
Image processing: image resizing

© 1999, Denis Zorin

How computer images work?

Continuous real image

↓ *Digitization
(e.g. scanning)*

**square array of numbers
(abstract pixels)**

↓ *each physical pixel
covers an area*

display (physical pixels)

↓ *The eye blurs pixels into
continuous image*

perceived image

© 1999, Denis Zorin

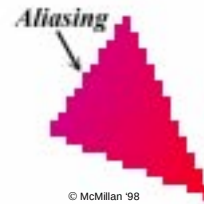
What can go wrong?

Pipeline: sample - process - reconstruct

All kinds of artifacts can appear

- jaggies
- alias patterns
- moire patterns
- temporal aliasing (wheels going wrong way)

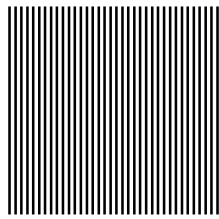
Question: how do we avoid all this?



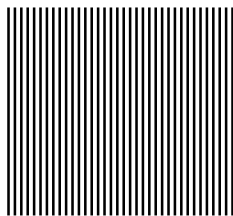
© McMillan '98

© 1999, Denis Zorin

Aliasing



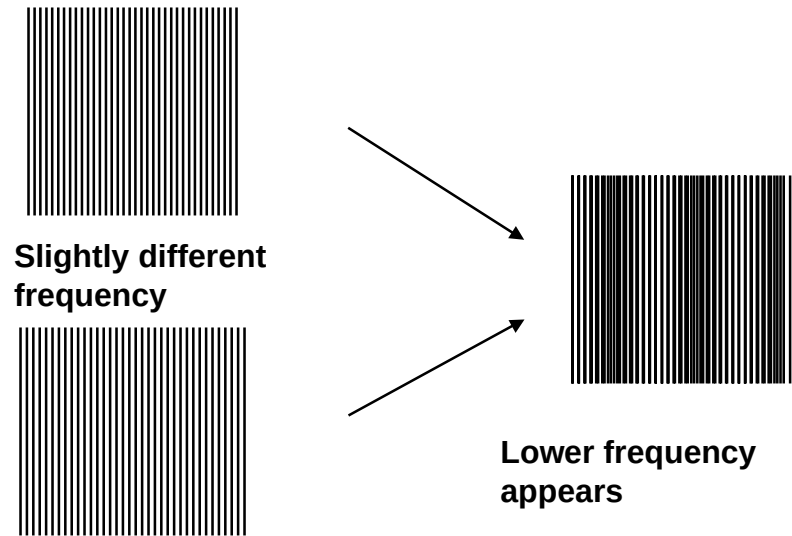
Slightly different
frequency



What can go wrong?

© 1999, Denis Zorin

Aliasing



© 1999, Denis Zorin

Shrinking

Naïve 1.5x shrinking : drop 2 out of 3

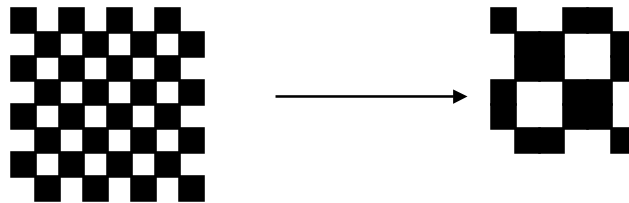


What do we get?

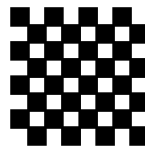
© 1999, Denis Zorin

Shrinking

Naïve 1.5x shrinking: drop 2 out of 3



But we want



(impossible, not enough samples)

© 1999, Denis Zorin

Shannon theorem

(Almost) any function or signal can be decomposed into a sum of sine waves with different frequencies:

$$f(t) = \sum_i a_i \sin(\omega_i t + \phi_i)$$

amplitude frequency phase

A function can be reconstructed from samples if the sampling frequency is at least twice the max. frequency in the decomposition of the function.

© 1999, Denis Zorin

General idea

Keep the part of the signal (image) we can represent
(low frequencies).

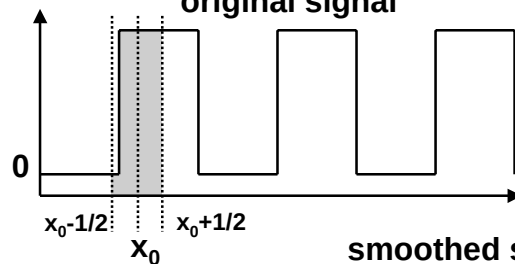
Eliminate the part of the signal (image) we cannot
represent (high frequencies).

Smooth slowly changing signals have lower high
frequency components, smoothing eliminates
high frequencies,

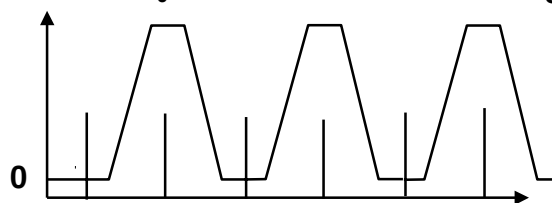
© 1999, Denis Zorin

Continuous smoothing

Smoothing can be done by local integration:
original signal



slide the integration
interval along the
x-axis

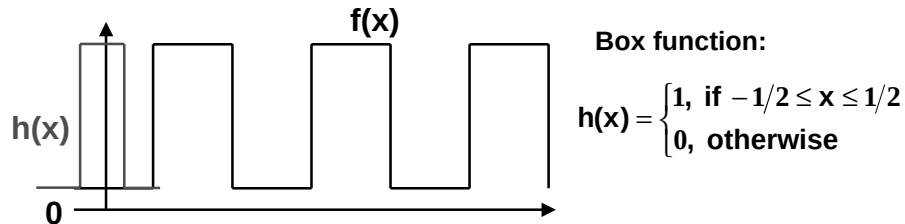


$$f^{\text{smoothed}}(x_0) = \int_{x_0 - 1/2}^{x_0 + 1/2} f(x) dx$$

© 1999, Denis Zorin

Continuous smoothing

Another way to look at the same integral:



$$f^{\text{smoothed}}(x_0) = \int_{x_0-1/2}^{x_0+1/2} f(x) dx = \int_{-\infty}^{\infty} h(x_0 - x) f(x) dx$$

Can replace box with some other bump-like function $h(x)$, resulting in better smoothing.

© 1999, Denis Zorin

Image shrinking

For simplicity, derive everything in 1D; shrinking of a two dimensional image is done in two steps: first in x direction, then in y direction.

Problem: after shrinking have too few pixels to represent all pixels of the original.

Solution: do local averaging; similar to the continuous case, but instead of integration do summation.

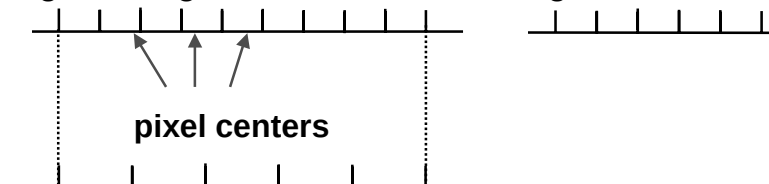
© 1999, Denis Zorin

Image shrinking

Shrinking by a factor $a < 1$:

original image, size w

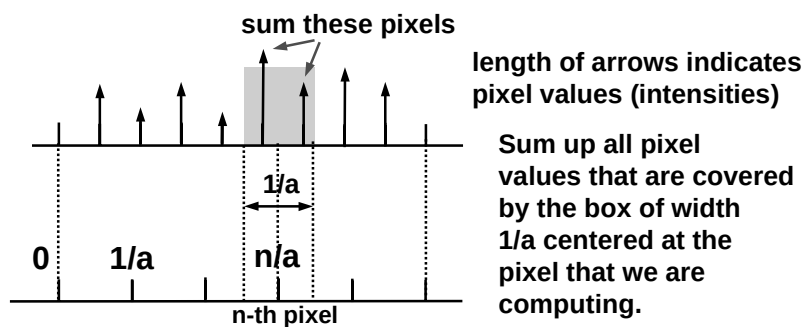
target size, aw



think about pixels of the target as “fat pixels”;
the size of a fat pixel is $1/a$; the the size of a the
(rescaled) target image is $aw(1/a) = w =$ size of the
original (but the pixel size is different!)

© 1999, Denis Zorin

Image shrinking



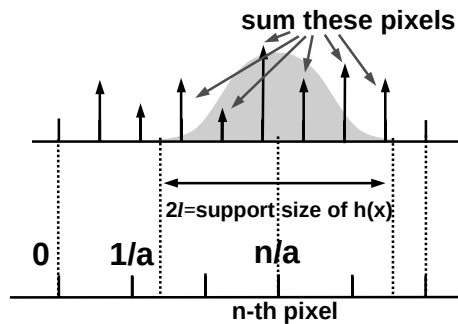
If box function of width 1 is $h(x)$, then:

- box function of width $1/a$ is $h(ax)$
- box function of width $1/a$ centered at n/a is $h(a(x-n/a)) = h(ax-n)$
- add a scale factor a , so that we do not get high intensities:

$$p^{\text{shrunked}}[n] = \sum_i h(ai - n)p[i]$$

© 1999, Denis Zorin

Image shrinking



Can replace the box by other functions (filters) that result in better images. The formula is the same:

$$p^{\text{shrunk}}[n] = \sum_i h(ai - n)p[i]$$

We have to sum only over pixels for which $h(ai - n)$ is not zero. If $h(x)$ is zero outside $[-l, l]$, The range for i is determined from

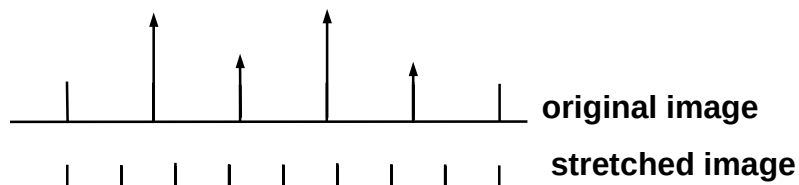
$$-l \leq ai - n \leq l$$

© 1999, Denis Zorin

Image stretching

Stretching by the factor of $a > 1$.

Same approach: make images the same size, use "tiny pixels" of size $1/a$.

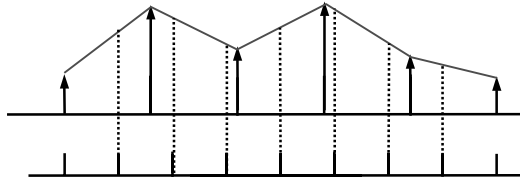


How do we determine values for points between the original pixels? Need to interpolate, that is, find a continuous function coinciding with the original at discrete values.

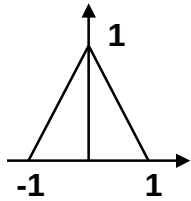
© 1999, Denis Zorin

Interpolation

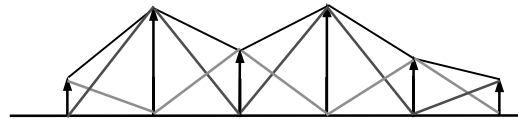
Simplest interpolation: linear.



How do we write expression for the interpolating function? Use hat functions (one of possible “bumps”).



© 1999, Denis Zorin



Put a hat of height $p[n]$ centered at n .
Add all hats.

Interpolation

Hat functions are linear on integer intervals; when we sum them we get a linear function with values $p[n]$ at integers:

$$f(x) = \sum_i \text{hat}(x - i)p[i]$$

Can use other functions instead of $\text{hat}(x)$:

Just need:

- $h(0) = 1$
- $h(n) = 0$ for n not equal to zero
- interpolating function: $f(x) = \sum_i h(x - i)p[i]$

© 1999, Denis Zorin

Image stretching

Now we only need to resample the interpolating function at “tiny pixel” intervals $1/a$:

$$p[n] = \sum_i h\left(\frac{n}{a} - i\right)p[i]$$

Again, the interval over which to sum is determined by the interval $[-1,1]$, on which the $h(t)$ is not zero:

$$-1 \leq \frac{n}{a} - i \leq 1$$

© 1999, Denis Zorin

Filter quality

Why some filters $h(t)$ are better than others?

Recall that we want to eliminate high frequency components, and keep low-frequency components.

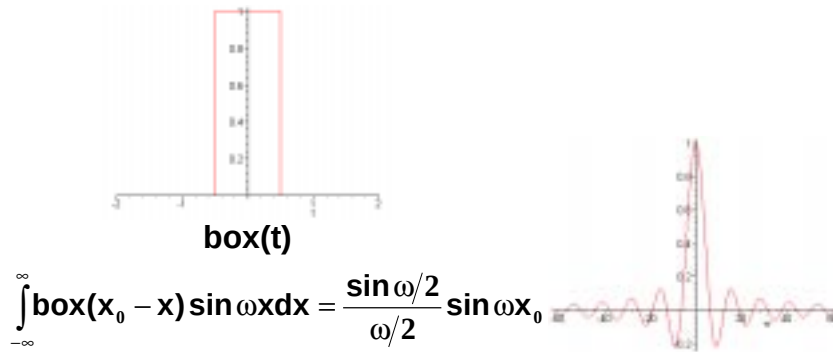
We can evaluate the filter by looking what it does to sine waves of different frequencies (frequency response). For an ideal filter,

$$\int_{-\infty}^{\infty} h(x_0 - x) \sin(\omega x) dx$$

should be zero for frequencies $> \text{sampling freq.}/2$
and 1 for frequencies $< \text{sampling freq.}/2$

© 1999, Denis Zorin

Box filter response



We have got the same wave back, but with different amplitude. The higher the frequency, the lower the amplitude. But the amplitude decreases as $1/\omega$, which is not very fast.

© 1999, Denis Zorin

Ideal filter

A function called sinc has the ideal response, that is, for low frequencies 1 for high frequencies zero:

$$\text{sinc}(x) = \frac{\sin \pi x}{\pi x}$$

$$\int_{-\infty}^{\infty} \text{sinc}(x_0 - x) \sin \omega x dx = \sin \omega x_0 \quad \text{if } \omega < \pi$$

$$\int_{-\infty}^{\infty} \text{sinc}(x_0 - x) \sin \omega x dx = 0 \quad \text{if } \omega > \pi$$

Why not use it? It is not zero anywhere, except a few points -- we would have to do infinite summations!

© 1999, Denis Zorin

Practical filters

Try to approximate sinc. At the same time, keep short the interval where the filter is not zero.

In addition to box and hat, here is a couple of useful filters:

Lanczos filter:

$$h(x) = \begin{cases} \text{sinc}(x) \text{sinc}(x/3), & \text{if } |x| < 3 \\ 0 & \text{otherwise.} \end{cases}$$

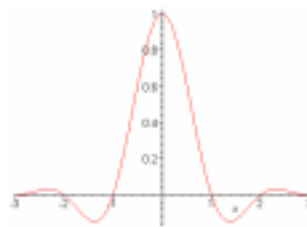
Mitchell filter:

$$h(x) = \frac{1}{6} \begin{cases} 7|x|^3 - 12|x|^2 + 16/3, & \text{if } |x| < 1 \\ -7/3|x|^3 + 12|x|^2 - 20|x| + 32/3, & \text{if } 1 \leq |x| < 2 \\ 0 & \text{otherwise} \end{cases}$$

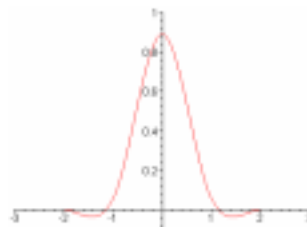
© 1999, Denis Zorin

Practical filters

Lanczos filter, $l=3$:



Mitchell filter, $l=2$:



© 1999, Denis Zorin