Rychlá Fourierova Transformace

Filtrování ve freknvenční oblasti ve 2D

Jan Bařtipán

Fourierova Transforace

- Jean Baptiste Joseph Fourier (1768–1830)
- Vyjádření posloupnsoti jako součet sinusových vln
- Časová oblast – původní posloupnost
- Frekvenční oblast – transformovaná posloupnost

Fourierova Transformace

- Prvky posloupnosti v časové i frekvenční oblasti jsou komplexní čísla
- Z prvků ve frekvenční oblasti jsme schopni určit zastoupení jednotlivých frekvencí v signálu

DFT

- Dopředná transformace

$$F_k = \sum_{n=0}^{N-1} f_n \cdot e^{\frac{-i \cdot 2\pi \cdot k\, n}{N}}$$

- Zpětná transformace

$$f_k = \frac{1}{N} \cdot \sum_{n=0}^{N-1} F_n \cdot e^{\frac{i \cdot 2\pi \cdot k n}{N}}$$

- Složitost O(n^2)

FFT

- J. W. Cooley a J. W. Tukey v roce 1965
- Složitost stlačena na O(n log n)
- Princip
  - Rekurentní rozklad posloupností na sudé a liché prvky
  - Výpočet spočívá ve skládání těchto podposloupností

FFT

- Platí totiž:

$$F_k = \sum_{n=0}^{N-1} f_n \cdot e^{\frac{-i \cdot 2\pi \cdot k n}{N}}$$

$$F_k = \sum_{n=0}^{N/2-1} f_{2\mathrm{n}} \cdot e^{\frac{-i \cdot 2\pi \cdot k 2\mathrm{n}}{N}} + \sum_{n=0}^{N/2-1} f_{2\mathrm{n}+1} \cdot e^{\frac{-i \cdot 2\pi \cdot k(2\mathrm{n}+1)}{N}}$$

$$F_k = \sum_{n=0}^{N/2-1} f_n^{even} \cdot e^{\frac{-i \cdot 2\pi \cdot k n}{N/2}} + e^{\frac{-j \cdot 2\pi}{N} \cdot k} \cdot \sum_{n=0}^{N/2-1} f_n^{odd} \cdot e^{\frac{-i \cdot 2\pi \cdot k n}{N/2}}$$

$$F_k = F_k^e + e^{\frac{-j \cdot 2\pi}{N} \cdot k} \cdot F_k^o$$

FFT

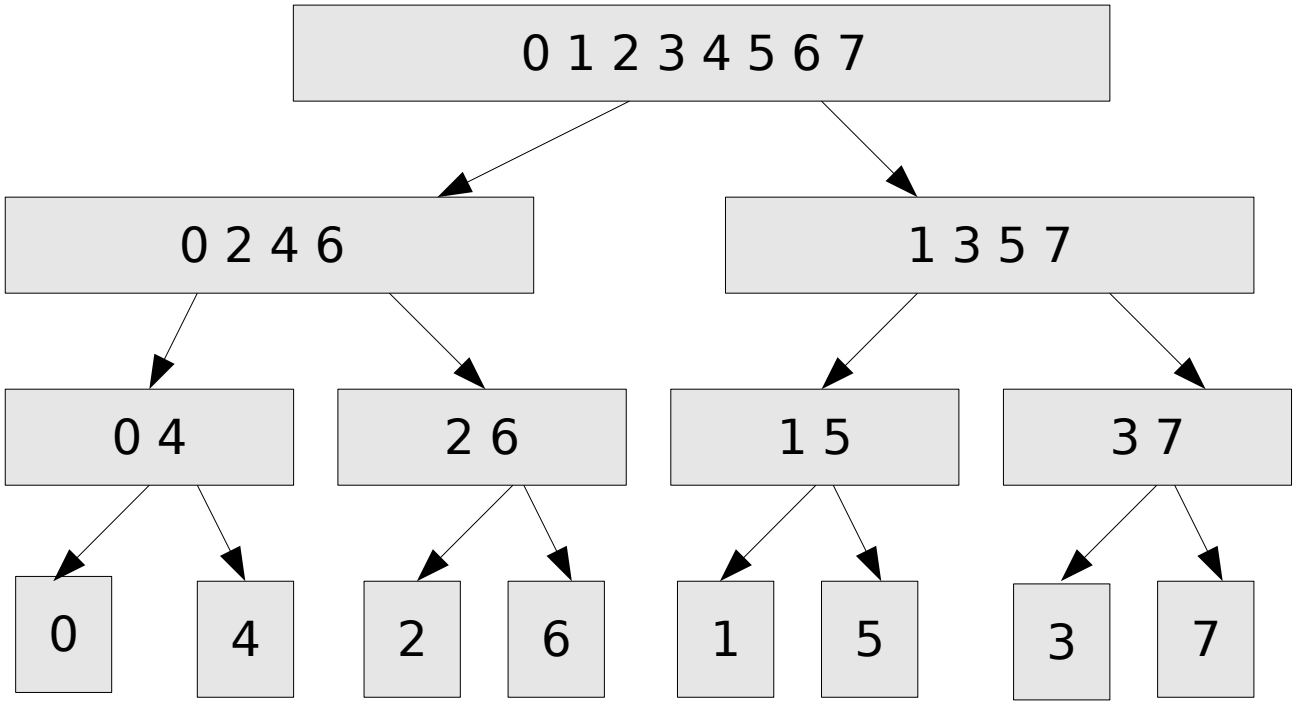
- Fourierova transformace posloupnosti o jednom prvku je tento prvek
- Abychom mohli rozložit rekurentně posloupnost na sudé a liché podposloupnosti, potřebujeme právě 2^n vzorků
- Jiná faktorizace je také možná, je však pomalejší než rozklad na 2 podposloupnosti

In place implementace

- FFT ačkoliv je ve své podstatě rekurentní, je často implementována jako in-place algoritmus.
- Prvním krokem je přeskládání posloupnosti tak aby spolu sousedili posloupnosti, které budem skládat dohromady

In-place implementace

- Přeskládání posloupnosti o osmi prvcích



Implementace přeskládání

Původní index		***Nový index***	
Dec.	***Bin.***	***Dec***	***Bin***
0	000	0	000
1	001	4	100
2	010	2	010
3	011	6	110
4	100	1	001
5	101	5	101
6	110	3	011
7	111	7	111

Implementace přeskládání

```
size_t n = arr.size();

// Do the bit reversal
u32 i2 = n >> 1;
u32 j = 0;
for (u32 i=0; i < n-1; i++)
{
   if (i < j)
   {
      Complex tmp = arr[i];
      arr[i] = arr[j];
      arr[j] = tmp;
   }
   u32 k = i2;
   while (k <= j)
   {
      j -= k;
      k >>= 1;
   }
   j += k;
}
```

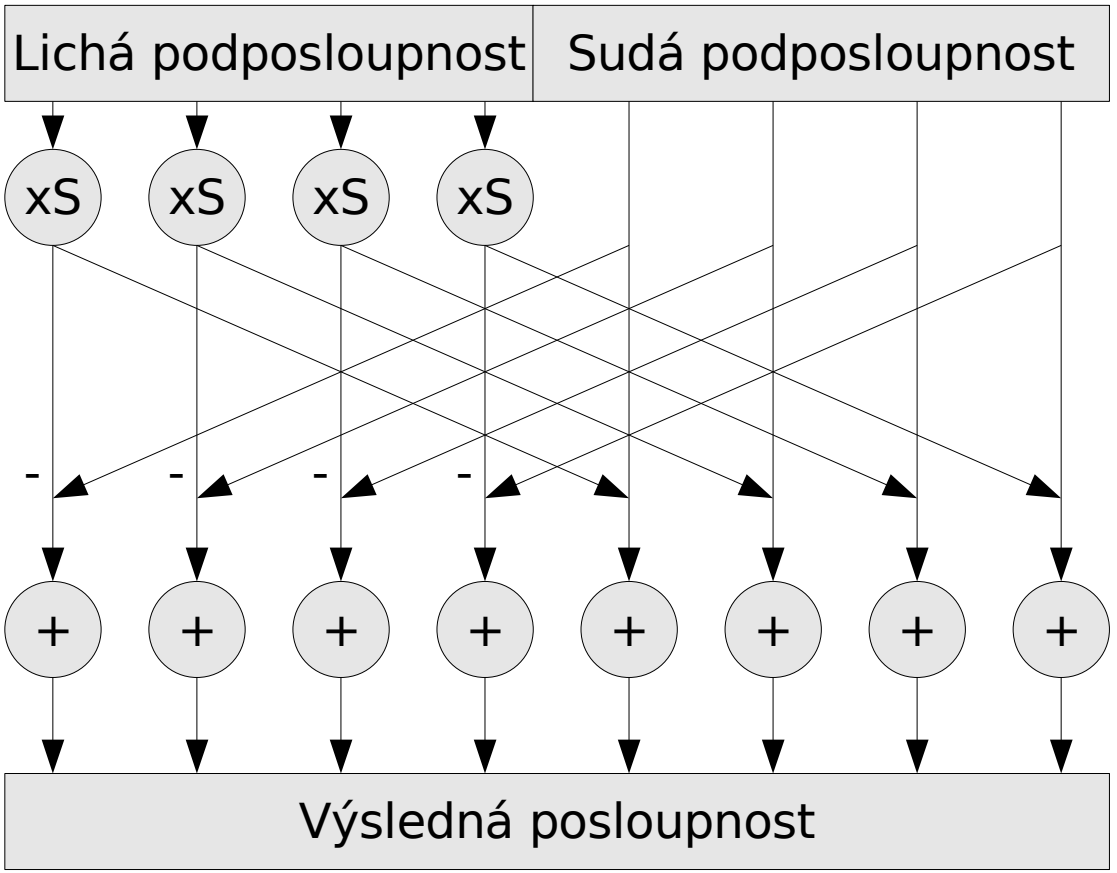
Motýlkový algoritmus

- Samotný výpočet probíhá skládáním transformovaných podposloupností do výsledné posloupnosti.
- Vztah pro nový prvek posloupnosti:

$$F_k = F_k^e + e^{\frac{-j \cdot 2\pi}{N} \cdot k} \cdot F_k^o$$

$$S_k = e^{\frac{-j \cdot 2\pi}{N} \cdot k}$$

Motýlkový algoritmus



Implementace motýlkového algoritmu

```
u32 l1;
u32 l2 = 1;
u32 m = static_cast<u32>(log(n)/log(2));
// sign of exponent
f64 sign = (forward) ? -1.0 : 1.0;
// base component of exponent of C
f64 cExpBase = sign * M_PI;
// initial
//Complex c = -1.0;
// loop for each level
for (u32 l = 0; l < m; l++)
{
   // loop for each sub DFT element
   // count of sub frequency spectras
   l1 = l2;
   // sub DFT stride
   l2 <<= 1;
   // W ^ 0
   Complex w(1,0);
   // W ^ 1
   Complex c(cos(cExpBase/l1), sin(cExpBase/l1));
   for (u32 j = 0; j < l1; j++)
   {
      // loop for each butterfly (sub DFT)
      for (u32 i = j; i < n; i += l2)
      {
         u32 i2 = i + l1;
         Complex t = w * arr[i2];
         arr[i2] = arr[i] - t;
         arr[i] += t;
      }
      // compute new w
      w *= c;
   }
}
```

2D DFT

- Dvourozměrná DFT je definována jako:

$$X_{k,l} = \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} x_{m,n} \cdot e^{-j2\pi\left(\frac{km}{M} + \frac{ln}{N}\right)}$$

- Zpětná 2D DFT je definována jako:

$$x_{k,l} = \frac{1}{MN} \sum_{m=0}^{M-1} \sum_{n=0}^{N-1} X_{m,n} \cdot e^{-j\pi\left(\frac{km}{M} + \frac{\ln}{N}\right)}$$

2D DFT pomocí vnoření 1D DFT

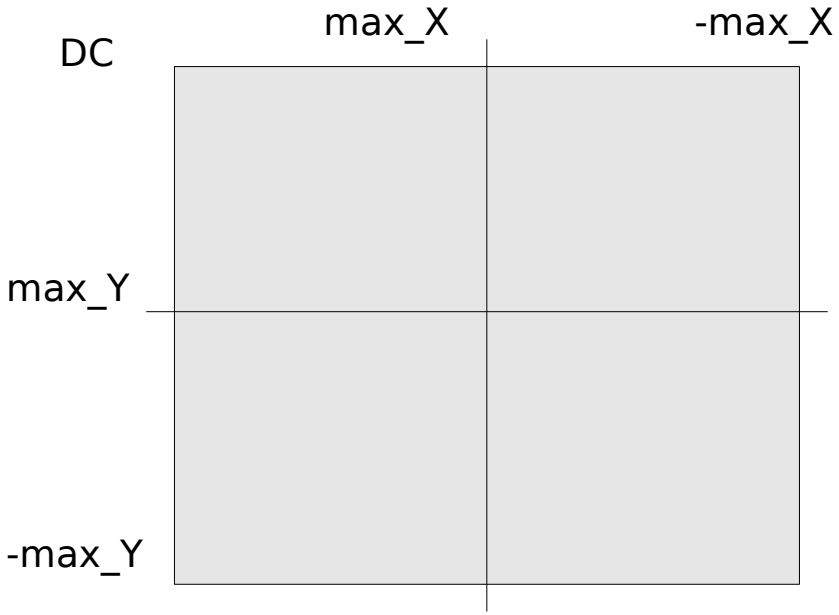
- Dopředná 2D DFT lze vyjádřit:

$$X_{k,l} = \sum_{m=0}^{M-1} \left(\sum_{n=0}^{N-1} x_{m,n} \cdot e^{-j2\pi \frac{km}{M}} \right) \cdot e^{-j2\pi \frac{ln}{N}}$$

- Z takto upraveného vzorce je již patrné, že lze vypočíst 2D DFT za pomocí 1D DFT. Proto nám nic nebrání použít algoritmus FFT v 1D pro výpočet v 2D transformováním jednotlivých sloupců a následně jednotlivých řádků obrázku.

FFTShift

- Výstup 2D FFT vypadá takto:



- Pro intuitivnější práci s výsledky FFT se přeskládájí kvadranty tak, aby DC bylo uprostřed a jednotlivé kvadranty odpovídaly osám grafu.

Reprezentace výstupu Fourierovy transformace

- Výstupem z Fourierovy transformace je posloupnost komplexní čísel (i pro reálný vstup)
- Abychom mohli pracovat s tímto výstupem, je vhodné pracovat s polárním vyjádřením komplexních hodnot výstupu.

$$\phi_i = atan(\frac{\Re X_i}{\Im X_i})$$

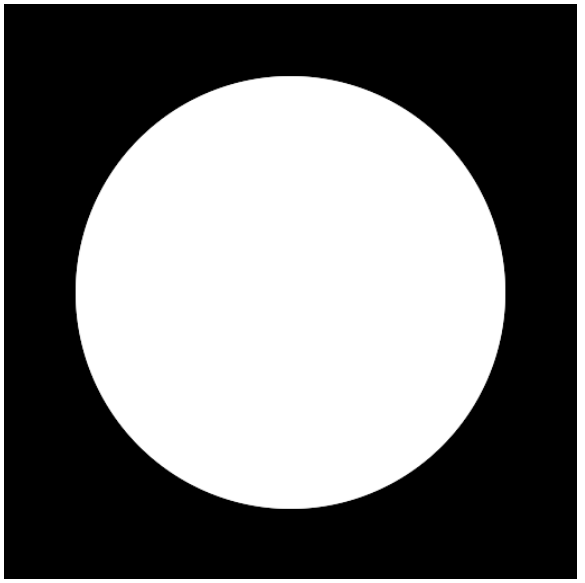
$$M_i = |X_i| = \sqrt{\Re X_i^2 + \Im X_i^2}$$

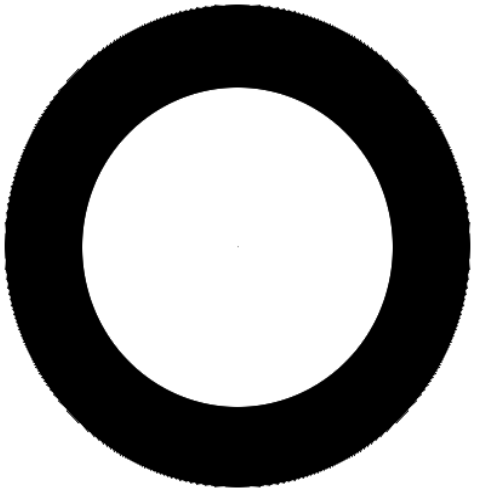
Filtrování ve frekvenční oblasti

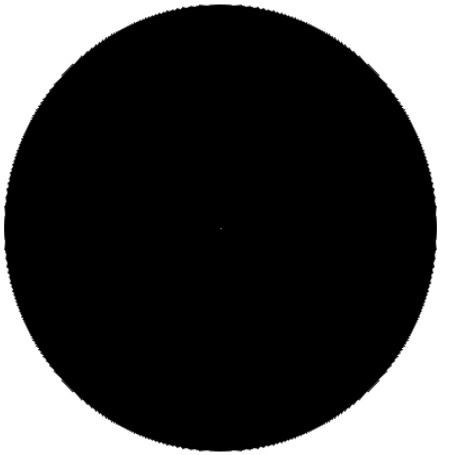
- Filtrování probíhá v těchto krocích:
  - Transformace z časové do frekcenční oblasti
  - Aplikování masky
  - Zpětná transformace do časové oblasti
- Aplikování masky nahradí prvky na některých místech (které odpovídají příslušným prostorovým frekvencím)

Příklady masek

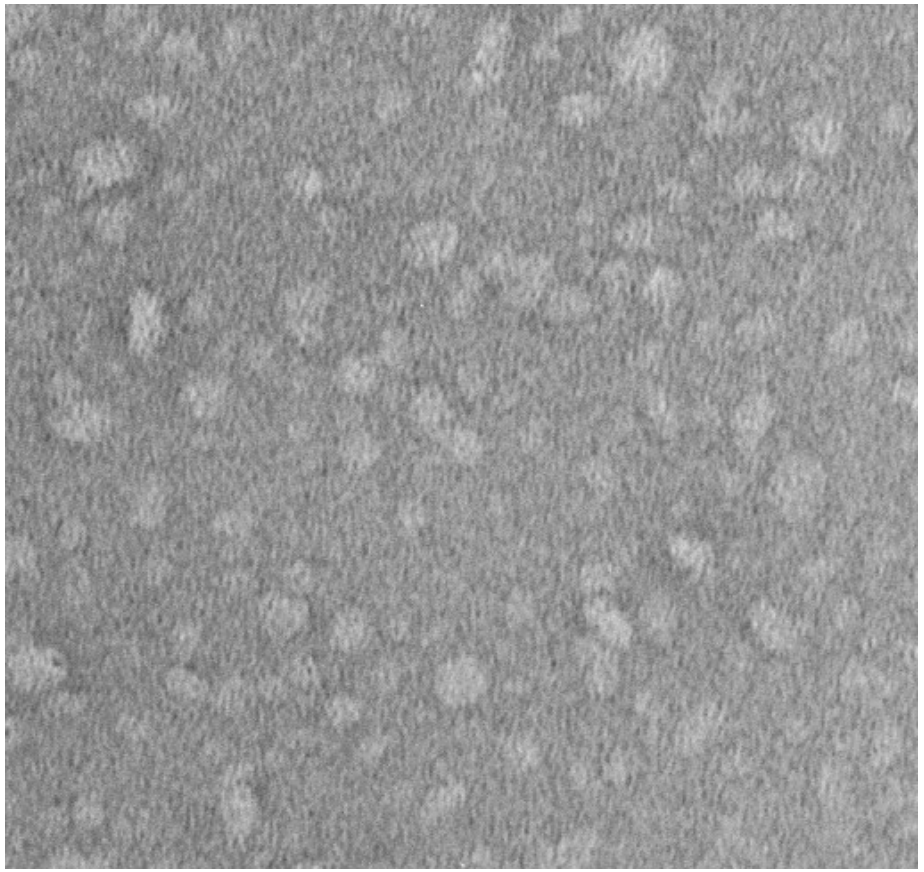
- Dolní propust
- Horní propust
- Pásmová zádrž

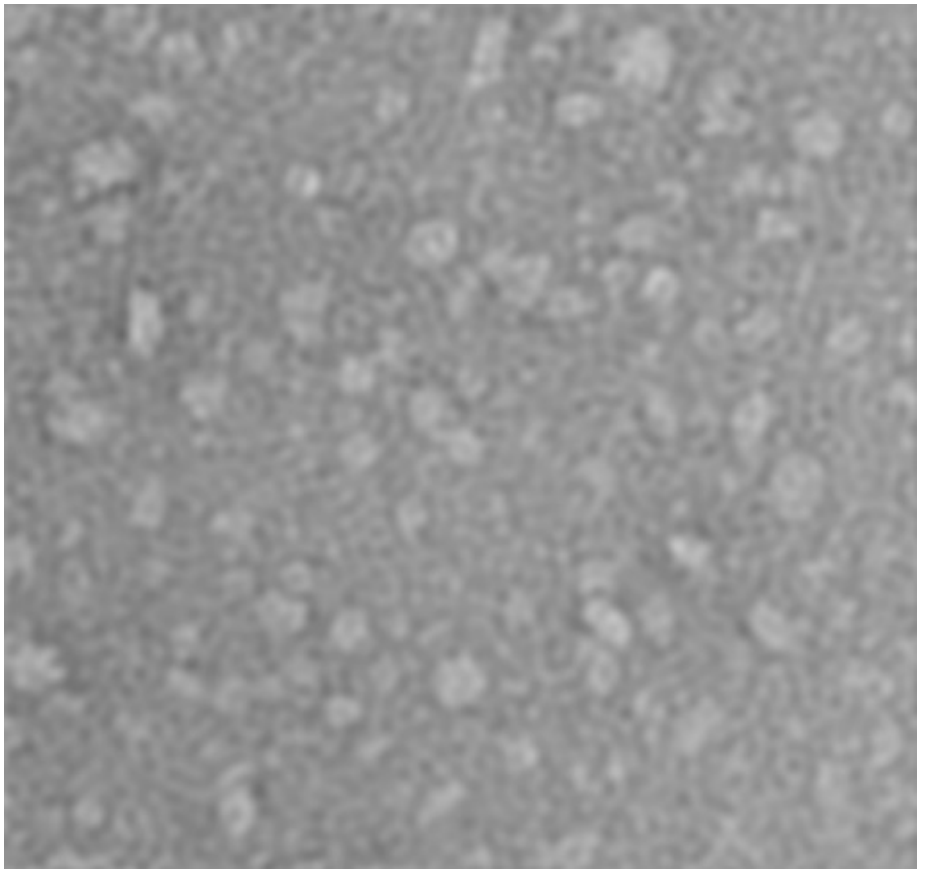




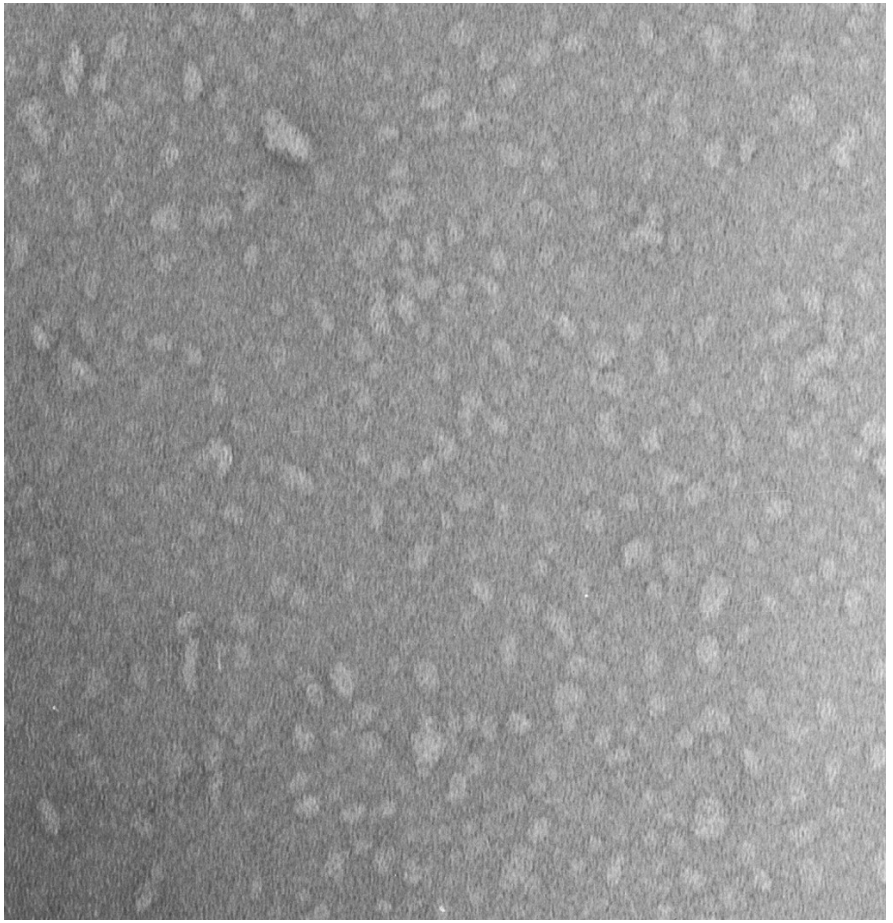


Dolní propust: Odstranění šumu

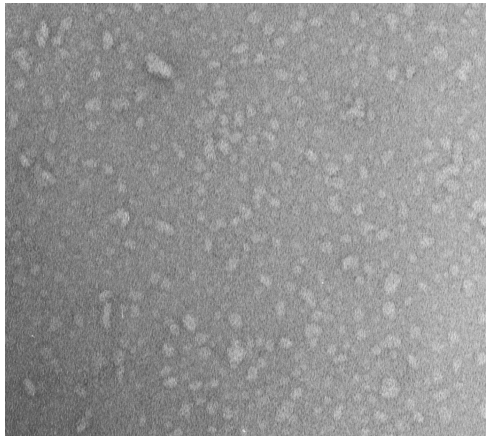




Dolní propust: konstantní pozadí

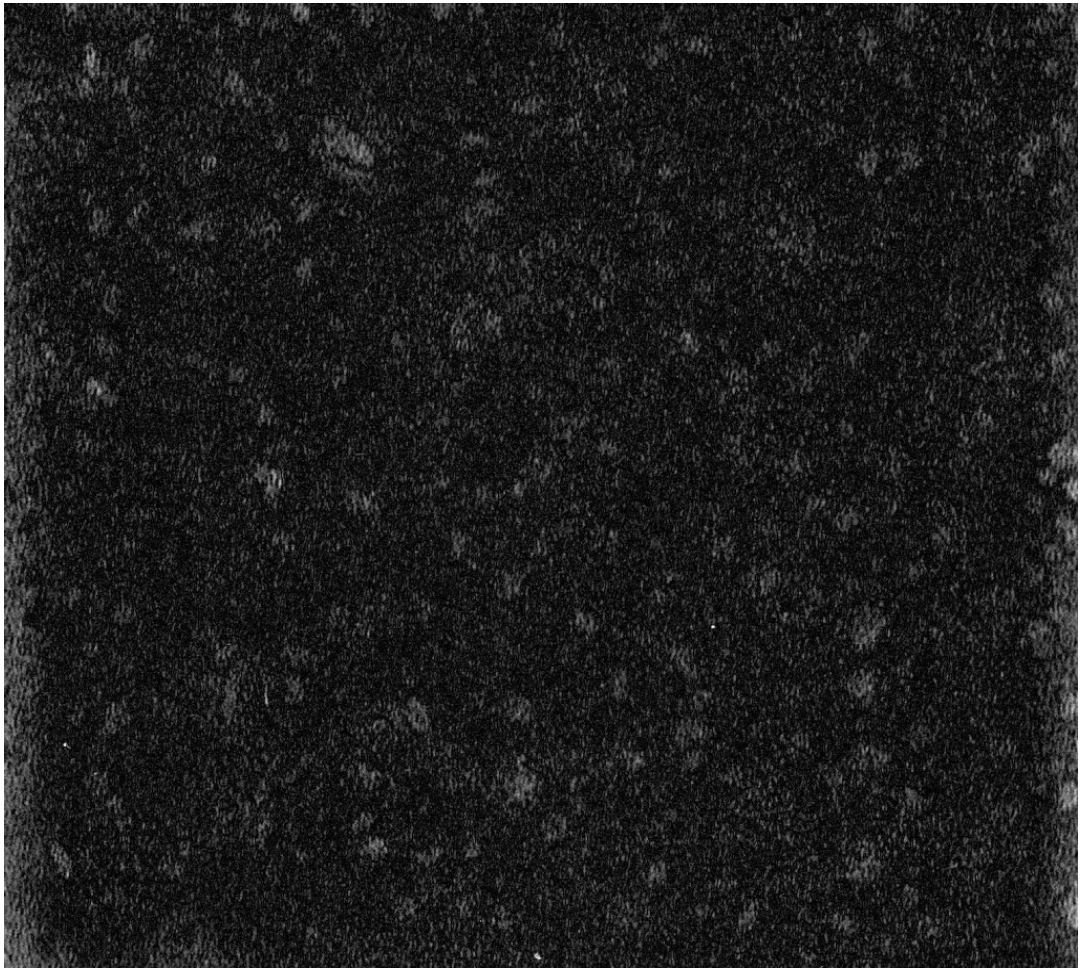


Dolní propust: konstantní pozadí

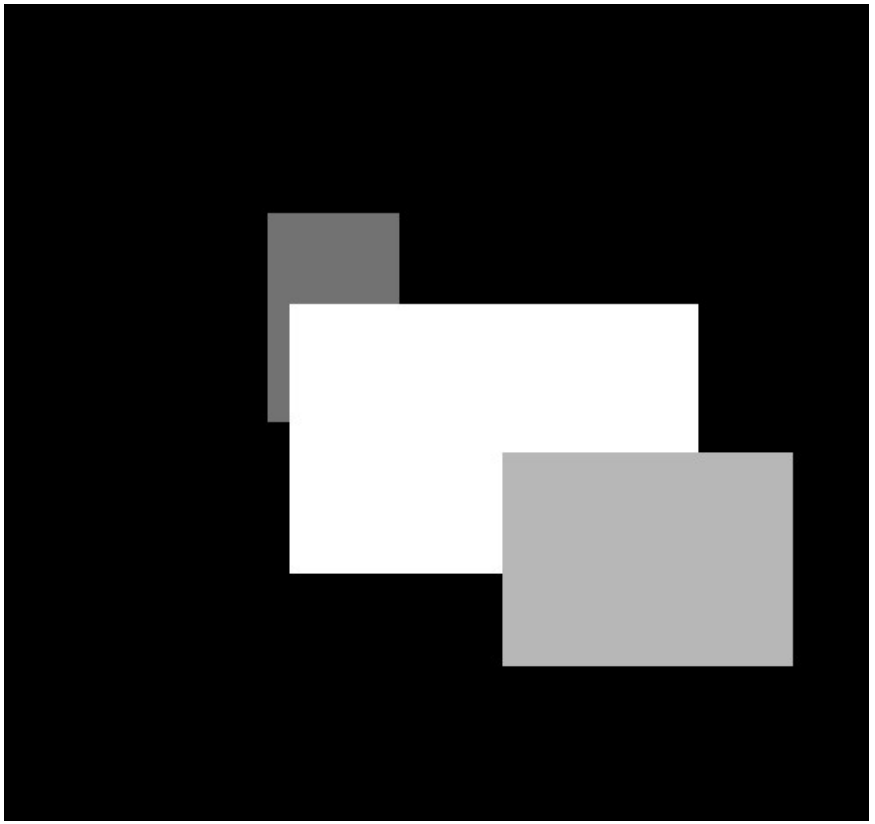


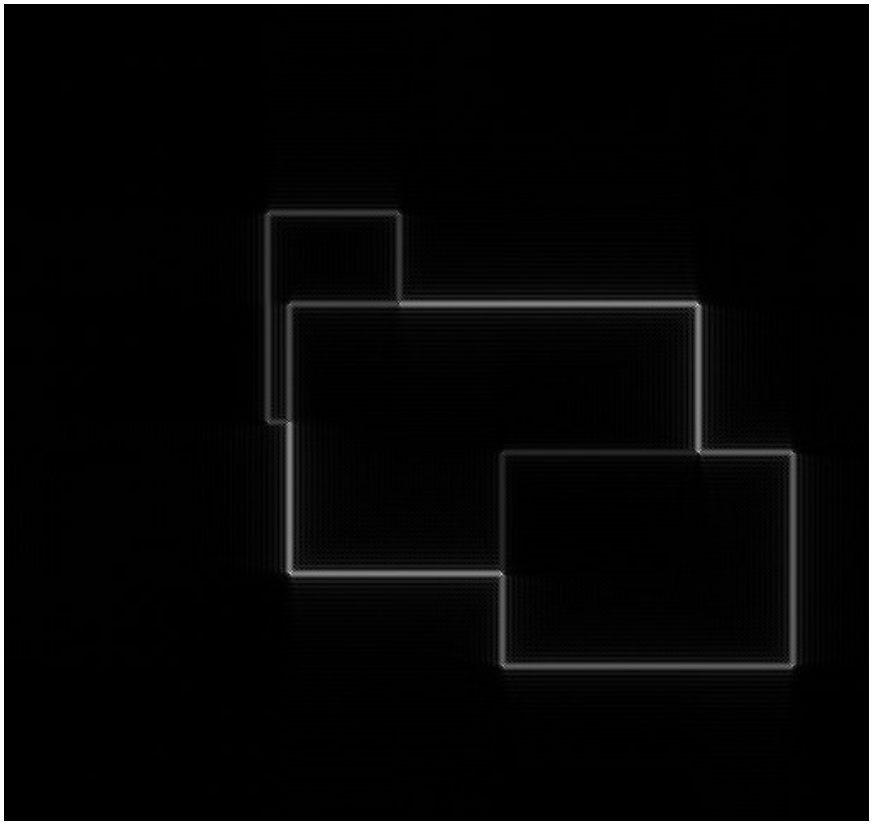
-

=



Horní propust: detekce hran





Horní propust: detekce hran

Reference

- 1.[DSPGuide] – The Scientist guide to DSP, http://www.dspguide.com/pdfbook.htm
- 2.[Wolfram] – http://mathworld.wolfram.com/FastFourierTransform.html
- 3.[RecipC] – Numerical Recipes in C, http://www.library.cornell.edu/nr/bookcpdf.html
- 4.[FFT2] – http://astronomy.swin.edu.au/~pbourke/other/fft2d/

Reference

- 5.[Impl] – How to implement the FFT algorithm – http://www.codeproject.com/cpp/howtofft.asp