ELEG404/604: Digital Imaging & Photography

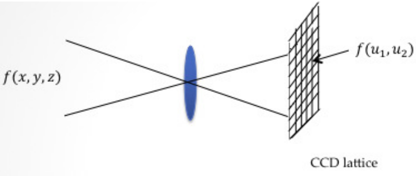
Gonzalo R. Arce

Department of Electrical and Computer Engineering
University of Delaware

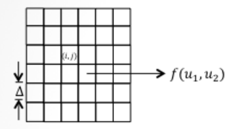
Chapter IV

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:

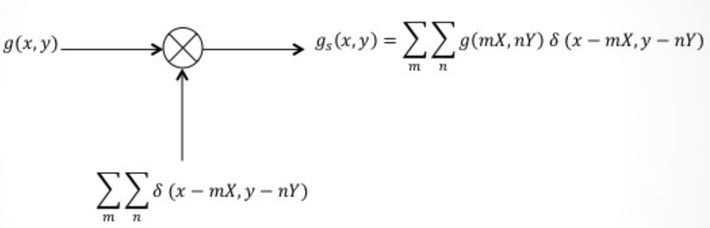


where charge couple devices (CCDs) consist on an array of detectors

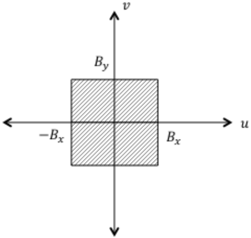


Common resolution range from $(256)^2 \longleftrightarrow (2000)^2$

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:



Assuming g(x,y) is band limited, $G(u,v) = 0$ for $u > B_x$ and $v > B_y$



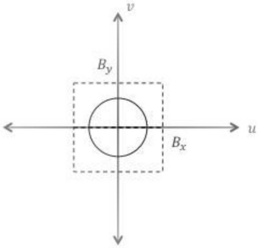
The Fourier transform of the sampled image is

$$\begin{aligned} F\{g_s(x,y)\} &= F\left\{g(x,y)\sum_m\sum_n \delta(x-mX, y-uY)\right\} \\ &= G(u,v) * \frac{1}{X}\frac{1}{Y}\sum_m\sum_n \delta\left(u-\frac{m}{X}, v-\frac{n}{Y}\right) \end{aligned}$$

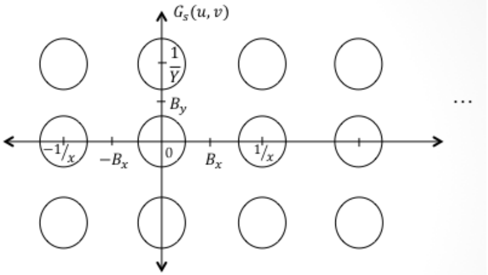
The Fourier transform of the sampled image is

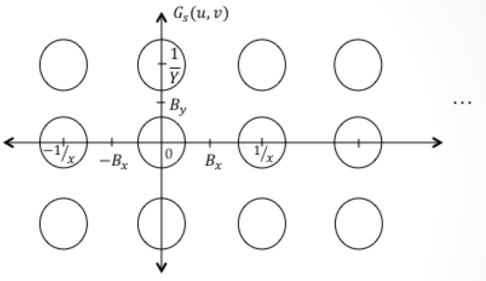
$$
\begin{aligned}
F\{g_s(x,y)\} &= F\left\{g(x,y)\sum_m\sum_n \delta(x-mX, y-uY)\right\} \\
&= G(u,v) * \frac{1}{X}\frac{1}{Y}\sum_m\sum_n \delta\left(u-\frac{m}{X}, v-\frac{n}{Y}\right) \\
&= \frac{1}{XY}\sum_m\sum_n G\left(u-\frac{m}{X}, v-\frac{n}{Y}\right) \\
&= \tfrac{1}{XY}\left(\mathrm{rep}_{\frac{1}{X}\frac{1}{Y}}(G(u,v)\right)
\end{aligned}
$$

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,

$$
\begin{aligned}
\hat{g}(x,y) &= g_s(x,y) * \text{sinc}[\tfrac{x}{X}, \tfrac{y}{Y}] \\
&= \left[\sum_m \sum_n g(mX, nY)\delta(x - mX, y - uY)\right] * \text{sinc}\left[\frac{x}{X}, \frac{y}{Y}\right] \\
\hat{g}(x,y) &= \sum_m \sum_n g(mX, nY)\text{sinc}\left(\frac{x - mX}{X}, \frac{y - nY}{Y}\right)
\end{aligned}
$$

Hence,

$$\begin{aligned}\hat{g}(x,y) &= g_s(x,y) * \mathrm{sinc}[\tfrac{x}{X},\tfrac{y}{Y}] \\ &= \left[\sum_m\sum_n g(mX,nY)\delta(x-mX,y-uY)\right] * \mathrm{sinc}\left[\frac{x}{X},\frac{y}{Y}\right] \\ \hat{g}(x,y) &= \sum_m\sum_n g(mX,nY)\mathrm{sinc}\left(\frac{x-mX}{X},\frac{y-nY}{Y}\right)\end{aligned}$$

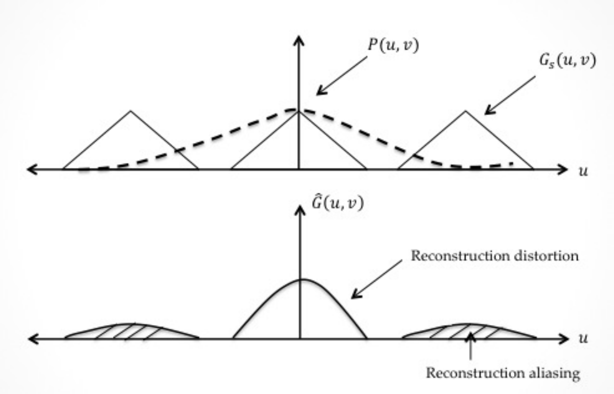
The ideal LPF is difficult to obtain, hence, other filters are generally designed. For instance, if we are to obtain a continuous image by proyecting into a CRT display, we are effectively replacing the 2-D function by a Gaussian function

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

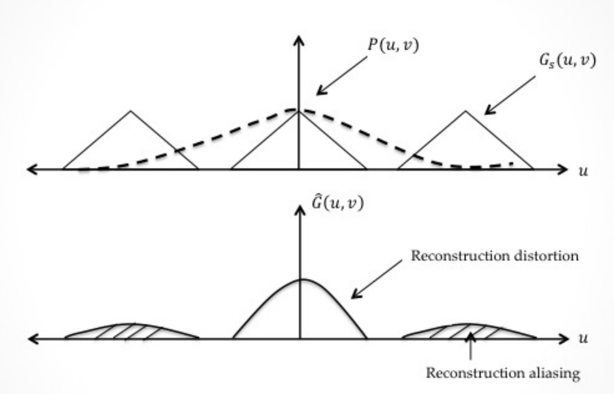
Hence the reconstructed image is:

$$\hat{g}(x,y) = \sum_m\sum_n g(mX,nY)\left(\frac{1}{2\pi\sigma^2}\right)\exp\left[\frac{-(x-mX)^2-(y-nY)^2}{2\sigma^2}\right]$$

The effect is the introduction of aliasing. To illustrate, consider a slice of $G_s(u,v)$ and $\hat{G}(u,v)$.

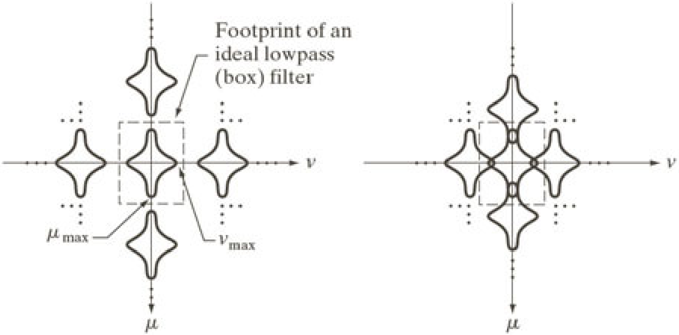


The effect is the introduction of aliasing. To illustrate, consider a slice of $G_s(u,v)$ and $\hat{G}(u,v)$.



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 comercial applications.

Aliasing

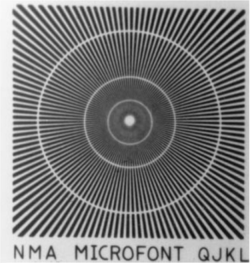


a b

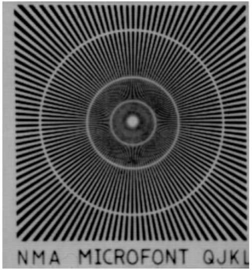
FIGURE 4.15 Two-dimensional Fourier transforms of (a) an over-sampled, and (b) under-sampled band-limited function.

Example of Aliasing

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

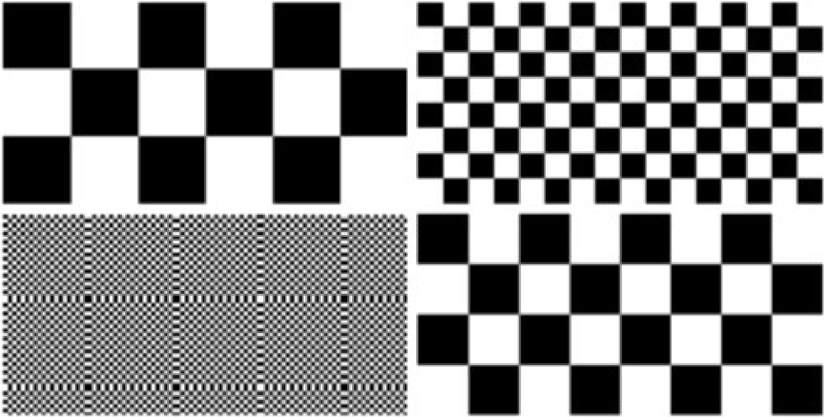


(a) Original image



(b) Sampled image

Aliasing



a b
c d

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

a b c

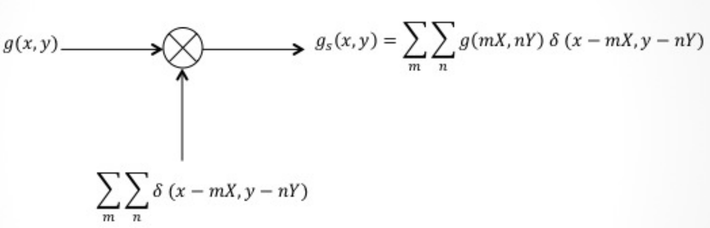
FIGURE 4.17 Illustration of aliasing on resampled images. (a) A digital image with negligible visual aliasing. (b) Result of resizing the image to 50% of its original size by pixel deletion. Aliasing is clearly visible. (c) Result of blurring the image in (a) with a 3×3 averaging filter prior to resizing. 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

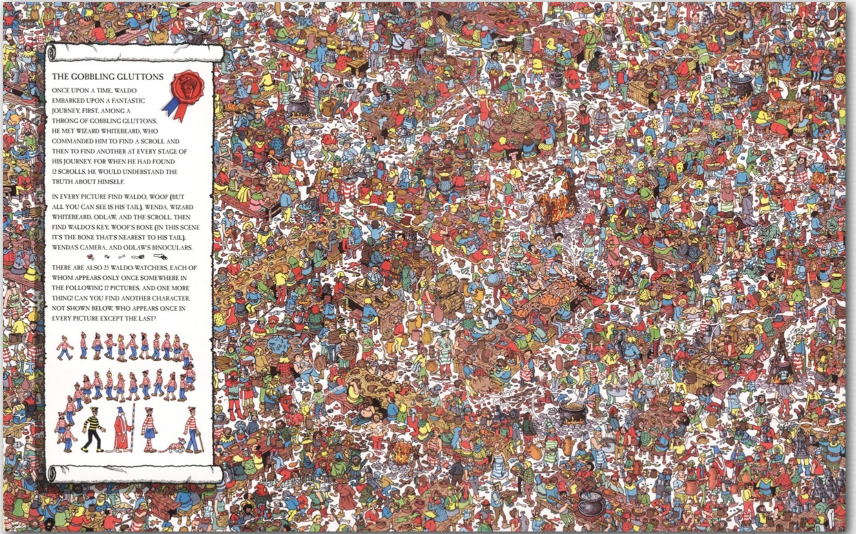
Courtesy of Scientific Volume Imaging - http://www.svi.nl/antialiasing

Aliasing in Photography

- a lens creates a focused image on the sensor
- suppose the sensor measured this image at points on a 2D grid, but ignored the imagery between points?
  - a.k.a. *point sampling*



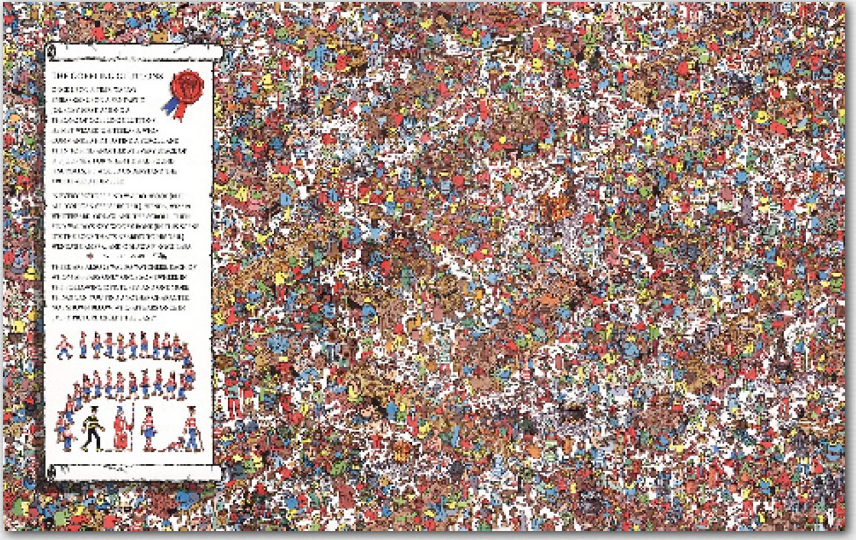
Simulation of Point Sampling



(Classic Media)

digital image, 1976 x 1240 pixels

Simulation of Point Sampling



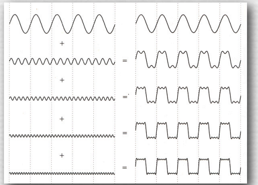
every 4^{th} pixel in x and y, then upsized using pixel replication

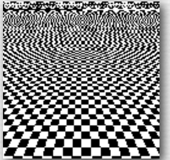
Prefiltering to Avoid Aliasing

- before sampling, remove (or at least attenuate) sine waves of frequency greater than half the sampling rate

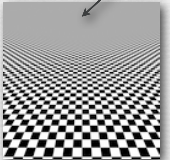
$$f_{cutoff} < \frac{1}{2} f_{sampling}$$

replace removed waves with their average intensity (gray in this case)

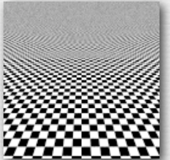




unfiltered



prefiltered



partially pre-filtered

Method for Prefiltering

- method #1: frequency domain
  1. convert image to frequency domain
  2. remove frequencies above f_{cutoff} (replace with gray)
  3. convert back to spatial domain
  4. perform point sampling as before

  - conversions are slow
  - not clear how to apply this method to images as they enter a camera

- method #2: spatial domain
  1. blur image using *convolution*
  2. perform point sampling as before

  - direct and faster
  - equivalent to method #1 (proof is beyond scope of this course)

9

Convolution in 1D

- replace each input value with a weighted sum of itself and its neighbors, with weights given by a filter function

$$f[x] * g[x] = \sum_{k=-\infty}^{\infty} f[k] \cdot g[x-k]$$

input signal $f[x]$	1	3	0	4	2	1
filter g[x]			2	1		
output $f[x] * g[x]$						

Convolution in 1D

- replace each input value with a weighted sum of itself and its neighbors, with weights given by a filter function

$$f[x] * g[x] = \sum_{k=-\infty}^{\infty} f[k] \cdot g[x-k]$$

input signal $f[x]$

1	3	0	4	2	1

1	2

notice that the filter gets flipped when applied

output $f[x] * g[x]$

7					

1

Convolution in 1D

- replace each input value with a weighted sum of itself and its neighbors, with weights given by a filter function

$$f[x] * g[x] \;=\; \sum_{k=-\infty}^{\infty} f[k] \cdot g[x-k]$$

input signal $f[x]$

1	3	0	4	2	1

1	2

output $f[x] * g[x]$

7	3				

2

Convolution in 1D

- replace each input value with a weighted sum of itself and its neighbors, with weights given by a filter function

$$f[x] * g[x] = \sum_{k=-\infty}^{\infty} f[k] \cdot g[x-k]$$

input signal $f[x]$

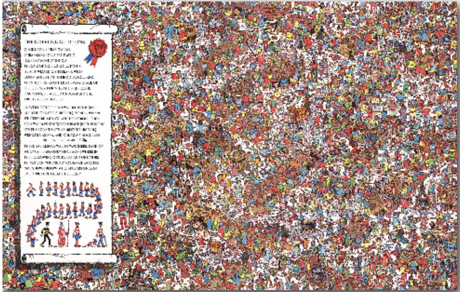
1	3	0	4	2	1

1	2

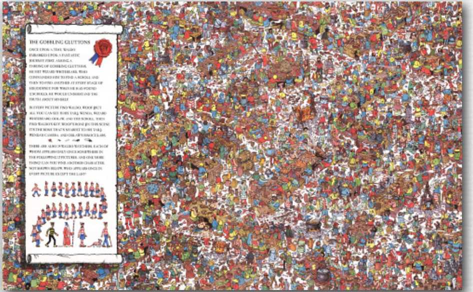
output $f[x] * g[x]$

7	3	8			

Prefiltering Reduce Aliasing



every 4^{th} pixel in x and y



convolved by 4×4 pixel rect,
then sampled every 4th pixel

Prefiltering & Sampling in Photography

- photography consists of convolving the focused image by a 2D rect filter, then sampling on a 2D grid
  - each point on this grid is called a *pixel*
- if convolution is followed by sampling, you only need to compute the convolution at the sample positions
  - for a rect filter of width equal to the sample spacing, this is equivalent to measuring the average intensity of the focused image in a grid of abutting squares
  - this is exactly what a digital camera does
- the rect width <u>should</u> roughly match the pixel spacing
  - much narrower would leave aliasing in the image
  - much wider would produce excessive blurring in the image

7

Aliasing

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

Aliasing

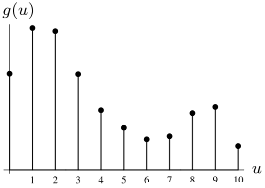
Interpolation

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

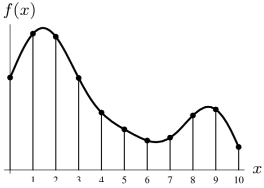
Nearest-neighbor interpolation

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

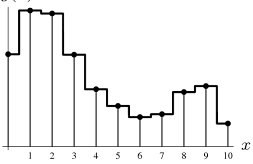
$$\begin{aligned} \hat{g}(x) &= g(u_0) \\ \text{where } u_0 &= \text{round}(x) = \lfloor x + 0.5 \rfloor \end{aligned}$$



(a)



(b)

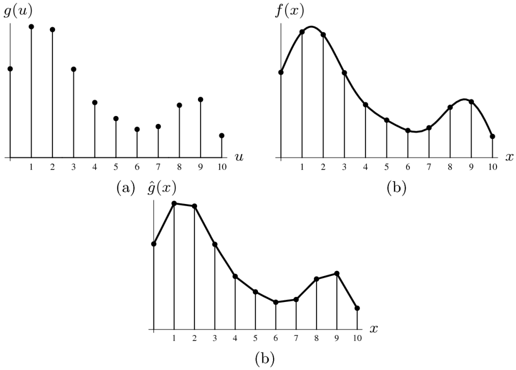


(a)

Linear Interpolation

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

$$
\begin{aligned}
\hat{g}(x) &= g(u_0) + (x - u_0)\cdot(g(u_0+1) - g(u_0)) \\
&= g(u_0)\cdot(1-(x-u_0)) + g(u_0+1)\cdot(x-u_0)
\end{aligned}
$$

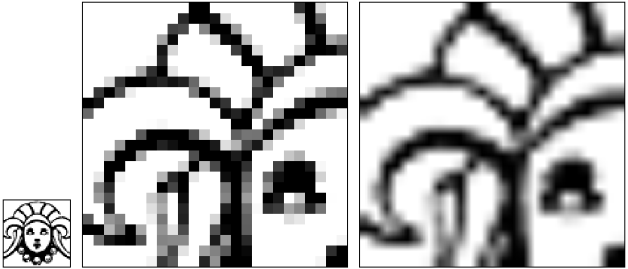


Result is a piecewise linear function

Nearest-neighbor interpolation in 2D

$$
\begin{aligned}
\hat{\mathsf{I}}(x_0, y_0) &= \mathsf{I}(u_0, v_0) \\
\text{with } u_0 &= \text{round}(x_0) = \lfloor x_0 + 0.5 \rfloor \\
v_0 &= \text{round}(y_0) = \lfloor y_0 + 0.5 \rfloor
\end{aligned}
$$

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

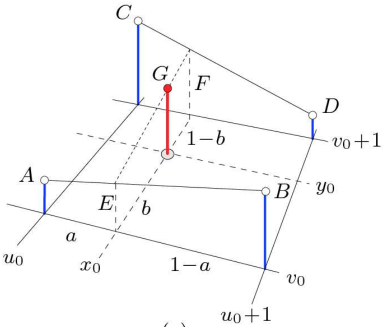


(a) (b) (c)

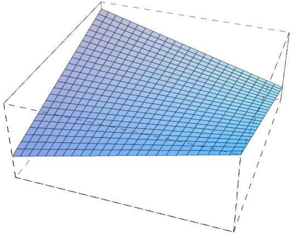
(a) Original. Image enlargement (8x), (b) nearest neighbor, (c) bilinear interpolation.

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)



(b)

$A = I(u_0, v_0)$
$B = I(u_0+1, v_0)$
$u_0 = \lfloor x_0 \rfloor$

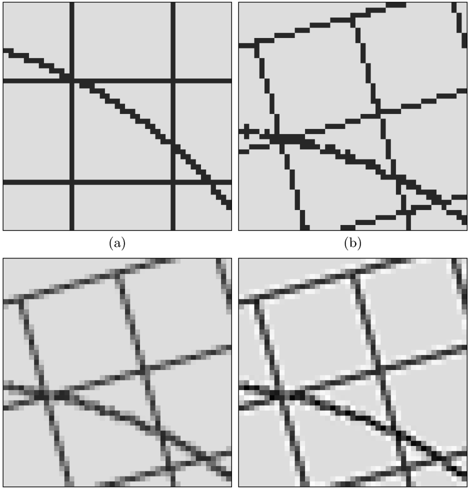
$C = I(u_0, v_0+1)$
$D = I(u_0+1, v_0+1)$
$v_0 = \lfloor y_0 \rfloor$

$$
\begin{aligned}
E &= A + (x_0 - u_0)\cdot(B - A) \\
E &= A + a\cdot(B - A) \\
F &= C + (x_0 - u_0)\cdot(D - C) \\
F &= C + a\cdot(D - C)
\end{aligned}
$$

Interpolation

Original

Nearest Neighbor



(a) (b)

Bilinear

Bicubic

(c) (d)

Upsizing by 16:1

nearest neighbor
(a.k.a. pixel replication)

bilinear

bicubic

Downsizing by 1:6

nearest neighbor
(point sampling)

aliasing!

bicubic

Upsizing by 16:1

nearest neighbor
(a.k.a. pixel replication)

bilinear

bicubic

Zero order interpolation (zero order hold)

$$\hat{\mathsf{f}}_c(x, y) = f(n_1, n_2)$$

$$n_1 = \text{Round to int}\left(\frac{x}{T_1}\right) \qquad n_2 = \text{Round to int}\left(\frac{y}{T_2}\right)$$

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:

$$\begin{pmatrix} - & - & - & - \\ - & - & - & - \\ - & - & - & - \end{pmatrix}$$

- Interlace with zeros:

$$\begin{pmatrix} - & 0 & - & 0 & - & 0 & - & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ - & 0 & - & 0 & - & 0 & - & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ - & 0 & - & 0 & - & 0 & - & 0 \end{pmatrix}$$

- 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.

$$\begin{pmatrix} \cdot & 0 & \cdot & 0 & \cdot \\ 0 & 0 & 0 & 0 & 0 \\ \cdot & 0 & \cdot & 0 & \cdot \\ 0 & 0 & 0 & 0 & 0 \\ \cdot & 0 & \cdot & 0 & \cdot \end{pmatrix}$$

.

- Convolve with the kernel:

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

.

P^{th} Order Interpolation

Another way to do higher order interpolation:

- Interlace image with p $0's$
  i.e. $p = 2$

$$\begin{pmatrix} x & x & x \\ x & x & x \\ x & x & x \end{pmatrix}$$

.

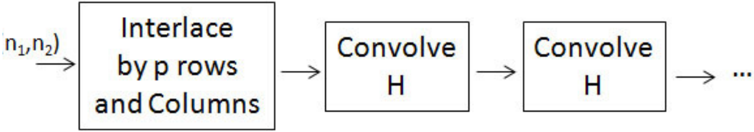
$$\begin{pmatrix} x & 0 & 0 & x & 0 & 0 & x & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ x & 0 & 0 & x & 0 & 0 & x & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ x & 0 & 0 & x & 0 & 0 & x & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \end{pmatrix}$$

.

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.

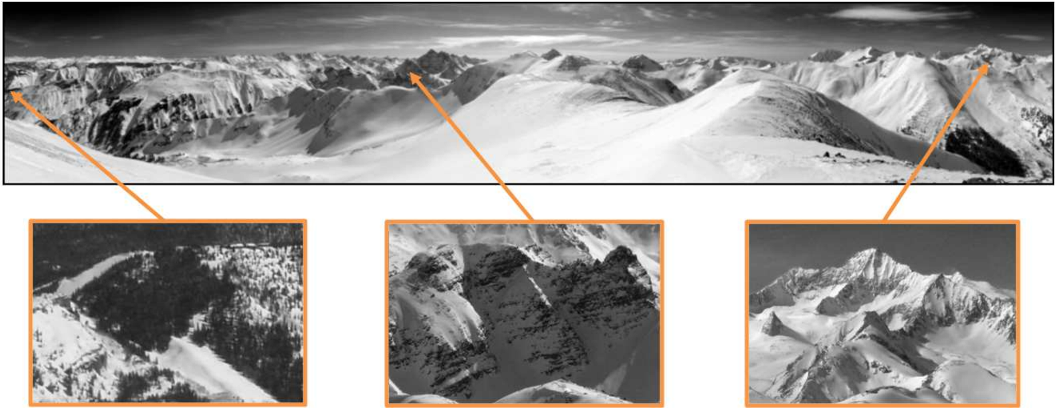
Interpolation

http:www.paris-26-gigapixels.comindex-en.html


Maximize


mize

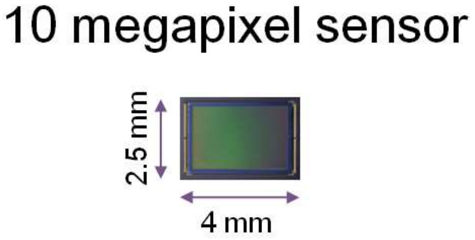
Images with a Billion Pixels



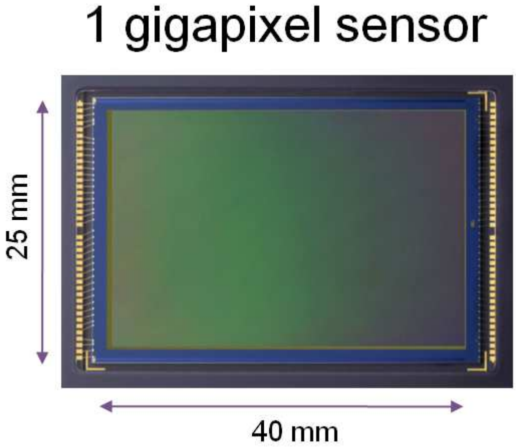
Why is there no gigapixel camera today?

Is it image sensor resolution?

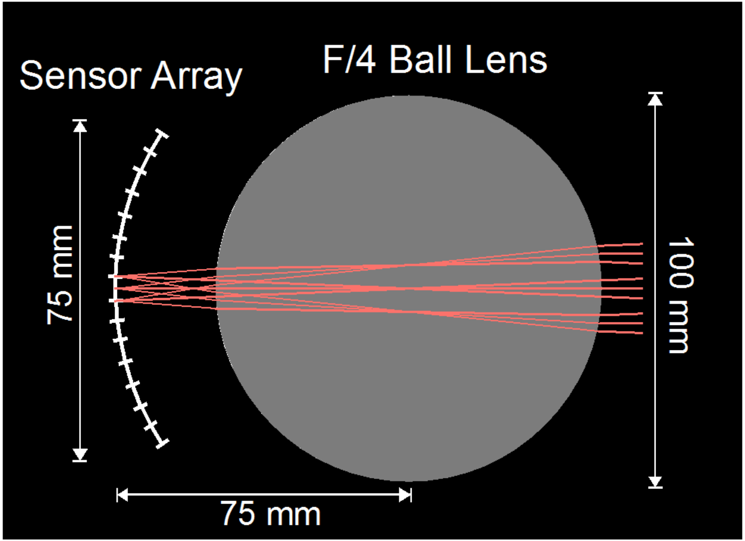
Assume 1 micron pixels (Fife et al 08)



Is it image sensor resolution?

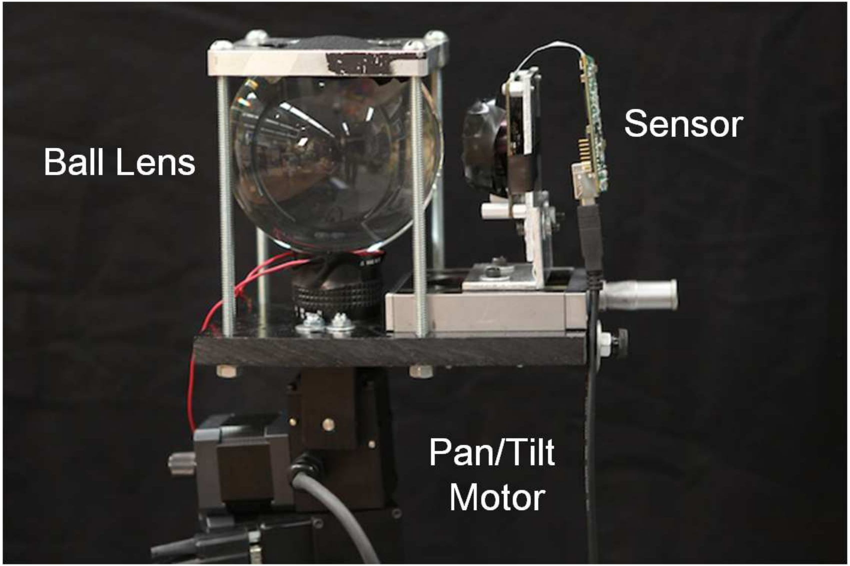


A ball lens gigapixel camera



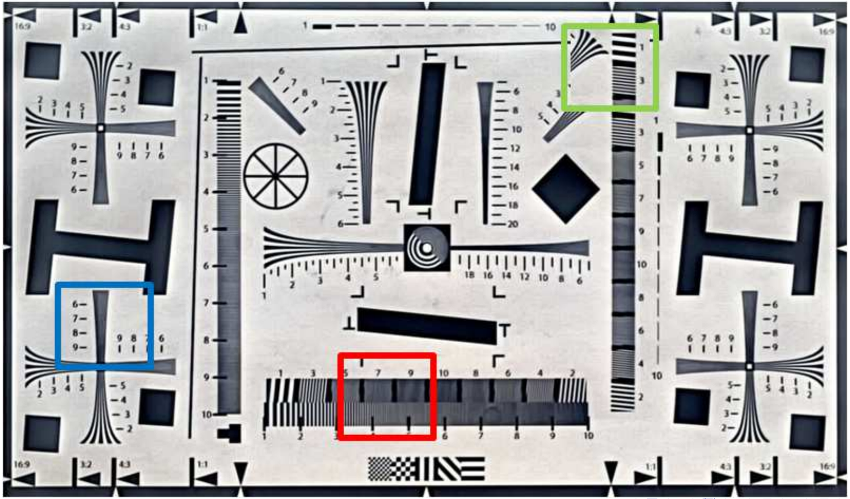
15 x 15 Array of 5Mpix ½" Lumenera Sensors

Proof of concept

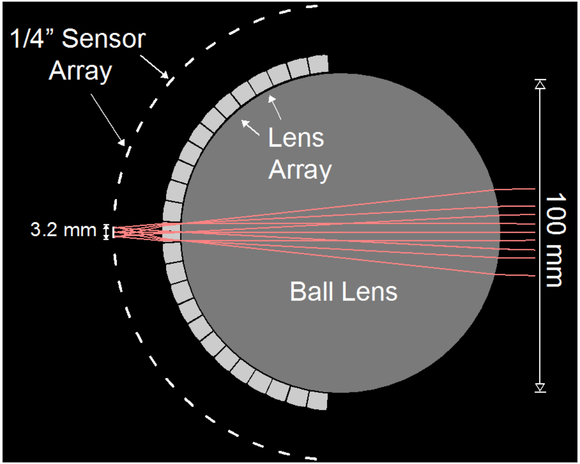


Proof of concept: Image Quality

Deblurred

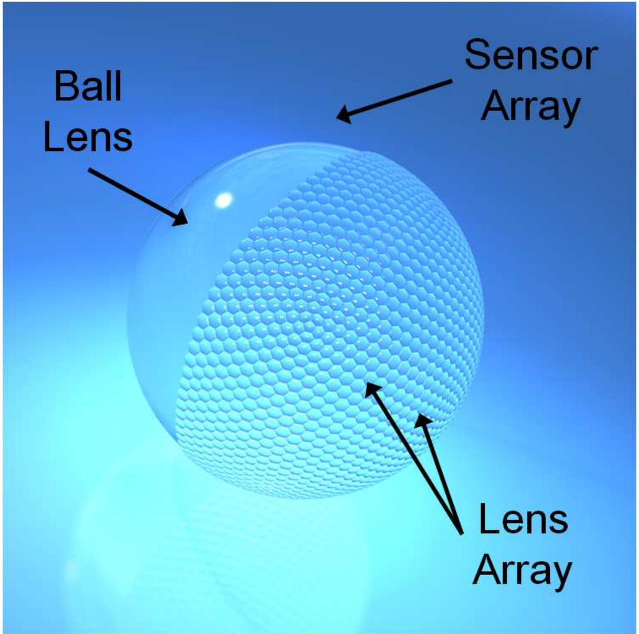


A single element design

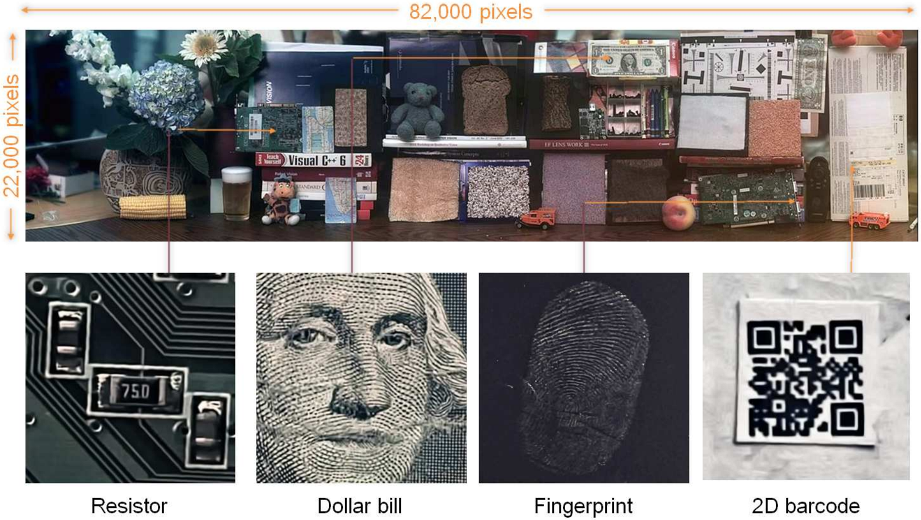


Parallel effort by DARPA MOSAIC Program led by D. Brady
(Brady and Hagen ‘09)(Marks and Brady ‘10)

A single element design



Still Life (1.7 Gigapixels)



URL: http://gigapan.org/gigapans/0dca576c3a040561b4371cf1d92c93fe/

Real flower

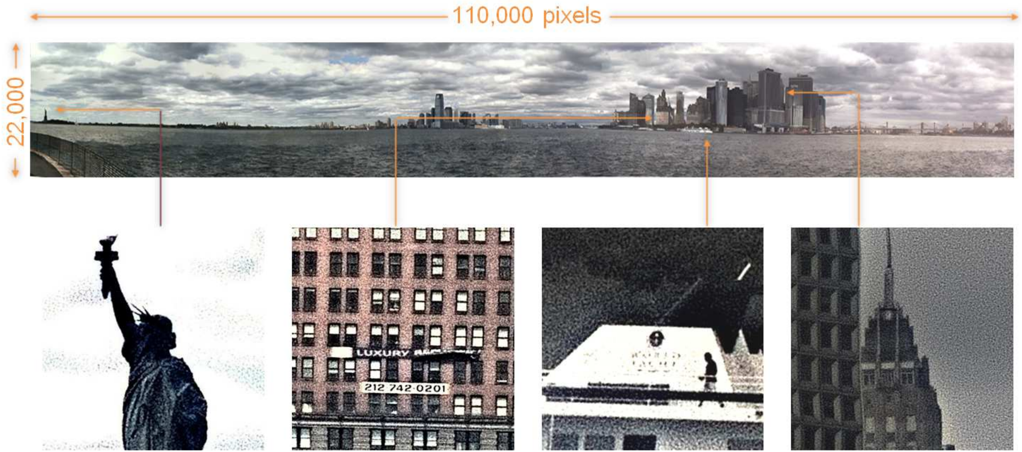
Plastic flower

Loofah

Fabric weave

URL: http://gigapan.org/gigapans/0dca576c3a040561b4371cf1d92c93fe/

New York and New Jersey Skyline (1.4 Gigapixels)



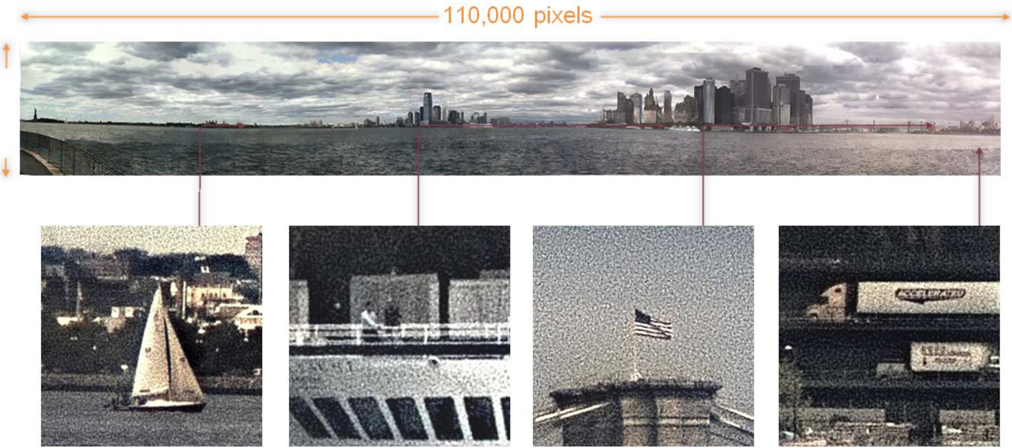
Statue of Liberty (~2.32km)

Apartments (~860m)

Person on boat deck (~860m)

Empire State Building (~6.45km)

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)