

**FIGURE 4.22**
A newspaper image and an enlargement showing how halftone dots are arranged to render shades of gray.
Moiré Pattern Effect

- Each grate is periodic
- Their superposition breaks the periodicity
- The problem is common in scanning of printed material
- Periodicities do not line up causing aliasing
Aliasing

Courtesy of Scientific Volume Imaging - http://www.svi.nl/antialiasing
Aliasing
Interpolation

- Estimating intermediate values of sampled function
- Obtain estimate of image $l(x, y)$ at any continuous position $(x, y) \in \mathbb{R}^2$

**Nearest-neighbor interpolation**
- Round coordinate $x$ to the closest integer $u_0$ and use $g(u_0)$ as the value $\hat{g}(x)$

$$\hat{g}(x) = g(u_0)$$

where $u_0 = \text{round}(x) = \lfloor x + 0.5 \rfloor$
Linear Interpolation

Estimated $\hat{g}(x)$ is the sum of the two closest samples $g(u_0)$ and $g(u_0 + 1)$, with $u_0 = \lfloor x \rfloor$,

$$\hat{g}(x) = g(u_0) + (x - u_0) \cdot (g(u_0 + 1) - g(u_0))$$

$$\hat{g}(x) = g(u_0) \cdot (1 - (x - u_0)) + g(u_0 + 1) \cdot (x - u_0)$$

Result is a piecewise linear function
\[ \hat{g}(x_0) = [\text{Sinc} \ast g](x_0) = \sum_{u=-\infty}^{\infty} \text{Sinc}(x_0 - u) \cdot g(u) \]

\( \ast \) is the linear convolution operator.

Ideal interpolation of \( g(u) \) at position \( x_0 \) involves \textit{infinitely many} values of \( g(u) \).
Interpolation by convolution

In general, express interpolation as a convolution of discrete function $g(u)$ with continuous interpolation kernel $w(x)$ as:

$$(\hat{g})(x_0) = [w * g](x_0) = \sum_{u=-\infty}^{\infty} w(x_0 - u) \cdot g(u).$$

Convolution kernels for nearest-neighbor interpolation $w_{nn}(x)$ and linear interpolation $w_{lin}(x)$
Cubic Interpolation

(a) Function $w_{cub}(x, a)$ with parameter $a = 0.25$ (dashed curve), $a = 1$ (continuous curve), and $a = 1.75$ (dotted curve). (b) Sinc function.

For $a = 1$:

$$w_{cub}(x) = \begin{cases} 
|x|^3 - 2|x|^2 + 1 & \text{for} \quad 0 \leq |x| < 1 \\
|x|^3 + 5|x|^2 - 8|x| + 4 & \text{for} \quad 1 \leq |x| < 2 \\
0 & \text{for} \quad |x| \geq 2
\end{cases}$$
"ideal" (low-pass filter) interpolation requires a two-dimensional Sinc function:

$$\text{SINC}(x, y) = \frac{\sin(\pi x)}{\pi x} \cdot \frac{\sin(\pi y)}{\pi y}$$

(a) $\text{SINC}(x, y)$ and (b) nearest-neighbor kernel $W_{nn}(x, y)$ for $-3 \leq x, y \leq 3$
Nearest-neighbor interpolation in 2D

\[ \hat{I}(x_0, y_0) = I(u_0, v_0) \]
with
\[ u_0 = \text{round}(x_0) = \lfloor x_0 + 0.5 \rfloor \]
\[ v_0 = \text{round}(y_0) = \lfloor y_0 + 0.5 \rfloor \]

Bilinear interpolation The 2D counterpart to the linear interpolation is the so-called *bilinear* interpolation.

(a) Original. Image enlargement (8x), (b) nearest neighbor, (c) bilinear interpolation.
Bilinear Interpolation

For \((x_0, y_0)\). (a) E and F are computed by linear interpolation in the horizontal direction. E, F are interpolated in the vertical direction.

\[
A = I(u_0, v_0) \quad C = I(u_0, v_0 + 1) \\
B = I(u_0 + 1, v_0) \quad D = I(u_0 + 1, v_0 + 1) \\
u_0 = \lfloor x_0 \rfloor \quad v_0 = \lfloor y_0 \rfloor
\]

\[
E = A + (x_0 - u_0) \cdot (B - A) \\
F = C + (x_0 - u_0) \cdot (D - C)
\]
Bilinear Interpolation

Using the vertical distance $b = y_0 - v_0$:

$$\hat{I}(x_0, y_0) = G = E + (y_0 - v_0) \cdot (F - E) = E + b \cdot (F - E)$$

$$= (a - 1)(b - 1)A + a(1 - b)B + (1 - a)bC + abD$$

Corresponding 2D kernel $W_{bil}(x, y)$ is the product of the two one-dimensional kernels $w_{lin}(x)$ and $w_{lin}(y)$,

$$W_{bil}(x, y) = \begin{cases} 
1 - x - y - x \cdot y & \text{for } 0 \leq |x|, |y| < 1 \\
0 & \text{otherwise}
\end{cases}$$
Interpolation

Original

(a) Nearest Neighbor

(b)

Bilinear

(c) Bicubic

(d)
Geometric Operations

Aliasing caused by image contraction.

Original

Bilinear

Bicubic

Nearest Neighbor
Zooming Example

FIGURE

Top row: images zoomed from 128 × 128, 64 × 64, and 32 × 32 pixels to 1024 × 1024 pixels, using nearest neighbor gray-level interpolation. Bottom row: same sequence, but using bilinear interpolation.
Zero order interpolation (zero order hold)

\[ \hat{f}_c(x, y) = f(n_1, n_2) \]

\[ n_1 = \text{Round to int} \left( \frac{x}{T_1} \right) \quad n_2 = \text{Round to int} \left( \frac{y}{T_2} \right) \]

This is, nearest pixel

- Typical application: Zoom by factor of two.
- Zero order interpolation: pixel replication.
You can also do this with mask:

- Take $n_1 \times n_2$ image:

\[
\begin{pmatrix}
\vdots & \vdots & \vdots & \vdots & \vdots \\
\vdots & \vdots & \vdots & \vdots & \vdots \\
\end{pmatrix}
\]

- Interlace with zeros:

\[
\begin{pmatrix}
\vdots & 0 & \vdots & 0 & \vdots & 0 & \vdots & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\vdots & 0 & \vdots & 0 & \vdots & 0 & \vdots & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\vdots & 0 & \vdots & 0 & \vdots & 0 & \vdots & 0 \\
\end{pmatrix}
\]

- Convolve with

\[
H = \begin{pmatrix} 1 & 1 \\ 1 & 1 \end{pmatrix}
\]
Can also do by a 2 zooming (with bilinear interpolation) as convolution create a $2N_1 \times 2N_2$ 0-interlaced image.

\[
\begin{pmatrix}
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 \\
\end{pmatrix}
\]

Convolve with the kernel:

\[
H = \begin{pmatrix}
\frac{1}{4} & \frac{1}{2} & \frac{1}{4} \\
\frac{1}{2} & 1 & \frac{1}{2} \\
\frac{1}{4} & \frac{1}{2} & \frac{1}{4} \\
\end{pmatrix}
\]
Another way to do higher order interpolation:

- Interlace image with \( p \) 0’s

i.e. \( p = 2 \)

\[
\begin{pmatrix}
\times & \times & \times \\
\times & \times & \times \\
\times & \times & \times \\
\end{pmatrix}
\]

\[
\begin{pmatrix}
\times & 0 & 0 & \times & 0 & 0 & \times & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\times & 0 & 0 & \times & 0 & 0 & \times & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\times & 0 & 0 & \times & 0 & 0 & \times & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\
\end{pmatrix}
\]
Convolve new interlaced image with

\[ H = \begin{pmatrix} 
1/4 & 1/2 & 1/2 \\
1/2 & 1 & 1/2 \\
1/4 & 1/2 & 1/4 \\
\end{pmatrix} \]

\( p \) times.

This gives \( p^{th} \) order interpolation.
Interpolation

Images with a Billion Pixels
Images with a Billion Pixels

Why is there no gigapixel camera today?
Is it image sensor resolution?

Assume 1 micron pixels (Fife et al 08)

10 megapixel sensor
Is it image sensor resolution?

1 gigapixel sensor

25 mm

40 mm
Defining Optical Resolution

Spatial Resolution

$\delta$: minimum resolvable spot size
Is resolution limited by diffraction

\[ \delta_d \approx \frac{\lambda F}{\#} \]

Diffraction Spot Size
Compromise: trading off light

10^9 pixels

Resolution

Telephoto
F/10 1000mm FL

SLR Lens
F/5 125mm FL

Wide Angle
F/3 27mm FL

Microscope
F/1 1mm FL

400 mm sensor

Scale (M)

R_{tradeoff}
(Lohmann '89)
Compromise: trading off light

The F/22 90° FOV Assymagon Lens

Graham Flint courtesy of Wired.com
Proposed solution: Computational Imaging

Reduce complexity with computations

(Cathey and Dowski '96)
(Robinson et al. '09)
(Guichard et al. ‘09)
(Cossairt and Nayar ’10)
Proposed solution: Computational Imaging
A ball lens gigapixel camera

Sensor Array

F/4 Ball Lens

75 mm

100 mm

75 mm

15 x 15 Array of 5Mpix ½” Lumenera Sensors
Proof of concept

Ball Lens

Sensor

Pan/Tilt Motor
Proof of concept: Image Quality

Deblurred
A single element design

Parallel effort by DARPA MOSAIC Program led by D. Brady
(Brady and Hagen ‘09)(Marks and Brady ‘10)
A single element design

- Ball Lens
- Sensor Array
- Lens Array
Still Life (1.7 Gigapixels)

URL: http://gigapan.org/gigapans/0dca576c3a040561b4371cf1d92c93fe/
URL: http://gigapan.org/gigapans/0dca576c3a040561b4371cf1d92c93fe/
New York and New Jersey Skyline (1.4 Gigapixels)

Statue of Liberty
(~2.32km)

Apartments
(~860m)

Person on boat
deck
(~860m)

Empire State Building
(~6.45km)

URL: http://gigapan.org/gigapans/7173ad0acace87100a3ca728d40a3772/
New York and New Jersey Skyline (1.4 Gigapixels)

Sailboat (~1.61km)
People on Boat (~860m)
Flag on Brooklyn Bridge (~2.19km)
Cars (~1.5km)

URL: http://gigapan.org/gigapans/7173ad0acace87100a3ca728d40a3772/