Noordelijke Hogeschool Leeuwarden

Computer Vision

Fourier Transform

27 August 2008

j.van.de.loosdrecht@tech.nhl.nl, jaap@vdlmv.nl

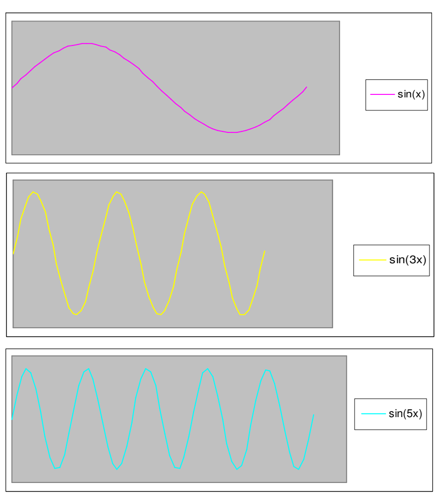
Fourier Transform (FT)

- Introduction
- Applications
  - Finding periodic structures
  - Convolution theorem
  - Removing defocus and motion blur (deconvolution)
  - Correlation
  - Calculation of sharpness in the image

Fourier Transform

- Introduction
  - Frequency
  - Fourier analysis
  - 1D Fourier transform
  - Complex numbers
  - 2D Fourier transform
  - Displaying FT of images
  - Some examples of synthetic images
  - Interpreting the frequency domain
  - Demonstration image shift (*)
  - Demonstration exchange of magnitude and phase (*)

Frequency



low

middle

high

Fourier analysis

Every periodic signal can be written as a summation of a series of sine shaped signals with an increasing frequency:

- **DC component (harmonic 0)**
- **h1 * harmonic 1 (base frequency)**
- **h2 * harmonic 2 (2 x base frequency)**
- **h3 * harmonic 3**

  ….
- **hn * harmonic n**

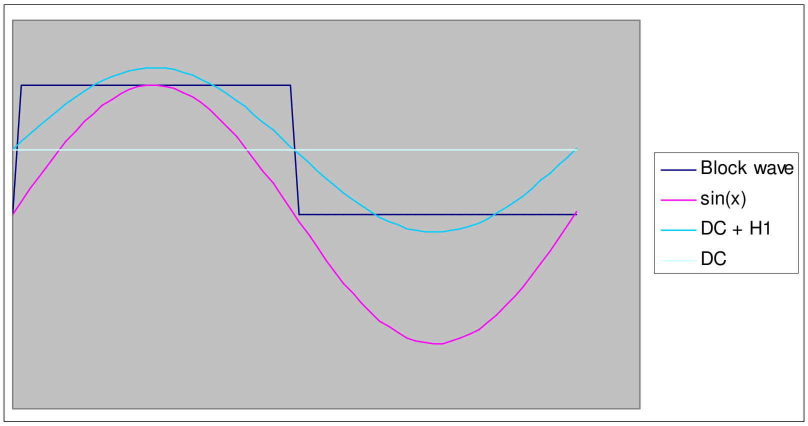
Fourier analysis of block wave

A block wave can be approximated with:

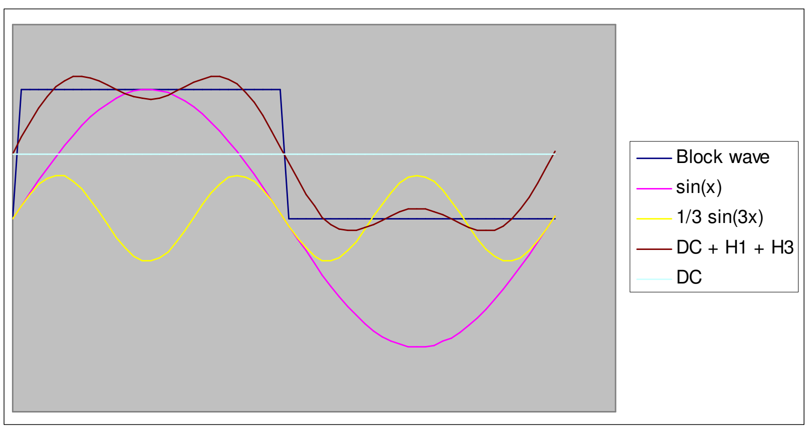
$$f(x) = \frac{h}{2} + \frac{2*h}{\pi}\left(\frac{\sin(x)}{1} + \frac{\sin(3x)}{3} + \frac{\sin(5x)}{5} + ...\right)$$

Where the block wave has an amplitude of h and a period of 2π.

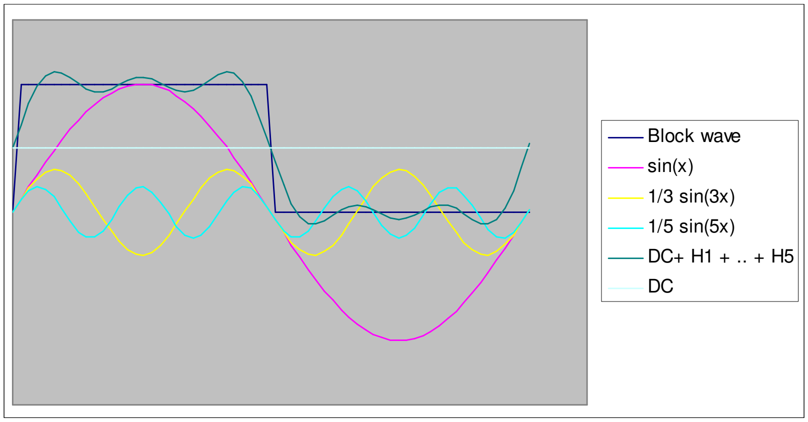
Block wave approximation



Block wave approximation

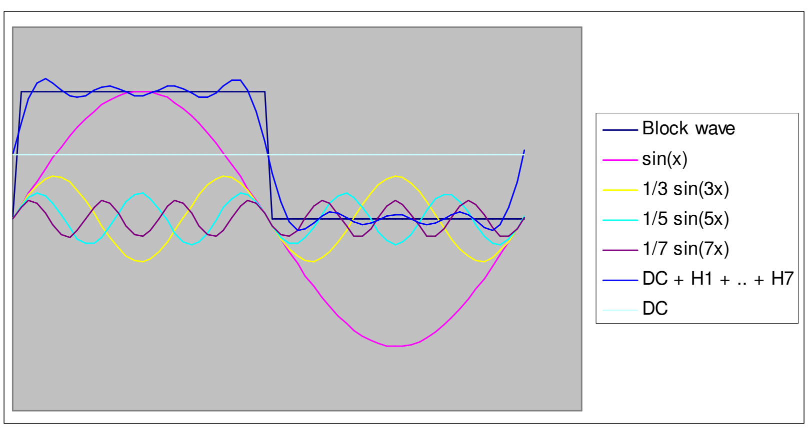


Block wave approximation



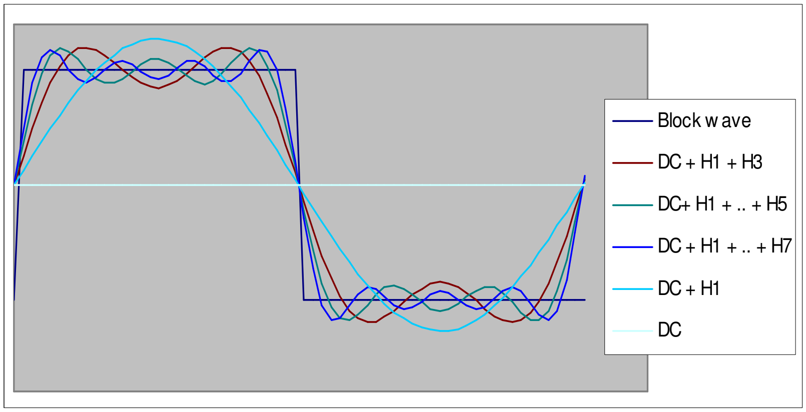
28-aug-08 Fourier Transform 9

Block wave approximation



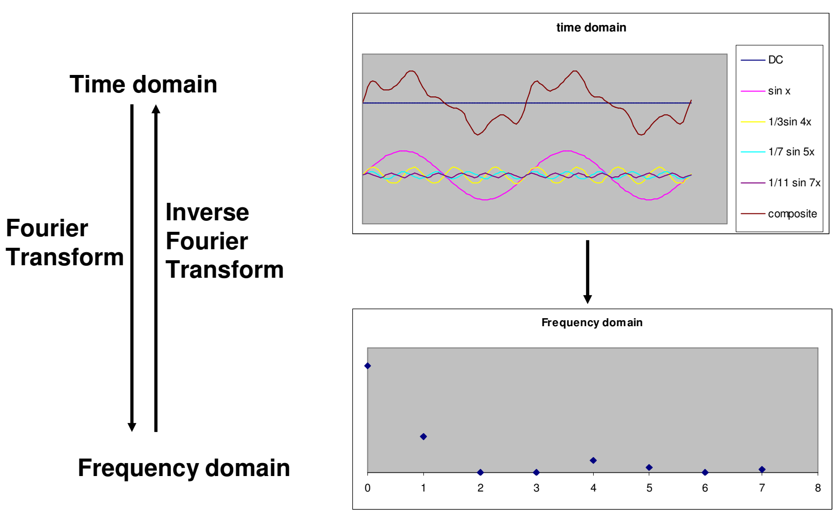
28-aug-08 Fourier Transform 10

Block wave approximation



28-aug-08 Fourier Transform 11

Fourier analysis



28-aug-08 Fourier Transform 12

Fourier analysis

Example:

by analyzing the noise of a machine, anti-noise can be generated by inverting the phase of the noise

1D Fourier transform

Fourier transform

$$FT(u) = \int_{-\infty}^{+\infty} f(x) e^{-2\pi iux} dx$$

Inverse Fourier transform

$$f(x) = \int_{-\infty}^{+\infty} FT(u) e^{2\pi iux} du$$

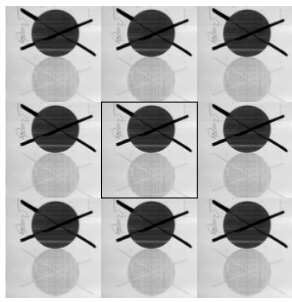
$$e^{-2\pi iux} = \cos(2\pi ux) - i\sin(2\pi ux)$$

$$i = \sqrt{-1}$$

Fast Fourier Transform (FFT): fast implementation for signals with a length of a power of two

2D Fourier transform

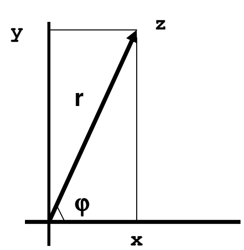
Images:
- **are 2D signals**
- **are interpreted as continuous signals**



Complex numbers

Cartesian representation
c = x + i y, where x and y are real numbers
- **x: real part**
- **y: imaginary part**

Polar representation
- **r: magnitude (amplitude)**
- **phi: phase (angle)**



Typical framework for FT operations

IntImage

↓ Conversion

ComplexImage

↓ FT

ComplexImage

↓ operations

ComplexImage

↓ FT^{inv}

ComplexImage

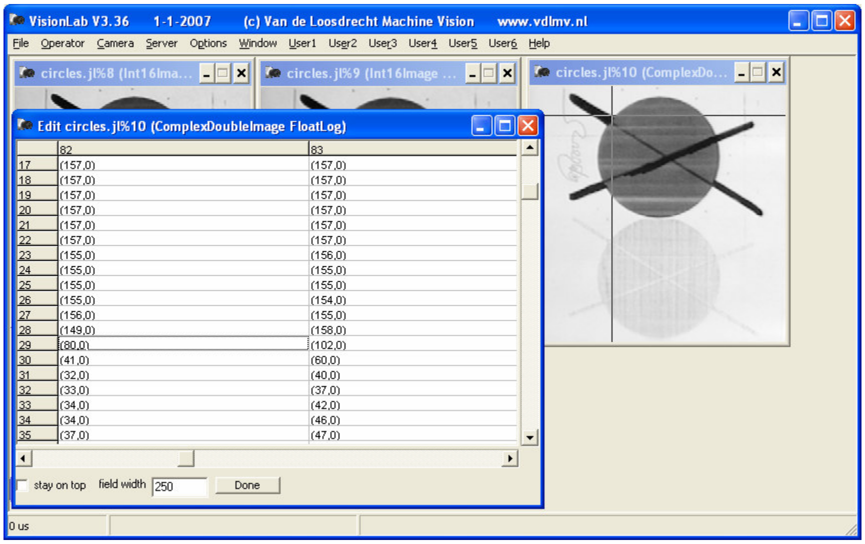
↓ Conversion

IntImage

Demonstration FT

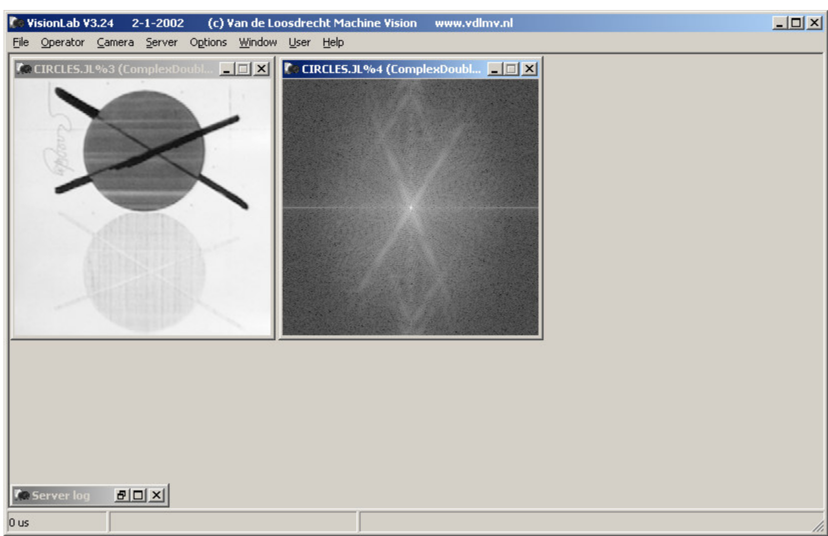
- **Open image circles.jls**
- ***Zoom 256 256 BilinearPixelInterpolation***
- **convert ComplexDoubleImage**
- **Show with edit convert image that pixels are complex numbers with imaginary part = 0**
- **FT complex image and use display LUT FloatLog**
- **Analyse edit result**
- **Perform the inverse FT on FT image**
- **Analyse edit result = almost original image (some rounding errors)**

Convert to ComplexDouble image



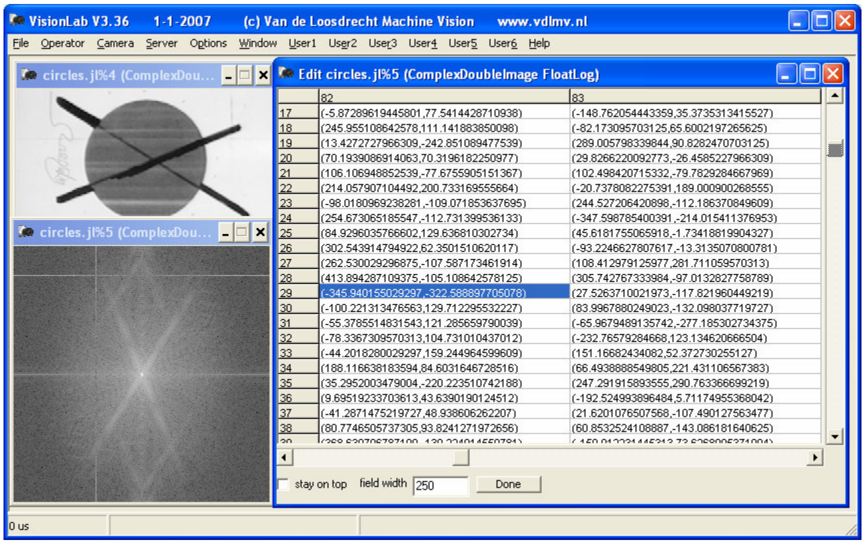
28-aug-08 Fourier Transform 19

FT of circles



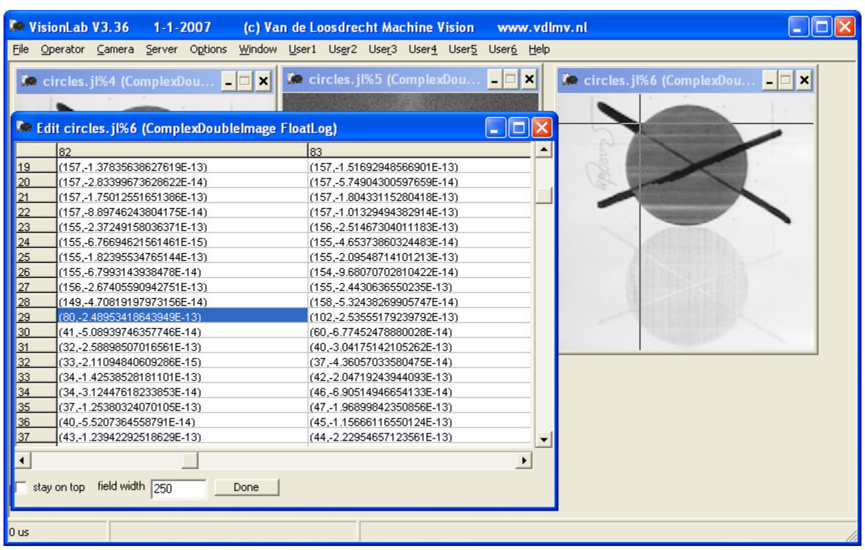
28-aug-08 Fourier Transform 20

FT of circles



28-aug-08 Fourier Transform 21

Inverse FT



28-aug-08 Fourier Transform 22

Displaying complex images

In VisionLab the power spectrum of a complex image is displayed

The power spectrum is the square of the magnitude

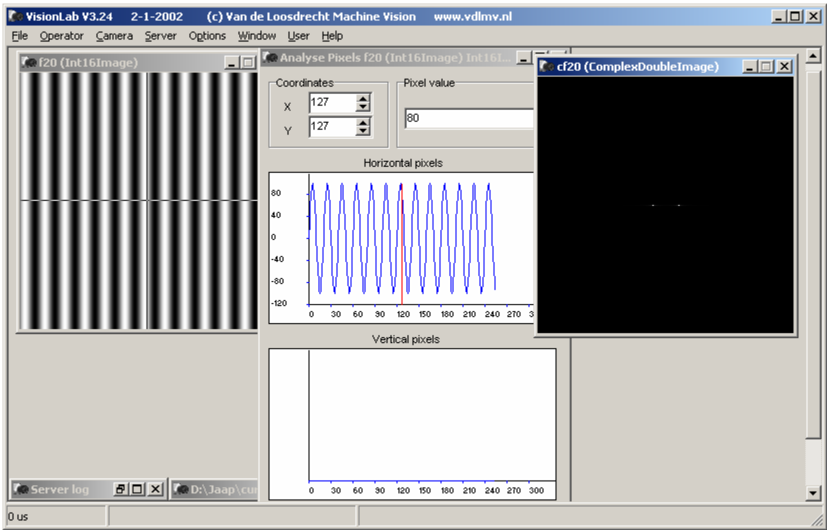
The power spectrum of the FT of an image is symmetric to the origin and has often a high dynamical range

There are two display LUT's:
- FloatLog
- FloatLinear

Demonstration sinus patterns

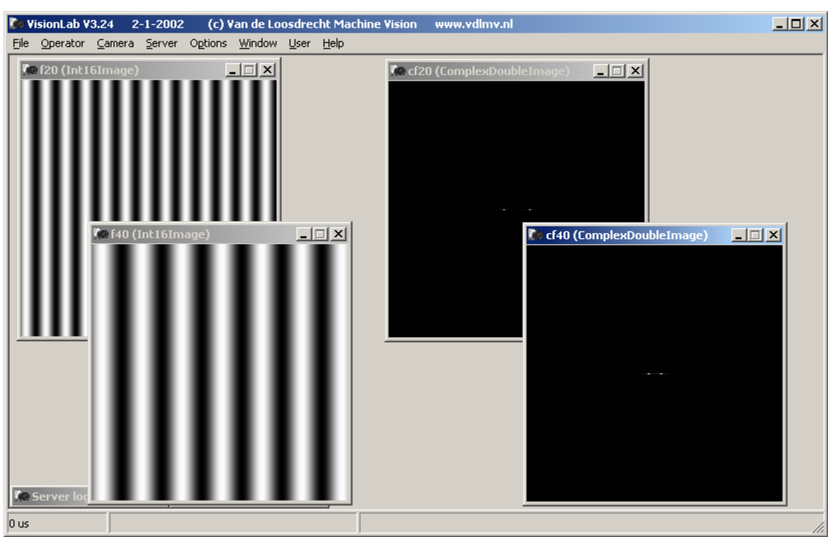
- Open script fftd1_2.jls
- Execute first part to generate a vertical sinus pattern and its FT,
  note: direction in frequency domain (FT) is perpendicular to direction in space domain
- Continue script for second part in order to generate a vertical sinus pattern with a lower frequency and its FT,
  note: maxima in frequency domain are closer to the origin
- Continue script for third part in order to generate a vertical sinus pattern with is 45 degrees rotated and its FT,
  note: maxima frequency domain are perpendicular to patterns in space domain
- Continue script for fourth part in order to generate a 2D sinus pattern and its FT

Vertical sinus pattern



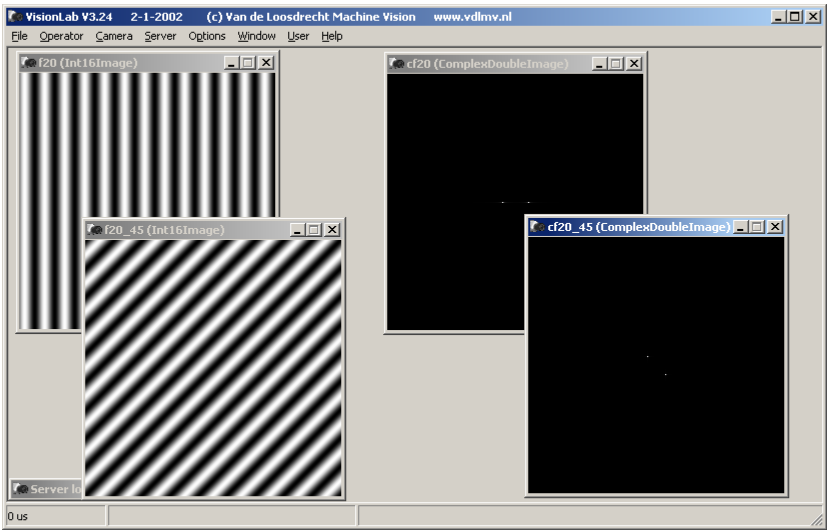
28-aug-08 Fourier Transform 25

Vertical sinus pattern with a lower frequency



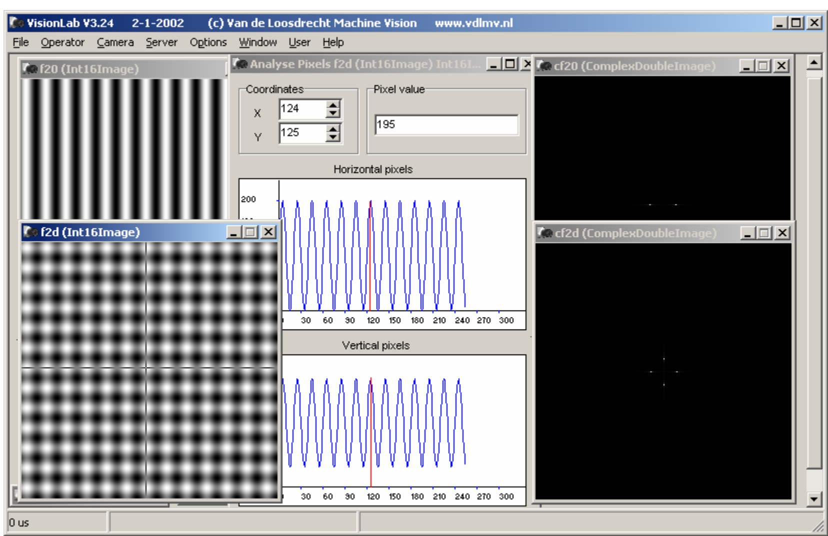
28-aug-08 Fourier Transform 26

Sinus pattern is rotated 45 degrees



28-aug-08 Fourier Transform 27

2D sinus pattern



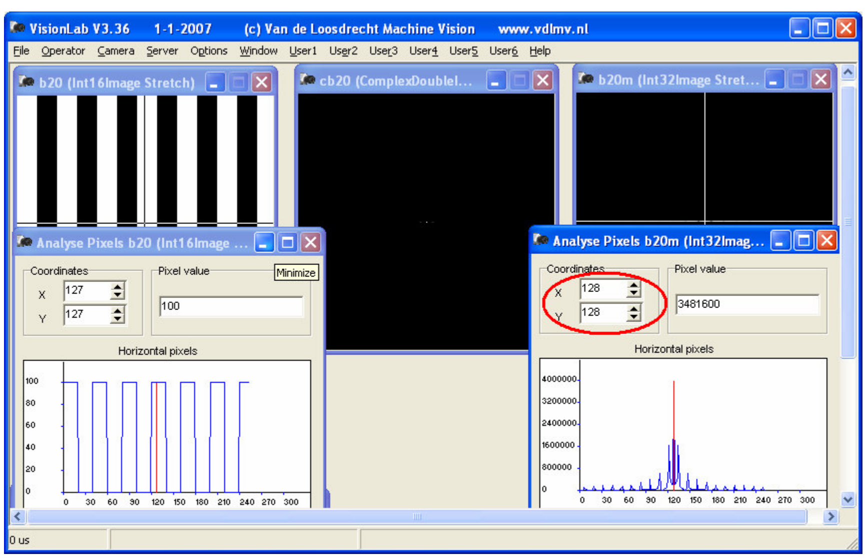
28-aug-08 Fourier Transform 28

Demonstration block patterns

- Open script fft_b.jls
- Execute first part to generate a vertical block pattern and its FT,
  note: only odd frequencies are present in frequency domain
- Execute second part to generate a vertical block pattern with a higher frequency and its FT,
  note: peaks in frequency domain have a bigger distance
- Execute third part to generate a 2D block pattern and its FT

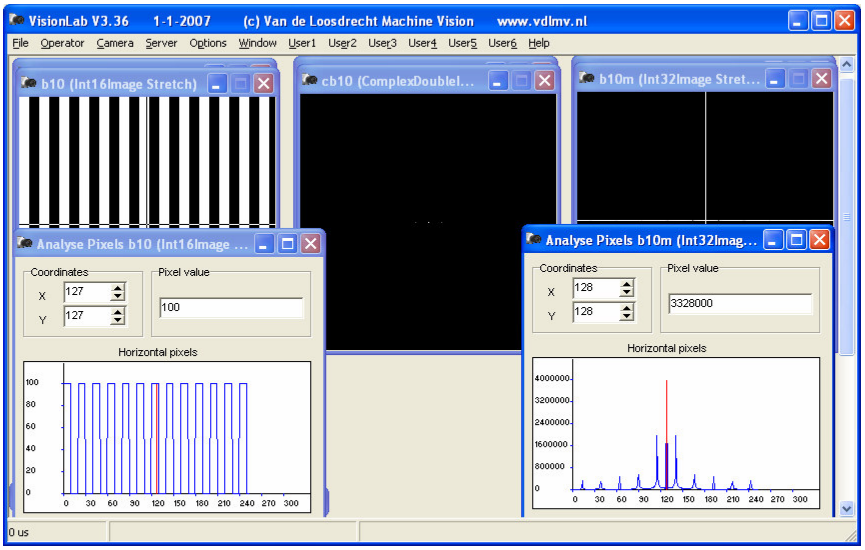
28-aug-08 Fourier Transform 29

Vertical block pattern



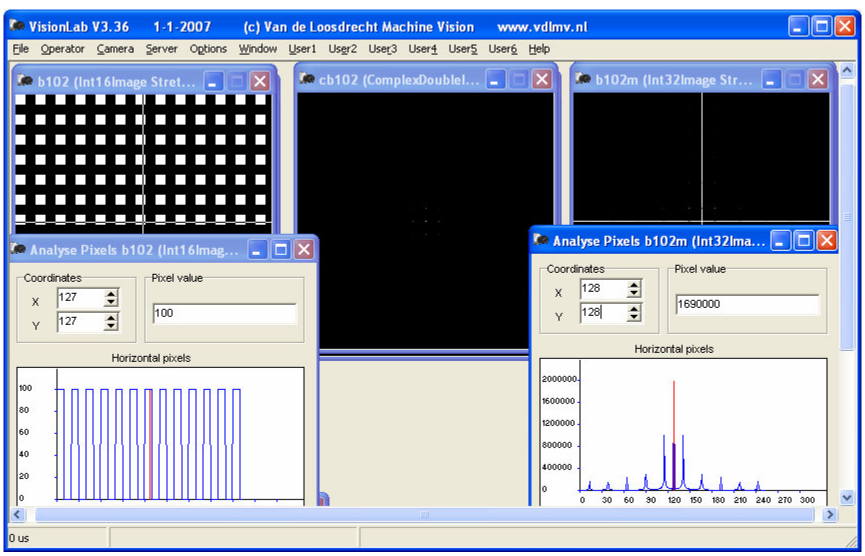
28-aug-08 Fourier Transform 30

Vertical block pattern with a higher frequency



28-aug-08 Fourier Transform 31

2D block pattern



28-aug-08 Fourier Transform 32

Interpreting the power spectrum

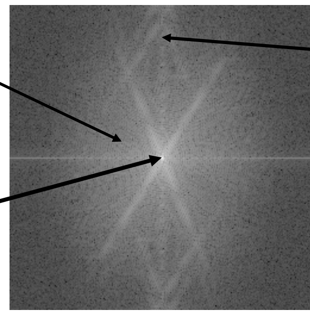
Frequencies increase from the origin

low frequency

high frequency

DC component

= sum pixels (origin)



28-aug-08 Fourier Transform 33

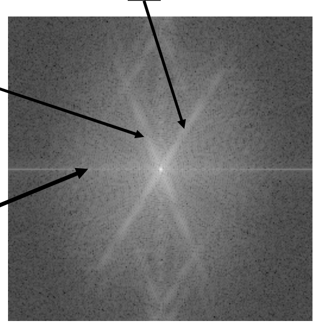
Interpreting the power spectrum

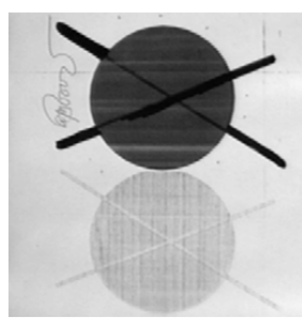
Explain structures in power spectrum ??

1

2

3

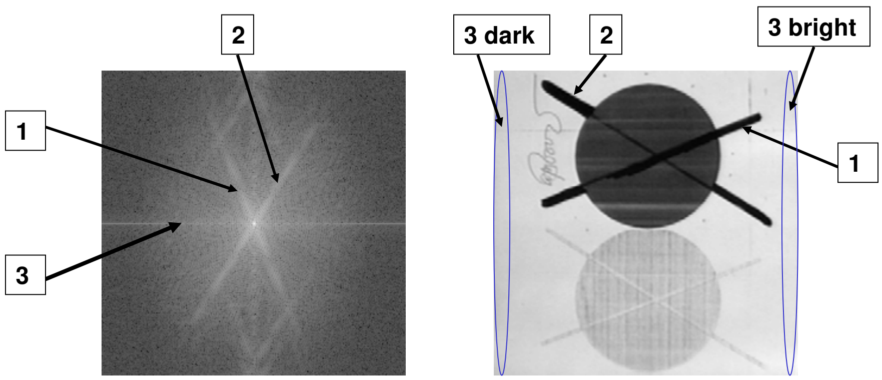




28-aug-08 Fourier Transform 34

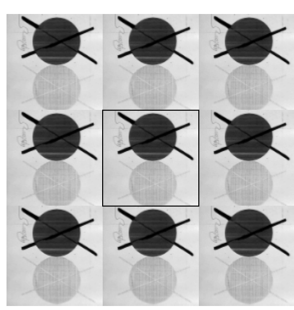
Interpreting the power spectrum

Structures in space domain are perpendicular to the corresponding structures in the frequency domain



Interpreting the power spectrum

Explanation for structure nr 3



Interpreting the frequency domain

It is not easy possible to interpret the Cartesian representation.

Some interpretation can be given in the Polar representation:

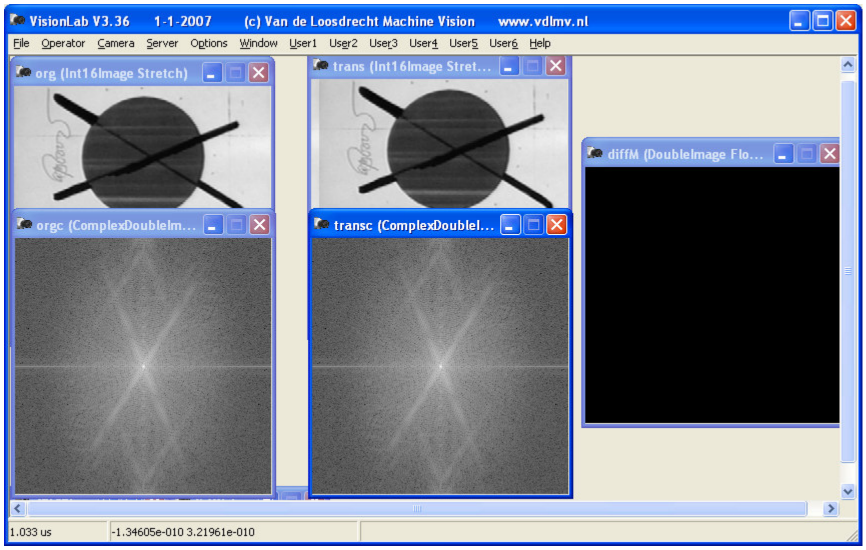
- The magnitude information gives information on the frequencies of the periodic structures contained in the image
- The phase information contains information about the position of those structures

The power spectrum (square of the magnitude) gives the distribution of the energy of the frequencies in the image

Demonstration image shift (*)

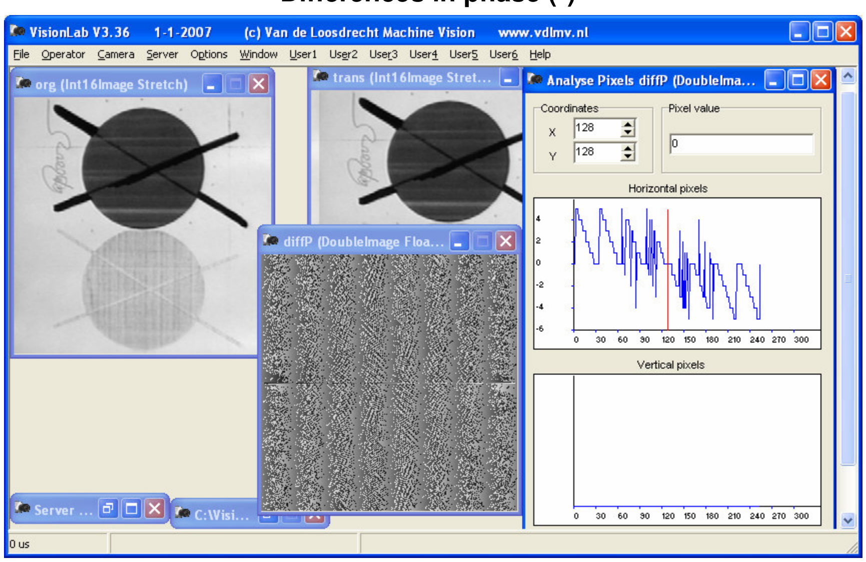
- Use script fft_pshift.jls
  - image circles.jl is cyclic shifted 8 pixels to the right
  - Original and shifted are Fourier transformed
  - Compare magnitudes of both transformed images
  - Compare phases of both transformed images

Differences in magnitudes (*)



28-aug-08 Fourier Transform 39

Differences in phase (*)

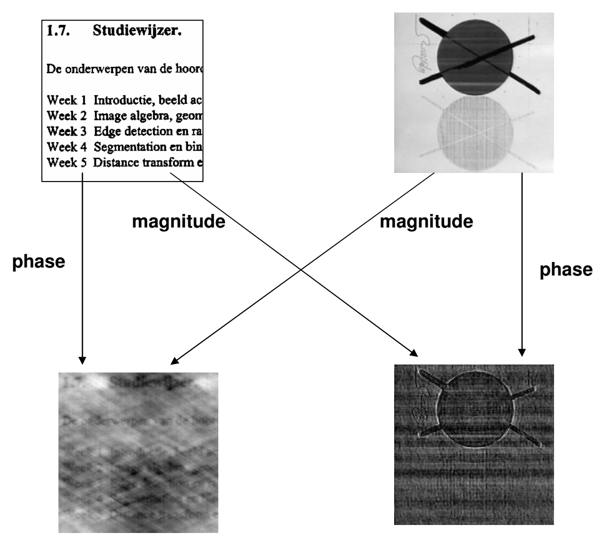


28-aug-08 Fourier Transform 40

Demonstration exchange of magnitude and phase (*)

- Use script fft_phase.jls
  - Image circles.jl is Fourier transformed
  - Image text.jl is Fourier transformed
  - Generate for both transformed images an image with magnitude information and an image with phase information
  - A complex image with magnitude information from circles and phase information from text is created
  - A complex image with phase information from circles and magnitude information from text is created
  - Both new complex image are inversed Fourier transformed
- Conclusion the phase information determines the content of the image

Demonstration exchange of magnitude and phase (*)



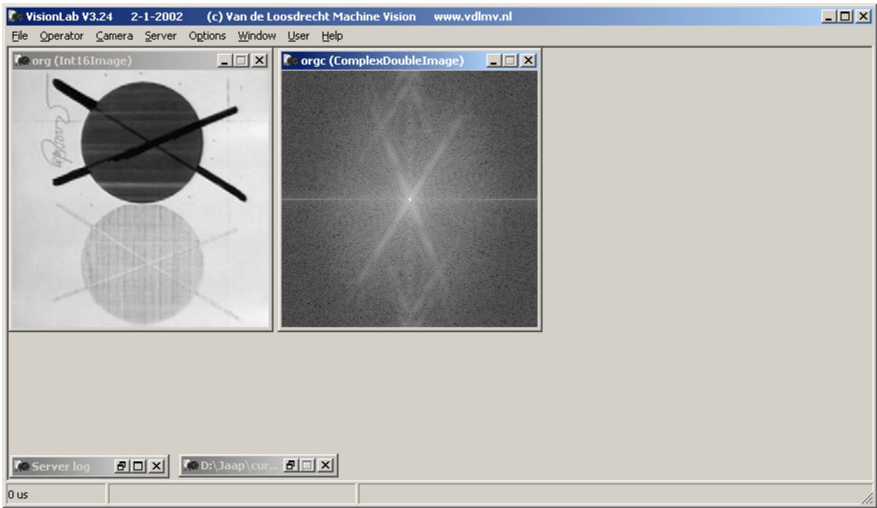
Applications of Fourier transform

- Applications:
  - Finding periodic structures
    - Low and high pass filter
    - Band reject filter
  - Convolution
  - Removing defocus and motion blur (deconvolution)
  - Correlation

Demonstration low and high pass filter

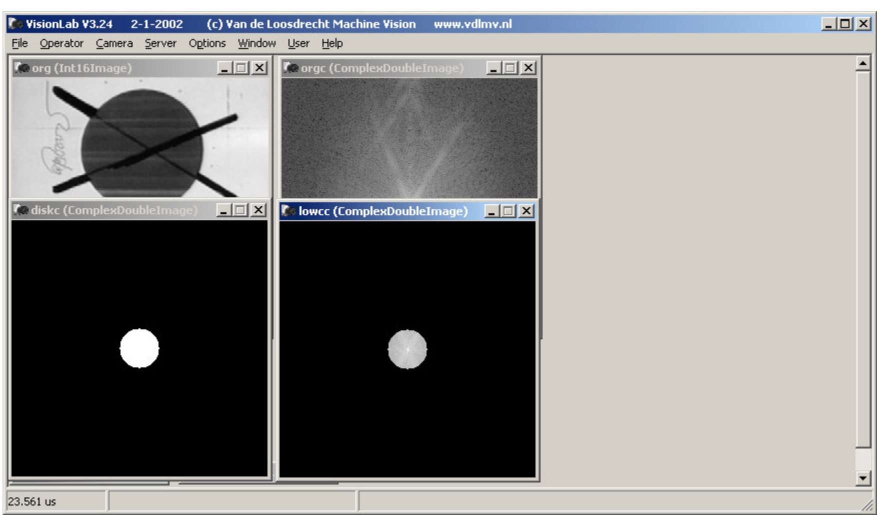
- Use script fft_filter.jls
  - Image circles.jl is Fourier transformed
  - Ideal low pass filter:
    - A circular mask image is created and multiplied with Fourier transform, this means that only the low frequencies are selected
    - Result is inversed Fourier transformed, note artifacts in image due to ideal (theoretical) low pass filter
  - Practical low pass filter:
    - A 'soft' circular mask with a Gaussian distribution is created and multiplied with Fourier transform, this means that only the low frequencies are selected
    - Result is inversed Fourier transformed and shows only the low frequencies
  - High pass filter:
    - A mask for the high pass filter is created and multiplied with Fourier transform, this means that the low frequencies are blocked
    - Result is inversed Fourier transformed and shows only the high frequencies

Image circles.jl and it's FT



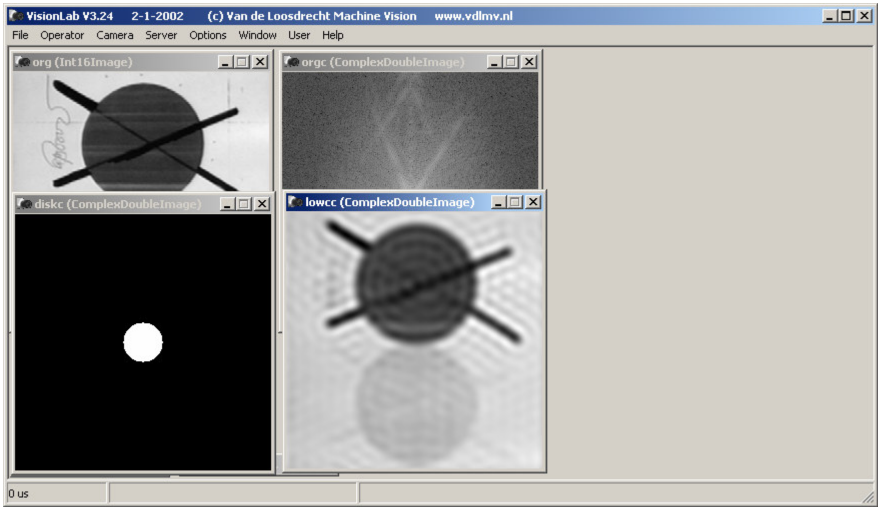
28-aug-08 Fourier Transform 45

A circular mask image is created and multiplied with Fourier transform



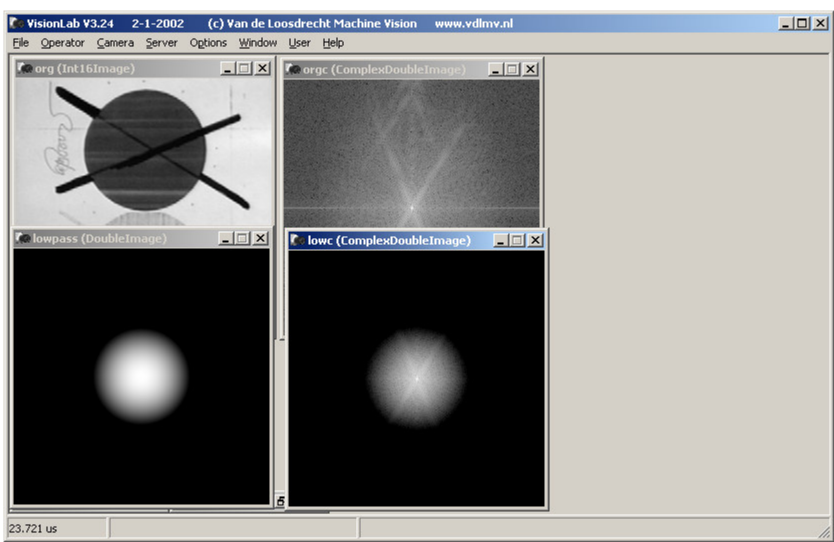
28-aug-08 Fourier Transform 46

Result of ideal low pass filter



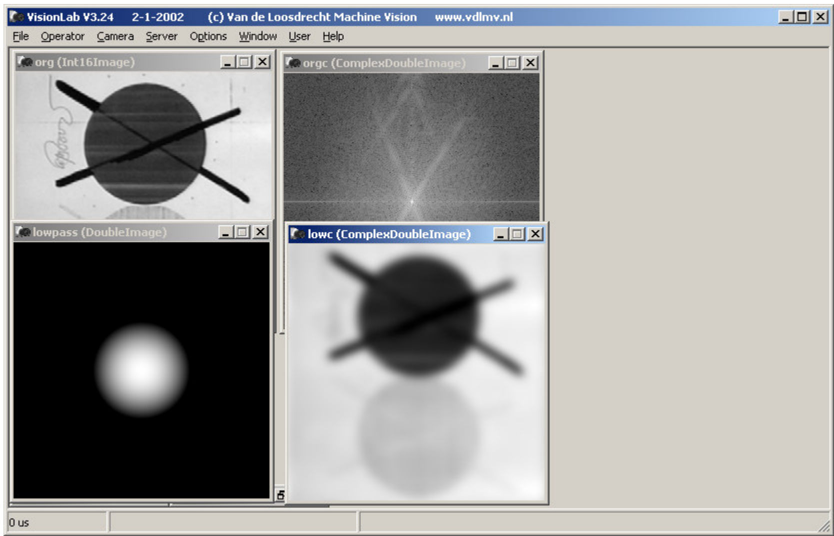
28-aug-08 Fourier Transform 47

A 'soft' circular mask image is created and multiplied with Fourier transform



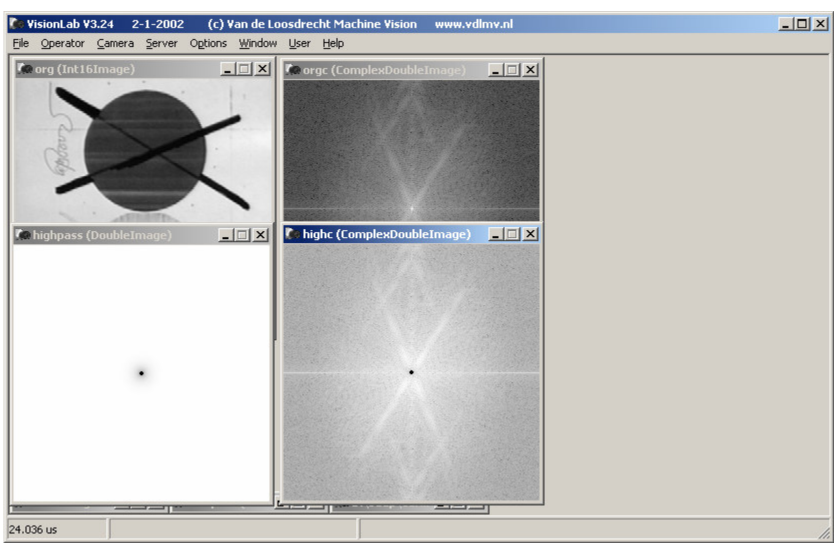
28-aug-08 Fourier Transform 48

Result of Gaussian low pass filter



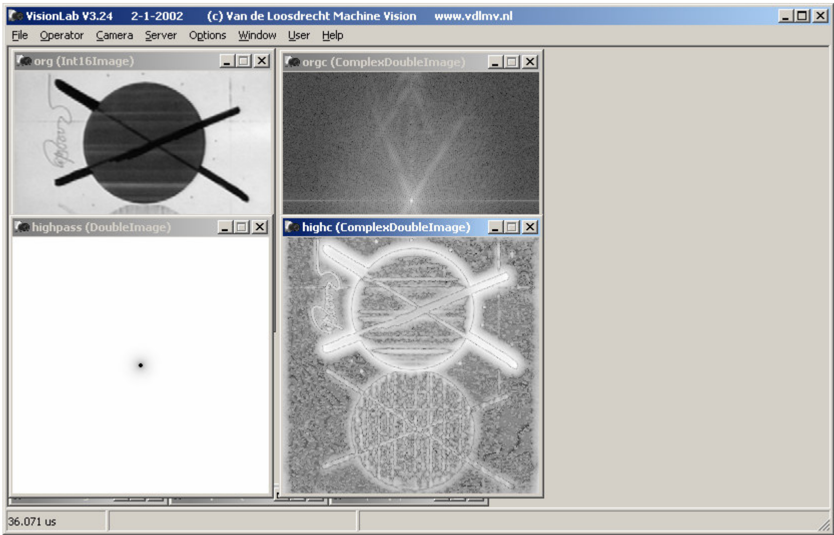
28-aug-08 Fourier Transform 49

Mask for high pass filter



28-aug-08 Fourier Transform 50

Result of high pass filter



28-aug-08 Fourier Transform 51

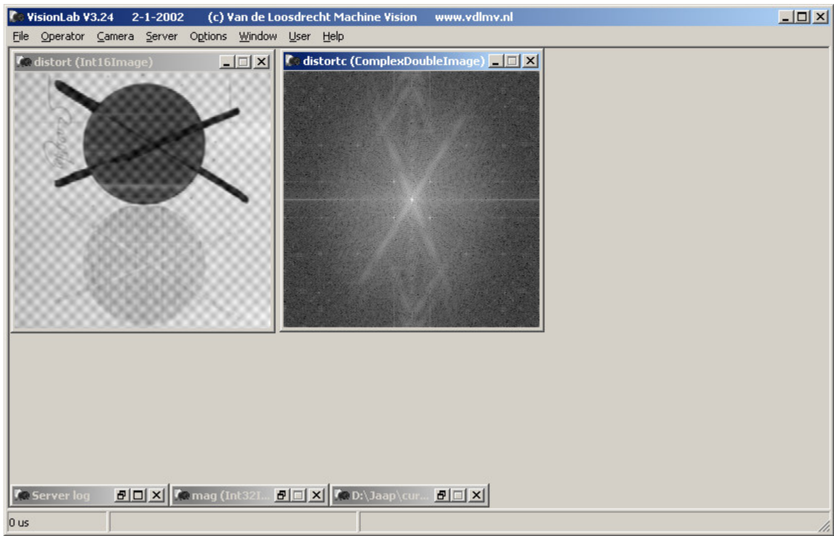
Demonstration band reject filter

Purpose is to demonstrate how periodic structures (interference) can be removed from an image

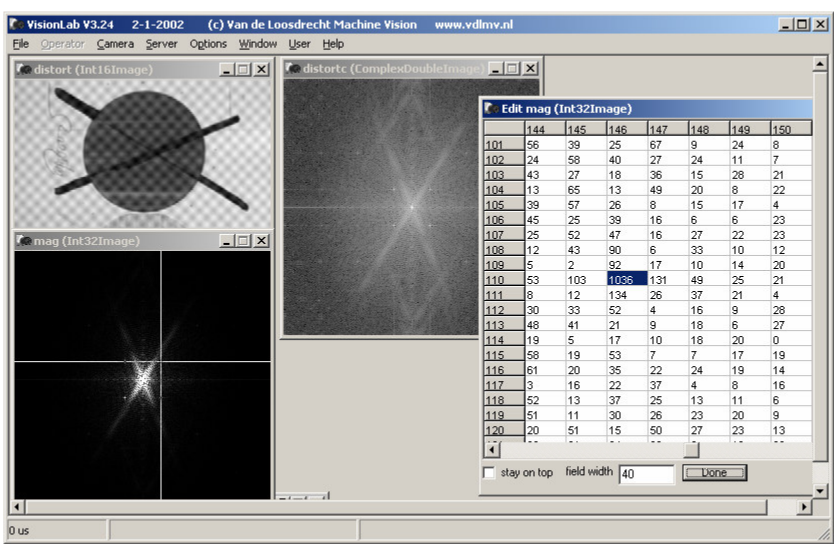
- **Use script fft_bandreject.jls**
  - **A distorted image circles.jl is created by adding a 45 degrees rotated sinus pattern. The distorted image is Fourier transformed.**
  - **The magnitude information is extracted and the positions of the maxima is searched for**
  - **A filter image with small Gaussian circles is created on the position of the maxima, note: the higher order components are not selected !**
  - **The filter image is inverted because the frequencies must be blocked**
  - **Inverted filter is multiplied with FT**
  - **Result is inversed FTed, note still artefacts in image, result is not perfect at the borders, due to high order components**

28-aug-08 Fourier Transform 52

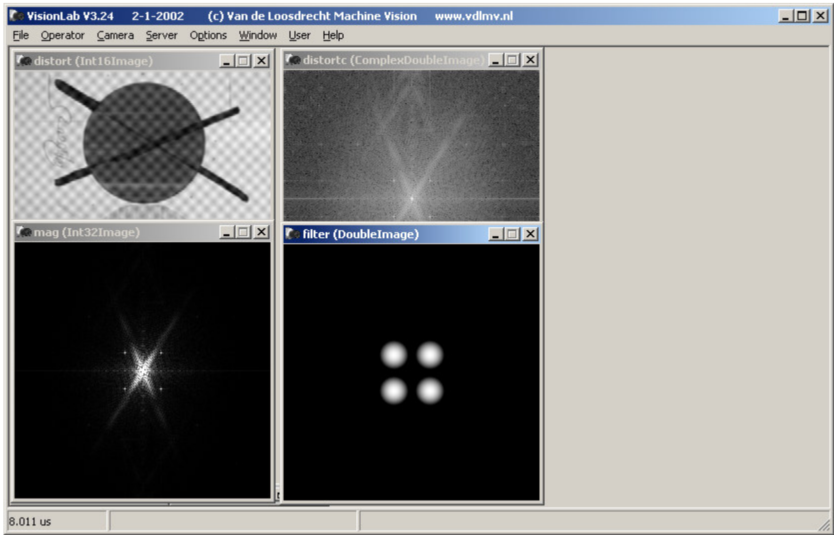
Distorted image and it's FT



Search for the maxima

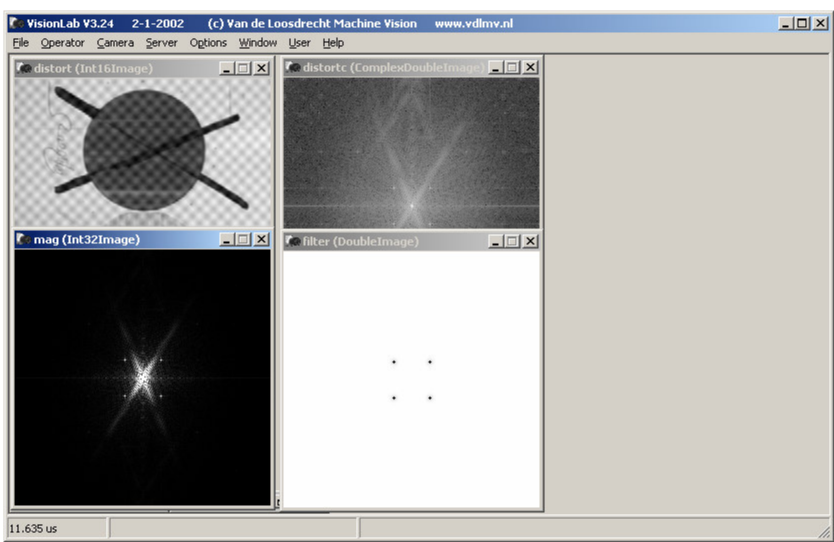


Filter image with small Gausian circles



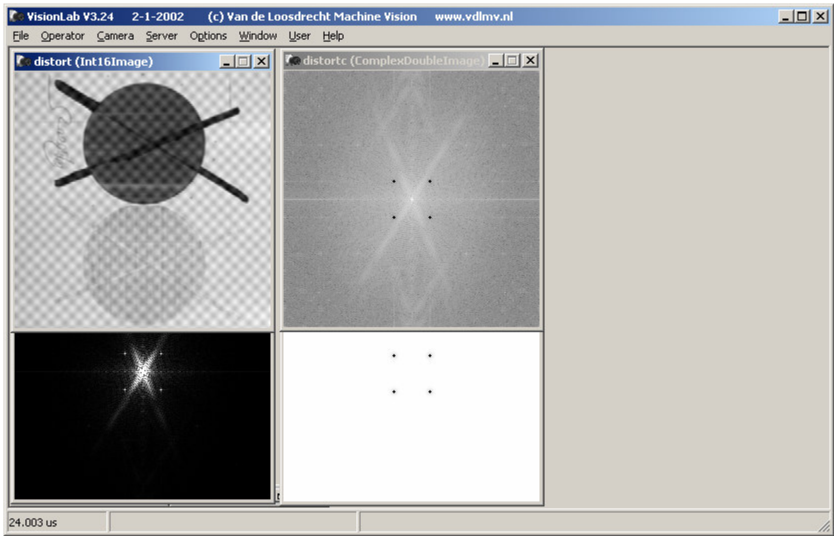
28-aug-08 Fourier Transform 55

Inverted filter image



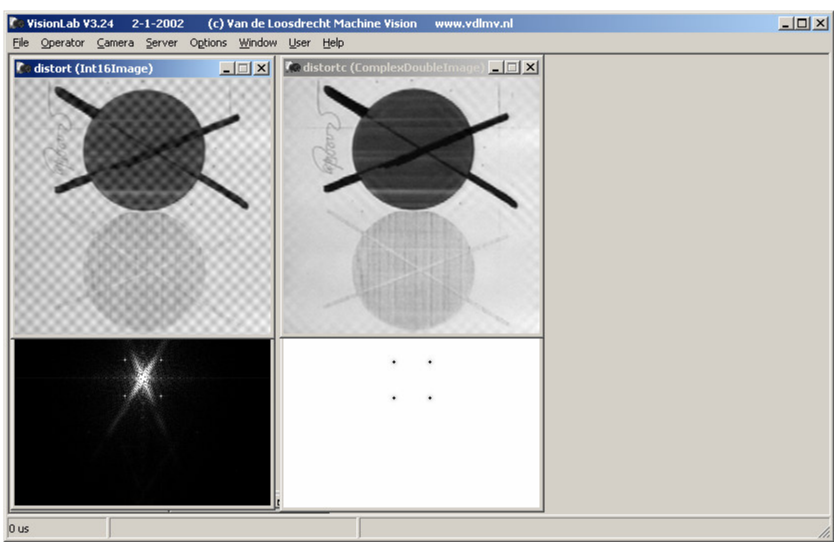
28-aug-08 Fourier Transform 56

Inverted filter is multiplied with FT



28-aug-08 Fourier Transform 57

Inverse FT



28-aug-08 Fourier Transform 58

Convolution theorem

Notation:

$$\otimes = \text{convolution}$$

- **Spatial domain**

$$image \otimes mask = convoluted\_image$$

- **Frequency domain**

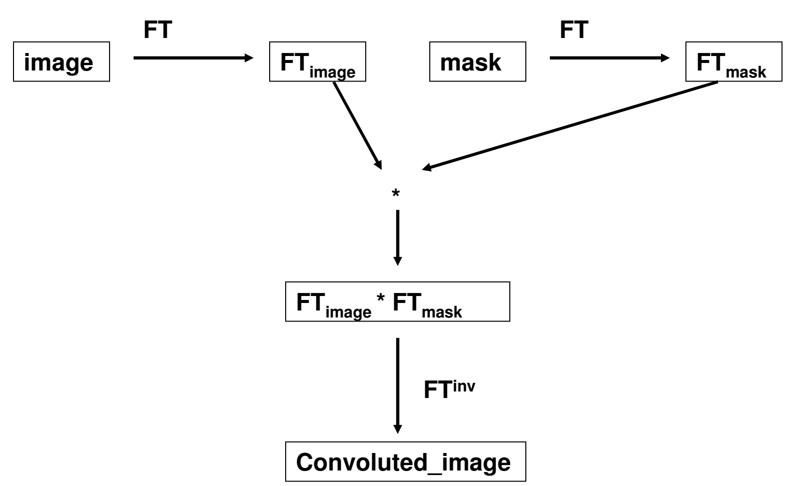
$$FT(image) * FT(mask) = FT(convoluted\_image)$$

- **Convolution using FT**

$$convoluted\_image = FT^{inv}(FT(image) * FT(mask))$$

28-aug-08 Fourier Transform 59

Convolution theorem



28-aug-08 Fourier Transform 60

Convolution theorem

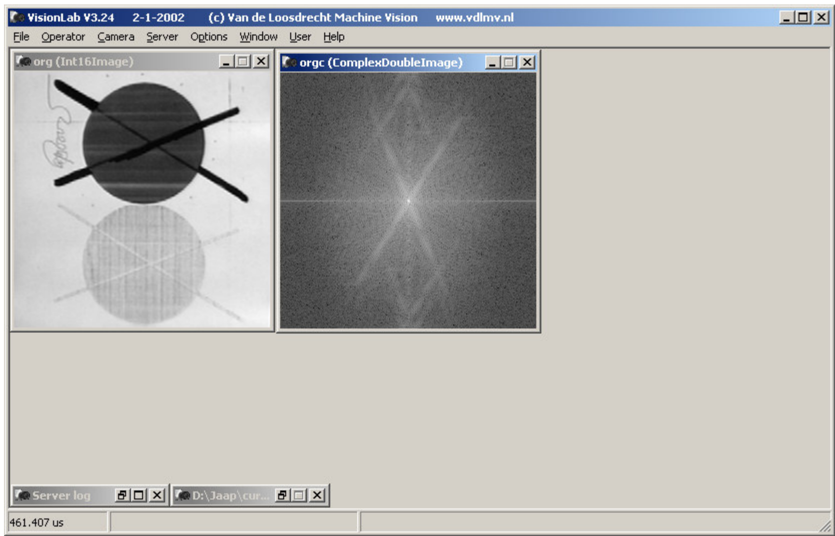
A convolution in the spatial domain is exactly equivalent to a multiplication in the frequency domain.

If the kernel is smaller then the image, it is padded with zeroes to the full image size.
(some round off errors at the borders)

Demonstration Convolution theorem

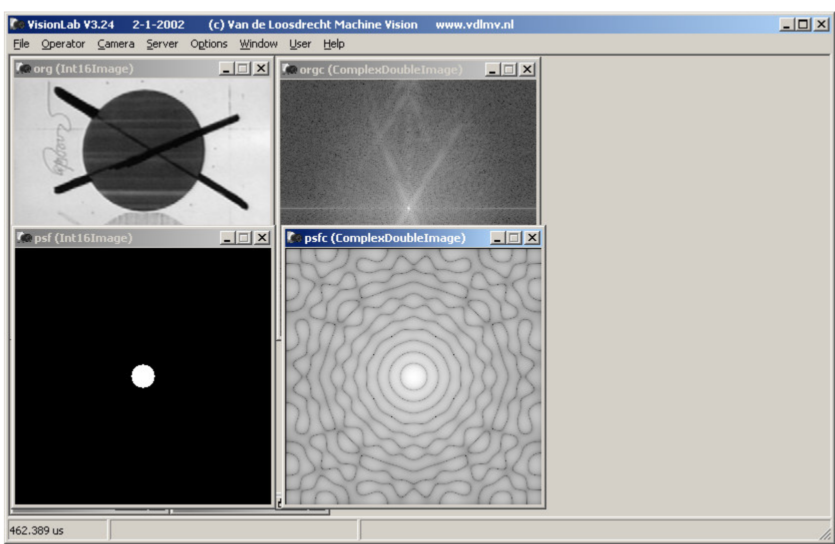
- **Use script fft_conv.jls**
  - **Image circles.jl and it's FT**
  - **A point spread function (psf) mask is created and it's FT**
  - **FT 's are multiplied and result is reversed FT,**
    note similar result as convolution in spatial domain with a big smoothing mask

Image circles and it's FT



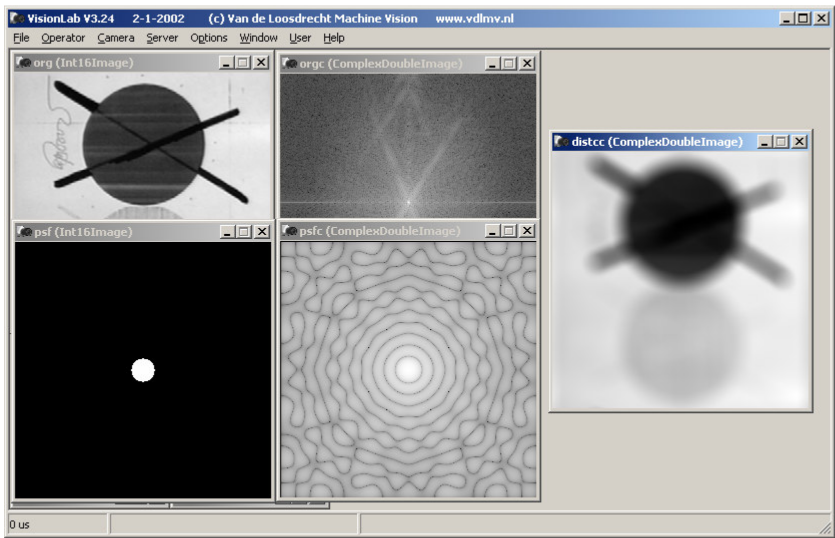
28-aug-08 Fourier Transform 63

psf mask and it's FT



28-aug-08 Fourier Transform 64

FT 's are multiplied and result is reversed FT



Removing defocus blur (deconvolution)

Notation:

$$\otimes = \text{convolution}$$

$$FT_{image} = FT(image)$$

Making unsharp:

- **Spatial domain**

  $$\text{image} \otimes \text{psf} = \text{blurr}$$

- **Frequency domain**

  $$\text{FT}_{\text{image}} \times \text{FT}_{\text{psf}} = \text{FT}_{\text{blurr}}$$

Removing defocus blur (deconvolution)

Making unsharp image sharp in frequency domain:

$$\frac{FT_{blurr}}{FT_{psf}} = FT_{image}$$

But FT_{psf} contains complex zeroes:

$$\frac{FT_{blurr}}{FT_{psf}} = \frac{FT_{blurr}}{FT_{psf}} \times \frac{\overline{FT_{psf}}}{\overline{FT_{psf}}} = \frac{FT_{blurr} \times \overline{FT_{psf}}}{FT_{psf} \times \overline{FT_{psf}}} = \frac{FT_{blurr} \times \overline{FT_{psf}}}{\left|FT_{psf}\right|^2}$$

Divider is now a real, but still can be zero

28-aug-08 Fourier Transform 67

Wiener filter

Idea: add a small constant to the divider

$$\frac{FT_{blurr}}{FT_{psf}} = \frac{FT_{blurr} \times \overline{FT_{psf}}}{\left|FT_{psf}\right|^2 + k}$$

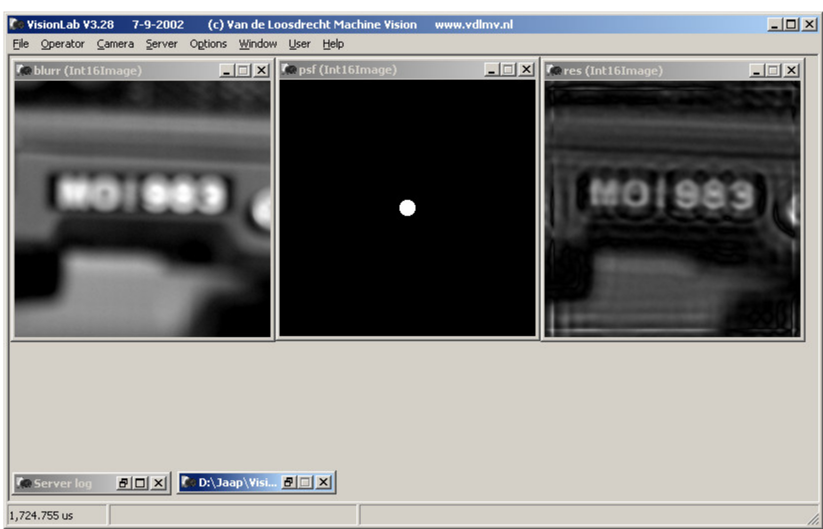
28-aug-08 Fourier Transform 68

Demonstration deconvolution

- Use script blurr.jls
  - lread blurr blurr.jl
  - display blurr
  - copy blurr psf
  - diskshape psf 128 128 8 1
  - display psf
  - copy blurr res
  - deconvolution res psf 0.01 // Wiener filter
  - display res
- Notes:
  - Result is not perfect
  - Parameters to tune: size of psf and estimate for k
  - Image restoration is a special branch of science and outside of scope of this course

28-aug-08 Fourier Transform 69

Deconvolution



28-aug-08 Fourier Transform 70

Exercise removing motion blur

- **Use image motion_blur.jl**
- **Try to remove the motion blur**
- **Hint: use script blurr.jls as template and determine suitable point spread function**

Answer: motion_blur.jls

28-aug-08 Fourier Transform 71

Correlation

Purpose: finding a specified pattern in an image

Idea: pattern is used as 'convolution mask', everywhere where the pattern fits to the image it will give a high convolution result

Theory:
find the peaks in $FT^{inv}(FT_{image} \times \overline{FT_{pattern}})$

Note: can not handle rotation and / or scaling

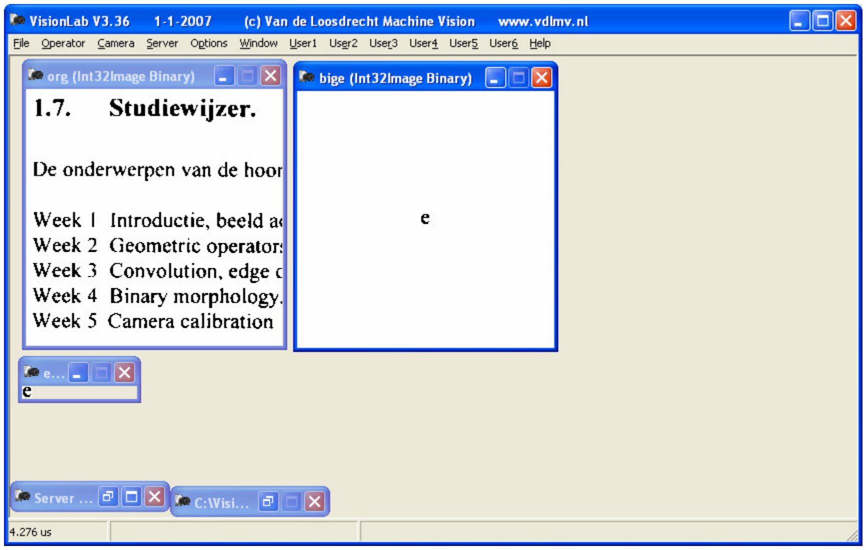
28-aug-08 Fourier Transform 72

Demonstration correlation

- Use script correlation.jls
  - Open image correlation.jl and select pattern
  - Add border to search pattern with average gray value
  - Correlate image with search pattern
  - Threshold on peaks of correlation
  - Map search result on original image,
    Notes:
    - One 'u' is seen for a 'e'

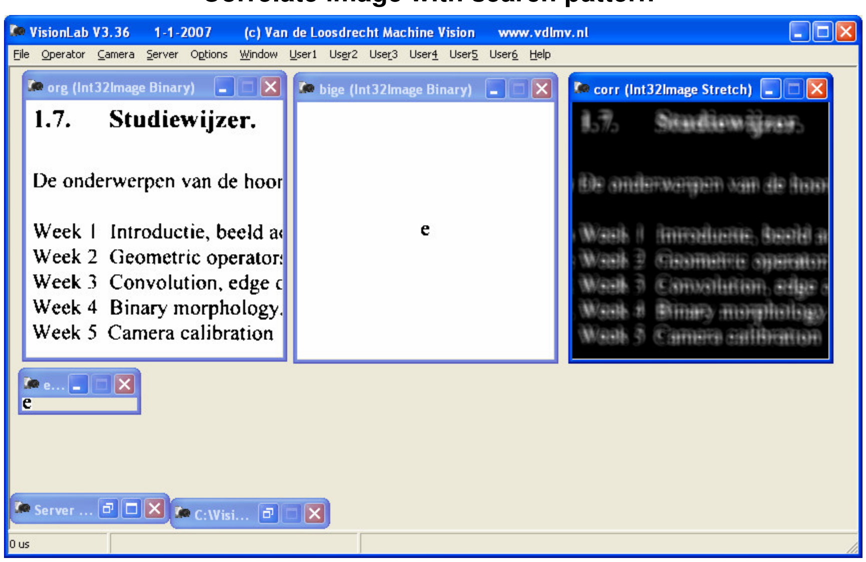
Image and search pattern

Add border to search pattern



28-aug-08 Fourier Transform 75

Correlate image with search pattern

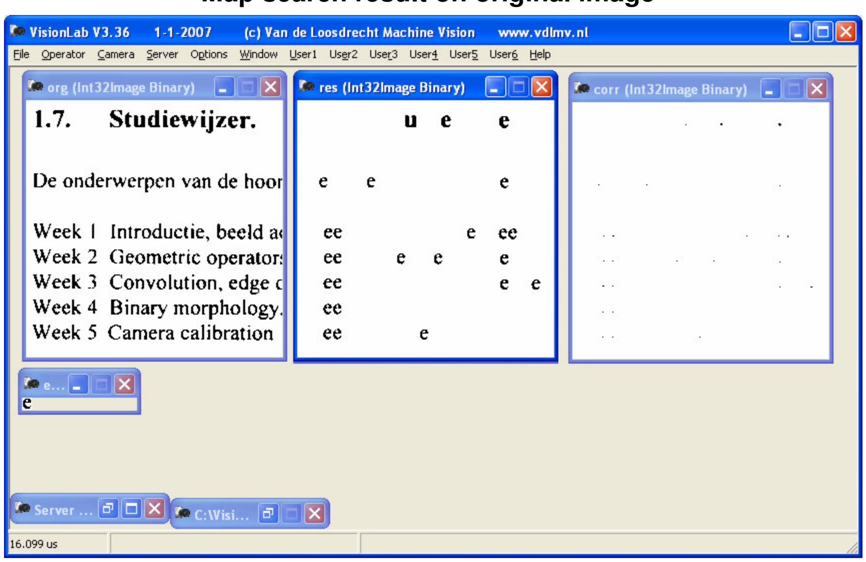


28-aug-08 Fourier Transform 76

Threshold on peaks of correlation

28-aug-08 Fourier Transform 77

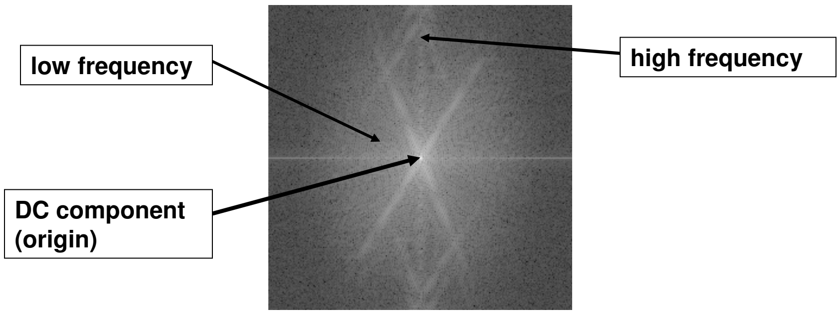
Map search result on original image



28-aug-08 Fourier Transform 78

Calculation of sharpness in the image

Frequencies increase from the origin



Idea: calculate the energy in the higher frequencies

28-aug-08 Fourier Transform 79

In focus value

InFocusValue (image, lowestfreqnr)

This operator calculates a value for how good the image is in focus (='sharpness'). The higher the returned value the more high frequencies are present in the image.

The parameter lowestfreqnr specifies the lowest frequency nr in the FT which is used in the calculation.

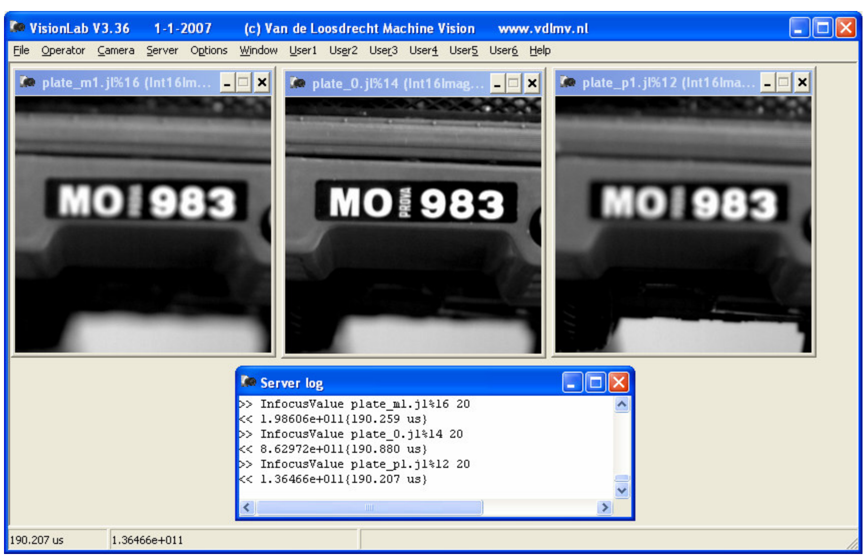
28-aug-08 Fourier Transform 80

Demonstration In focus value

- Apply operator InFocusValue on the images with lowestFreqNr = 20:
  - Plate_m1.jl (before focus)
  - Plate_0.jl (in focus)
  - Plate_p1.jl (after focus)

28-aug-08 Fourier Transform 81

Demonstration In focus value



28-aug-08 Fourier Transform 82

Exercise sharpness calculation

Make a script in order to calculate the sharpness in an image.
Use the following images to test the script:

- **Plate_m1.jl (before focus)**
- **Plate_0.jl (in focus)**
- **Plate_p1.jl (after focus)**

Modify the script in order to shoot continuously image with the camera and display for each image its sharpness

Answer first part: sharpness.jls