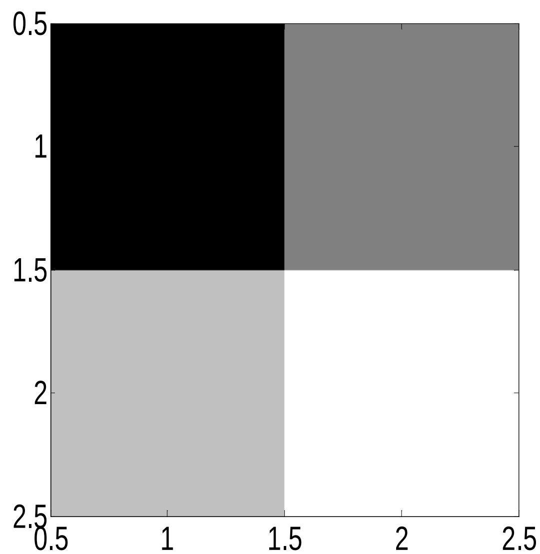
DIGITAL IMAGE PROCESSING

WHAT IS A (GRAY SCALE) DIGITAL IMAGE?

Matlab (command window) instructions:

```
>> A = [0 128; 192 255]                ENTER
 A =
      0      128
      192    255
>> image( A )                          ENTER
>> colormap( gray ( 256 ) )            ENTER
>> axis image                          ENTER
```



A gray scale digital image is a

- **matrix $f(x, y)$, where**
- **f is the intensity (gray scale value) of the image f at row x and column y**
- **and each coordinate (x, y) is represented by a pixel, where**
- **the pixel is coloured black if $f(x, y) = 0$ and coloured white if $f(x, y) = L - 1$**

(for the example on the previous page we have that $L = 256$)

Matlab instructions:

```
>> a = A(2,1)                                                    ENTER

 a =
       192

>> b = A(1,2)                                                    ENTER

 b =
       128
```

When `colormap( gray ( 256 ) )` is selected and an image (matrix) is **displayed** in Matlab with the `image` instruction, each entry (pixel) is internally "converted" to the `uint8` (unsigned 8-bit integer) data type. This implies that each pixel (matrix entry) is represented with an 8-bit byte, which implies $L = 2^8 = 256$ possible intensity values (gray scales). The instruction `imagesc` is the same as `image`, except that the data is scaled to use the full colormap, which can come in very handy. Images read into memory from a storage device are often in `uint8` format.

In order to be able to **manipulate** an `uint8` matrix (image) in double precision arithmetic, the matrix has to be converted into the `double` data type with `B = double( A )`. The inverse operation is achieved with `A = uint8( B )`. The Matlab default is `double`.

An image file, e.g. "lenna256.jpg" (JPEG format) can be **displayed** with the following Matlab instruction (IP Toolbox)

```
>> imshow( 'lenna256.jpg' )
```

ENTER

lenna256.jpg

To manipulate this image in Matlab, the following instructions are needed

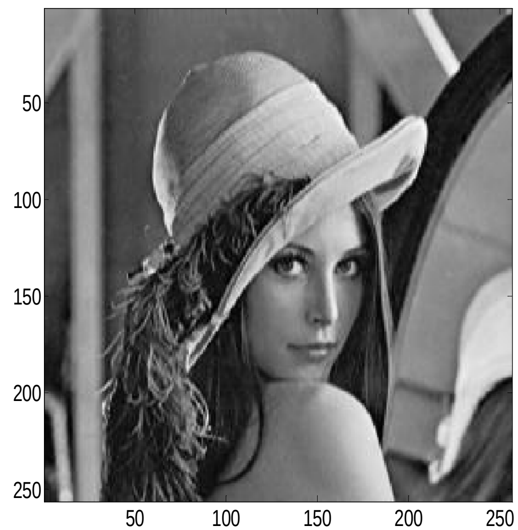
```
>> X = imread( 'lenna256.jpg' );                 ENTER
>> XX = double( X );                             ENTER
```

The variable `X` now represents a matrix that can be accessed/manipulated

```
>> mn = min( min( XX ) )                         ENTER
 mn =
      0
>> mx = max( max( XX ) )                         ENTER
 mx =
      239
>> sz = size( XX )                               ENTER
 sz =
      256    256
>> entry = XX(100,150)                           ENTER
 entry =
         173
```

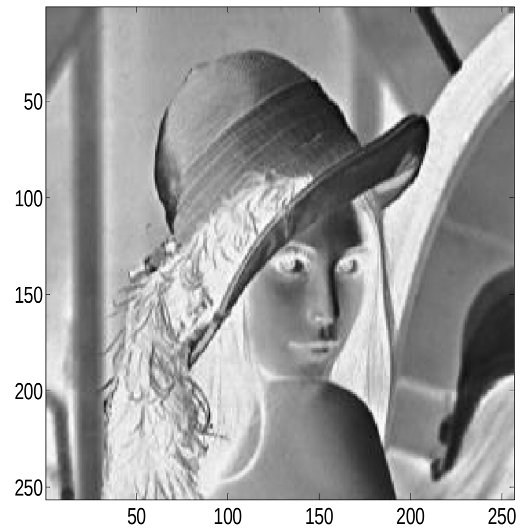
The matrix `XX` can again be displayed with the instructions

```
>> colormap( gray ( 256 ) );     ENTER
>> imagesc( XX )                 ENTER
>> axis image                    ENTER
```



We can now, e.g., calculate and display the negative of the image with the instructions

```
>> Y = max( max( XX ) ) - XX;    ENTER
>> colormap( gray ( 256 ) );     ENTER
>> imagesc( Y )                  ENTER
>> axis image                    ENTER
```



Any matrix can be saved to a disk. The following instruction saves the variable `Y` in the current directory as the Matlab file "lenna256neg.mat":

```
>> save lenna256neg.mat Y
```
ENTER

During subsequent Matlab sessions this file can be retrieved and automatically reassigned to the variable `Y` with the instruction

```
>> load lenna256neg.mat
```
ENTER

The matrix `Y` can also be saved in a variety of common image file formats, like JPEG, BMP, etc., with instructions like

```
>> imwrite(Y,gray(256),'lenna256neg.bmp','bmp')
```

The file "lenna256neg.bmp" (BMP format) can be displayed with the Matlab instruction (IP Toolbox): `>> imshow('lenna256neg.bmp')`

lenna256neg.bmp

In order to obtain a list of all the available Matlab functions in the Image Processing (IP) Toolbox, give the instruction: `>> help images`

E.g., the negative of an image can also be obtained with the `imcomplement` function, i.e.: `>> Y = imcomplement( X );`

For a demonstration of the IP Toolbox, enter `demo` in the command window and double-click on "Toolboxes" and then "Image Processing"...

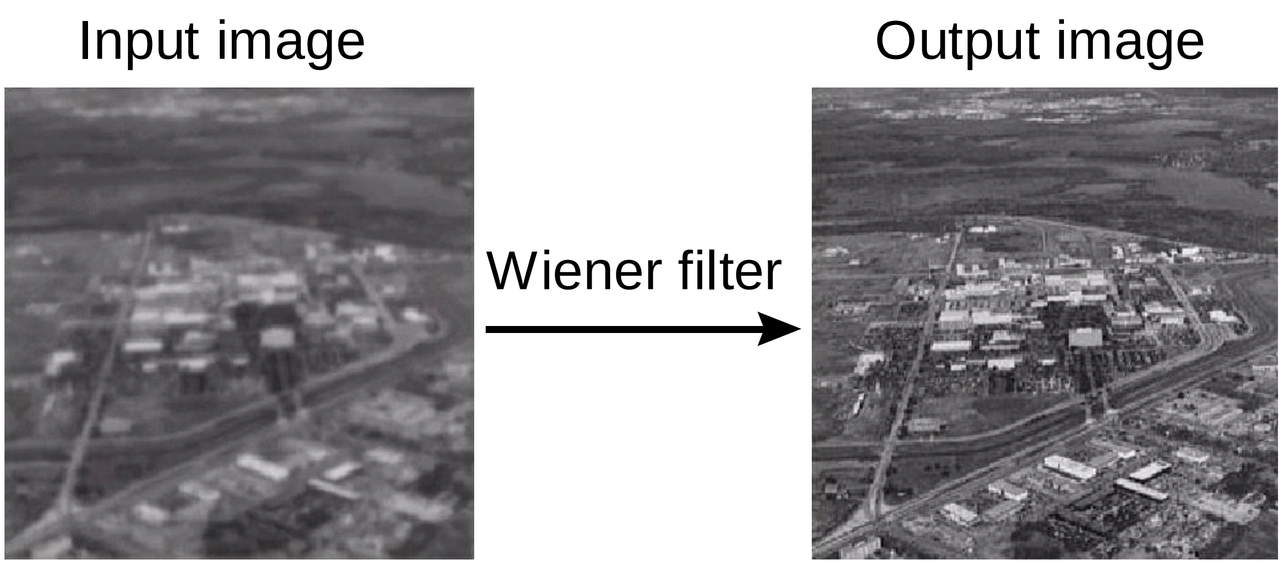
WHAT IS DIGITAL IMAGE PROCESSING?

1 Input image → Processing → Output image

- **Image enhancement and restoration**

Input → Filter → Output

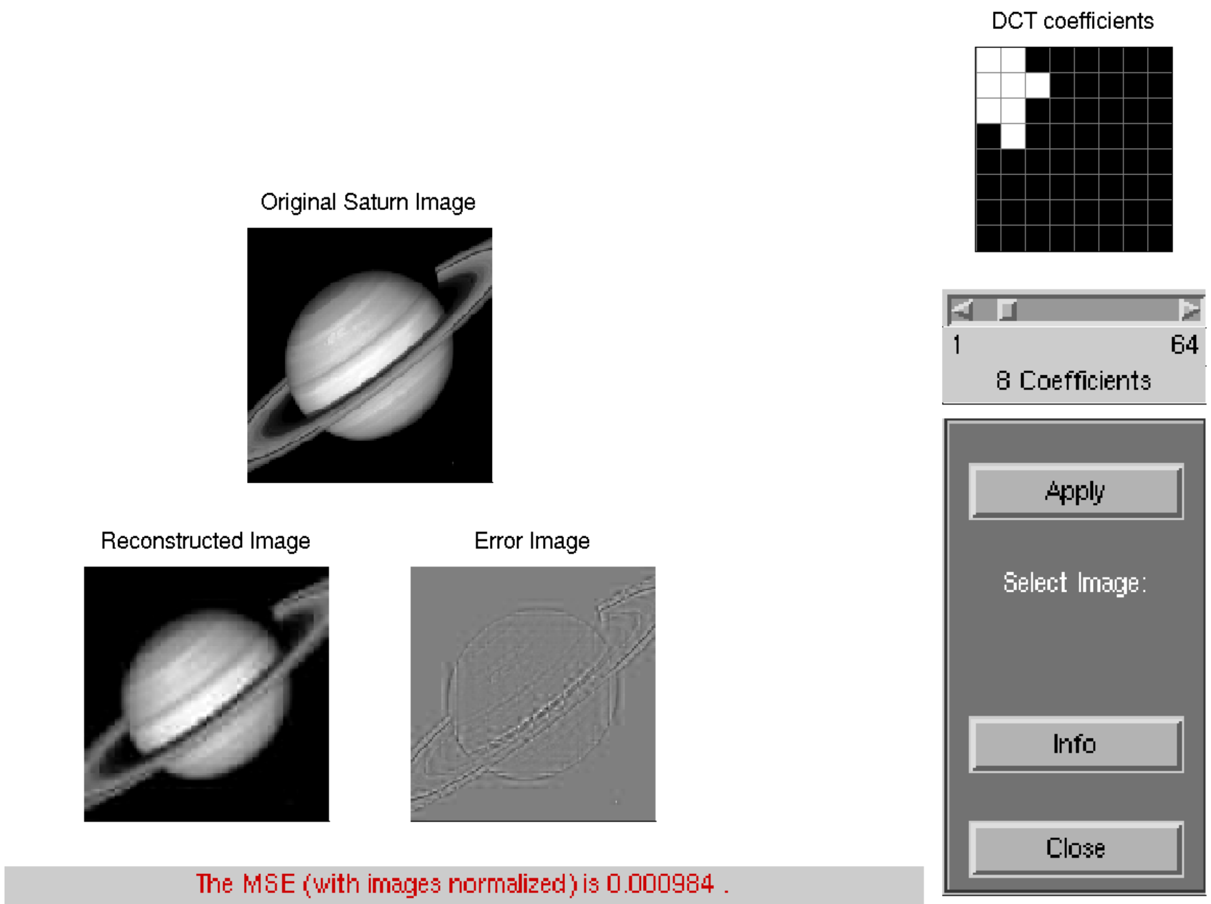
Input → Transform → Filter → Inverse transform → Output



- **Image compression**

Input $\longrightarrow$ **Coding** $\longrightarrow$ **Code** $\longrightarrow$ **Decoding** $\longrightarrow$ **Output**

JPEG Demo: 8 DCT coefficients retained



2 $\boxed{\text{Image}} \longrightarrow \boxed{\text{Processing}} \longrightarrow \boxed{\text{Features}} \longrightarrow \boxed{\text{Recognition}}$

- **Object recognition**

$\boxed{\text{Image}} \longrightarrow \boxed{\text{Segmentation}} \longrightarrow \boxed{\text{Object}}$

$\longrightarrow \boxed{\text{Representation and description}} \longrightarrow \boxed{\text{Features}} \longrightarrow \boxed{\text{Recognition}}$

Example: Signature verification

(a)

(b)

(c)

(d)

(the authentic signature is (a))

OVERVIEW OF COURSE

- **Fourier analysis**
- **Image enhancement**
  - **Spatial domain (Ch. 3)**
  - **Fourier domain (Ch. 4)**
- **Image restoration (Ch. 5)**
- **Colour IP (Ch. 6)**

Wavelets (Ch. 7)

- **Compression (Ch. 8)**
- **Morphological IP (Ch. 9)**
- **Pattern Recognition**
  - **Segmentation (Ch. 10)**
  - **Representation & description (Ch. 11)**

  Object recognition (Ch. 12)