Filters in Image Processing

Fourier transform in two dimensions & (Re)sampling

David Svoboda

email: svoboda@fi.muni.cz
Centre for Biomedical Image Analysis
Faculty of Informatics, Masaryk University, Brno, CZ

CBIA

September 26, 2019

Outline

2D Fourier Transform (discrete)

Definition

Given 2D discrete function f of (M, N) samples and two bases $(\varphi_k, k = \{0, \dots, M-1\})$ and $(\varphi_l, l = \{0, \dots, N-1\})$, let us define:

- forward 2D discrete Fourier transform:

$$\mathcal{F}(k,l) \equiv \frac{1}{\sqrt{MN}} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f(m,n) e^{-2\pi i \left(\frac{mk}{M} + \frac{nl}{N}\right)}$$

- inverse 2D discrete Fourier transform:

$$f(m,n) \equiv \frac{1}{\sqrt{MN}} \sum_{k=0}^{M-1} \sum_{l=0}^{N-1} \mathcal{F}(k,l) e^{2\pi i \left(\frac{mk}{M} + \frac{nl}{N}\right)}$$

2D Fourier Transform

Separability

The evaluation of 2D-(D)FT can be decomposed into two simpler tasks:

$$\begin{aligned}
\mathcal{F}(k,l) &= \frac{1}{\sqrt{MN}} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} f(m,n) e^{-2\pi i \left(\frac{mk}{M} + \frac{nl}{N}\right)} \\
&= \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} \left\{ \frac{1}{\sqrt{M}} \sum_{m=0}^{M-1} f(m,n) e^{-\frac{2\pi imk}{M}} \right\} e^{-\frac{2\pi inl}{N}} \\
&= \frac{1}{\sqrt{N}} \sum_{n=0}^{N-1} \mathcal{F}(k,n) e^{-\frac{2\pi inl}{N}}
\end{aligned}$$

Question

Let us have an image with dimensions $M \times N$ (M and N are powers of 2). As we want to apply discrete cosine transform (Fourier transform) to implement JPEG compression, we would like to know the efficiency of this process.

What is the complexity:

- when doing straightforward (naive) evaluation of DFT?
- when we utilize separability of FT and FFT?

2D Fourier Transform (continuous)

Definition

Given 2D integrable function f and two bases ($\varphi_{\omega_x}, \omega_x \in \mathbb{R}$) and ($\phi_{\omega_y}, \omega_y \in \mathbb{R}$), let us define:

- forward 2D continuous Fourier transform

$$\mathcal{F}(\omega_x, \omega_y) \equiv \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} f(x, y) e^{-2\pi i (x\omega_x + y\omega_y)} dxdy$$

- inverse 2D continuous Fourier transform

$$f(x, y) \equiv \int_{-\infty}^{\infty} \int_{-\infty}^{\infty} \mathcal{F}(\omega_x, \omega_y) e^{2\pi i (x\omega_x + y\omega_y)} d\omega_x d\omega_y$$

2D Discrete Fourier Transform Properties

All the properties from 1D DFT are valid:

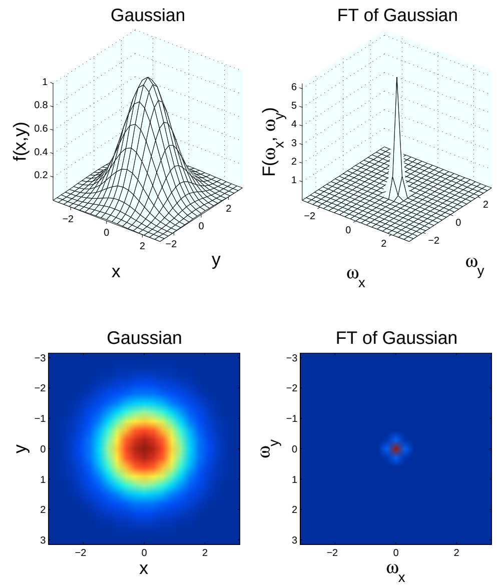
- scaling
- shift
- repetition
- convolution theorem

There are some more properties:

- separability
- rotation

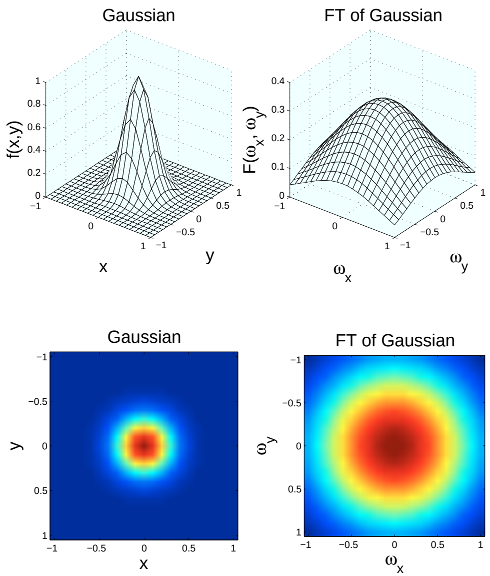
2D DFT

Scaling



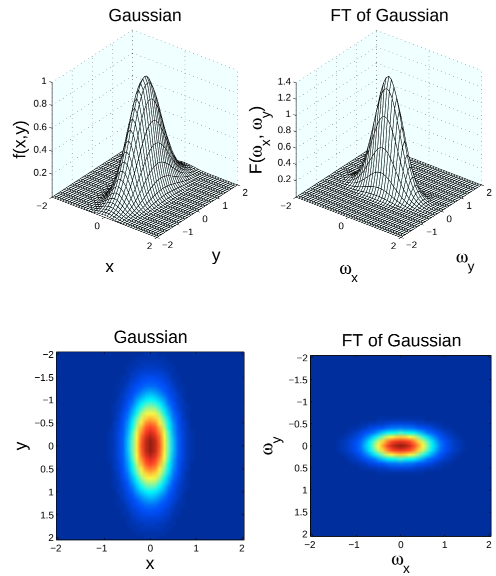
2D DFT

Scaling



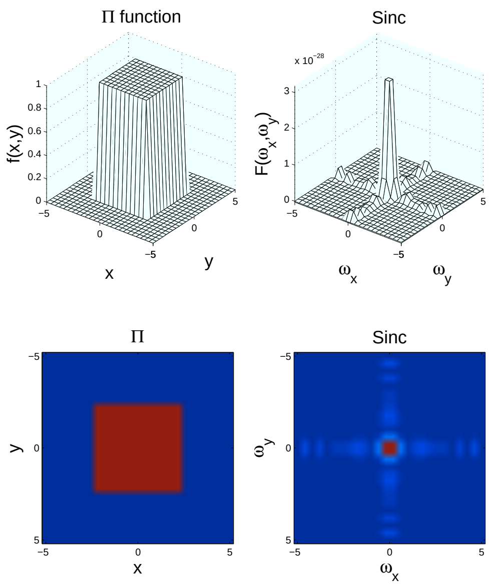
2D DFT

Scaling



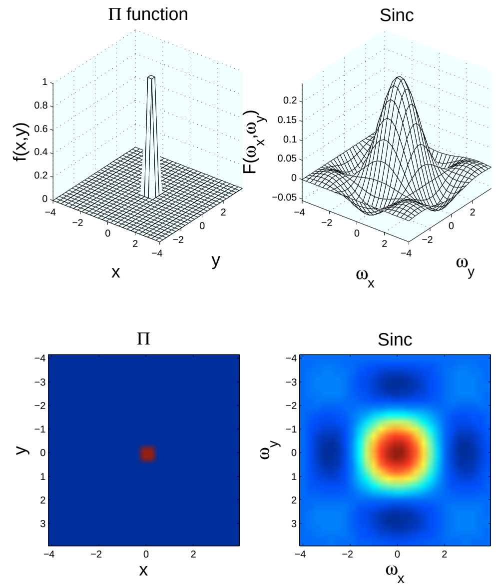
2D DFT

Scaling



2D DFT

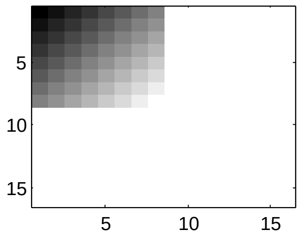
Scaling



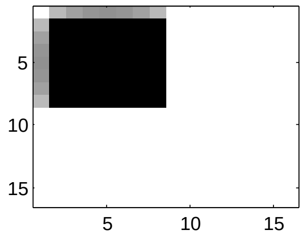
2D DFT

Repetition

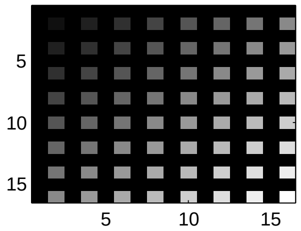
f(x,y)=x+y



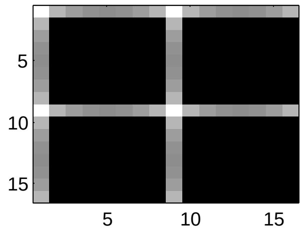
Fourier Transform (amplitude)



f(x,y) stretched by factor 2



Fourier Transform (amplitude)

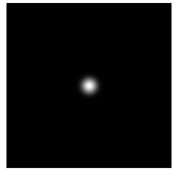


2D DFT

Convolution theorem

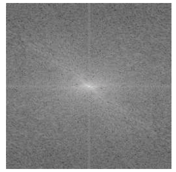
Input image "I"

Convolution kernel "G"

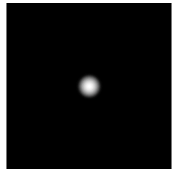


I * G

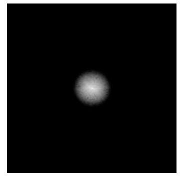
log(|FT(I)|)



log(|FT(G)|)



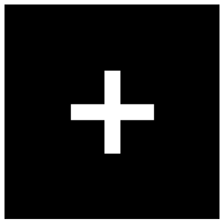
log(|FT(I)FT(G)|)



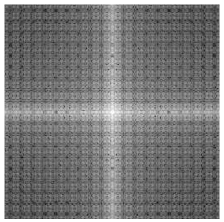
2D DFT

Rotation

Input image "I"

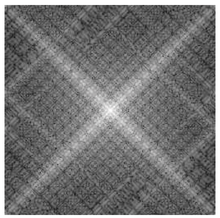


|FT(I)|



rotated I

|FT(rotated I)|



2D DFT

Rotation

Let us introduce the polar coordinates:

$$\begin{aligned} x &= r\cos\phi \\ y &= r\sin\phi \\ \omega_x &= R\cos\psi \\ \omega_y &= R\sin\psi \end{aligned}$$

Then

$$\begin{aligned} f(x,y) &\rightarrow f(r,\phi) \\ \mathcal{F}(\omega_x,\omega_y) &\rightarrow \mathcal{F}(R,\psi) \end{aligned}$$

2D DFT

Rotation

It is now clear to see that:

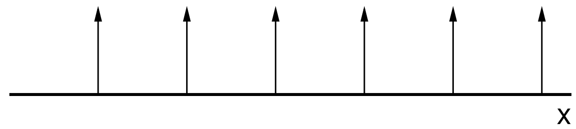
$$f(r, \phi + \phi_0) \supset \mathcal{F}(R, \psi + \phi_0)$$

Conclusion: Rotating $f(x, y)$ by an angle ϕ_0 rotates $\mathcal{F}(\omega_x, \omega_y)$ by the same angle, and vice versa.

Comb function

In 1D space

$$\text{III}(x) = \sum_{n=-\infty}^{\infty} \delta(x - n)$$



Notice: “III” is pronouced as *shah* (Cyrilic character).

Comb function

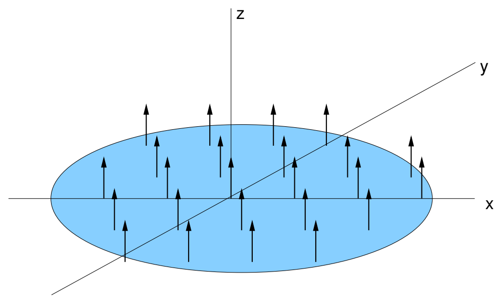
Some properties

- $\text{III}(-x) = \text{III}(x)$
- $\text{III}(x + n) = \text{III}(x)$
- $\text{III}(x - \frac{1}{2}) = \text{III}(x + \frac{1}{2})$
- $\text{III}(x) = 0 \quad x \neq n$
- $\text{III}(ax) = \frac{1}{|a|} \sum \delta(x - \frac{n}{a})$
- $\text{III}(\frac{x}{\tau}) \supset \tau \text{III}(\tau\omega)$

Comb function

In 2D space

$${}^2\text{III}(x, y) = \sum_{m=-\infty}^{\infty} \sum_{n=-\infty}^{\infty} \delta(x - m, y - n)$$



Separability of delta function implies:

$${}^2\text{III}(x, y) = \text{III}(x)\text{III}(y)$$

Sampling

Sampling = the process of converting a continuous signal into a discrete sequence.

- In 1D:

$$\mathrm{III}(x)f(x) = \sum_{n=-\infty}^{\infty} f(n)\delta(x-n)$$

- In 2D:

$$^{2}\mathrm{III}(x,y)f(x,y) = \sum_{m}\sum_{n} f(m,n)\delta(x-m, y-n)$$

Question: What happens in frequency (Fourier) domain?

Sampling

Comparison of image and Fourier domain

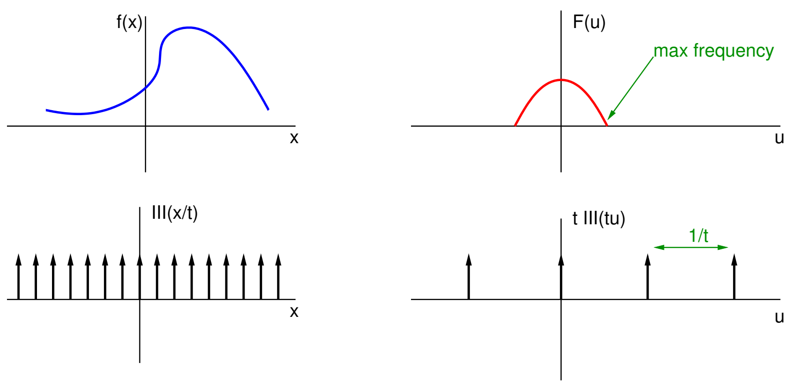
Image/Time domain:

- multiplication of the function f and III
- sampling

Fourier domain:

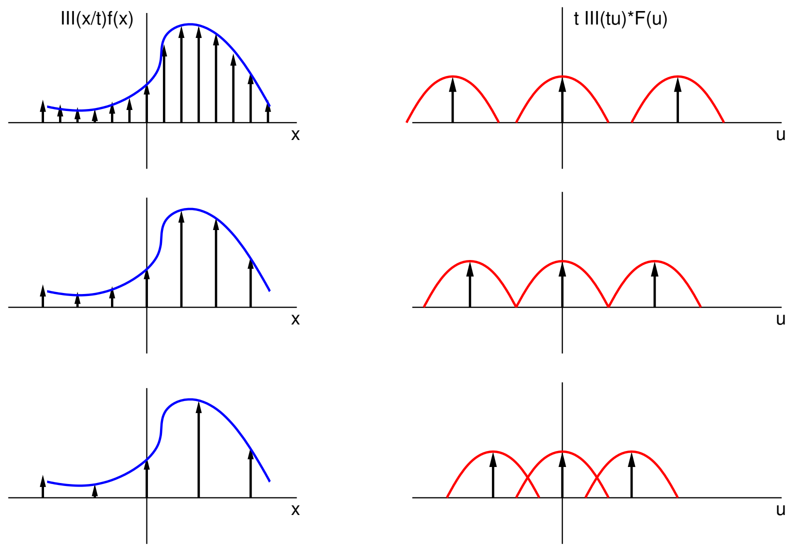
- convolution of the function $FT(f)$ and $FT(\mathrm{III})$
- convolution with Dirac impulses causes replication of $FT(f)$
- scaling property is also valid for III function

Sampling



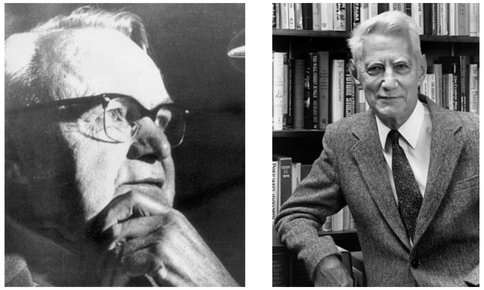
Notice: The comb function density must be high enough to guarantee proper sampling

Sampling



Nyquist-Shannon theorem

Exact reconstruction of a continuous signal from its samples is possible if the signal is bandlimited and the sampling frequency is greater than twice the signal maximal frequency



Harry Nyquist (1889 – 1976) & Claude Elwood Shannon (1916 – 2001)

Question: How to use N-S theorem, if the original signal is unlimited?

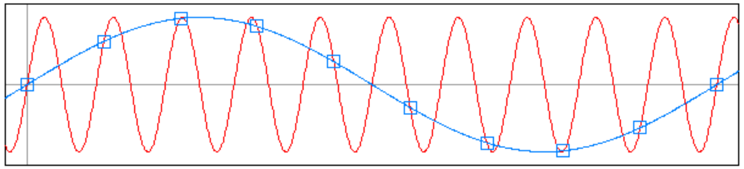
Sampling

Common problems – aliasing

The cause of aliasing:

when Nyquist-Shannon condition is broken, i.e.

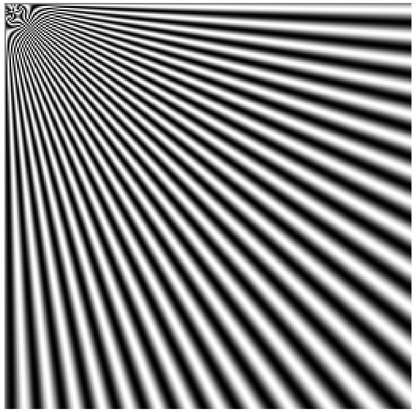
- sampling frequency is not high enough or (time alias – wagon wheel effect)
- the signal in not bandlimited (PC games – far horizon)



Sampling

Common problems – aliasing

An example



Sampling

Common problems – aliasing

An example

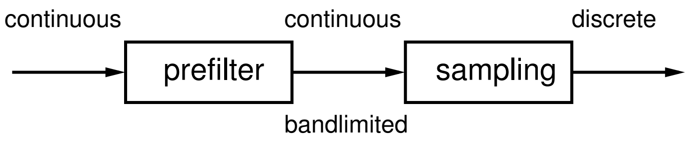
Sampling

Common problems – aliasing

How to eliminate aliasing?

- sampling at higher frequency
  - does it help if the signal is not band limited?
  - expensive for memory and time

  OR
- prefiltering
  - before sampling the input signal is "prefiltered" by lowpass filter



Sampling

Common problems – aliasing

Some lowpass filters

- Gaussian filter

$$f_\sigma(x) = \frac{1}{\sigma\sqrt{2\pi}} e^{-\frac{x^2}{2\sigma^2}}$$

- Sinc filter

$$f(x) = \frac{sin(x)}{x}$$

- B-spline filter

$$\begin{aligned} b_1(x) &= \begin{cases} 1 & |x| \leq 1/2 \\ 0 & |x| > 1/2 \end{cases} \\ b_n(x) &= b_1(x) * b_1(x) * \cdots * b_1(x) \quad \text{n-times} \end{aligned}$$

Reconstruction

Inverse process to sampling

The purpose: reconstruction of the original continuous signal from the sampled sequence.

Reconstruction $\equiv$ convolution with a *low-pass* filter.

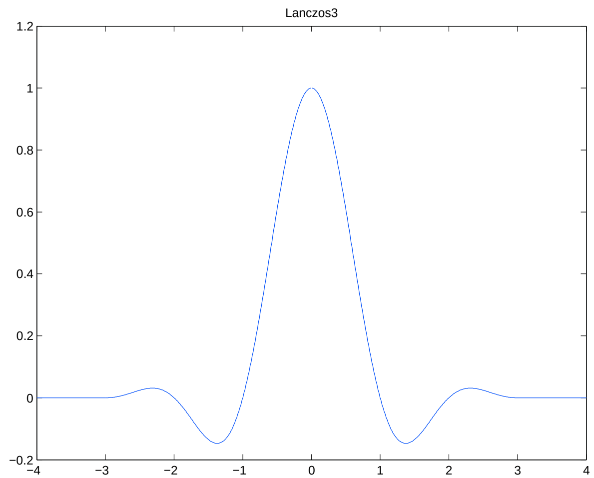
Common reconstruction filters:

- *box* (nearest neighbour)
- *tent* (linear interpolation)
- cubic B-spline (cubic polynomial interpolation)
- Gaussian
- *sinc* function
- Lanczos (windowed sinc function)

Notice: The unit area under the curve.

Reconstruction

Lanczos filter

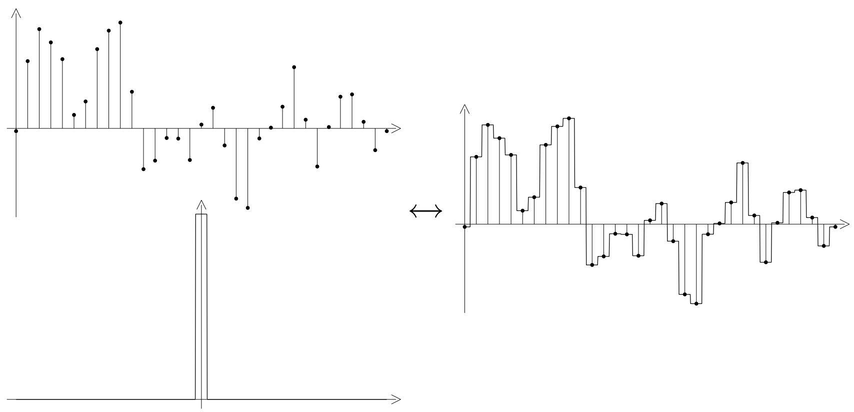


$$lanczos3(x) = \begin{cases} \frac{\sin \pi x}{\pi x} \frac{\sin \pi \frac{x}{3}}{\pi \frac{x}{3}} & |x| < 3 \\ 0 & |x| \geq 3 \end{cases}$$

Reconstruction

Examples of reconstruction

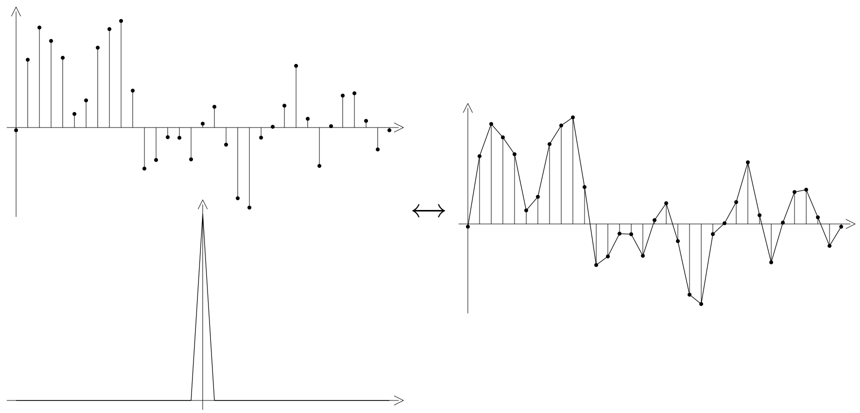
Box filter



Reconstruction

Examples of reconstruction

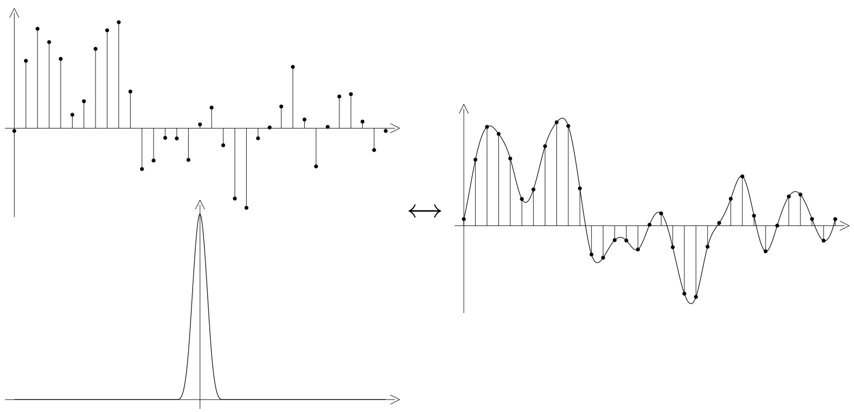
Tent filter



Reconstruction

Examples of reconstruction

Cubic B-spline filter



Resampling in 1D

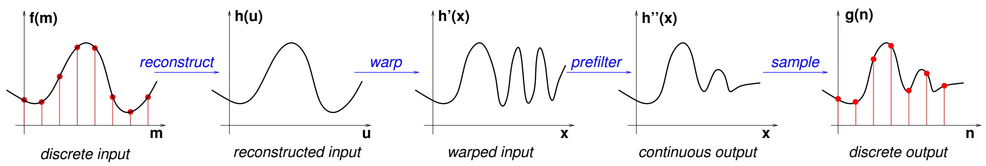
Let us design a 1D resampling filter

- The filter should be easy to implement and fast for computation.
- The filter should solve the alias problem.

Resampling in 1D

Design of the resampling filter

1. reconstruct the continuous signal from the discrete one
2. warp the domain of the continuous signal
3. prefilter the warped, continuous signal
4. sample this signal to produce the discrete output signal



Resampling in 1D

Important (implementation) notes

- During the resampling process we actually never construct a continuous signal $h(u)$, $h'(x)$ or $h''(x)$.
- We pick up the individual positions in the resampled image $g(n)$ and look for their corresponding positions and their neighbourhood in the original image $f(m)$.
- As the computation is inverted, we never use γ function. We use only γ^{-1}.

Resampling in 1D

Derivation of an ideal resampling filter

Computation of one sample point

$$\begin{aligned} g(n) &= h''(n) = \int h'(t)p(n-t)dt \\ &= \int h(\gamma^{-1}(t))p(n-t)dt \\ &= \int p(n-t)\sum_k f(k)r(\gamma^{-1}(t)-k)dt = \sum_k f(k)\rho(n,k) \end{aligned}$$

where

$$\rho(n,k) = \int p(n-t)r(\gamma^{-1}(t)-k)dt$$

- $\rho(n,k)$ is called a resampling filter.
- If γ is affine, we can derive: $\rho(n,k) = p(\gamma^{-1}(n)-k) * r(\gamma^{-1}(n)-k)$.

Resampling in 1D

Practical problems

- If the mapping γ is not affine, the filter $\rho(m,k)$ is space variant.

Solution (*postfiltering/supersampling*)

1. Reconstruct the continuous signal from the discrete input signal.
2. Warp the domain of the input signal.
3. Sample the warped signal at *very high resolution* to avoid alias.
4. Postfilter the signal to produce a lower resolution output signal.

Notice: The convolution is employed in the very end of this algorithm, i.e. it is discrete and space invariant.

Resampling in 2D

Task

Design a 2D resampling filter

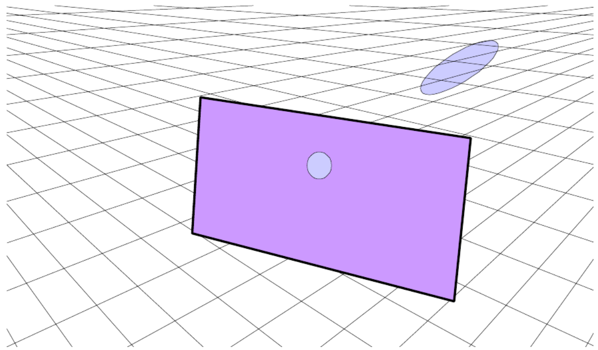
- Obey the rules that are valid for 1D resampling filter.
- The filter maps texels from texture space to screen space.
- The filter might be anisotropic.
- The filter should work fast.

Resampling in 2D

γ-mapping

Basic properties of γ-mapping

- γ converts coordinates from screen space to texture space.
- γ is projective (neither linear nor affine).
- Circular neighbourhood of one pixel (in screen space) is transformed into ellipse (in texture space).



Resampling in 2D

γ-mapping

Approximation of γ-mapping

- Let us approximate γ in the neighbourhood of $\mathbf{u}_0$ as the locally-affine mapping:

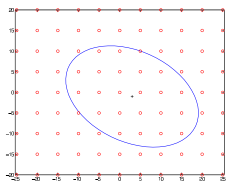
$$\gamma(\mathbf{u}) = \mathbf{u}_0 + J_{\mathbf{u}_0}(\mathbf{u} - \mathbf{u}_0)$$

where $J_{\mathbf{u}_0}$ is Jacobian and $\mathbf{u} = (u, v)$ is 2D vector in texture space.

Construction of ellipse in texture space

- The major and minor axis determining the ellipse shape correspond to partial derivatives of γ in the position $\mathbf{u}_0$, i.e. the rows of matrix $J_{\mathbf{u}_0}$:

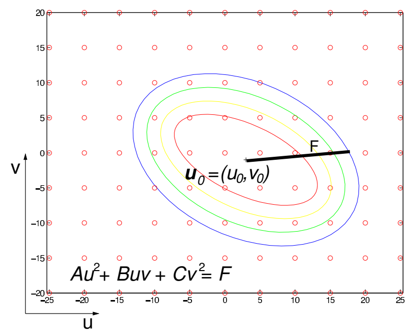
$$J_{\mathbf{u}_0} = \begin{bmatrix} \frac{\partial u}{\partial x} & \frac{\partial u}{\partial y} \\ \frac{\partial v}{\partial x} & \frac{\partial v}{\partial y} \end{bmatrix}$$



Resampling in 2D

γ-mapping

Construction of ellipse in texture space (continued)



$$\begin{aligned}
A &= \left(\frac{\partial v}{\partial x}\right)^2 + \left(\frac{\partial v}{\partial y}\right)^2 \\
B &= -2\left(\frac{\partial u}{\partial x}\frac{\partial v}{\partial x} + \frac{\partial u}{\partial y}\frac{\partial v}{\partial y}\right) \\
C &= \left(\frac{\partial u}{\partial x}\right)^2 + \left(\frac{\partial u}{\partial y}\right)^2 \\
F &= \left(\frac{\partial u}{\partial x}\frac{\partial v}{\partial y} - \frac{\partial u}{\partial y}\frac{\partial v}{\partial x}\right)^2
\end{aligned}$$

Resampling in 2D

γ-mapping

Image Pyramids (MIP map)

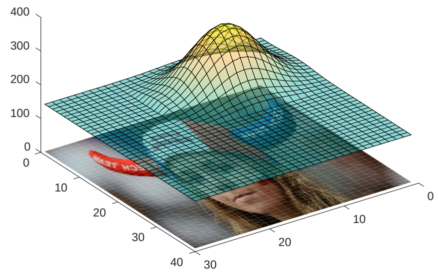
- Size of ellipse determines level of detail in MIP map pyramid that should be fetched from the memory.

Resampling in 2D

Computation in texture space

Collecting pixels from texture space:

1. Create the Gaussian with the elliptical support.
2. Attach the Gaussian to the texture image loaded from the MIP map pyramid.
3. Sum up the image pixels by using the Gaussian weights.



Resampling in 2D

Elliptical Weighted Average (EWA)

Implementation Notes

For each image pixel (x,y) from screen space:

1. Find corresponding point (u,v) in texture space.
2. Define the local affine transform γ.
3. Compute Jacobian J of this mapping.
4. Delineate the ellipse in texture space.
5. Using the ellipse size choose the appropriate MIP map level.
6. Build the Gaussian over the ellipse.
7. Evaluate direct convolution of MIP map image with Gaussian.
8. Store the result (one value) in the screen pixel (x,y).

Resampling in 2D

EWA – Properties

EWA fullfills the requirements applied to optimal resampling filter

$$g(\mathbf{n}) = \sum_{\mathbf{k}} f(\mathbf{k})\rho(\mathbf{n},\mathbf{k})$$

where

$$\rho(\mathbf{n},\mathbf{k}) = p(\gamma^{-1}(\mathbf{n}) - \mathbf{k}) * r(\gamma^{-1}(\mathbf{n}) - \mathbf{k})$$

- γ is locally affine: prefilter p and reconstruction filter r are Gaussians. Their convolution is again Gaussian.
- γ is locally affine: p and r have elliptical support. The product of their convolution has also elliptical support, as the ellipses are closed under affine transforms.

Resampling in 2D

EWA – Technical Notes

The quality of filtering corresponds to the resampling filter support

- Anisotropic filtering 1× . . . 8 texels (pixels from texture space)
- Anisotropic filtering 2× . . . 16 texels
- Anisotropic filtering 4× . . . 32 texels
- Anisotropic filtering 8× . . . 64 texels
- Anisotropic filtering 16× . . . 128 texels

Higher the quality ⇒ higher the computational cost (GPU usage)

Resampling in 2D

EWA – An Example

Texture filtering with naive MIP map (on the left) and anisotropic filtering with so called *EWA* based method (on the right)

source: wikipedia.org

Resampling in 2D

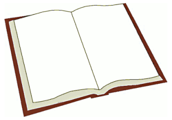
EWA – An Example

Texture filtering with so called *EWA* based method

source: shinvision.com

Bibliography

1. Bracewell, R. N., Fourier transform and its applications / 2nd ed. New York: McGraw-Hill, p 474. ISBN 0070070156
2. Paul Heckbert, Fundamentals of Texture Mapping and Image Warping, Master's thesis, UCB/CSD 89/516, CS Division, U.C. Berkeley, June 1989, 86 pp
3. Turkowski, K. Filters for common resampling tasks. In Graphics Gems, A. S. Glassner, Ed. Academic Press Professional, San Diego, CA, 147-165, 1990
4. McCormack, J., Perry, R.N., Farkas, K.I.; Jouppi, N.P. "Feline: Fast Elliptical Lines for Anisotropic Texture Mapping", ACM SIGGRAPH, pp 243-250, August 1999
5. Pharr, M., Humphreys, G. Physically Based Rendering: From Theory to Implementation. Morgan Kaufmann Publishers Inc., San Francisco, CA, USA, 2004



You should know the answers . . .

- Show that the 2D DFT is separable transform.
- Derive the complexity of 2D discrete FFT.
- Explain the reciprocity of wide and narrow shapes in time and frequency domain, respectively.
- Derive (dot not formulate) the Nyquist-Shannon theorem for 2D image data.
- Show an example of the aliasing effect.
- What is a prefilter?
- What is the difference between a screen space and texture space?
- Give an example of γ warping function both for 1D and 2D case.
- What is the difference between projective and affine mappings?
- Describe individual steps of EWA filter.