Obdelava digitalnih slik

Metode obdelave digitalnih slik

- Quantization
  - Uniform Quantization
  - Random dither
  - Ordered dither
  - Floyd-Steinberg dither
- Pixel operations
  - Add random noise
  - Add luminance
  - Add contrast
  - Add saturation
- Filtering
  - Blur
  - Detect edges
- Warping
  - Scale
  - Rotate
  - Warp
- Combining
  - Composite
  - Morph

Thomas A. Funkhouser, *Princeton Graphics Group*

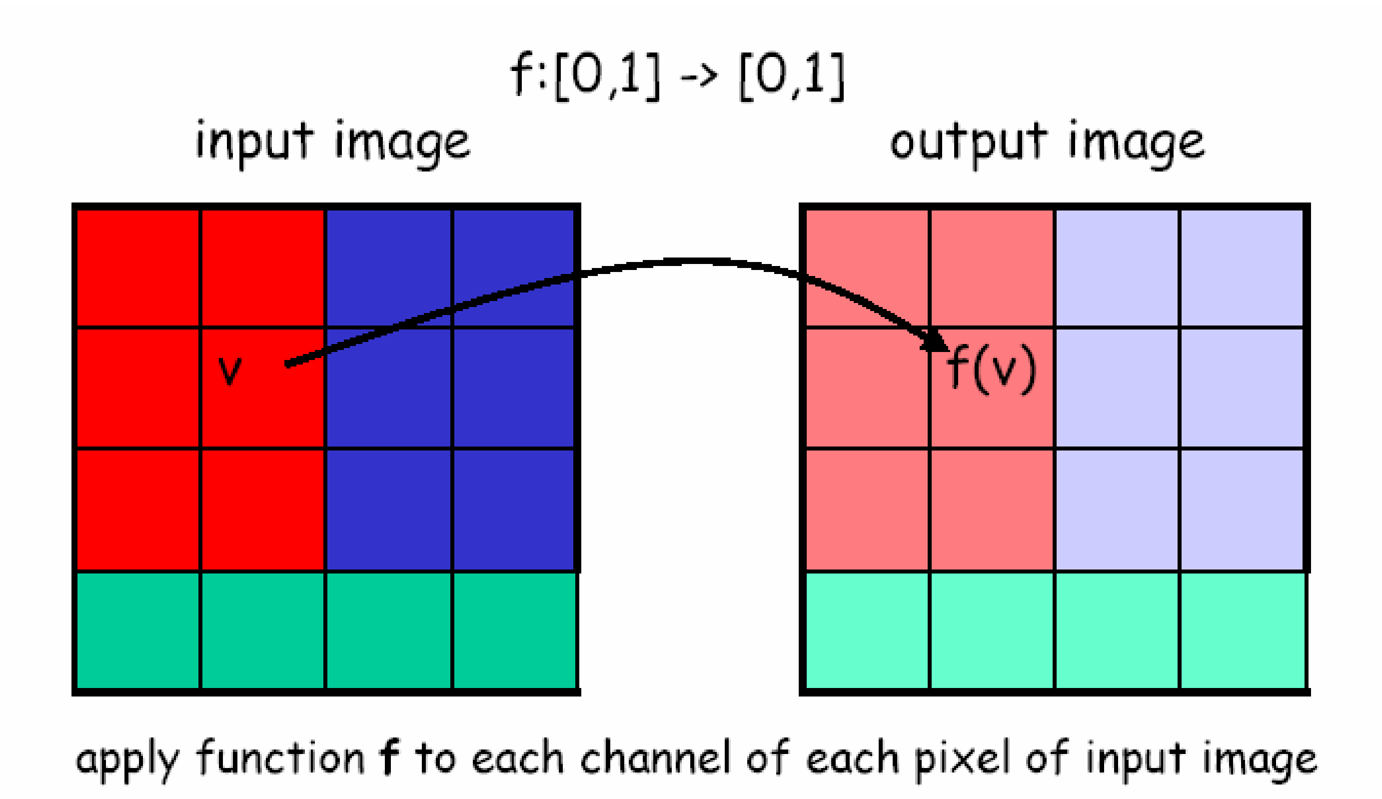
types of techniques

- simple pixel modification
- interpolation/extrapolation
- compositing
- convolution
- dithering
- warping
- morphing
- misc. effects

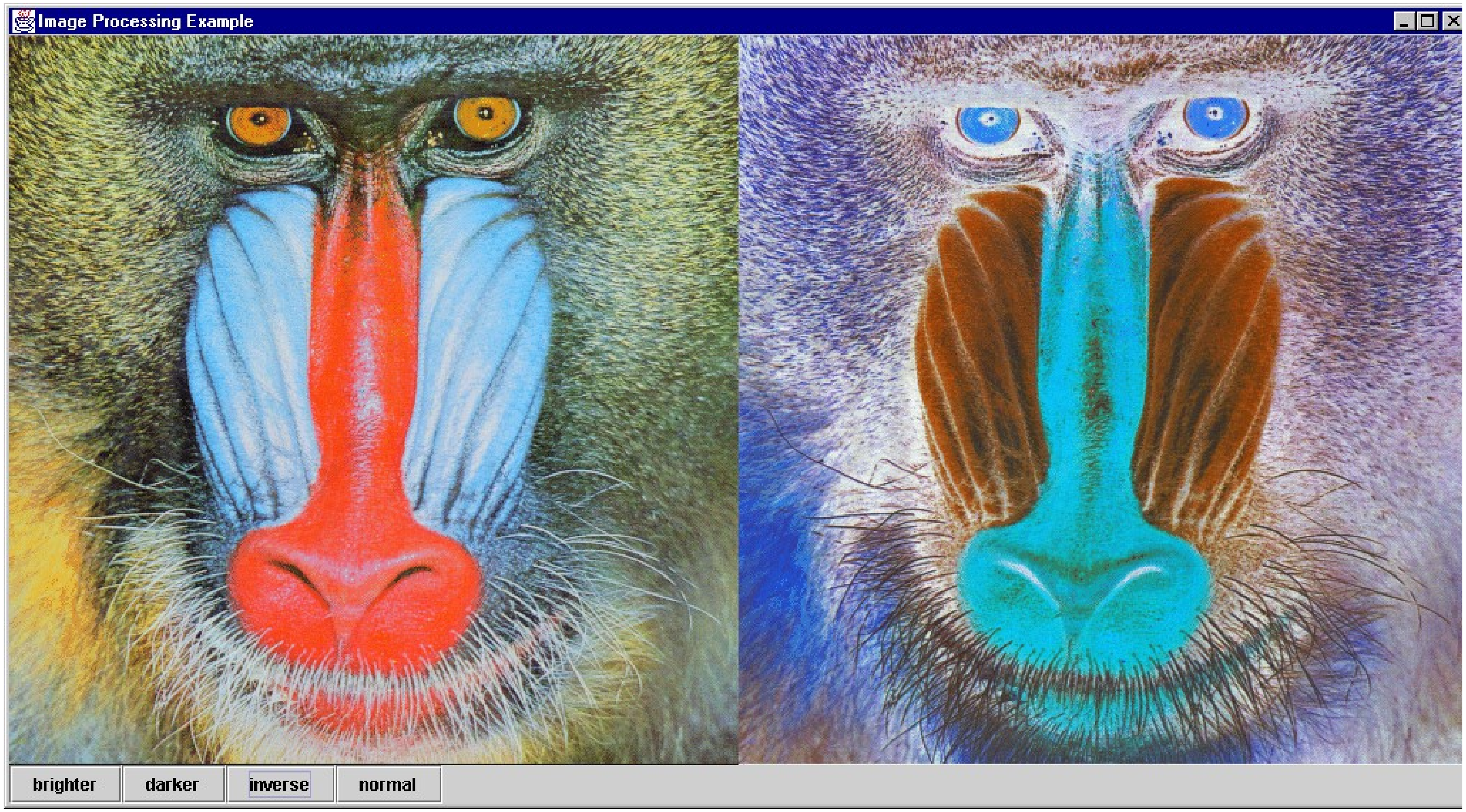
types of techniques

- **simple pixel modification**
- interpolation/extrapolation
- compositing
- convolution
- dithering
- warping
- morphing
- misc. effects

Preprosto spreminjanje pikslov



Primer



Preprosto spreminjanje pikslov

Homogeneous Point Processing--

An image processing that maps an input image to an output image without performing geometric changes.

Theory--

If an image is represented by an array of pixels, each with a color value, one can manipulate the color values to change the image.

Examples--

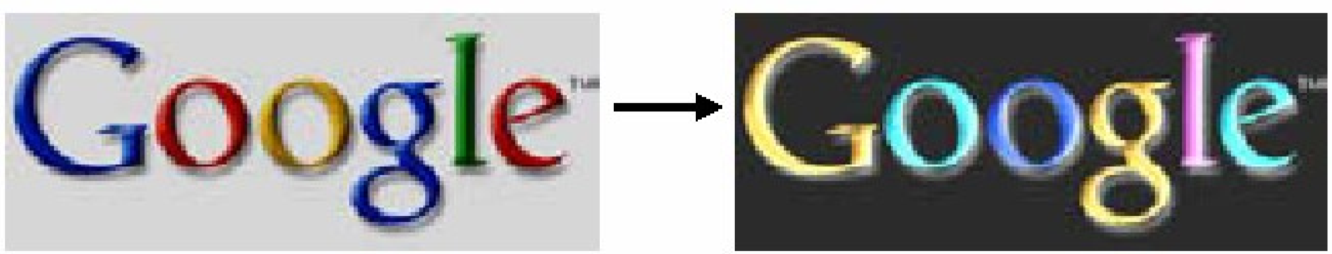
- washing out an image,
- brightening an image,
- adjusting an image’s contrast

Negativi slik

Princip: $f(v) = 1 - v$

A negative image may be created by taking a color, and finding its opposite. If 8 bits represent a given red, blue, or green value, the inverse would be 255 minus that color value.

Formula: $\boldsymbol{f\,(v_{ij}) = 255 - v_{ij}}$



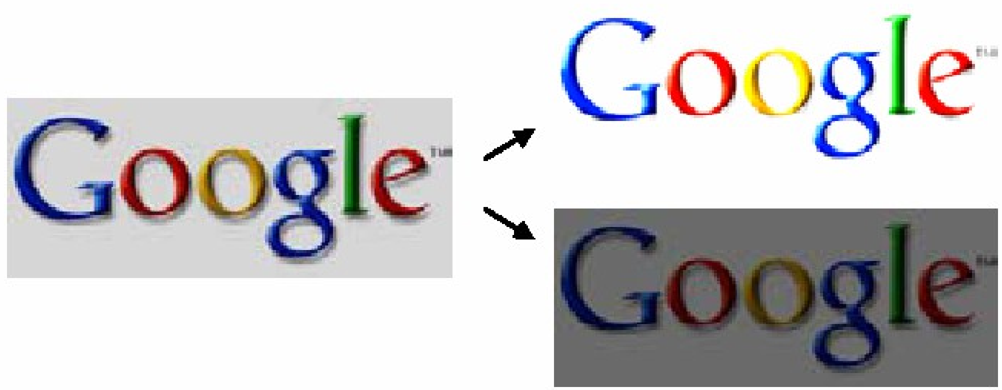
Popravljanje svetlosti (Brightness)

Princip: $f(v) = \alpha v$ for $\alpha \geq 0$ clamp to [0,1]

The brightness of each pixel may be adjusted by a similar use of the byte range:

$$\boldsymbol{f (v_{ij}) = 255 \, (v_{ij} / 255)^{pow}}$$

With pow > 1, the image darkens



Popravljanje kontrasta

Likewise, we can build a linear transform to alter contrast

$$v_{ij} = cv_{ij} + b,$$

where

c = delta D / delta V,

b = (Dmin*Vmax - Dmax*Vmin)/ deltaV

given

V = value in image

D = value that can be displayed

(This is also called linear gray level scaling)

Koda v Javi: naložimo sliko (in njene piksle)

```
public void loadImage(String strFile) {
       ImageIcon i = new ImageIcon(strFile);
       image = i.getImage();
       width = image.getWidth(this);
       height = image.getHeight(this);
   }

public void getData() {
      pixels = new int[width * height];
      PixelGrabber pg = new PixelGrabber
           (image, 0, 0, width, height, pixels, 0, width);
       try {
         pg.grabPixels();
       }
       catch (InterruptedException e){
               System.err.println("interrupted waiting for pixels!");
               return;
           }
       if ((pg.getStatus() & ImageObserver.ABORT) != 0){
               System.err.println("image fetch aborted or errored");
               return;
           }
   }
```

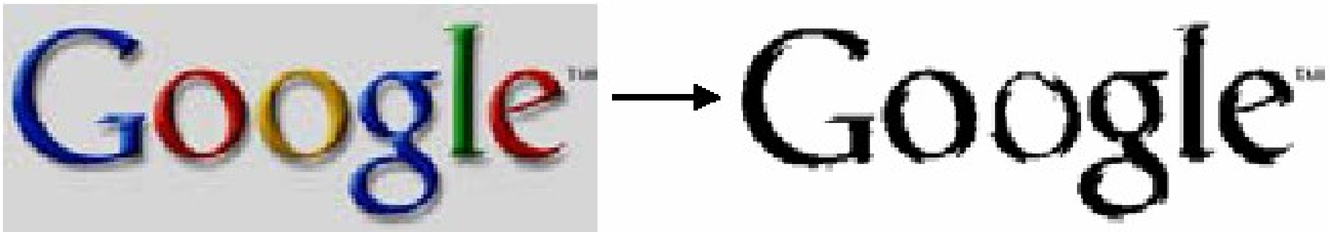
How does PixelGrabber work? Check your API

Obdelava pikslov

```
public int inversePixel(int x, int y, int pixel) {
    int alpha = (pixel >> 24) & 0xff;
    int red   = (pixel >> 16) & 0xff;
    int green = (pixel >>  8) & 0xff;
    int blue  = (pixel      ) & 0xff;
    red = Math.abs(255-red);
    green = Math.abs(255-green);
    blue  = Math.abs(255-blue);
    return (alpha << 24) | (red << 16) | (green << 8) | blue;
}

public int brightenPixel(int x, int y, int pixel, double p){
    int alpha = (pixel >> 24) & 0xff;
    int red   = (pixel >> 16) & 0xff;
    int green = (pixel >>  8) & 0xff;
    int blue  = (pixel      ) & 0xff;
    red   = (int)(255 * Math.pow(red/255.0, p));
    green = (int)(255 * Math.pow(green/255.0, p));
    blue  = (int)(255 * Math.pow(blue/255.0, p));
    return (alpha << 24) | (red << 16) | (green << 8) | blue;
}
```

Prag (treshold)



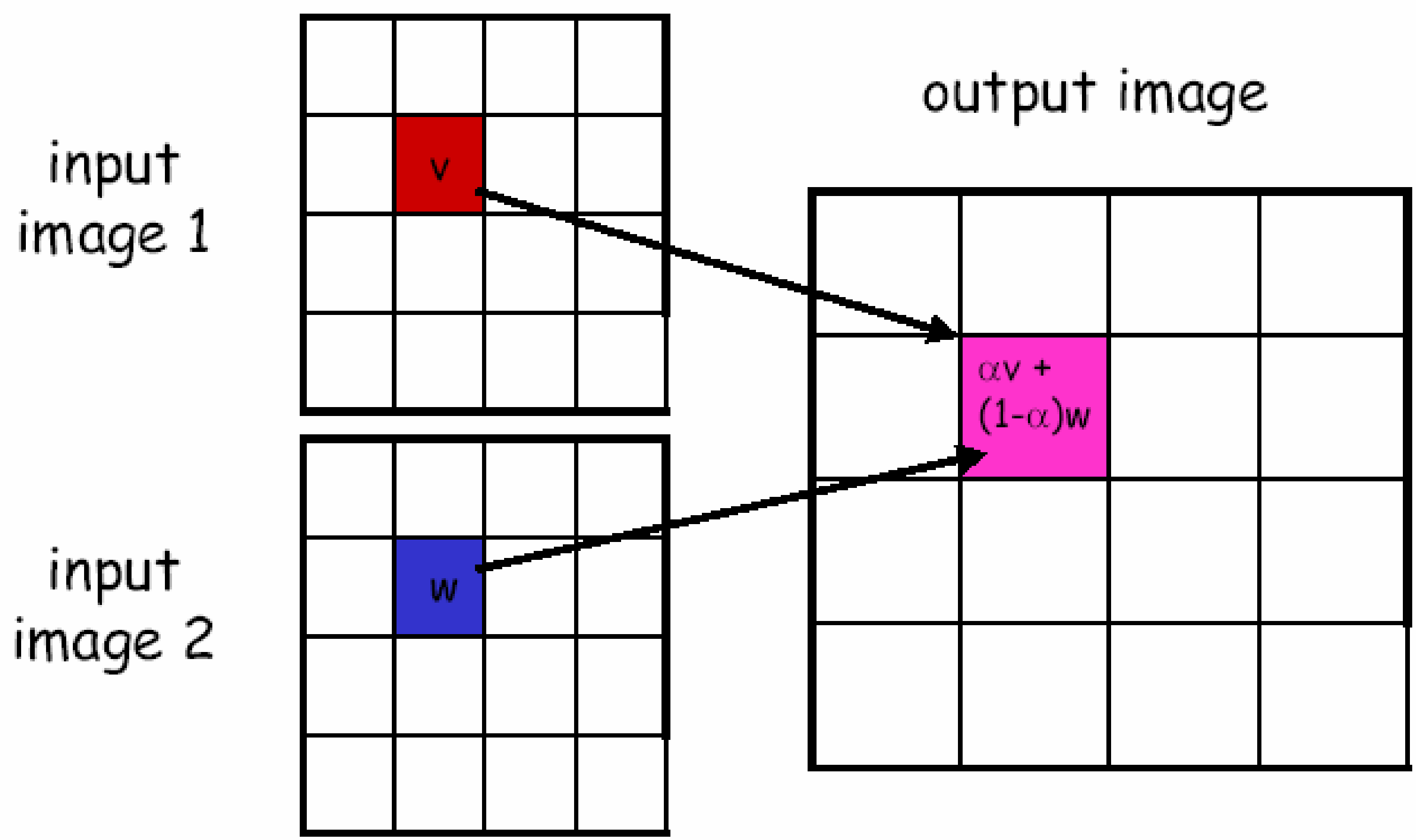
if $v>t$ then $f(v)=1$

else $f(v)=0$

types of techniques

- simple pixel modification
- **interpolation/extrapolation**
- compositing
- convolution
- dithering
- warping
- morphing
- non-photo-realistic effects

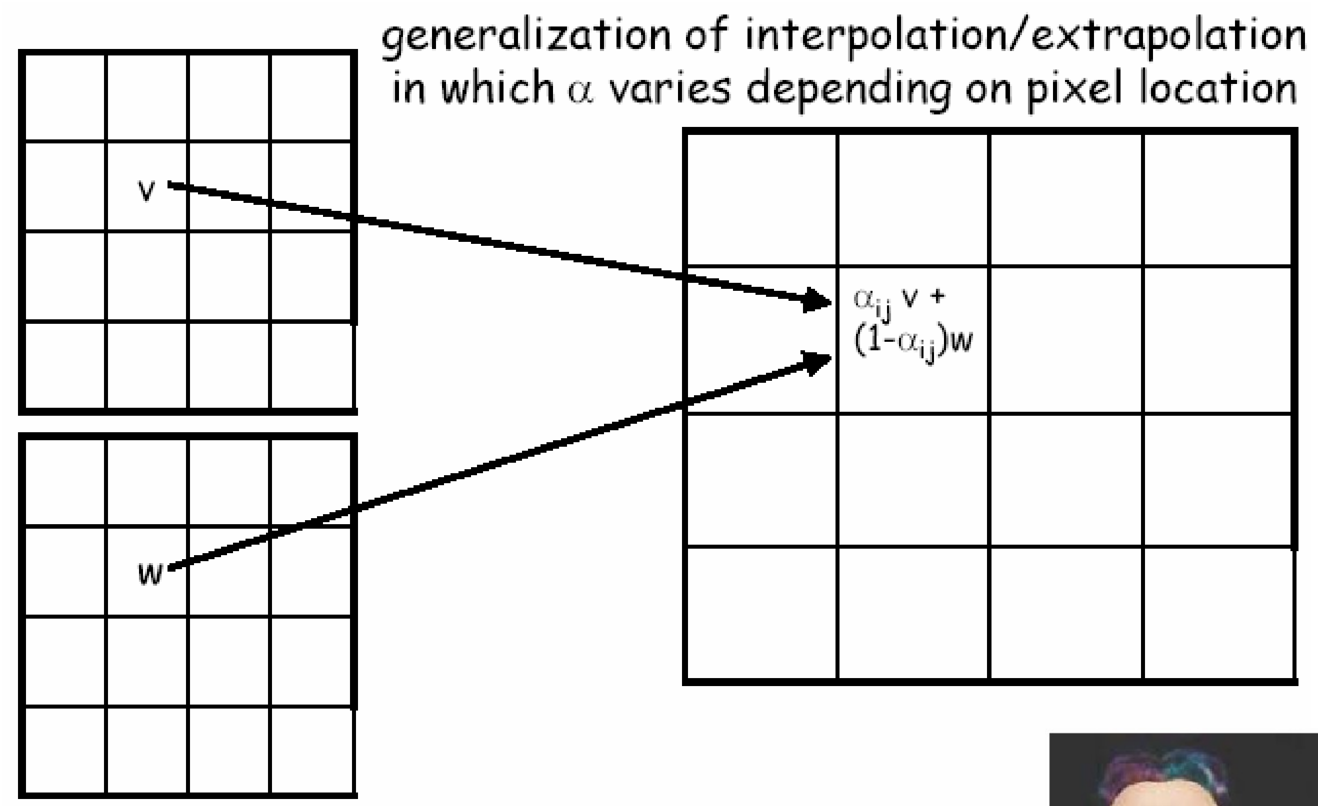
Interpolacija - ekstrapolacija



type of techniques

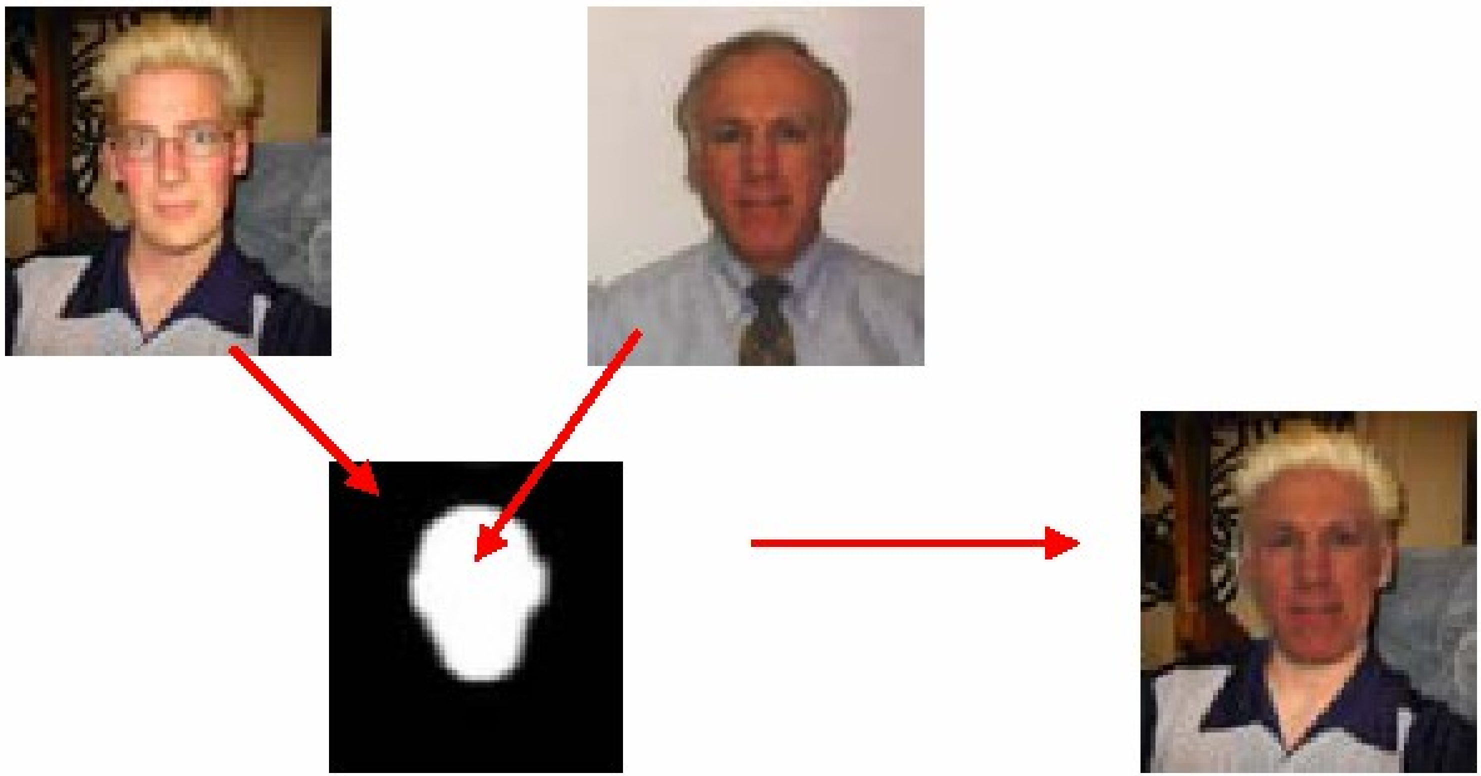
- simple pixel modification
- interpolation/extrapolation
- **compositing**
- convolution
- dithering
- warping
- morphing
- misc. effects

Kompozicija



Kompozicija

typically $\alpha \in [0,1]$ so the array of α values can represented by a single channel image called a *mask*



type of techniques

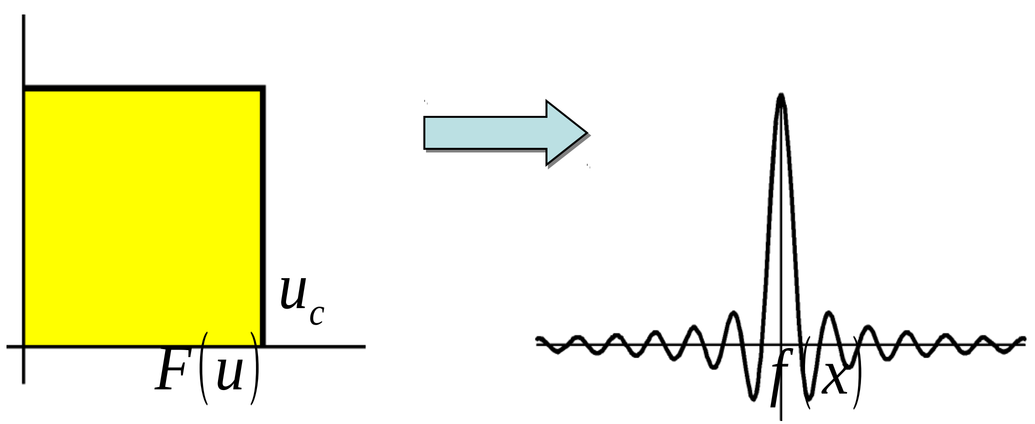
- simple pixel modification
- interpolation/extrapolation
- compositing
- **convolution**
- dithering
- warping
- morphing
- misc. effects

Filtri

- Filters are characterised by:
  - *frequency response* (e.g. high-pass, low-pass, band-pass, notch…)
  - *support* = region over which filter is non-zero. If non-zero over a finite domain, filter is said to have finite support.
  - *order*: most filters are *linear* (i.e. do not depend on the signal being filtered)
  - *variance*: a filter which does not alter with position is known as *shift-invariant* otherwise *spatially variant*.
- The frequency representation of a filter determines its frequency response.
- The spatial representation is used to characterise its support.

Jedra filtrov (Filter Kernels)

- The ideal filter is a box function in the frequency domain.
- In the spatial domain, this becomes the *sinc* filter kernel which has non-finite support.



$$F(u) = \begin{cases} 1 & |u| \le u_c \\ 0 & |u| > u_c \end{cases}$$

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

Filtri

- Kernels with larger support can have better frequency response properties (more effective filters).

  ⇒ *more memory* required per filter = $(2s+1)^2$ for discrete 2D filter

  ⇒ convolution is *more expensive* = $n(2s+1)^2$

- Filter *gain* is usually assumed to be 1, i.e. *unity gain* ⇒ filter is normalised:

$$g = \sum_{t=-s}^{s} g[t]$$

- If gain > 1, the filtered signal will be scaled by the gain
  - images will get brighter or darker
- We can compensate by dividing by the gain (= *scale*):

$$h(x) = f * g = \sum_{t=-s}^{s} \frac{g[t]}{gain} f[x-t] = \frac{1}{gain} \sum_{t=-s}^{s} g[t]\, f[x-t]$$

Konvolucija

- Multiplying the frequency representation of an image by a filter may be expressed in the spatial domain as *convolution*.
- If the image is *f(x)* and the filter (in the spatial domain) is *g(x)* then convolution is defined by:

$$f(x) * g(x) = \mathbf{F}(f(x))\mathbf{F}(g(x)) = \int_{-\infty}^{\infty} g(t) f(x-t) dt$$

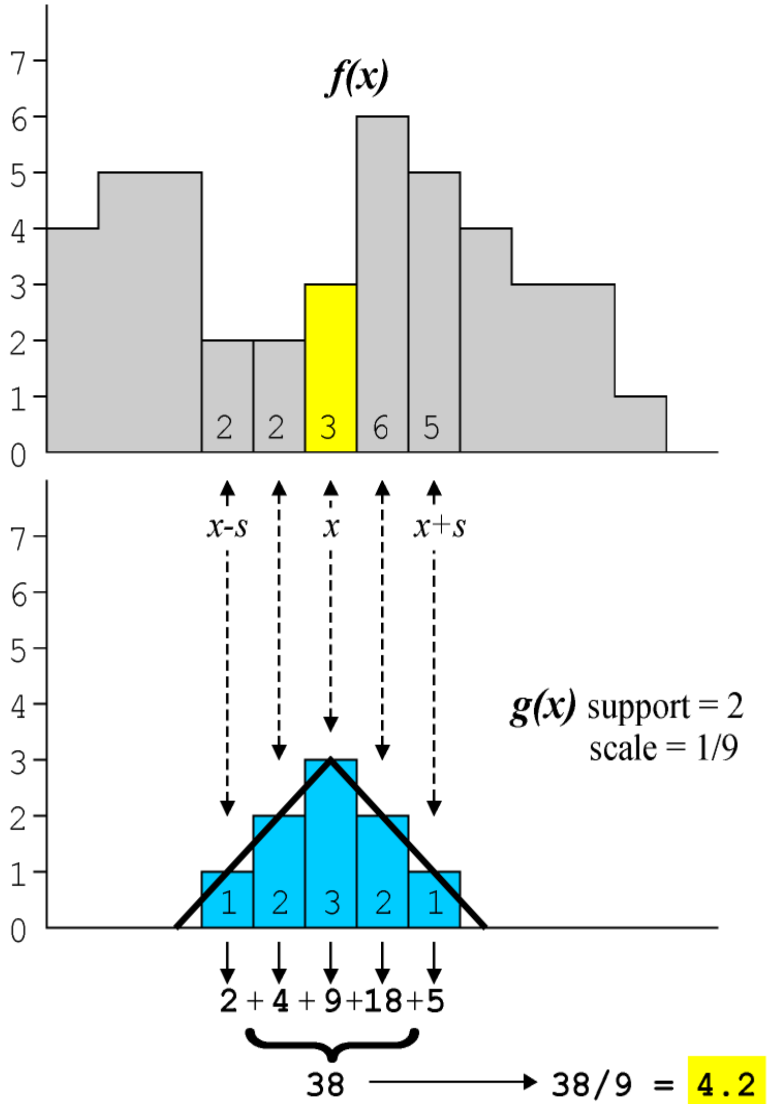
- Normally we are concerned with *discrete convolution*:

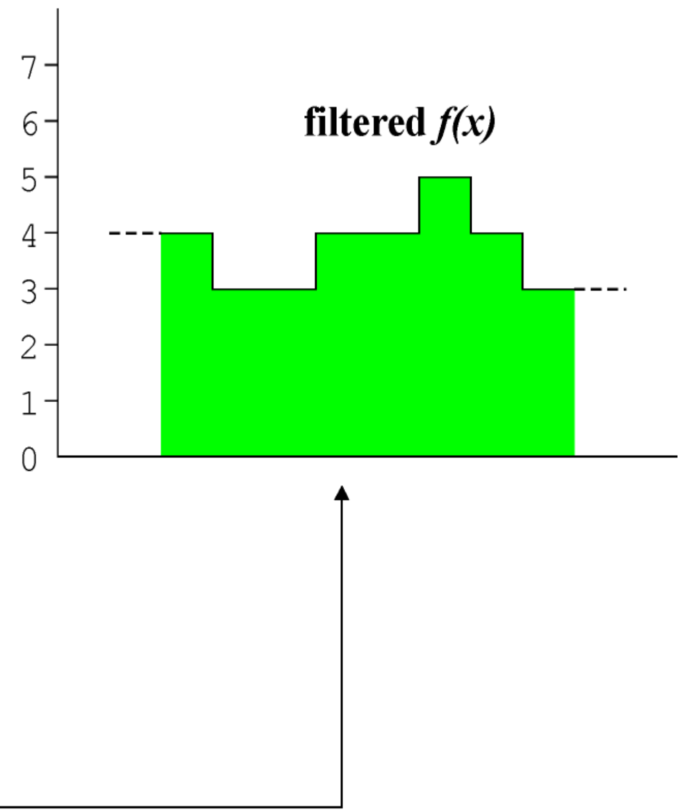
$$f * g = \sum_{t=-s}^{s} g[t] f[x-t]$$

- In this case, the filter *g(x)* is assumed to have finite discrete support [-*s*,*s*]

Diskretna konvolucija

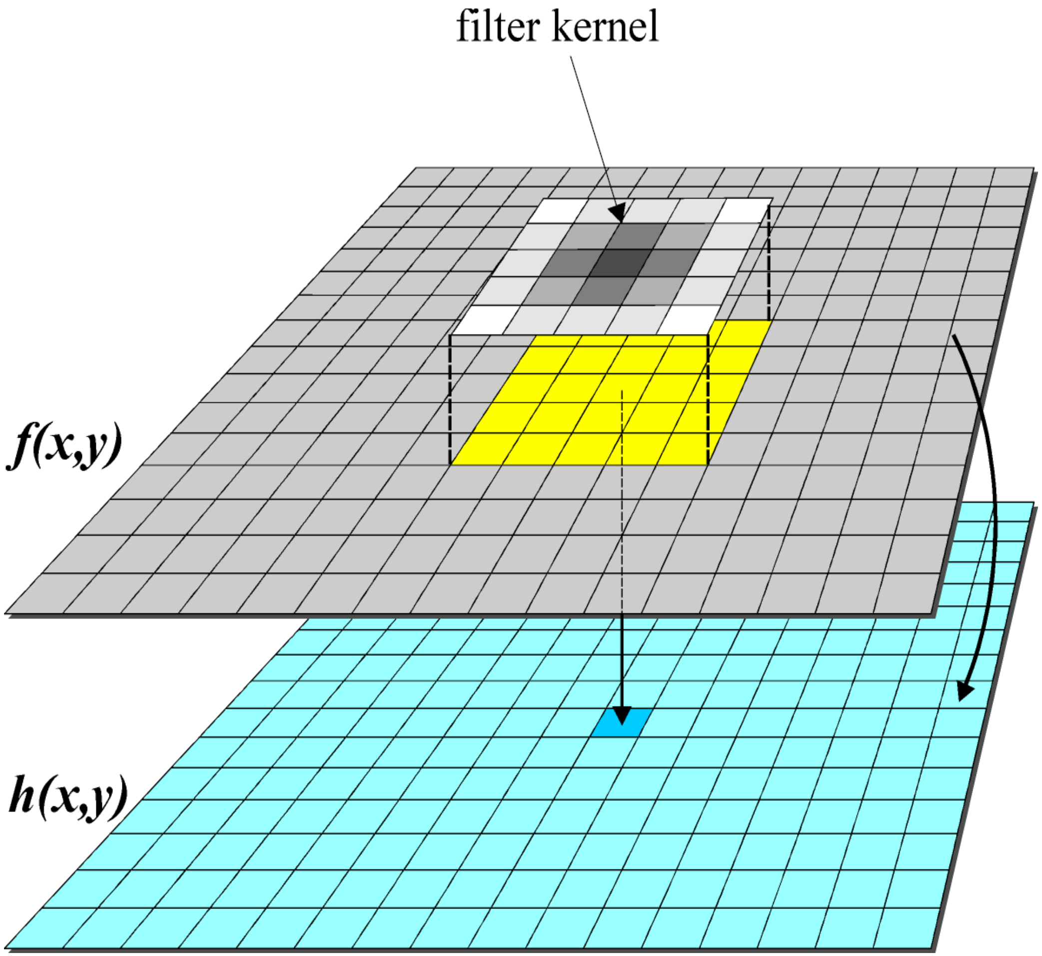
$$f * g = \frac{1}{gain} \sum_{t=-s}^{s} g[t]\, f[x-t]$$



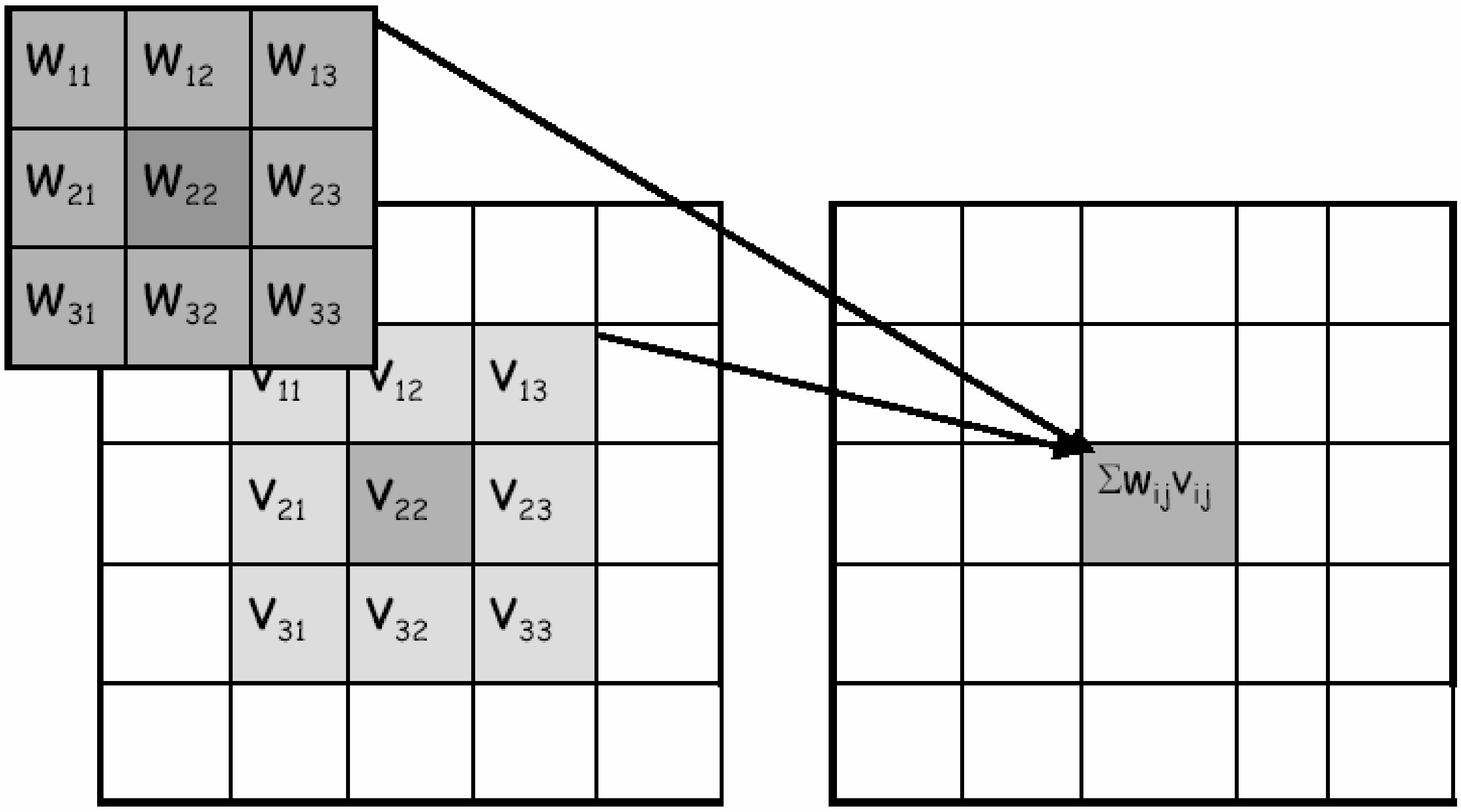


Diskretna 2D konvolucija

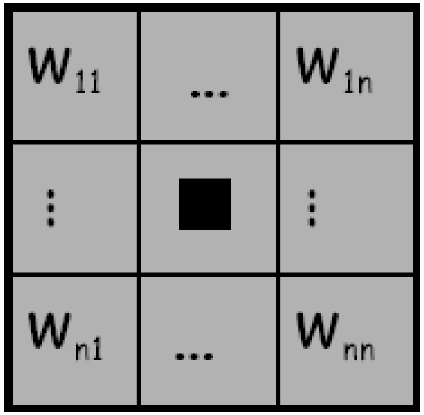
$$f * g = \frac{1}{gain} \sum_{v=-s}^{s} \sum_{u=-s}^{s} g[u,v]\, f[x-u, y-v]$$



Konvolucija

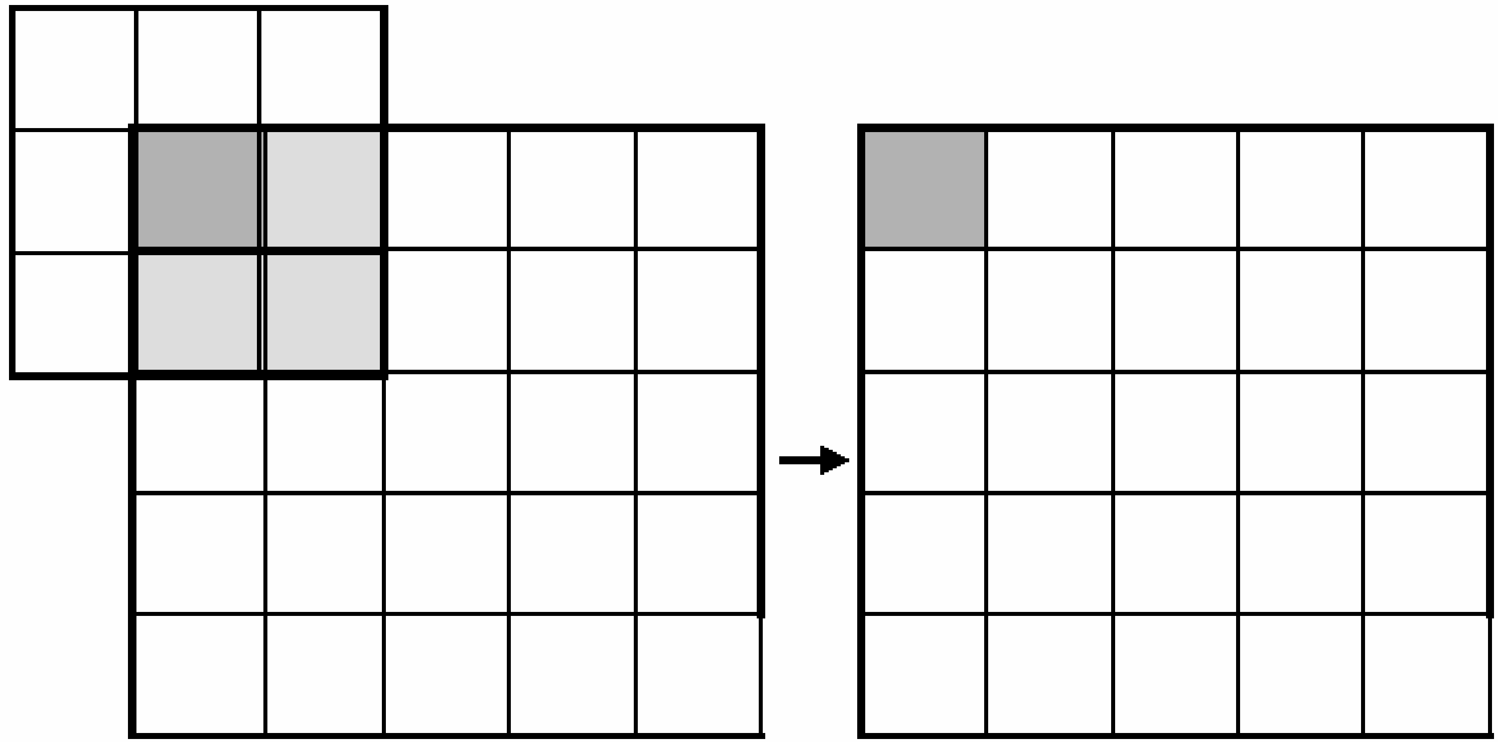


Splošna oblika jedra

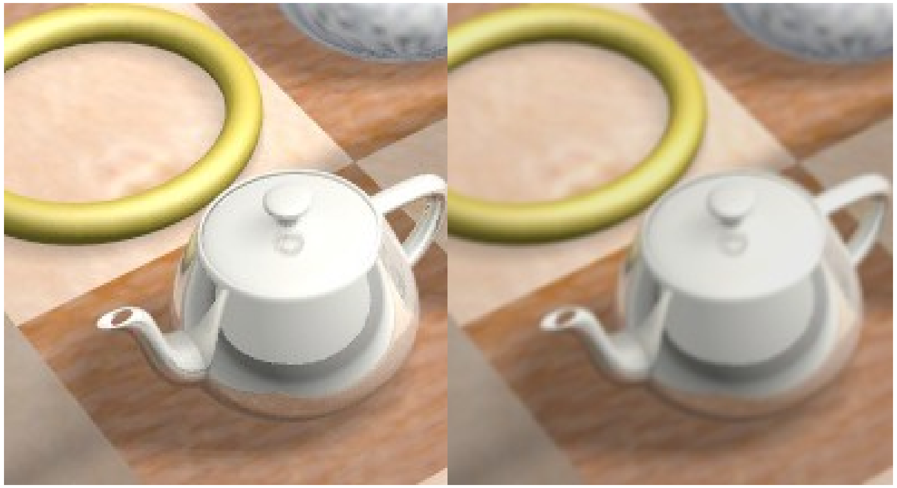


n.... liho število

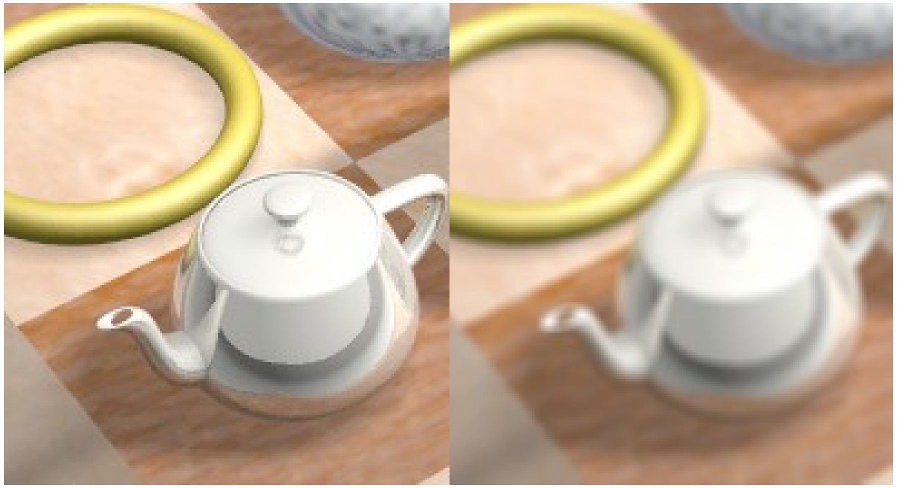
Kaj narediti na robovih?



Nizkopasovni filtri

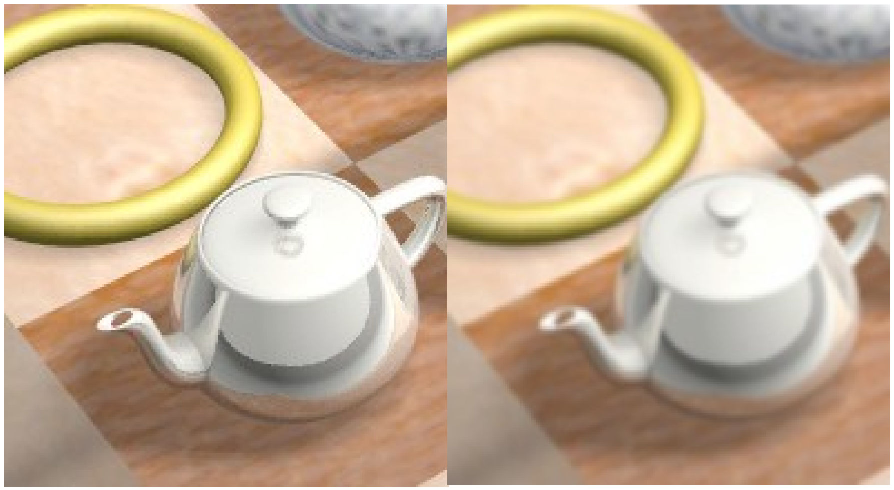


$$\text{Box (order 3)} = \frac{1}{9}\begin{bmatrix}1 & 1 & 1\\1 & 1 & 1\\1 & 1 & 1\end{bmatrix}$$

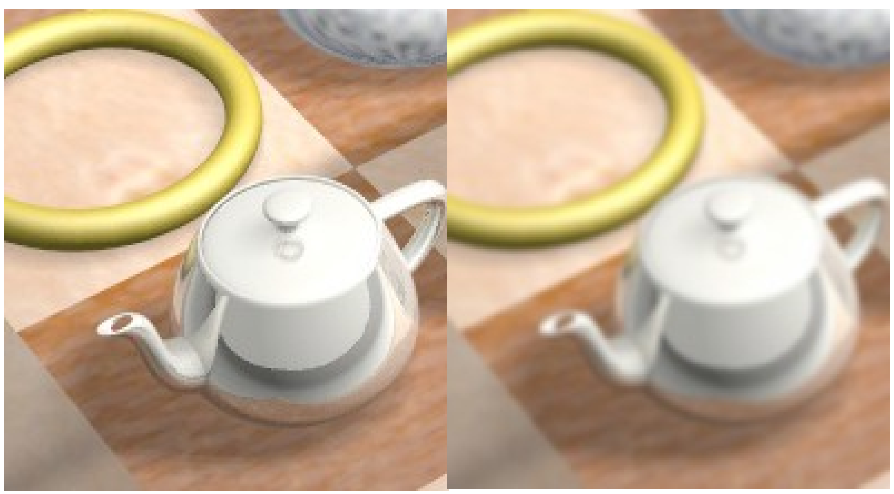


$$\text{Box (order 5)} = \frac{1}{25}\begin{bmatrix}1 & 1 & 1 & 1 & 1\\1 & 1 & 1 & 1 & 1\\1 & 1 & 1 & 1 & 1\\1 & 1 & 1 & 1 & 1\\1 & 1 & 1 & 1 & 1\end{bmatrix}$$

Nizkopasovni filtri

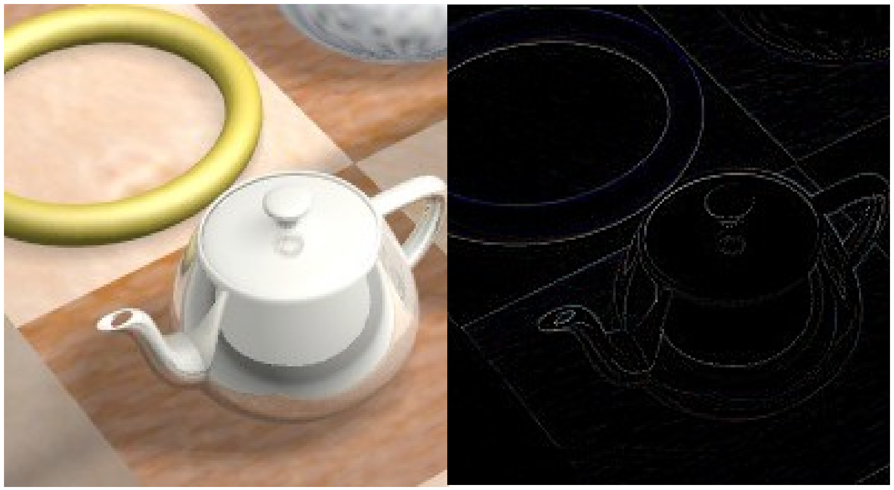


$$\text{Gaussian} = \frac{1}{256}\begin{bmatrix} 1 & 4 & 6 & 4 & 1 \\ 4 & 16 & 24 & 16 & 4 \\ 6 & 24 & 36 & 24 & 6 \\ 4 & 16 & 24 & 16 & 4 \\ 1 & 4 & 6 & 4 & 1 \end{bmatrix}$$



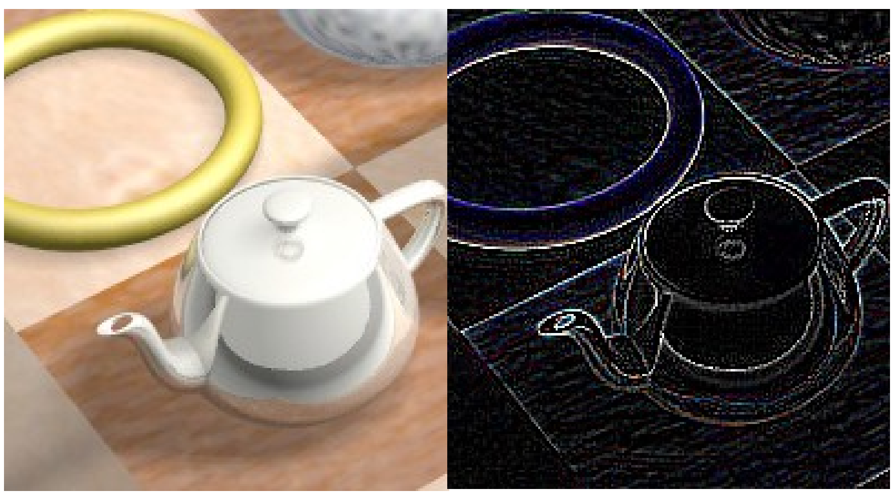
$$\text{Triangle (Bartlett)} = \frac{1}{81}\begin{bmatrix} 1 & 2 & 3 & 2 & 1 \\ 2 & 4 & 6 & 4 & 2 \\ 3 & 6 & 9 & 6 & 3 \\ 2 & 4 & 6 & 4 & 2 \\ 1 & 2 & 3 & 4 & 1 \end{bmatrix}$$

Visokopasovni filtri



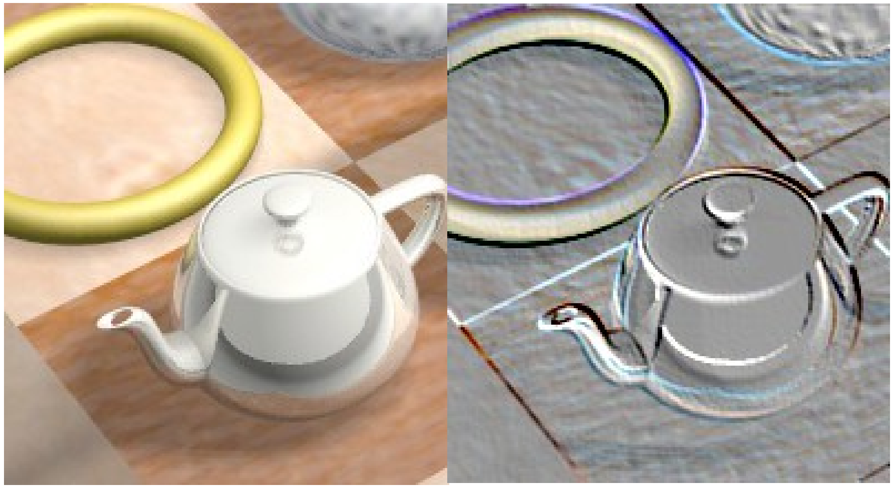
$$\text{Highpass} = \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & -12 & 2 \\ 1 & 2 & 1 \end{bmatrix}$$

$$= \frac{1}{16}\begin{bmatrix} 1 & 2 & 1 \\ 2 & 4 & 2 \\ 1 & 2 & 1 \end{bmatrix} - \begin{bmatrix} 0 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 0 \end{bmatrix}$$

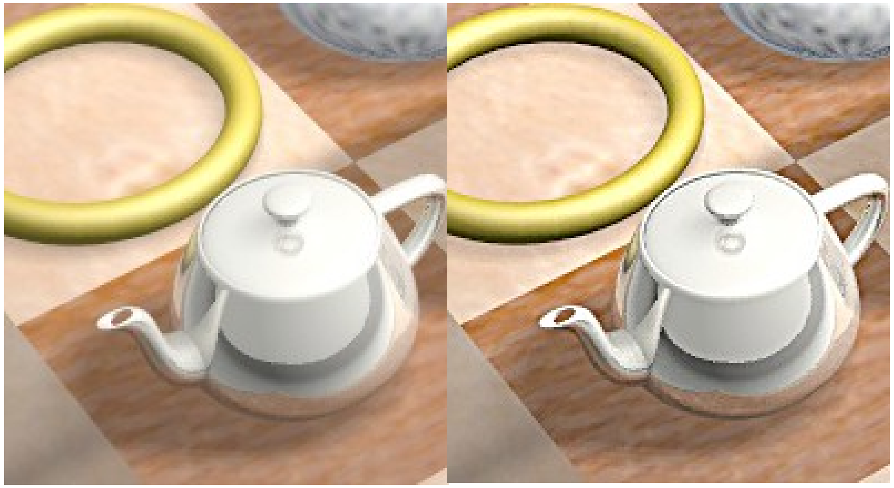


$$\text{Laplacian} = \frac{1}{1}\begin{bmatrix} 1 & 1 & 1 \\ 1 & -8 & 1 \\ 1 & 1 & 1 \end{bmatrix}$$

Drugi filtri

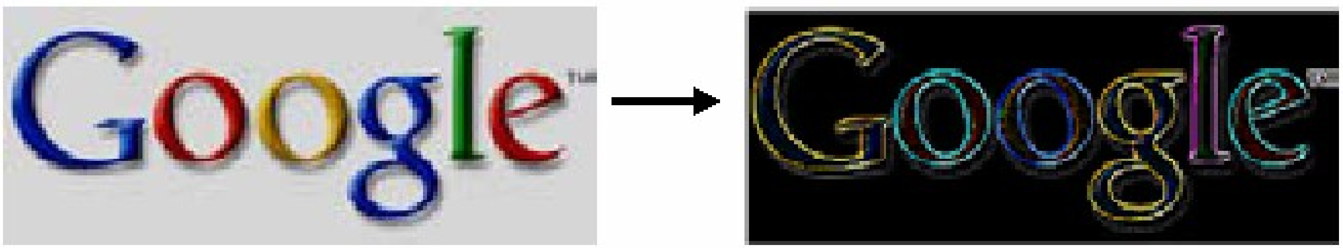


$$\text{Emboss} = \frac{1}{2} + \begin{bmatrix} 0 & 1 & 1 \\ -1 & 0 & 1 \\ -1 & -1 & 0 \end{bmatrix}$$



$$\text{unsharp} = f + \left(f - f * g_{LP} \right)$$

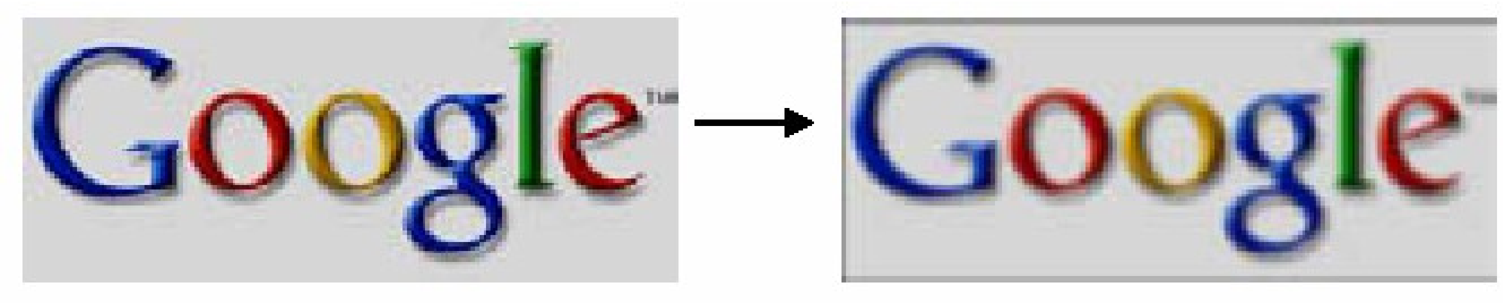
Odkrivanje robov



Jedro za odkrivanje robov

-1/8	-1/8	-1/8
-1/8	1	-1/8
-1/8	-1/8	-1/8

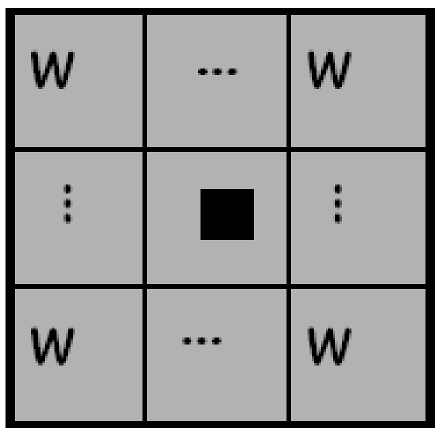
Zameglitev (blur)



3x3 box blur

1/9	1/9	1/9
1/9	1/9	1/9
1/9	1/9	1/9

nXn box blur



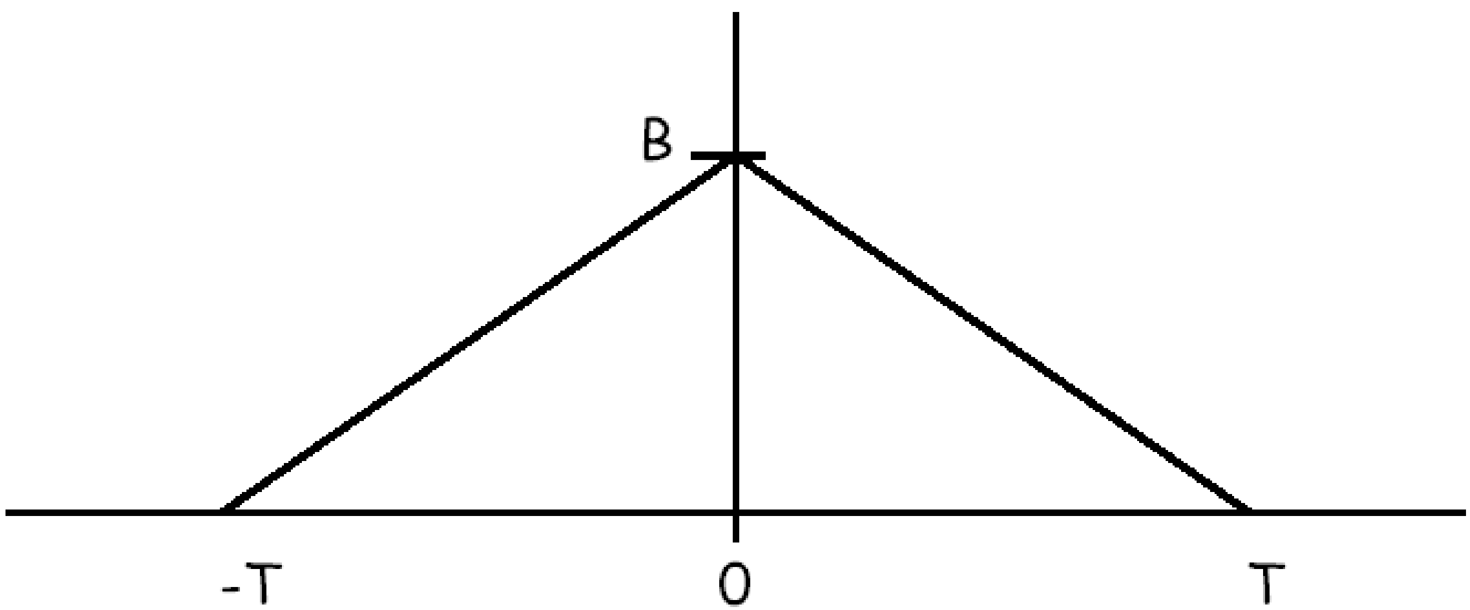
$w=1/n^2$

why is it important that the sum of the weights is 1?

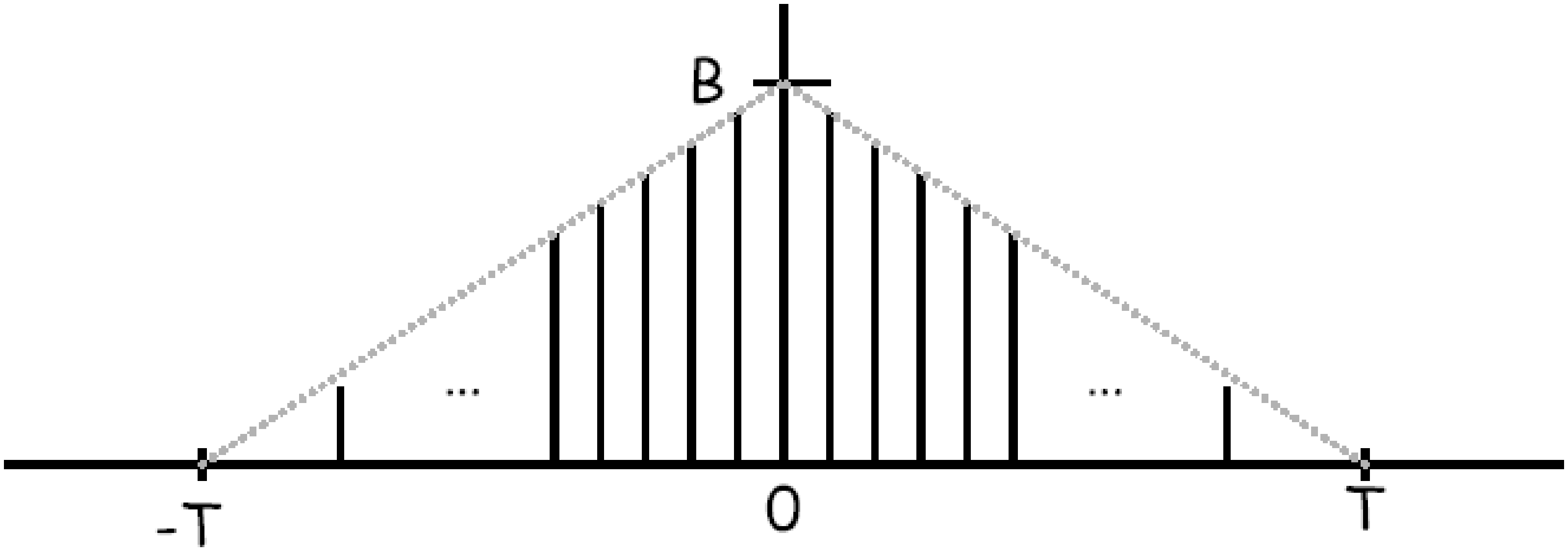
3x3 triangle blur

1/16	1/8	1/16
1/8	1/4	1/8
1/16	1/8	1/16

triangle function

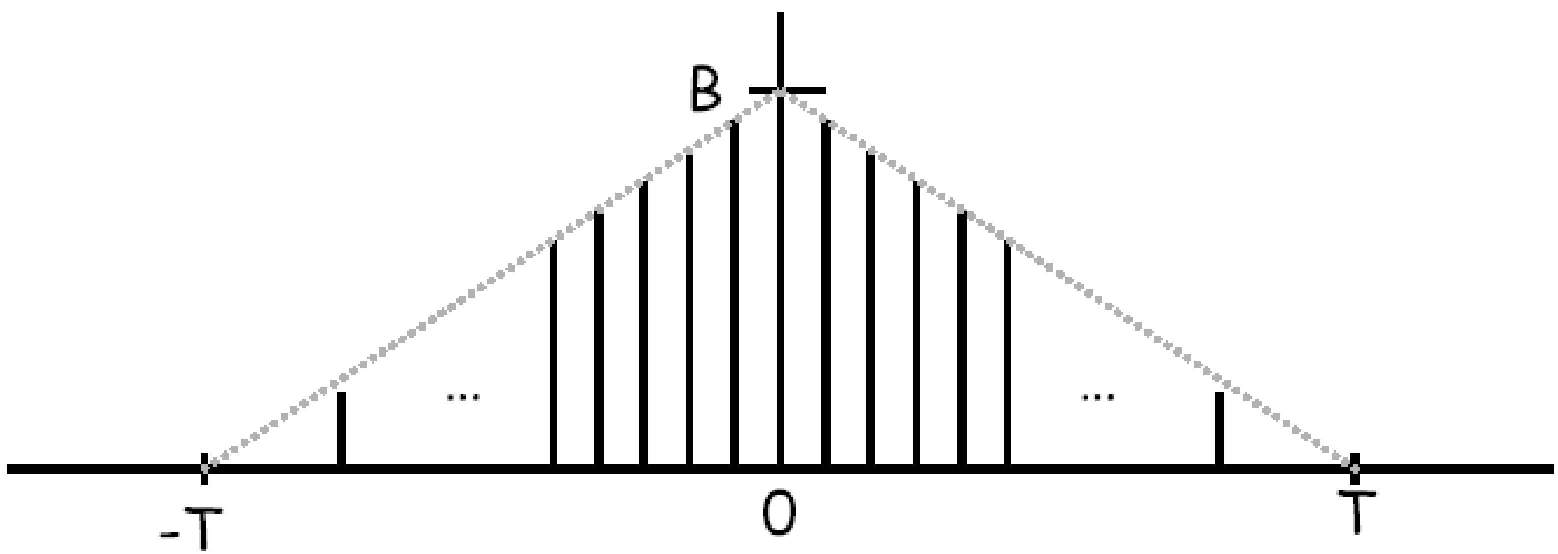


discrete triangle

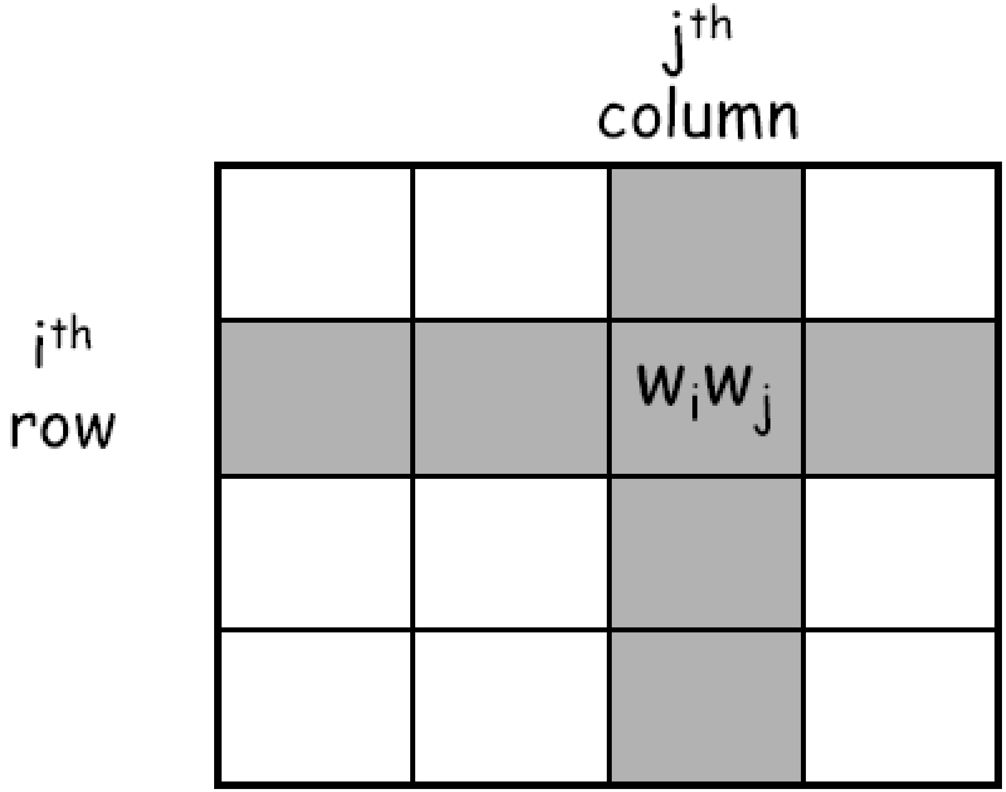


normalized, discrete triangle

1. T= (n+1)/2 gives n non-zero samples
2. $\Sigma_{j=-T \ldots T}$ f(j) = 1 provided B=2/(n+1)



triangle blur filter

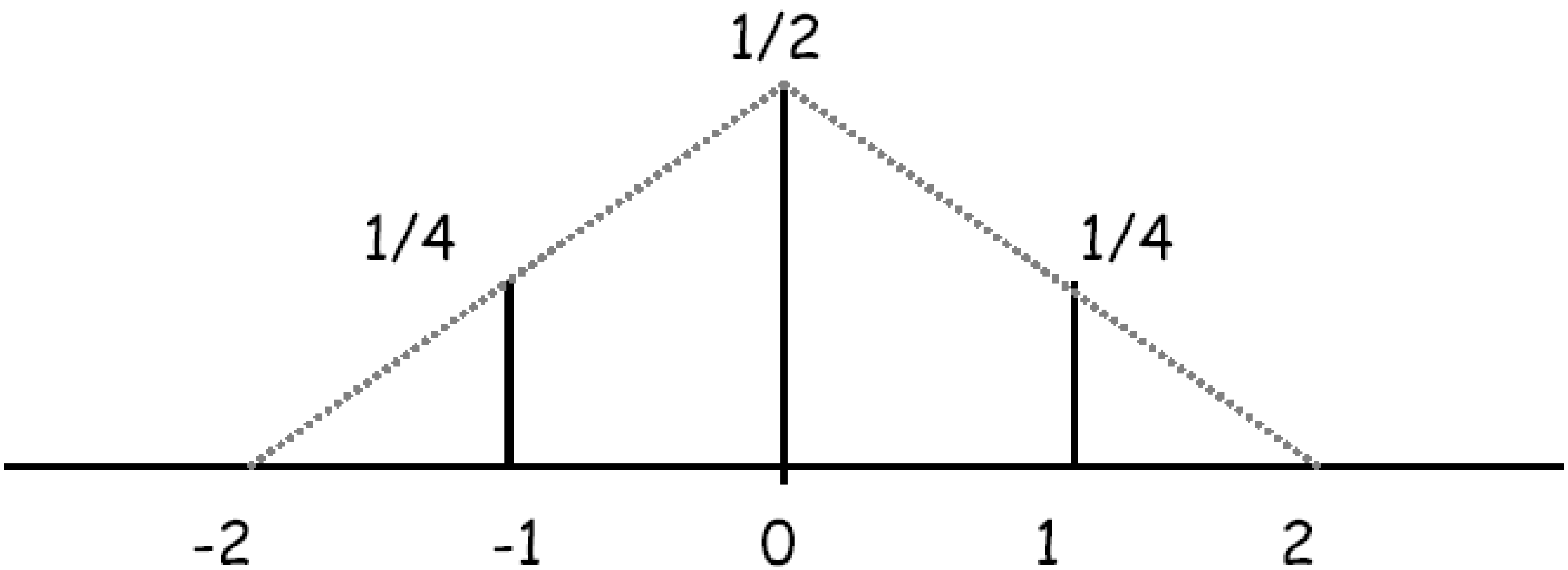


triangle samples: $w_1, w_2, \ldots, w_n$

example: n=3

$T=(n+1)/2 = 2$

$B = 2/(n+1) = 1/2$

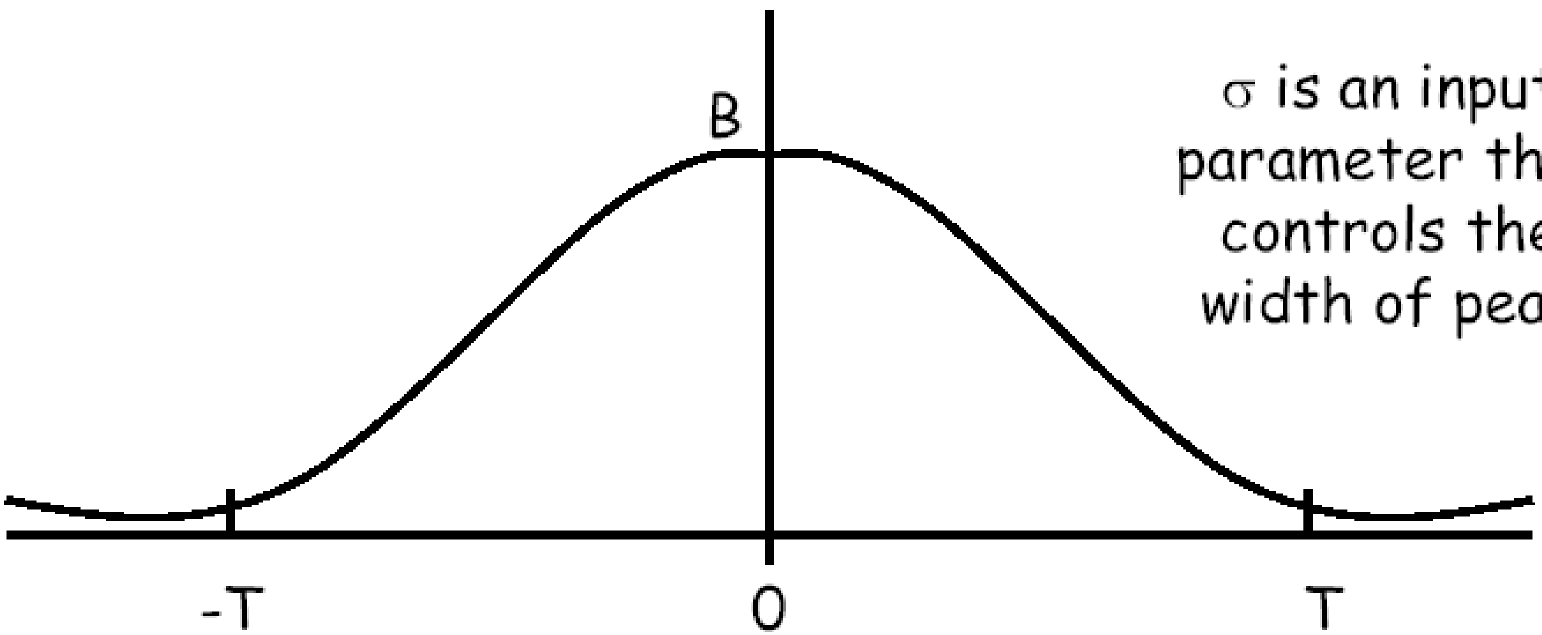


3x3 triangle blur filter

1/4	1/16	1/8	1/16
1/2	1/8	1/4	1/8
1/4	1/16	1/8	1/16
	1/4	1/2	1/4

gaussian function

$f(x) = Be^{-x^2/\sigma^2}$

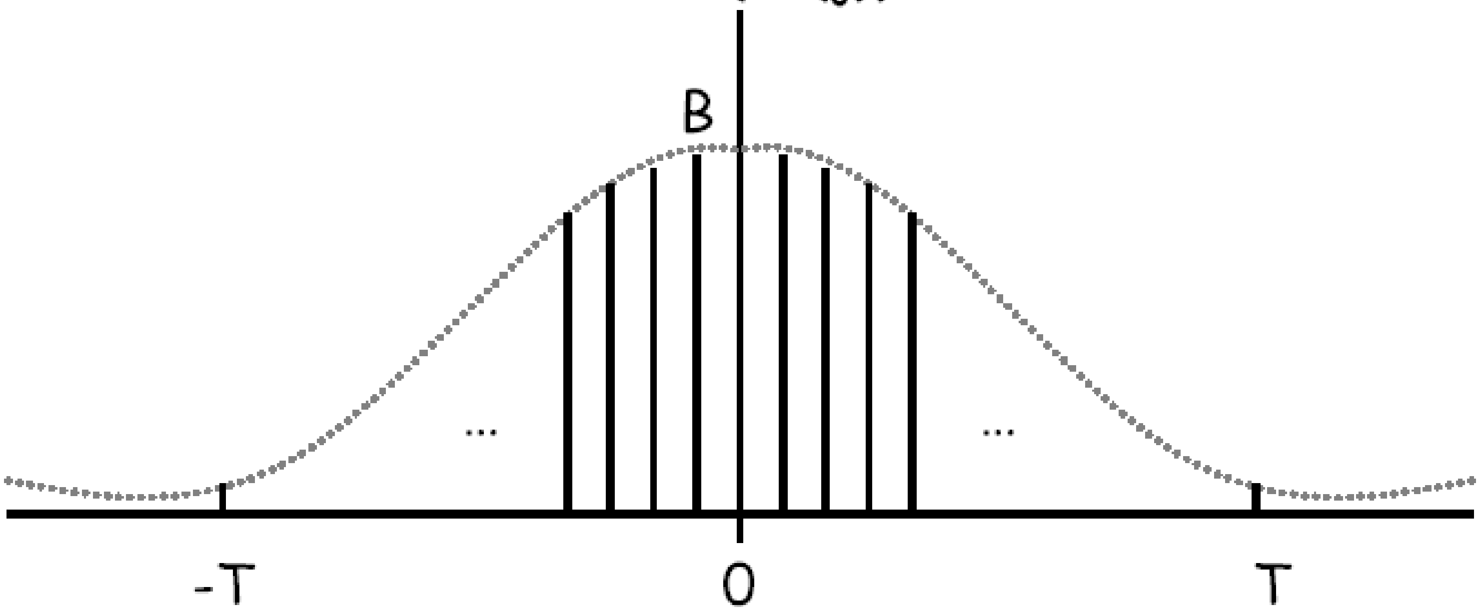


σ is an input parameter that controls the width of peak

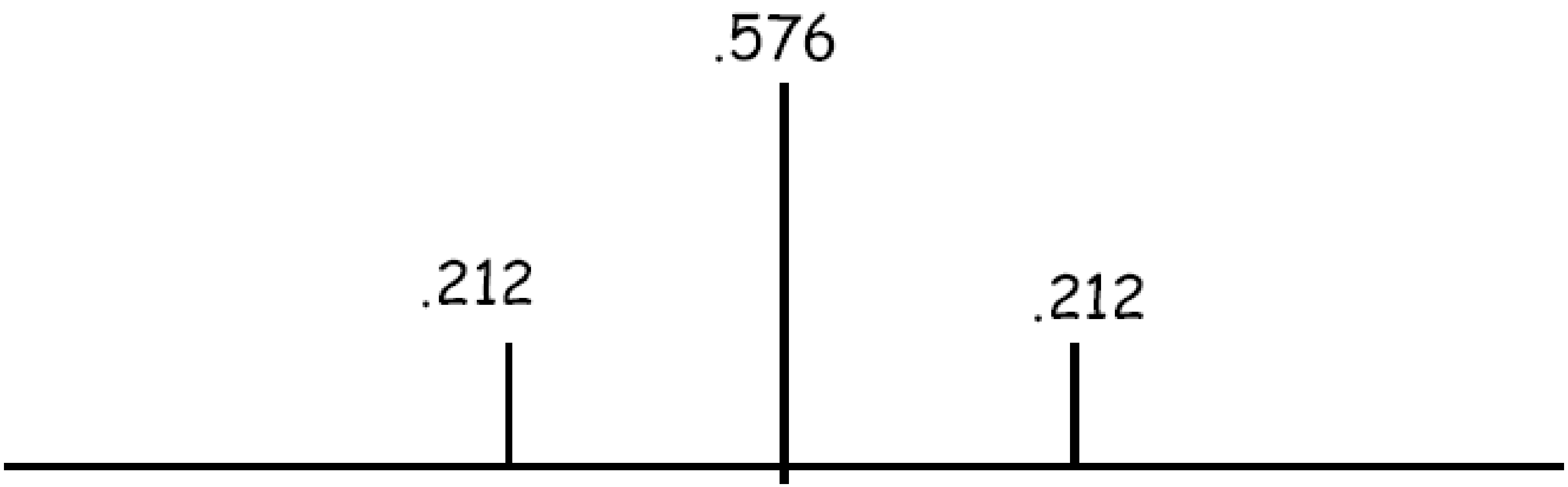
normalized, discrete version

T= (n+1)/2 gives n samples

$B = 1/(\Sigma f(j))$



example: n=3, σ=1



3x3 gaussian blur, $\sigma = 1$

.212	.045	.122	.045
.576	.122	.332	.122
.212	.045	.122	.045
	.212	.576	.212

Box, triangle, Gaussian blurr

Obdelava slik in java2D

type of techniques

- simple pixel modification
- interpolation/extrapolation
- compositing
- convolution
- **dithering** Razprševanje
- warping
- morphing
- misc. effects

Razprševanje in poltoni (dithering, halftoning)

Trade spatial for intensity resolution

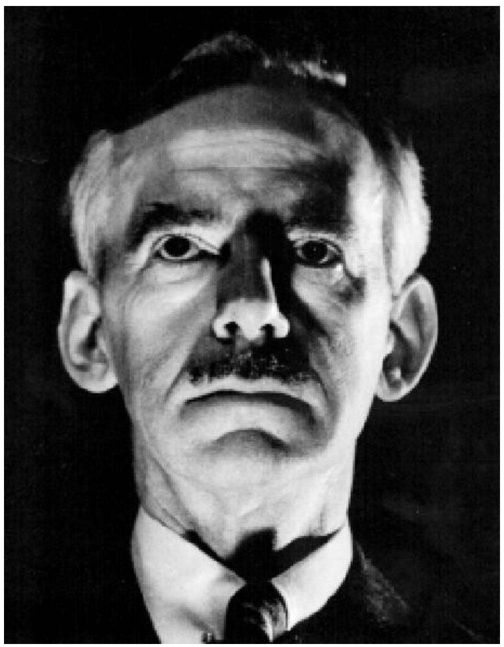
- Thresholding.
- Random dither; Robert's algorithm
- Ordered dither
- Error diffusion

Your eye will average over an area

- Spatial Integration

Prag (Thresholding)

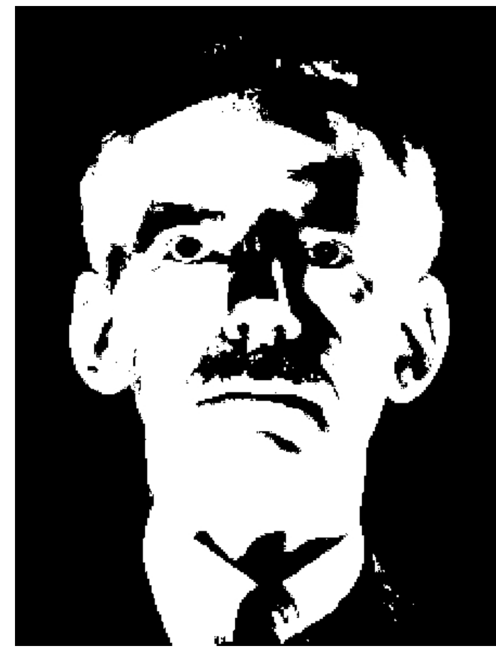
Original image.



n = 0.5

Simple threshold.

n= 0.7



$$v(x, y) = trunc(\tilde{v}(x, y) + n)$$

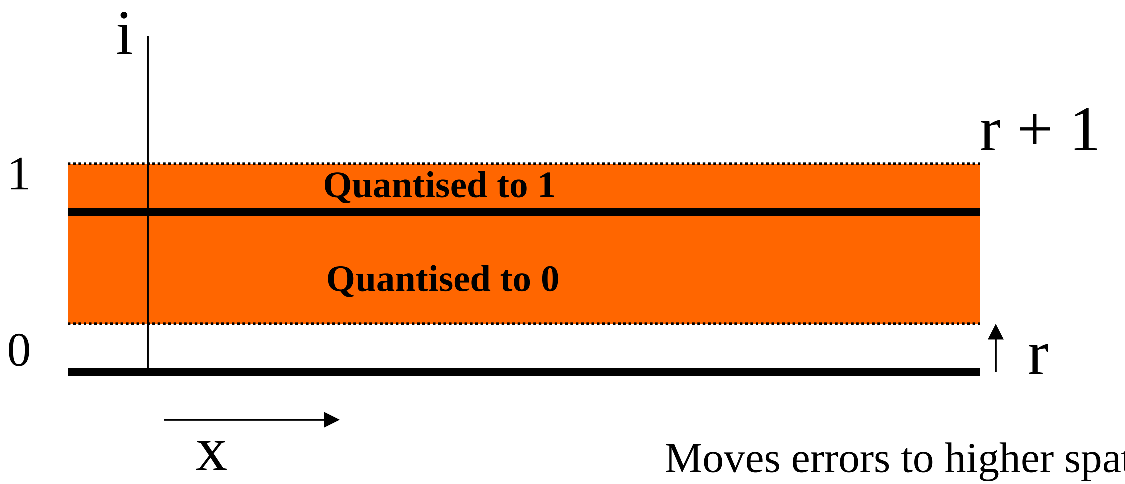
Errors are low spatial frequencies.

Robertov algoritem

- First add noise
- Then quantize

$$\overset{\sqcup}{v}(x,y) = trunc(K \times v(x,y) + noise(x,y))$$

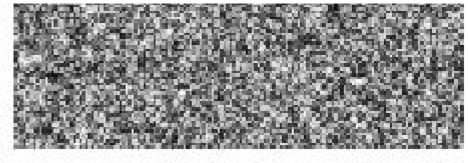
$$0 \le noise < 1$$



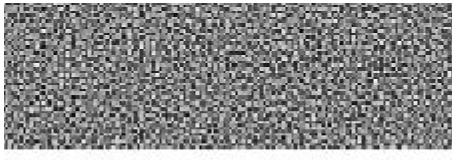
Moves errors to higher spatial frequencies.
-> eye averages over an area.

Spektri šumov

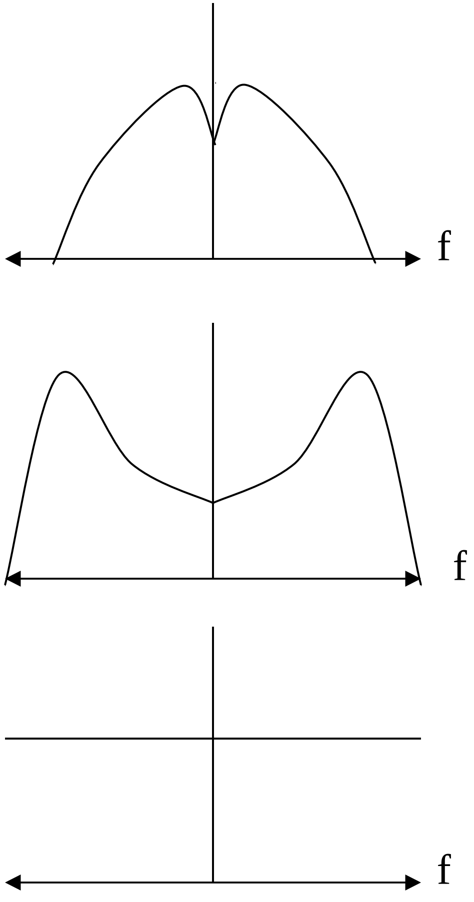
Pink Noise



Blue Noise



White Noise

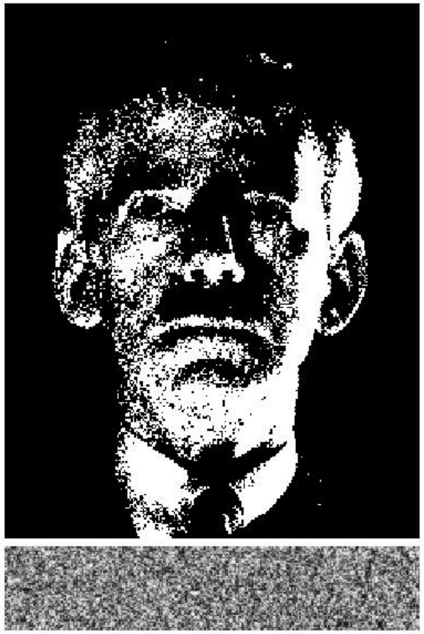


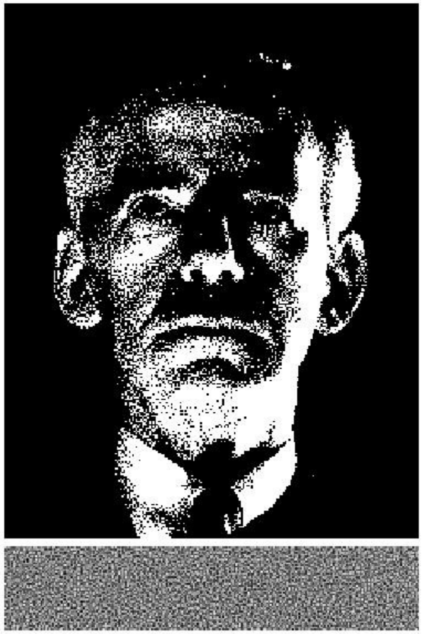
Robertov algoritem

Moves low frequency (average error) to high frequency

Pink(low), **Blue (high)**, White(all) frequency noise

Pink





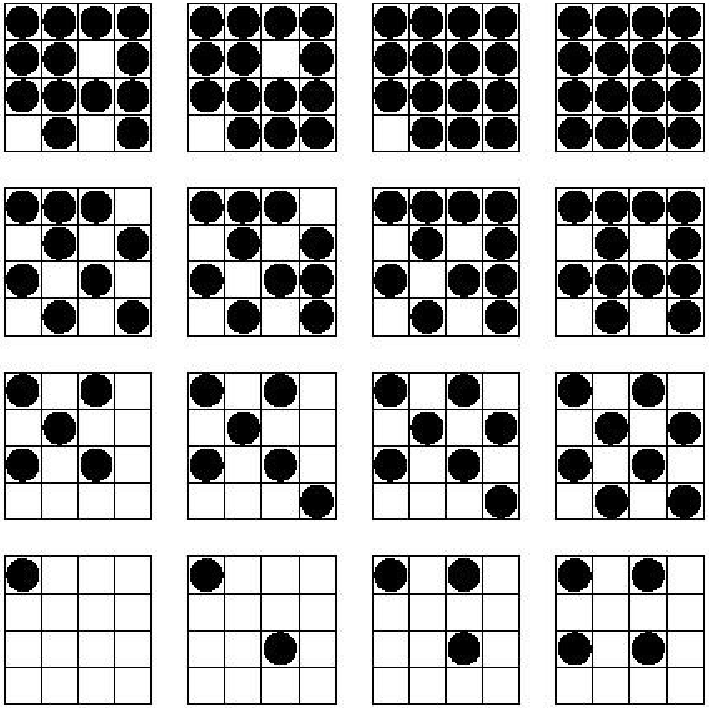
Blue

Problemi s šumom

- Difficult to compute quickly.
- Not reproducible.
- Pre-compute pseudo-random function and store in table.
- Small tiled patterns sufficient

Urejeno razprševanje (Ordered Dithering)

- Trade off spatial resolution for intensity resolution.
- Use dither patterns.
- Can be represented as a matrix.

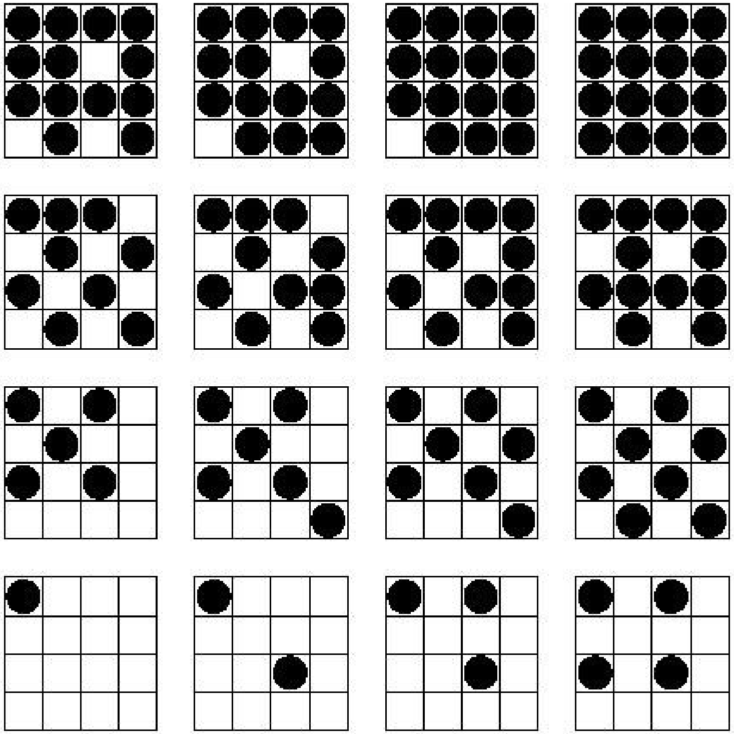


Bayerjevi vzorci urejenega razprševanja

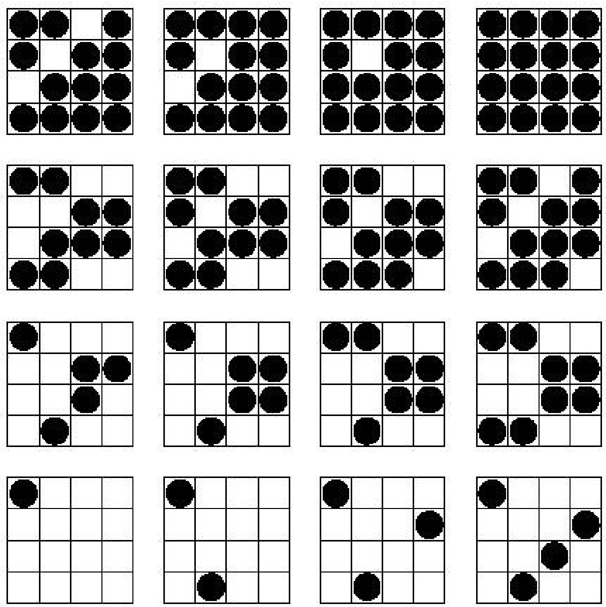
$$D_2 = \begin{bmatrix} 0 & 2 \\ 3 & 1 \end{bmatrix}$$

$$D_n = \begin{bmatrix} 4D_{n/2} + 0 & 4D_{n/2} + 2 \\ 4D_{n/2} + 3 & 4D_{n/2} + 1 \end{bmatrix}$$

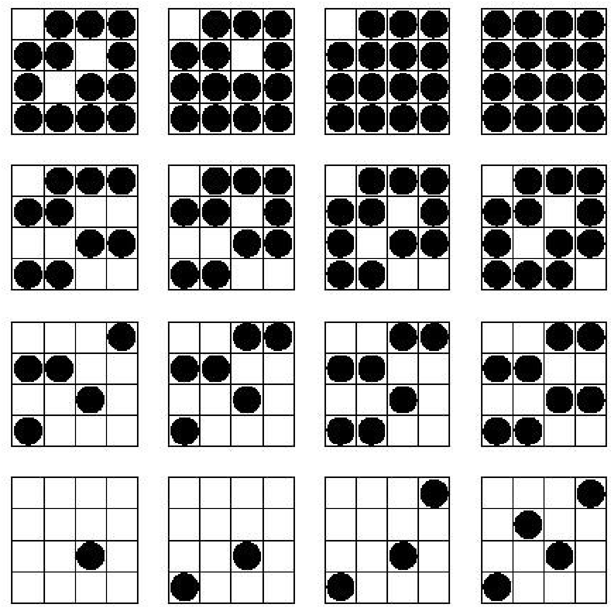
$$D_4 = \begin{bmatrix} 0 & 8 & 2 & 10 \\ 12 & 4 & 14 & 6 \\ 3 & 11 & 1 & 9 \\ 15 & 7 & 13 & 5 \end{bmatrix}$$



Druge možnosti



$$M_4 = \begin{bmatrix} 0 & 6 & 13 & 11 \\ 9 & 15 & 4 & 2 \\ 14 & 8 & 3 & 5 \\ 7 & 1 & 10 & 12 \end{bmatrix}$$



$$P_4 = \begin{bmatrix} 15 & 8 & 5 & 2 \\ 4 & 3 & 14 & 9 \\ 10 & 13 & 0 & 7 \\ 1 & 6 & 11 & 12 \end{bmatrix}$$

Primerjava

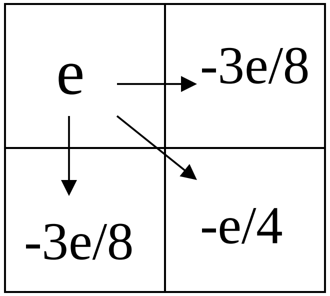
Blue Noise

Ordered Dither

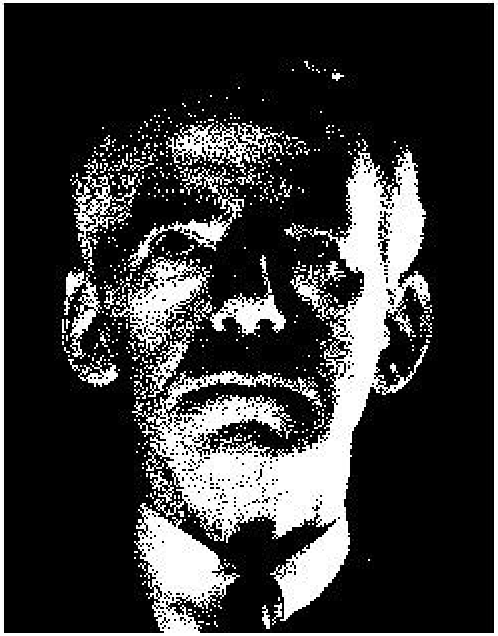
Širjenje napake

Idea: Quantize, then distribute error to neighbours

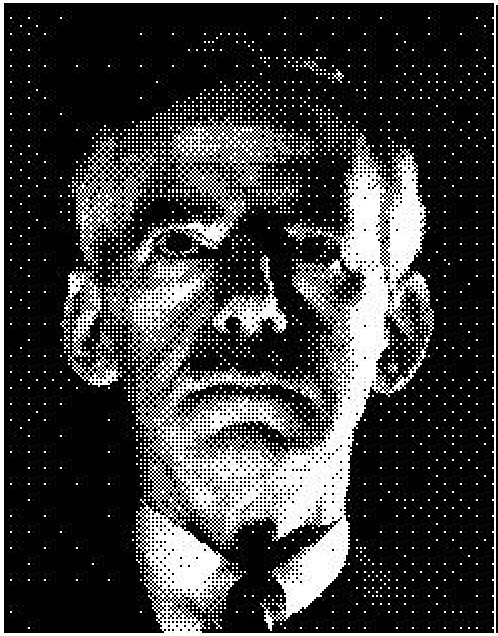
```
for(y=0; y<ny; y++)
 for(x=0; x<nx; x++){
  vq[x][y] = quantize(v[x][y]);
  e = v[x][y] - vq[x][y];
  i[x+1][y] += 3/8*e;
  i[x][y+1] += 3/8*e;
  i[x+1][y+1] += 1/4*e;
}
```



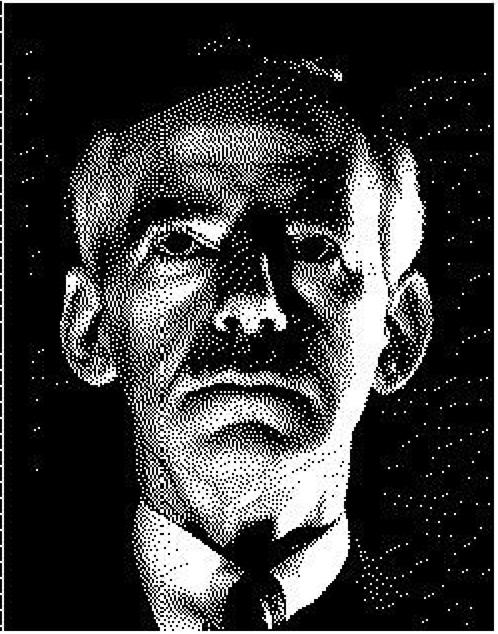
Primerjava



Blue Noise



Bayer Dither



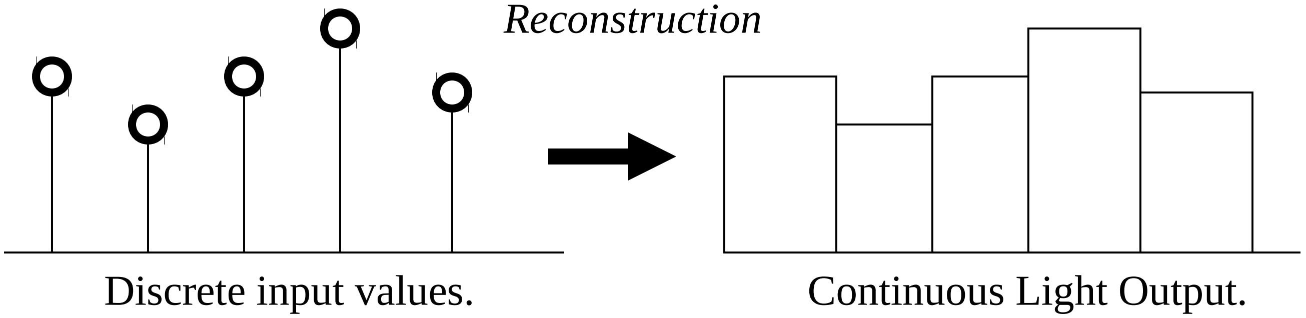
Floyd-Steinberg

Vzorčenje in aliasing

- Real world is continuous
- The computer world is discrete
- Mapping a continuous *function* to a discrete one is called *sampling*
- Mapping a continuous *variable* to a discrete one is called *quantizaion*
- To represent or render an image using a computer, we must both sample and quantize

Prikazovalniki→ rekonstrukcija signala

- All physical displays recreate a continuous image from a discrete sampled image by using a finite sized source of light for each pixel.



Spomnimo se: piksel je točka…

- It is NOT a box, disc or teeny wee light
- It has no dimension
- It occupies no area
- It can have a coordinate
- *More* than a point, it is a *SAMPLE*

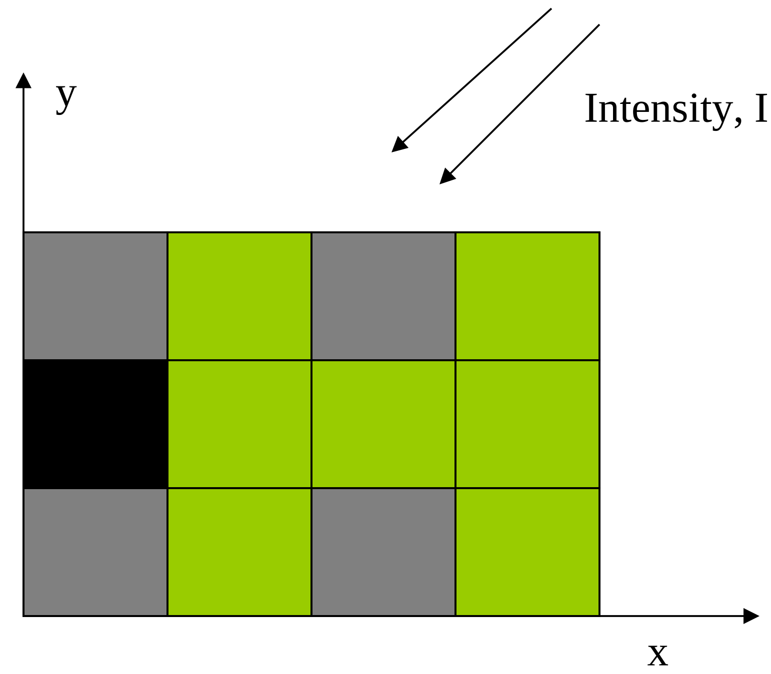
Vzorčenje v video kamerah

- Video camera : CCD array.

Regular array of imaging sensors.

Value sensed is an area integral over a pixel.

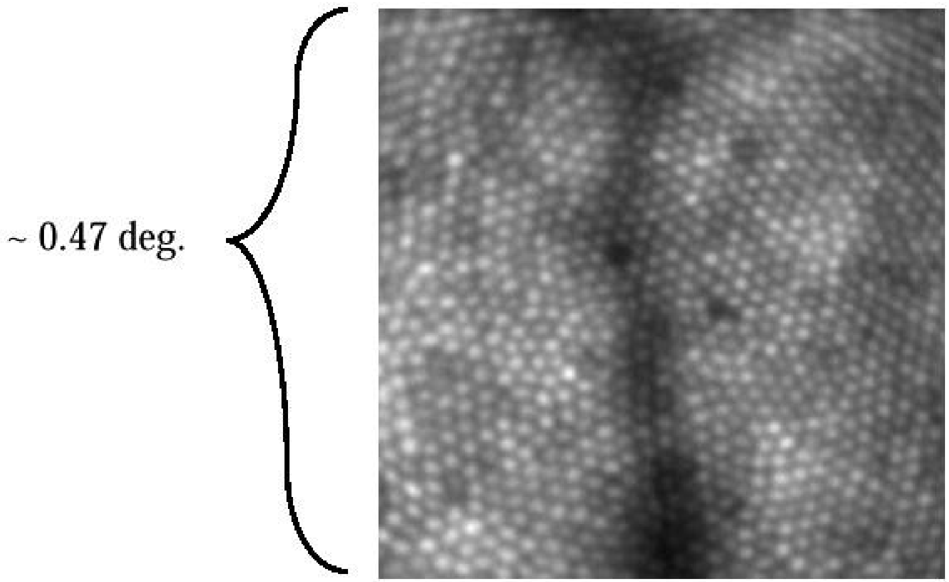
$$V = k \iint_{x,y} I\,dx\,dy$$



Vzorčenje v očesu

- The eye : photoreceptors

Film is similar : irregular array of receptors.



J. Liang, D. R. Williams and D. Miller, "Supernormal vision and high- resolution retinal imaging through adaptive optics," J. Opt. Soc. Am. A 14, 2884- 2892 (1997)

Ali je pri računalniški grafiki drugače ?

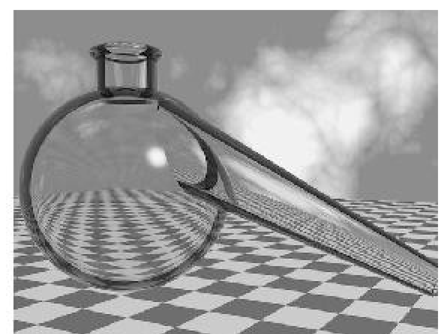
Model of Scene.

- points
- lines
- 3D objects
- lighting etc

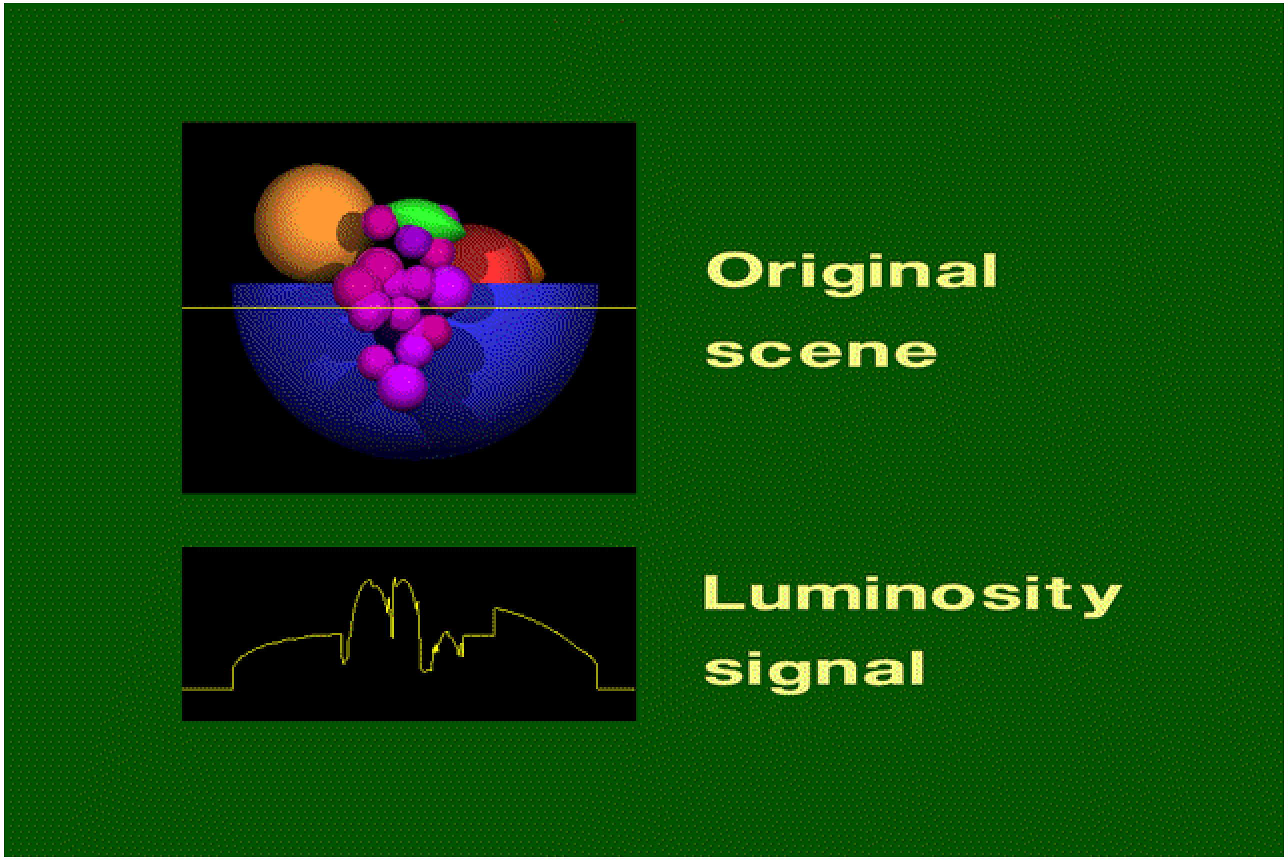
Query values of pixels

Frame Buffer

Point Sampling.

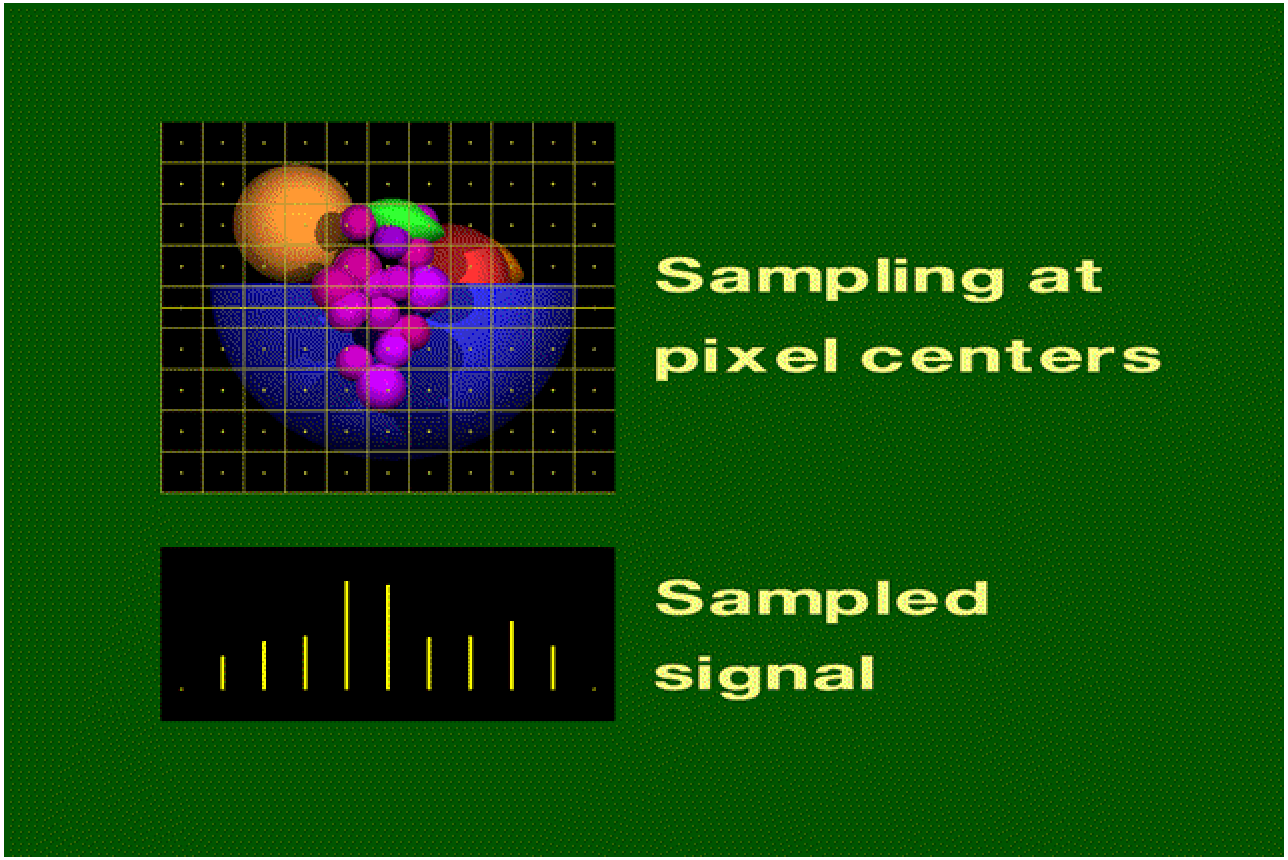


Zvezen svetlostni signal

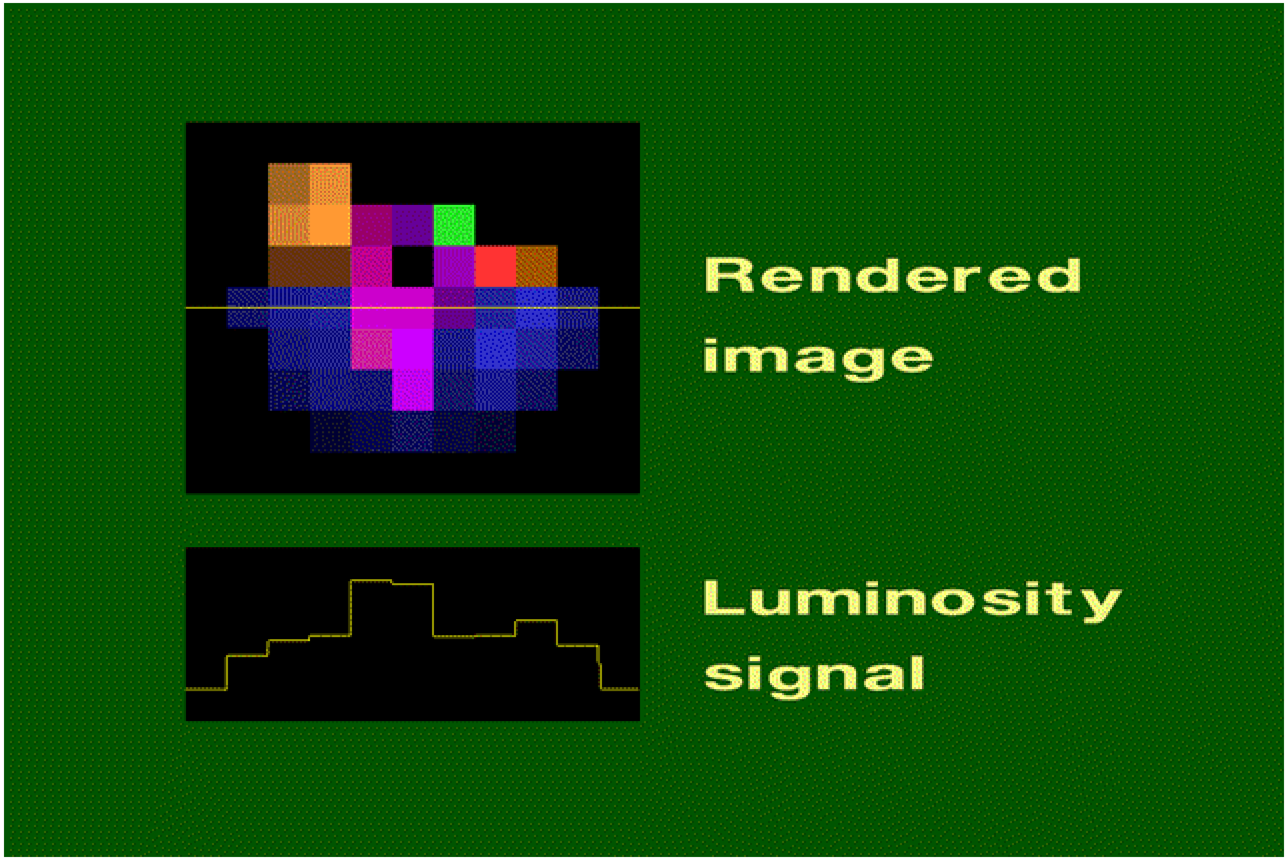


Slide © Rosalee Nerheim-Wolfe

Vzorčena svetlost

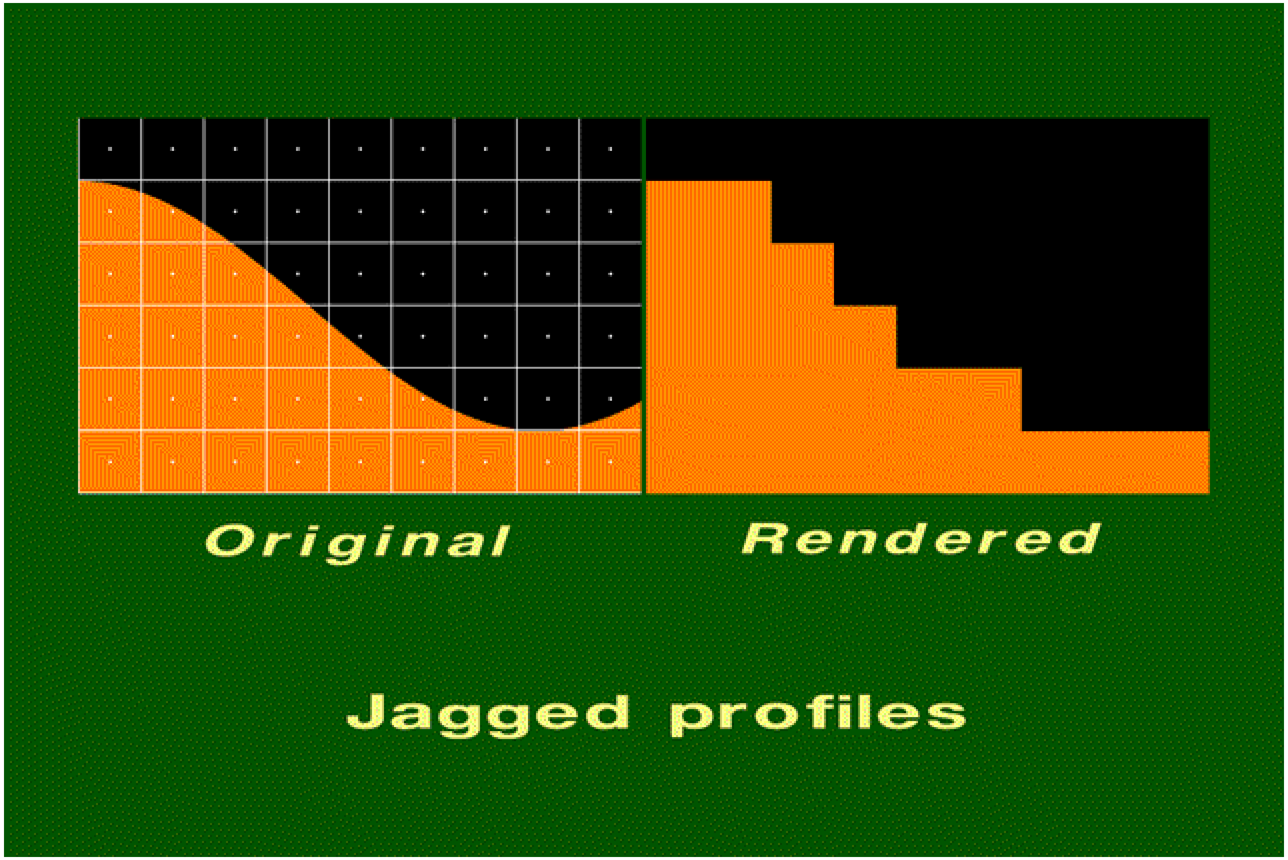


Rekonstruirana svetlost



Slide © Rosalee Nerheim-Wolfe

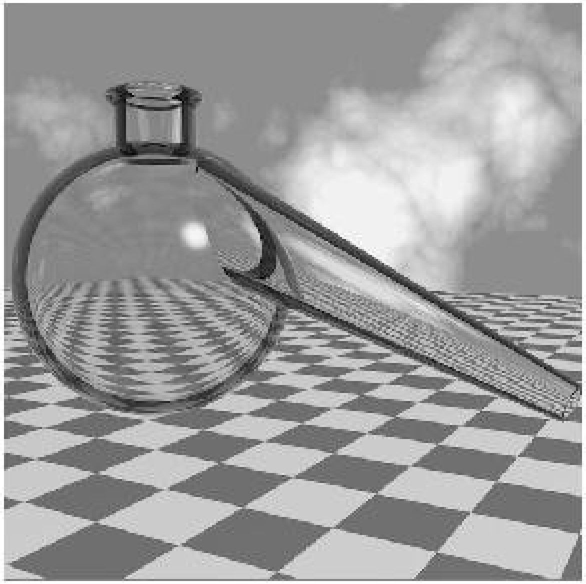
Rezultat rekonstrukcije

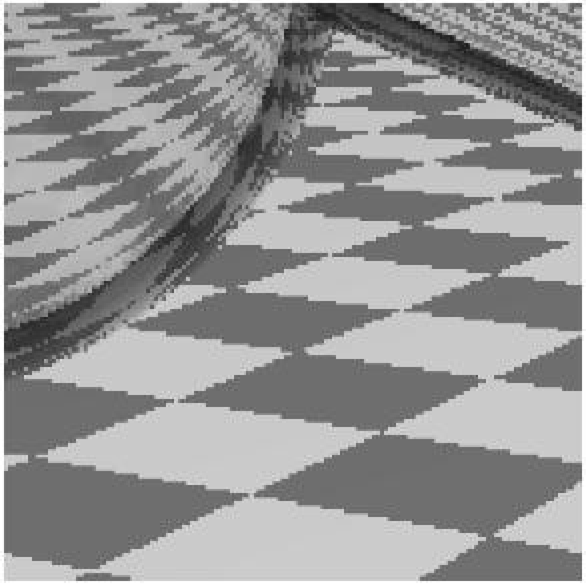


Slide © Rosalee Nerheim-Wolfe

Nazobčanost

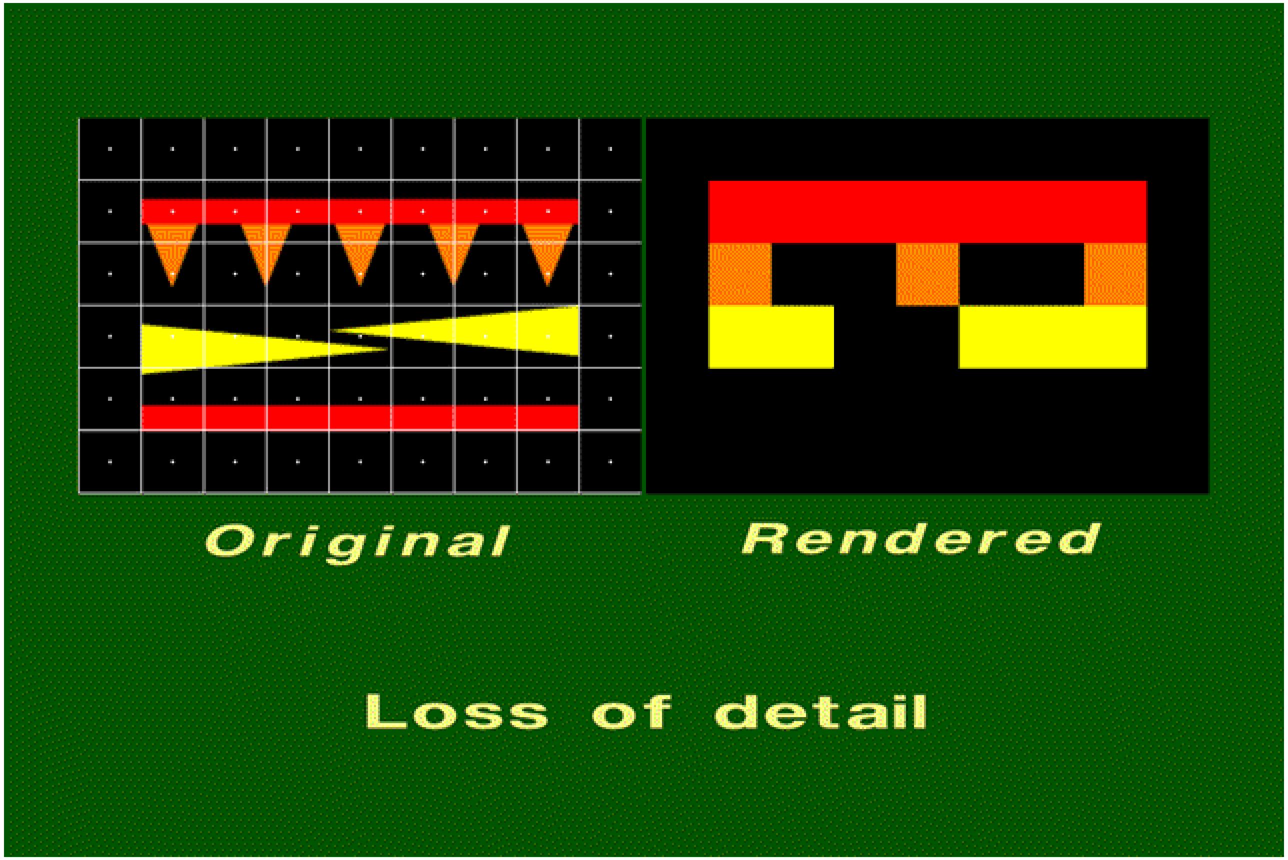
The raster *aliasing* effect – removal is called *antialiasing*





Images by Don Mitchell

Je lahko resen problem…



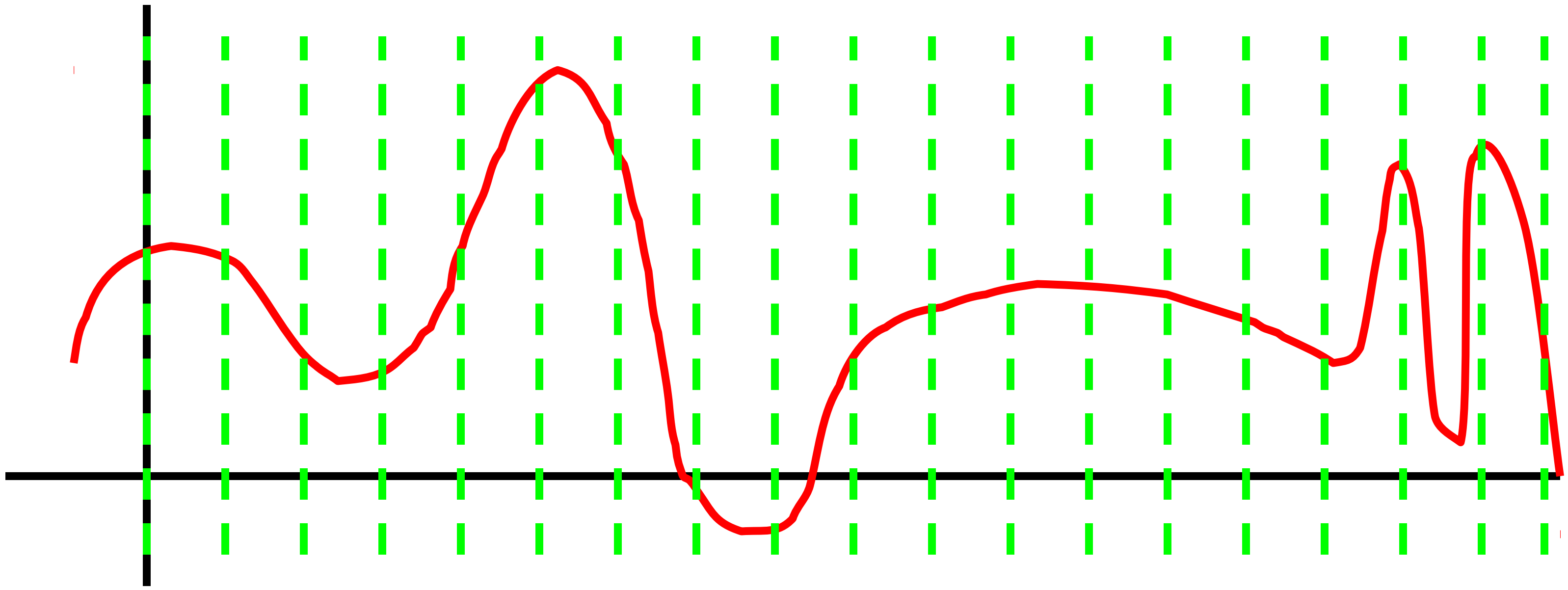
Slide © Rosalee Nerheim-Wolfe

...zelo resen problem!

Slide © Rosalee Nerheim-Wolfe

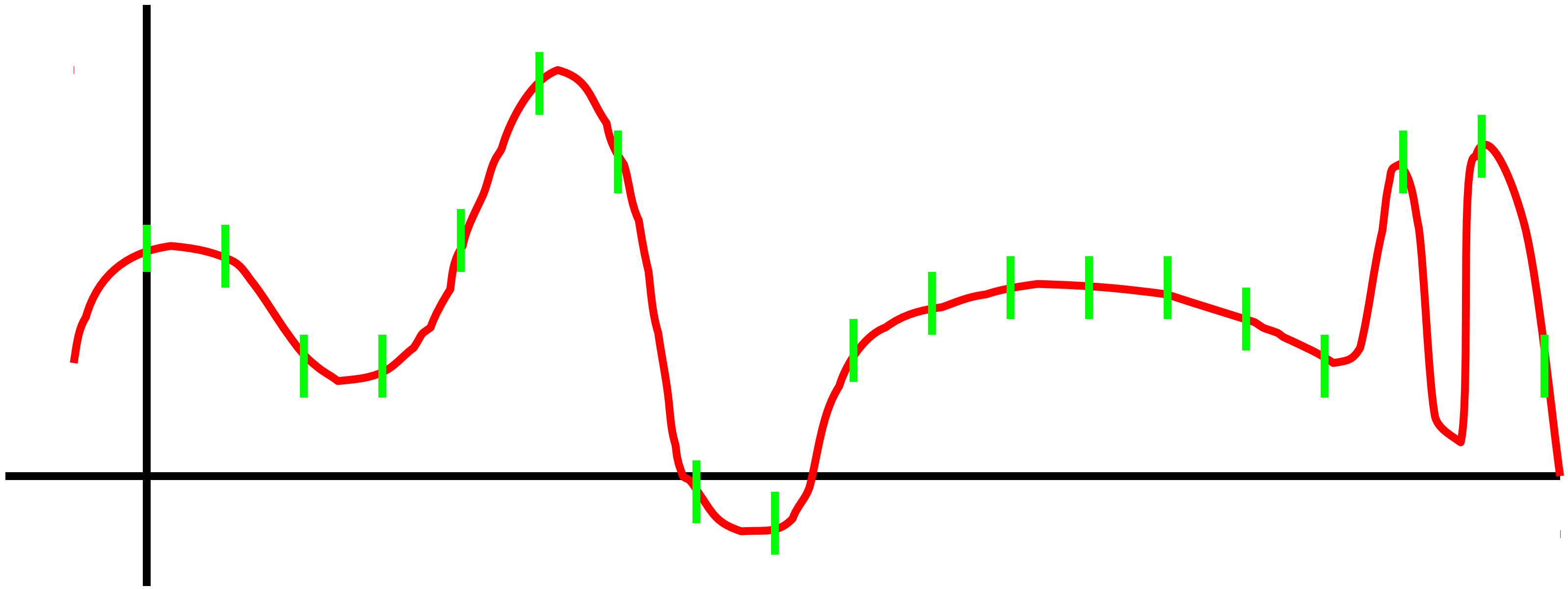
Podvzorčenje v 1D prostoru

- Sampling a 1-D function:



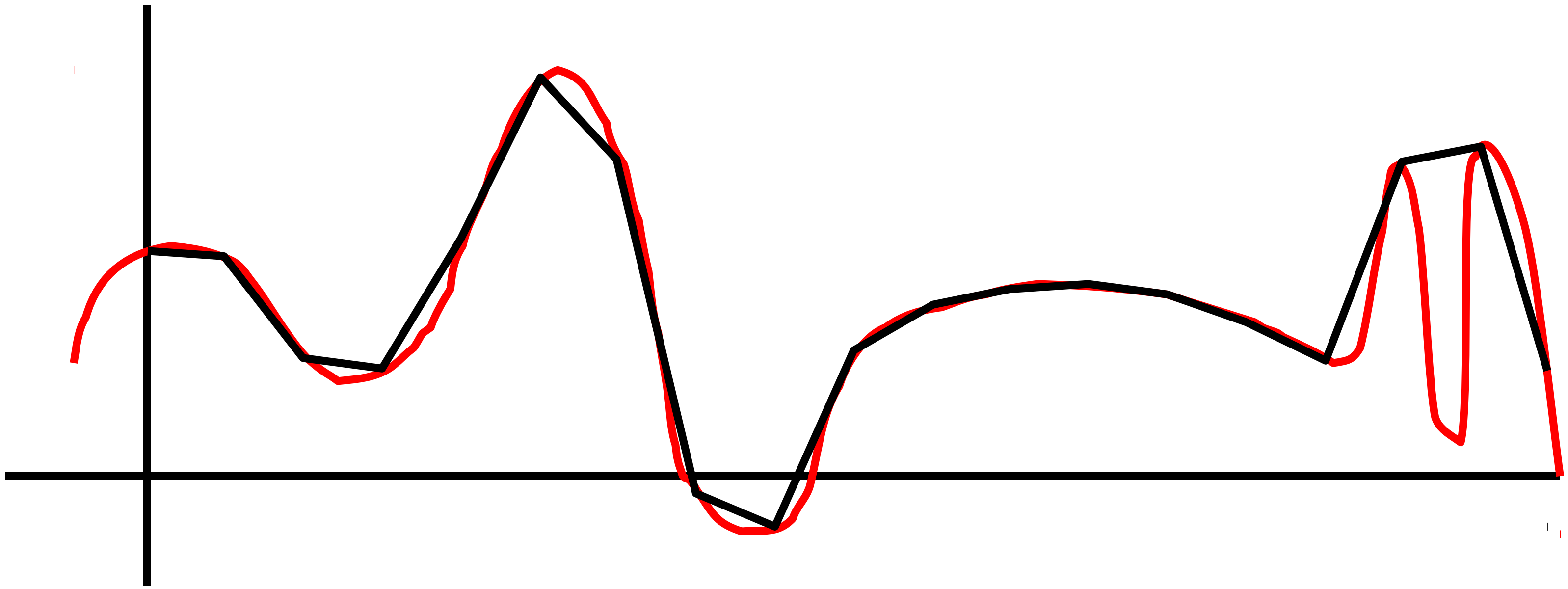
Podvzorčenje v 1D prostoru

- Sampling a 1-D function:



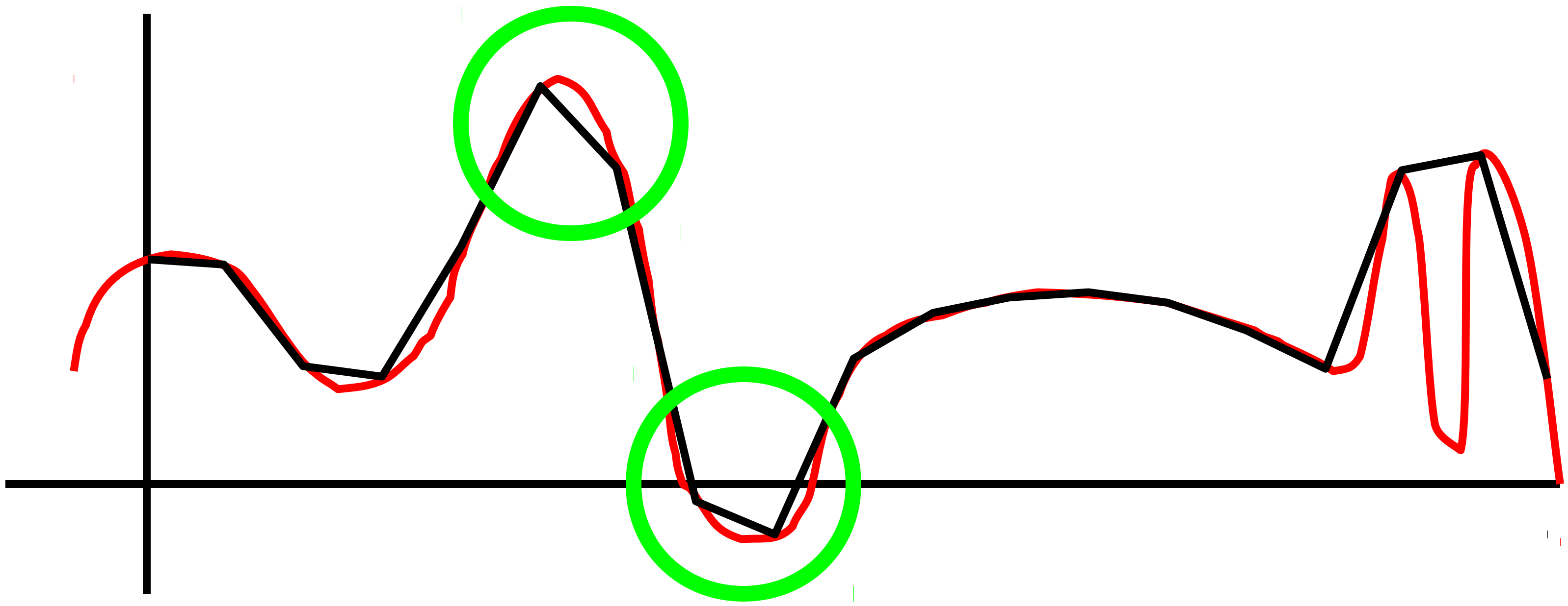
Podvzorčenje v 1D prostoru

- Sampling a 1-D function:
  - What do you notice?



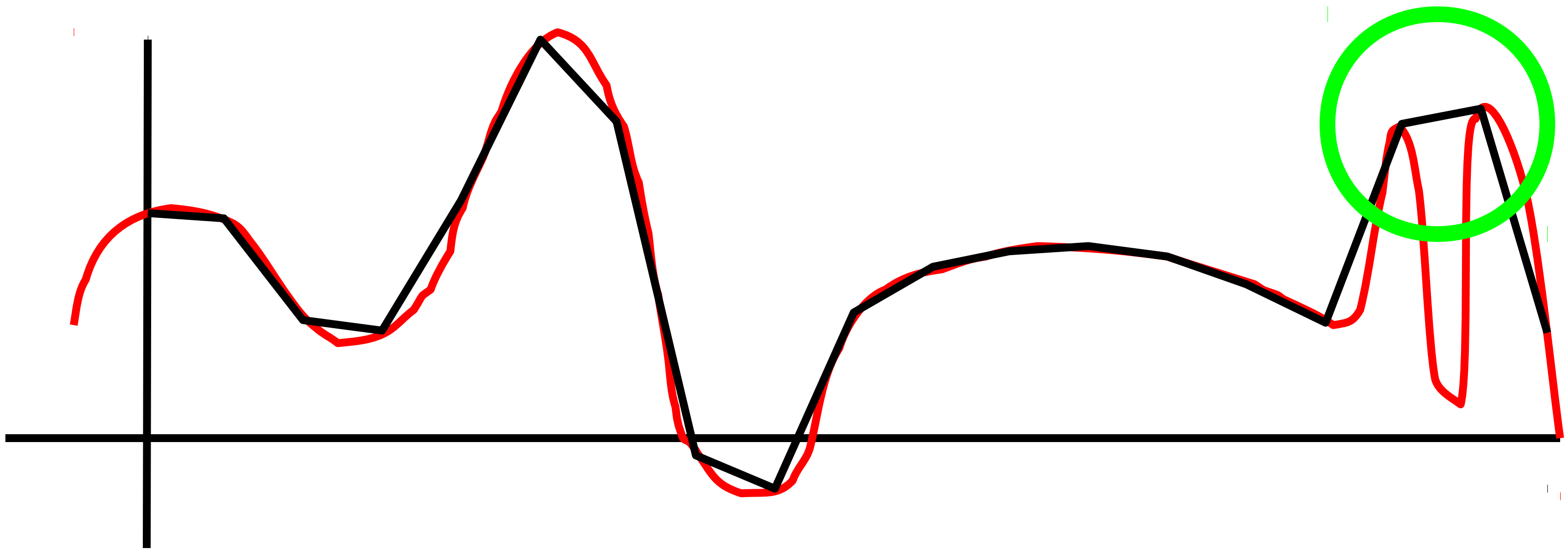
Podvzorčenje v 1D prostoru

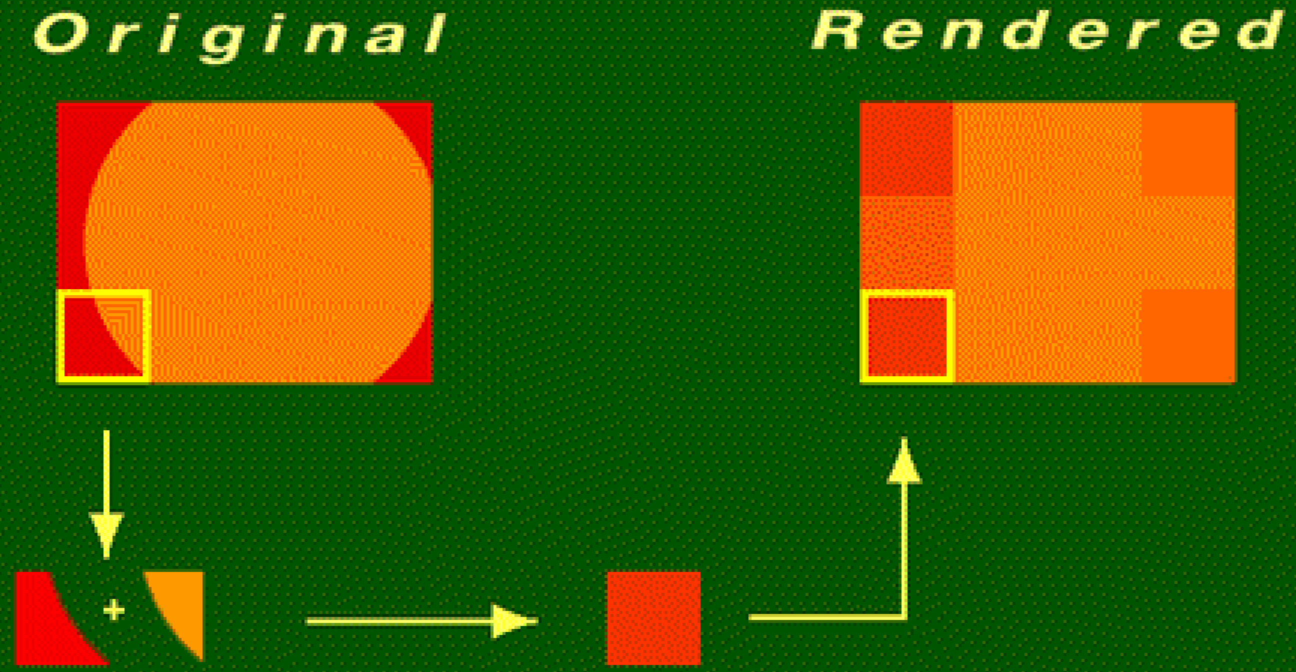
- Sampling a 1-D function: what do you notice?
  - Jagged, not smooth



Podvzorčenje v 1D prostoru

- Sampling a 1-D function: what do you notice?
  - Jagged, not smooth
  - Loses information!





Prefiltering methods examine areas of color within a pixel.

$\frac{1}{16}$	$\frac{1}{8}$	$\frac{1}{16}$
$\frac{1}{8}$	$\frac{1}{4}$	$\frac{1}{8}$
$\frac{1}{16}$	$\frac{1}{8}$	$\frac{1}{16}$

Combines nine samples

Filters combine samples to find a pixel's color.

Hello World

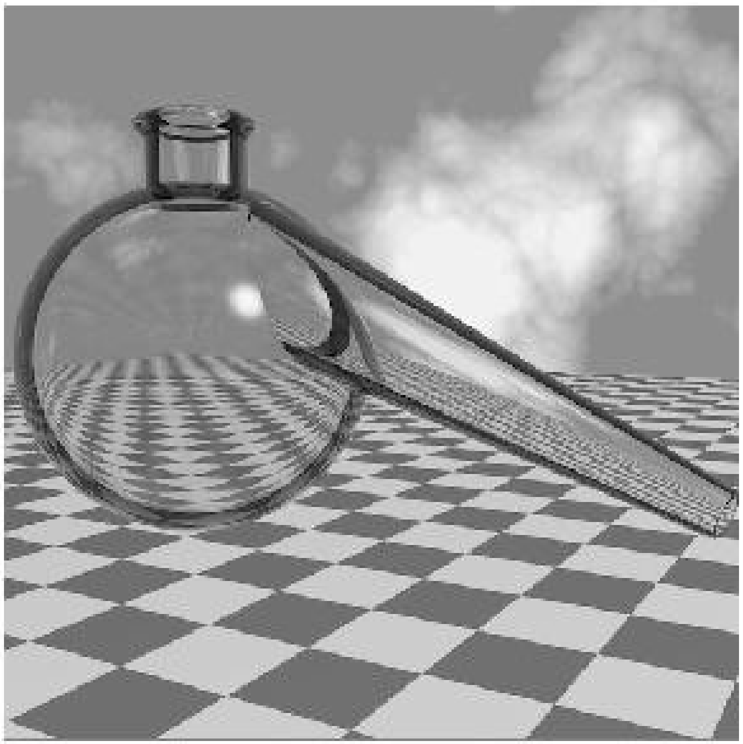
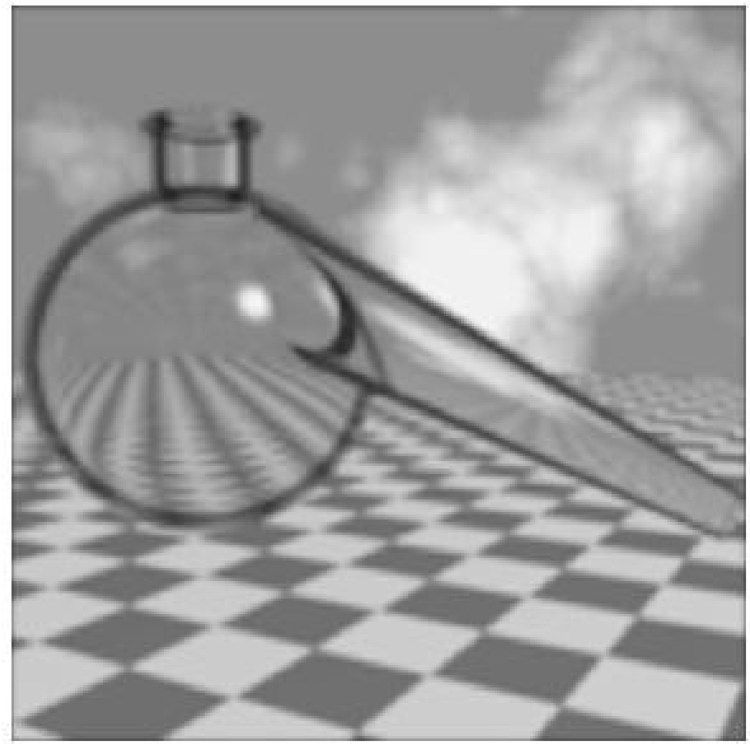
Hello World

A demonstration

No antialiasing

Prefiltering

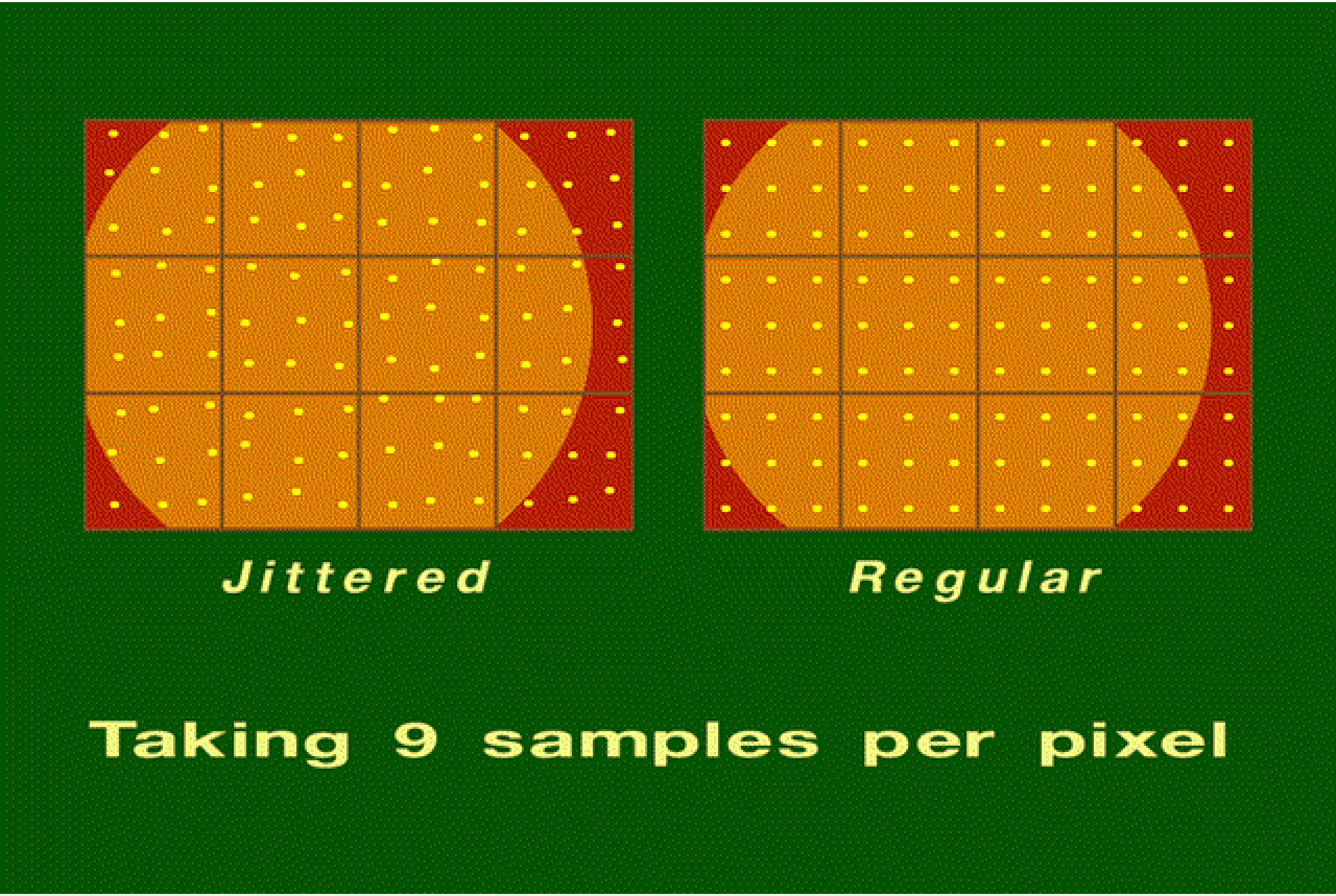
Zameglitev (blurring) ni dobra

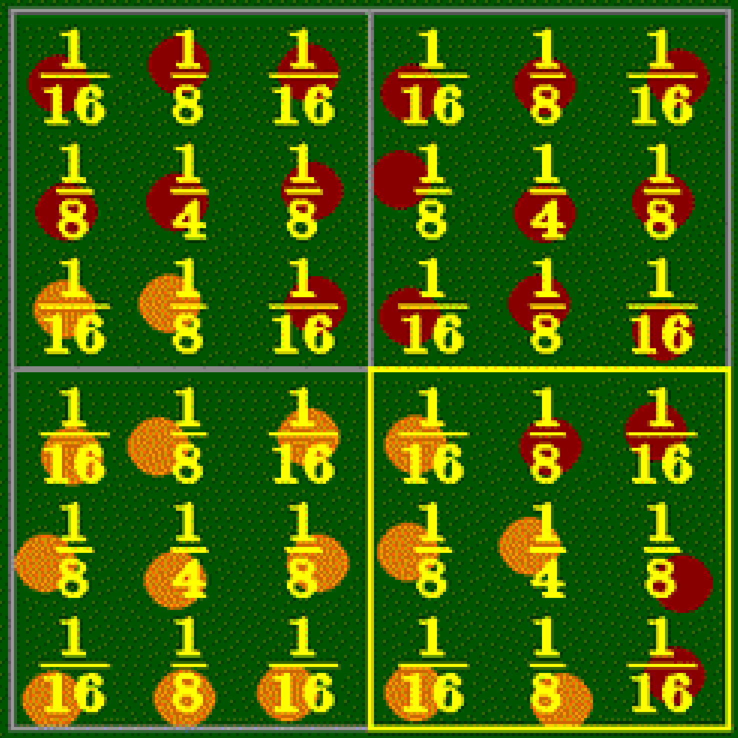
Removed the *jaggies*, but also all the detail ! $\longrightarrow$ Reduction in resolution

nadvzorčenje (supersampling) ?

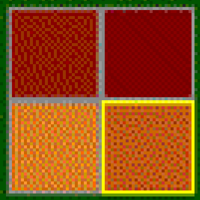
- solution : take multiple samples for each pixel and average them together → supersampling.
- Can weight them towards the centre → weighted average sampling
- Stochastic sampling


Jittered
Regular
Taking 9 samples per pixel

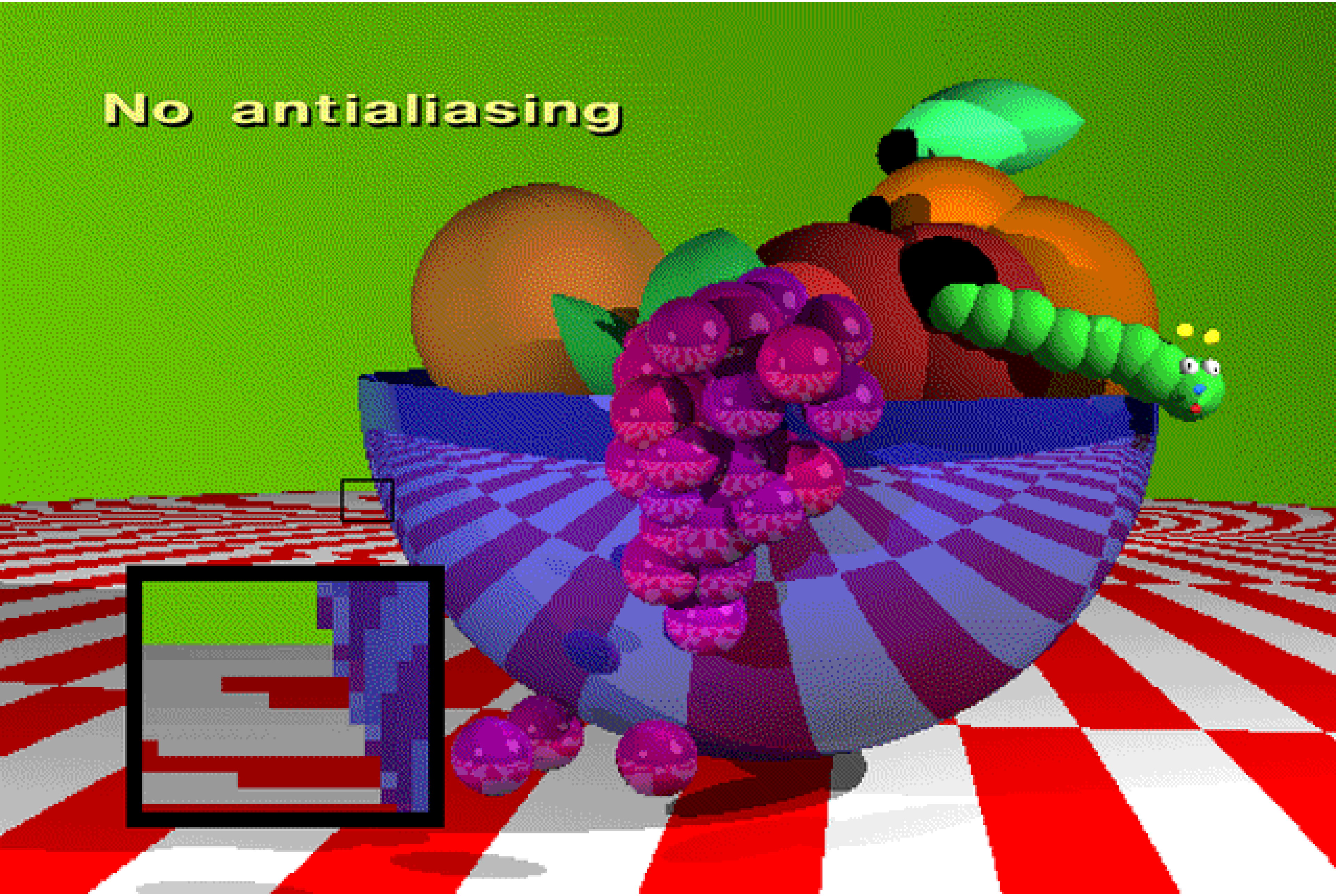
This filter computes a weighted average.

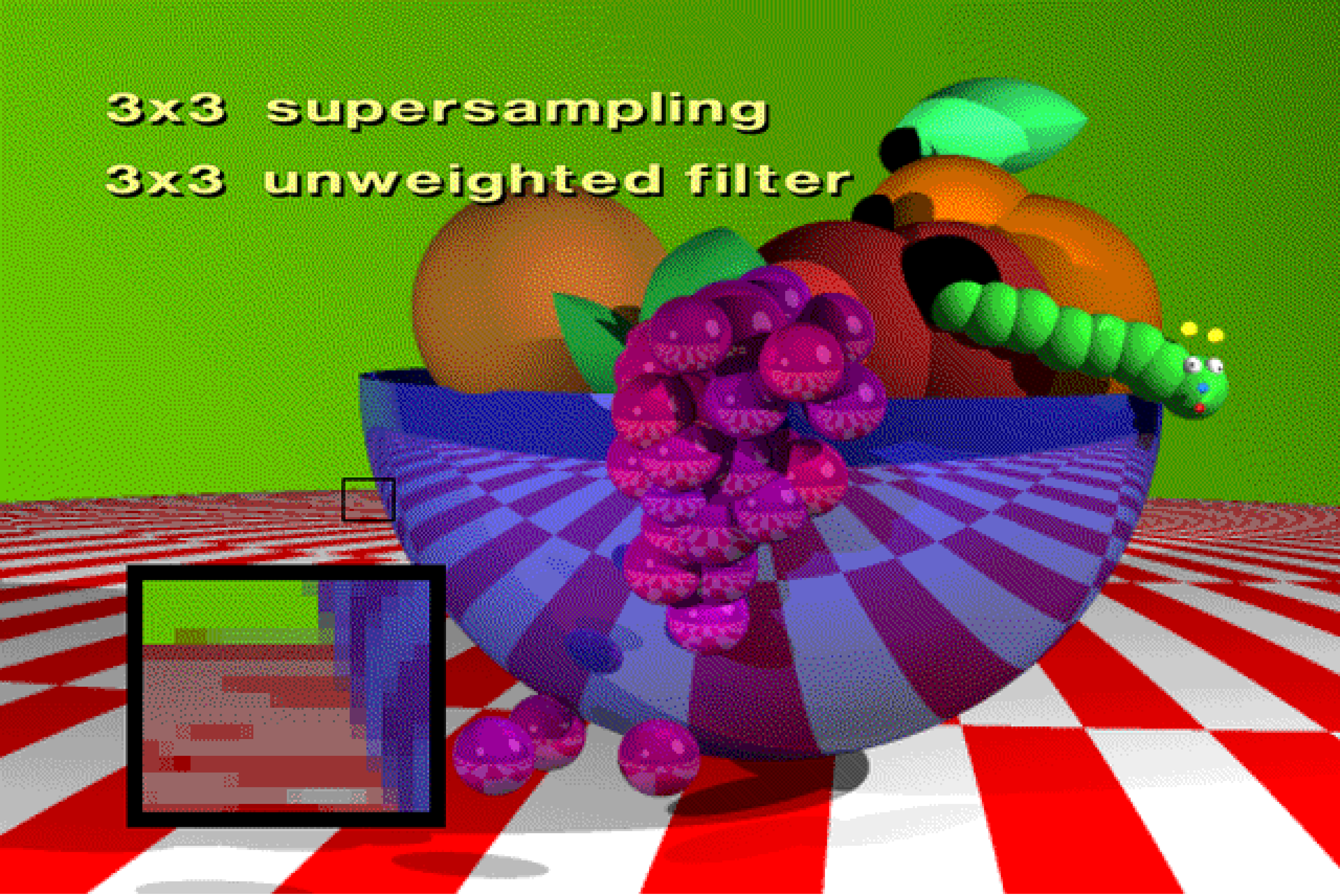


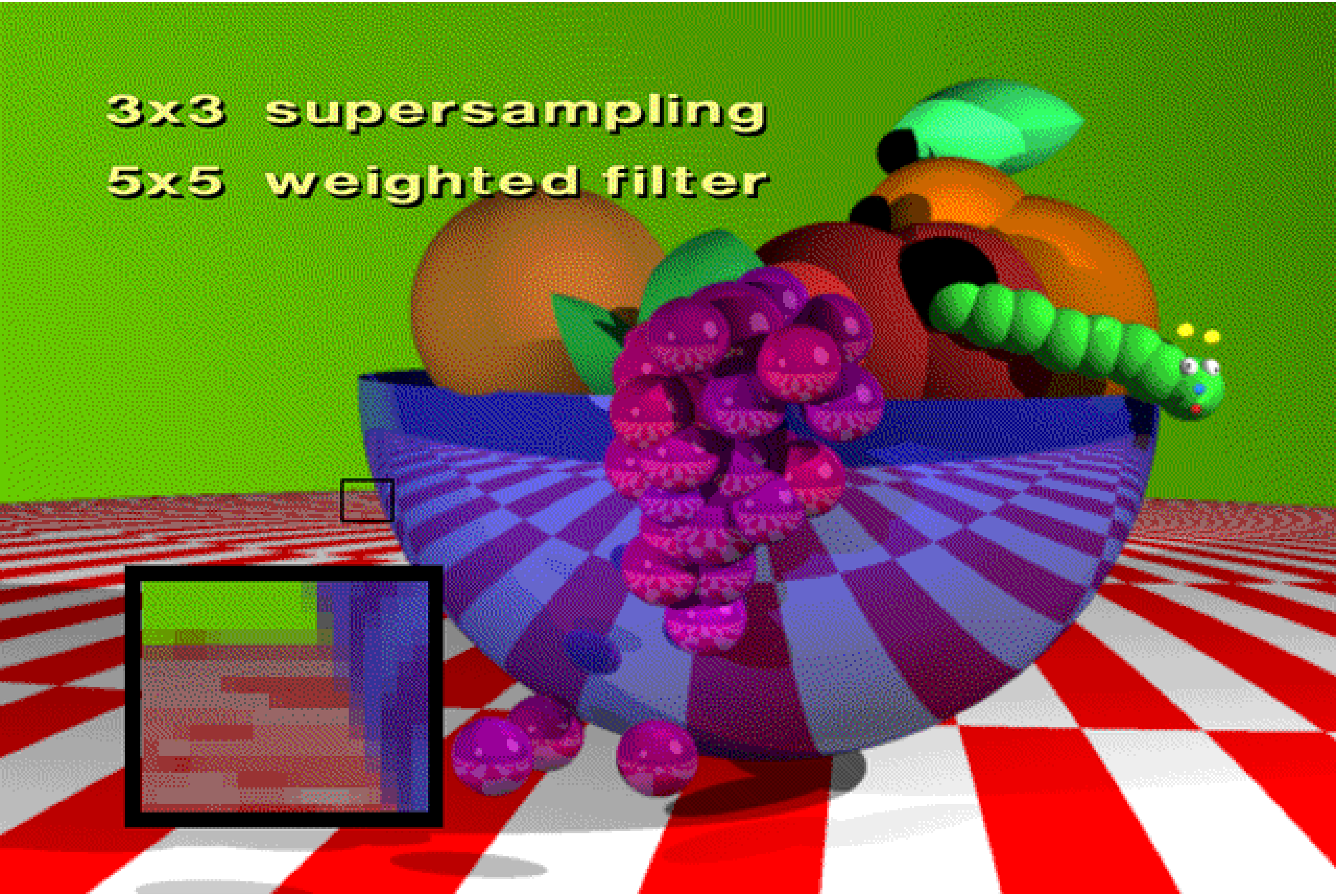
Samples



Pixels


No antialiasing


3x3 supersampling
3x3 unweighted filter


3x3 supersampling
5x5 weighted filter

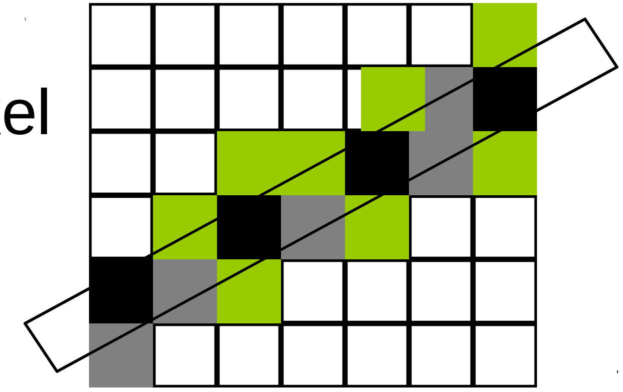

3x3 jittered supersampling
5x5 weighted filter

Vzorčenje prostora je drago

- Computationally expensive.
- Difficult to do analytically in a 3D environment
  - Single sample is a ray in 3D - easy computation of intersections.
- Area sample becomes a cone !
  - Reflections difficult.
- Is it an optimum solution ?

Antialiasing with Area Sampling

- A scan converted primitive occupies finite area on the screen
- Intensity of the boundary pixels is adjusted depending on the percent of the pixel area covered by the primitive. This is called weighted area sampling
- Intensities can be weighted depending on the distance of the area from the center of the pixel

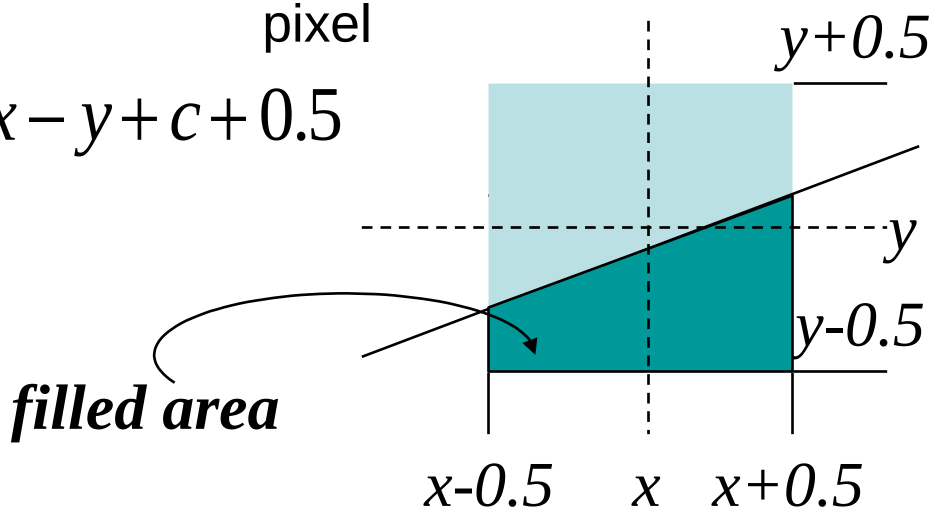


Antialiasing with Area Sampling

- Methods to estimate percent of pixel covered by the pixel
  - subdivide pixel into sub-pixels and determine how many sub-pixels are inside the boundary
  - Incremental line algorithm can be extended, with area calculated as

Efficient only when one edge of the area passes through the pixel

$$Area = m \cdot x - y + c + 0.5$$



Kako še naredimo antialiasing?

- We need some mathematical tools to
  - analyse the situation.
  - find an optimum solution.
- Tools we will use :
  - Fourier transform.
  - Convolution theory.
  - Sampling theory.

Nekaj matematike – spektralna analiza

- Spectral representation treats the function as a weighted sum of sines and cosines
- Every function has two representations
  - Spatial (time) domain - normal representation
  - Frequency domain - spectral representation
- The ✧❑◆❒❊❈❒ ▼❒❁■▲❈❑❒❍ converts between the spatial and frequency domains.

Nekaj matematike – spektralna analiza

- The ✧❑◆❒✱❈❒ ▼❒❆■▲❈❑❒❍ converts between the spatial and frequency domain.

Spatial domain

$$F(\omega) = \int_{-\infty}^{\infty} f(x) e^{-i\omega x} dx$$

$$f(x) = \frac{1}{2\Pi} \int_{-\infty}^{\infty} F(\omega) e^{i\omega x} d\omega$$

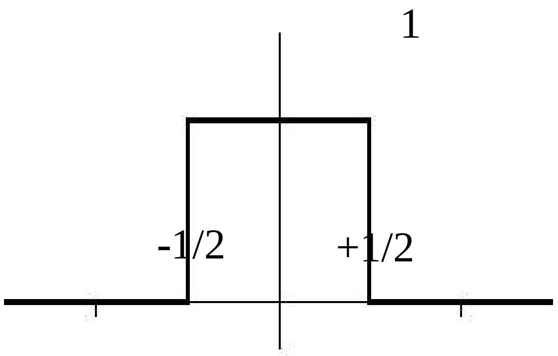
Frequency domain.

$$e^{it} = \cos t + i \sin t$$

- Note the Euler formula :
- Real and imaginary components.
- Forward and reverse transforms very similar.
  - Reversal in sign of imaginary component, scale constant.

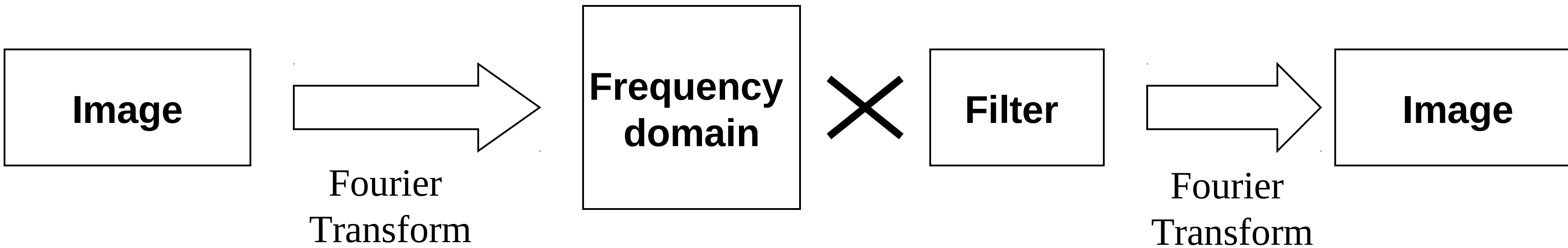
Some important properties of Fourier transforms.

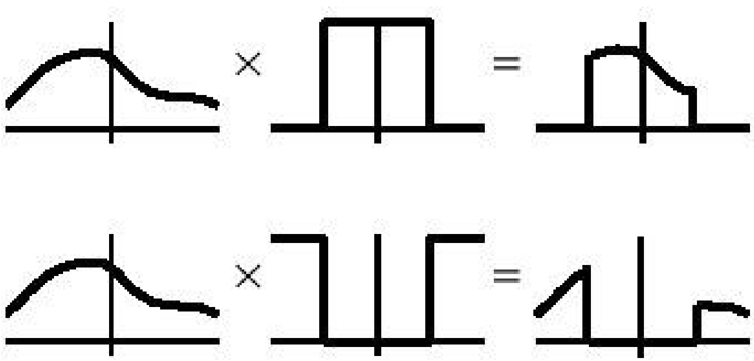
- Finite function ⇔ Infinite function.
  - eg. single sine wave ⇔ single point or ‘delta’ function.
- Square or *boxcar* function - corresponds to 1 pixel.



Function called :
Square(x)

Filtering in the frequency domain



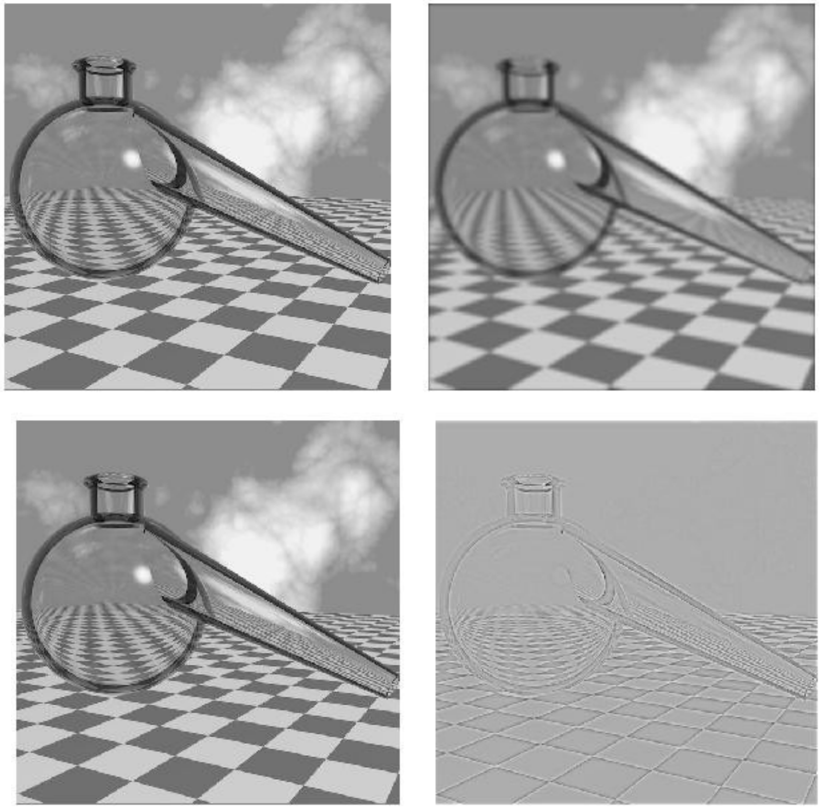


Lowpass filter

Highpass filter

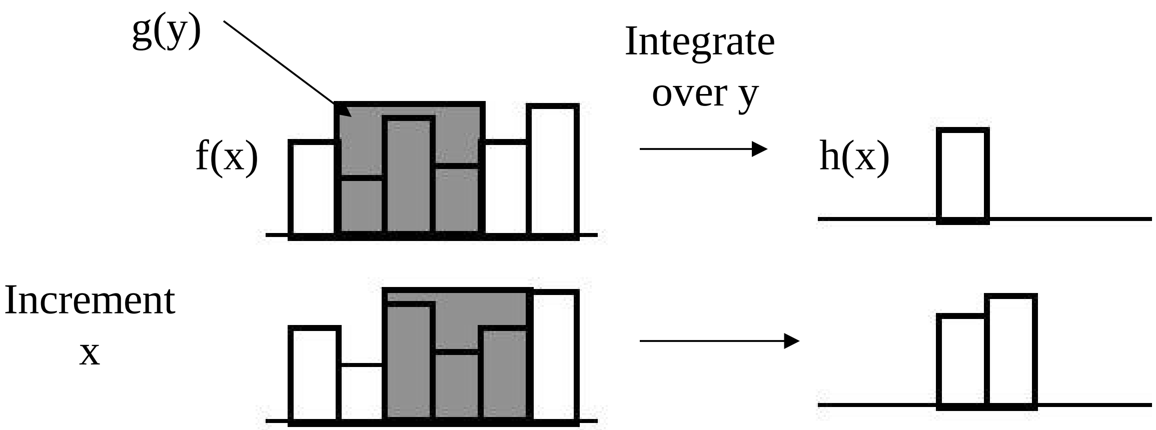
Filtriranje

- Low pass
- High pass



Filtriranje v časovnem prostoru

- Blurring or averaging pixels together.



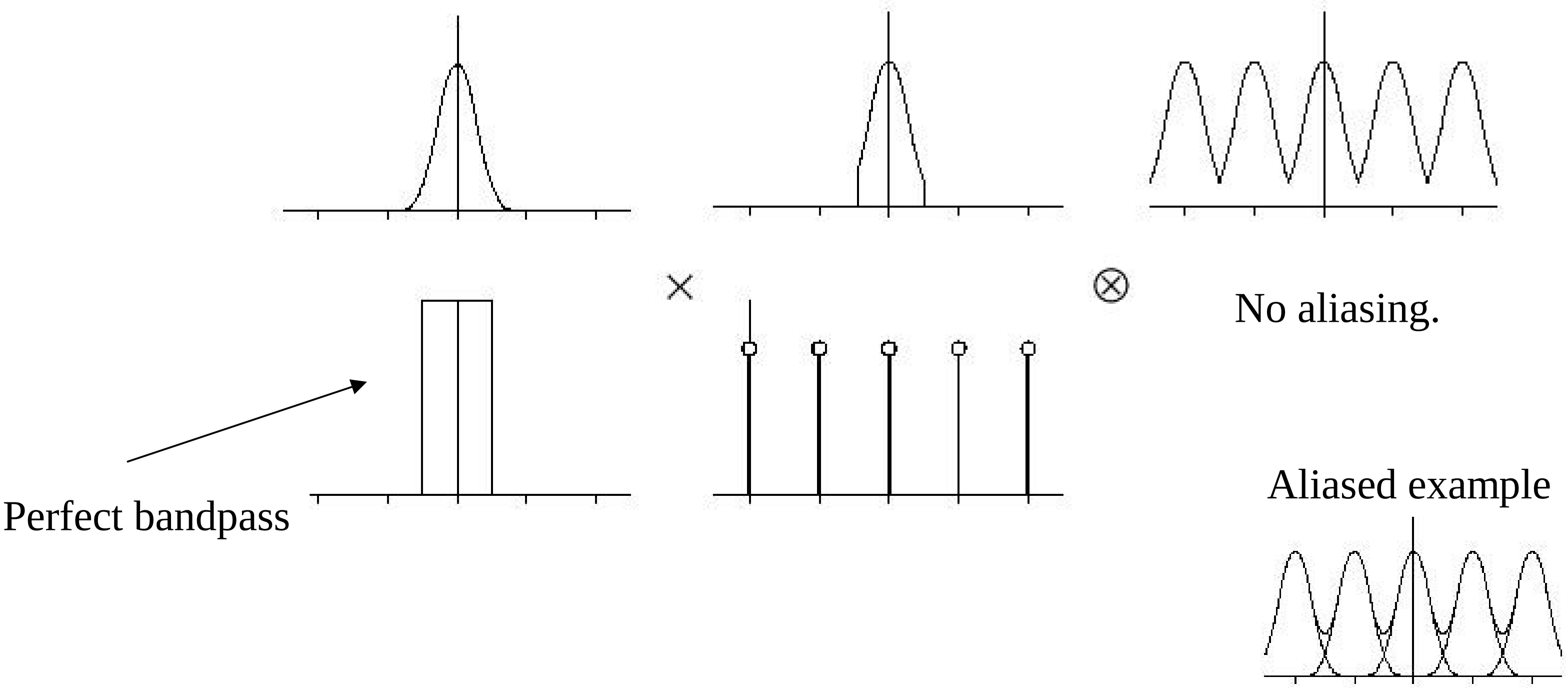
Calculate integral of one function, f(x) by a sliding second function g(x-y).

Known as *Convolution*.

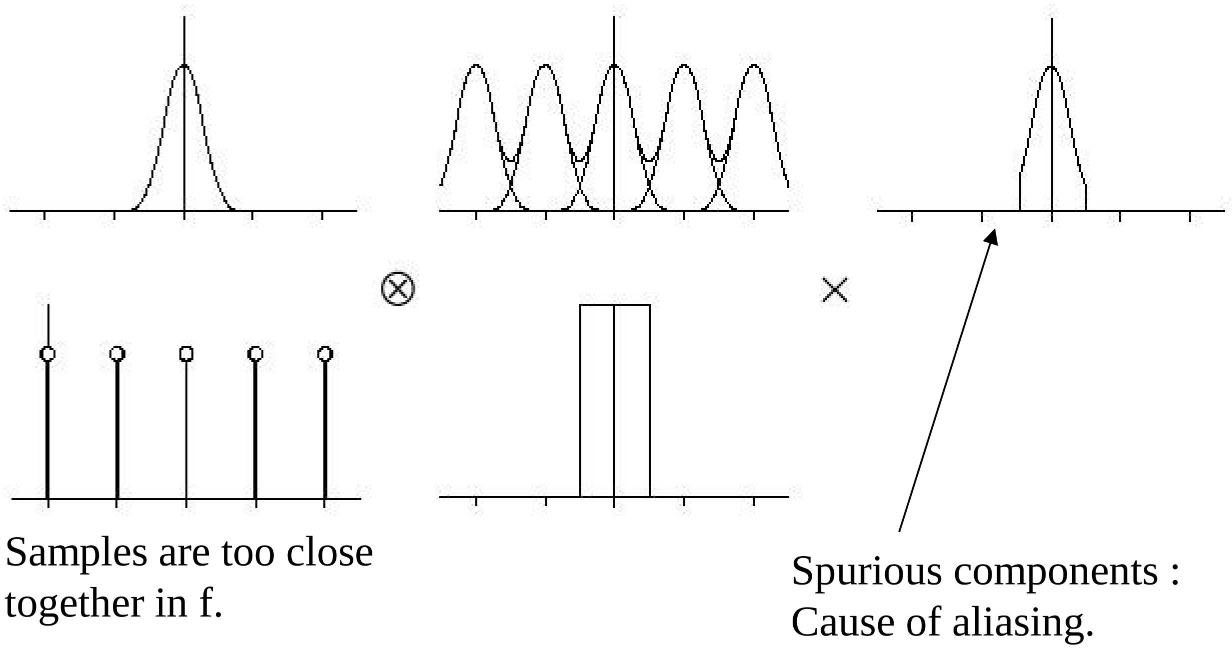
$$h(x) = f \otimes g = \int f(x) g(x-y) dy$$

Kako odstraniti aliasing ?

- Perfect solution - prefilter with perfect bandpass filter.

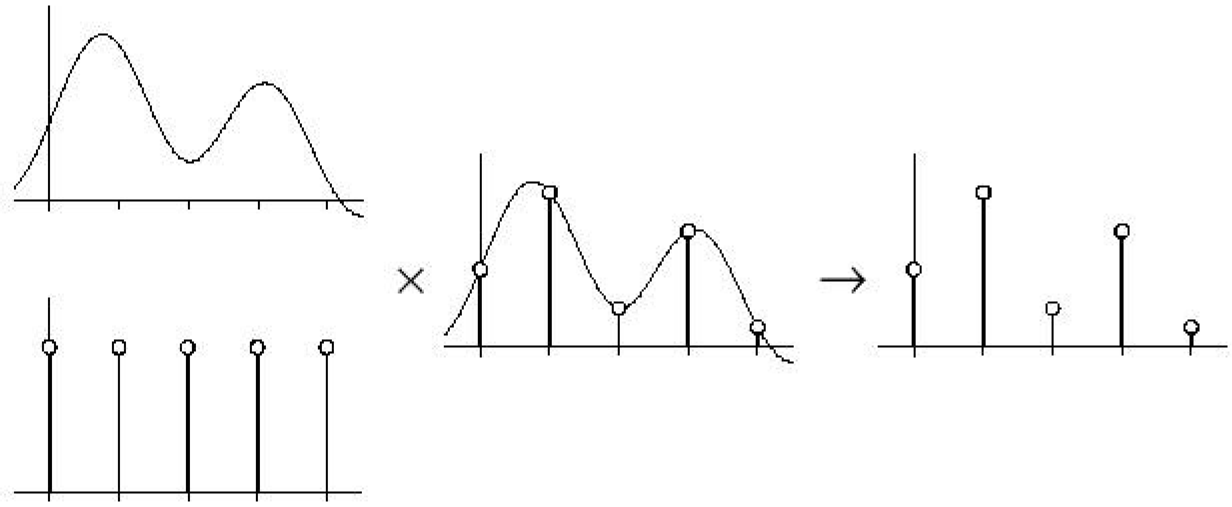


Do aliasing pripelje podvzorčenje



Kako predstavimo vzorčenje ?

- Multiplication of the sample with a regular train of delta functions.

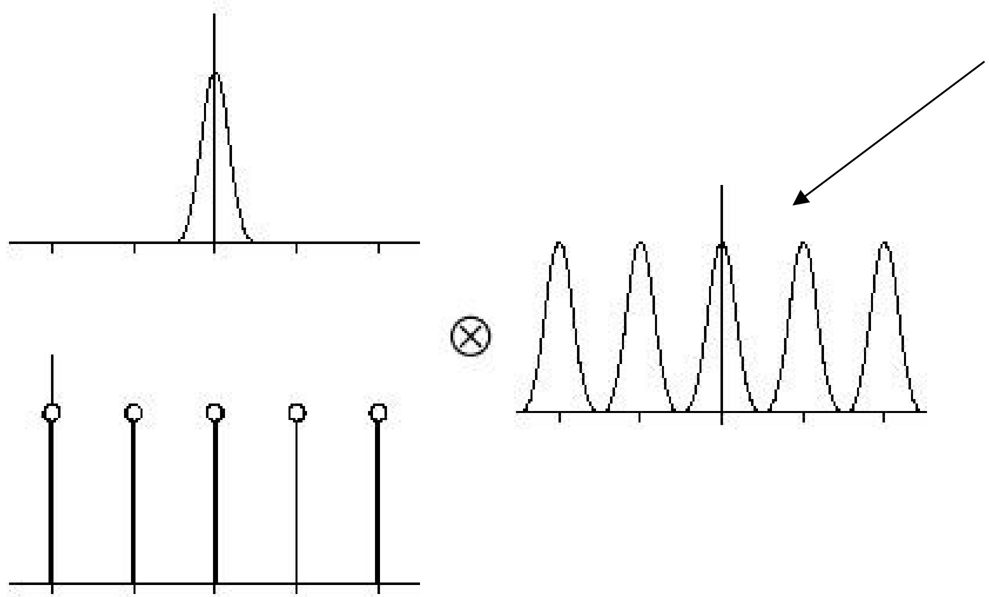


Vzorčenje – frekvenčni prostor

Need Fourier transform of regular train of delta functions.

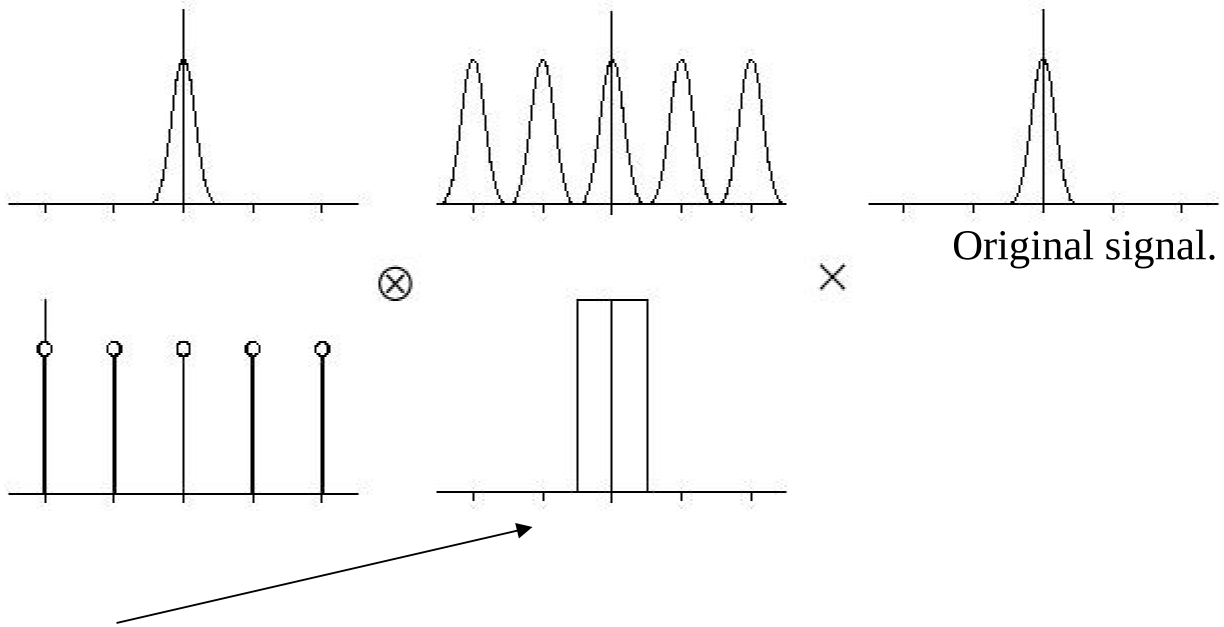
- A regular train of delta functions - spacing is inversely proportional

So convolve with this function.



Multiple solutions at regularly increasing values of f

Rekonstrukcija v frekvenčnem prostoru



Bandpass filter due to regular array of pixels.

Kako odstranimo aliasing ?

- Perfect solution - prefilter with perfect bandpass filter.
  - Difficult/Impossible to do in frequency domain
- Convolve with sinc function in space domain
  - Optimal filter - better than area sampling.
  - Sinc function is infinite !!
  - Computationally expensive.