Session 2

BASIC IMAGE PROCESSING

FUNCTIONS TO IMPLEMENT IN THIS SESSION

- **downsample.m** *
- **upsample.m** *
- **replicate.m** *
- **bilinear.m** *
- **quantizing.m**
- **psnr.m**
- **psnr_rgb.m**
- **psnr_ycbcr.m**

* Demanded functions for the coder-project

`downsample()`

1. Implement the function `downsample()` which reduces the input image by the factors `fh` (horizontally) and `fv` (vertically), creating a new image.

```
function im_sub = downsample(image, fh, fv)
```

Input:

- `image`: grey scale image of type `uint8` or `double`
- `fh`: horizontal subsampling factor (eliminating columns)
- `fv`: vertical subsampling factor (eliminating rows)

Output:

- `im_sub`: matrix containing subsampled image (`double`)

`downsample()`

Notes for implementation:

- As the input image could be of type `uint8` or `double` it is important to convert it first to `double`.
- The output image is `double`
- Use the "`:`" operator to access the matrix elements instead of a for-loop.

2. Write a **test program** which visualizes the image before and after subsampling.
3. Apply the test program to **Zoneplate_400_5.bmp** and **Barbara.bmp** reducing to **half** and **quarter size.**

`pre_filter()` Optional

Pre-filtering:

- To avoid the observed *aliasing* effects, we implement the function `pre_filter()` which should realize a low pass filter previously to the downscaling process to eliminate the frequencies above the critical Nyquist-frequency.

```
function im_filt = pre_filter(image,n,s)
```

Input:

- `image`: grey scale image of type `uint8` or `double`
- `n`: size of Gauss-filter H
- `s`: standard deviation of the Gauss-filter

Output:

- `im_filt`: matrix of prefiltered image (`double`)

`pre_filter()` Optional

Notes for implementation:

- The filtering is realized using `imfilter(I, H)`, where `I` is the image to filter and `H` the impulse response of the filter.
- To obtain a low-pass impulse response with good results, we use the Matlab function `fspecial('gaussian',n,s)`, where `n` is the filter order and `s` the standard deviation of a rotational symmetric gaussian filter.
- If `n` is a scalar, the result is a squared matrix

`pre_filter()`

Optional

Test program

- Try different low-pass impulse responses to obtain the optimum parameters.
- Modify the formerly created test program by inserting a call to `pre_filter()` previous to `downsample()`.
- Search for the optimum values of `n` and `s` for the case to subsample by factor 4 the images **Zoneplate_400_5.bmp** and **Barbara.bmp.** Program a loop with the following values:
  - `n = 10, s = 1.0, 1.5, 2.0`
- Visualize the results inside the loop.

`upsample()`

Implement the function `upsample()` as a step previous to interpolation:

```
function im_amp = upsample(image, fh, fv)
```

Input:

- `image`: grey scale image of type `uint8` or `double`
- `fh`: horizontal amplification factor
- `fv`: vertical amplification factor

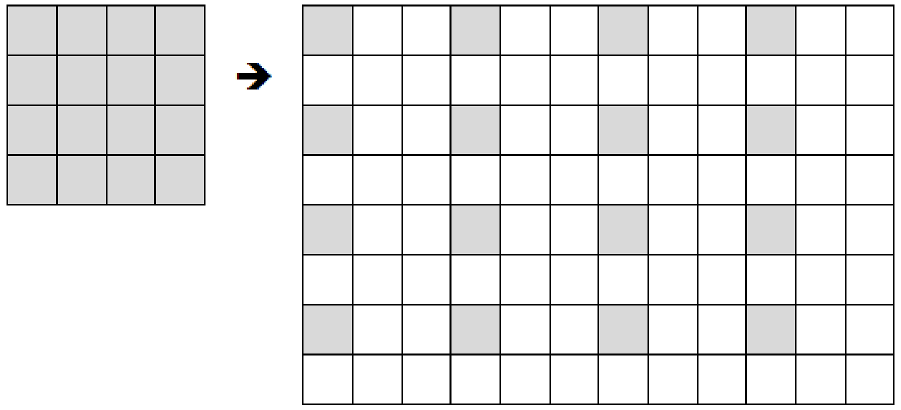
Output:

- `im_amp`: matrix containing resulting image (`double`)

`upsample()`

Notes for implementation:

- Create an empty matrix (zeros!) of size of magnified image
- Copy the known image pixels to the positions: *(`1+n*fv, 1+n*fh`)* of the zero-matrix, where `fv` and `fh` are the factors for vertical and horizontal amplification.
- Use the Matlab operator "`:`" specifying `startalue:increment:endvalue`



`replicate()`

1. Implement the function `replicate()` based on the function `upscale()` to fill the empty pixels. The known pixels should simply be copied (extrapolated) to the right and downwards.

```
function im_rep = replicate(image, fh, fv)
```

Input:

- `Image`: grey scale image of type `uint8` or `double`
- `fh`: horizontal amplification factor
- `fv`: vertical amplification factor

Output:

- `im_rep`: matrix containing resulting image (`double`)

2. Write a test program to amplify the image **Zoneplate_200_20.bmp** to double size. Compare the resulting image with **Zoneplate_400_5.bmp**.
3. Realice the same process with **Lena_128.bmp** and compare to **Lena_256.bmp.**

IMAGES: BILINEAL INTERPOLATION

- Linear interpolation in two steps:

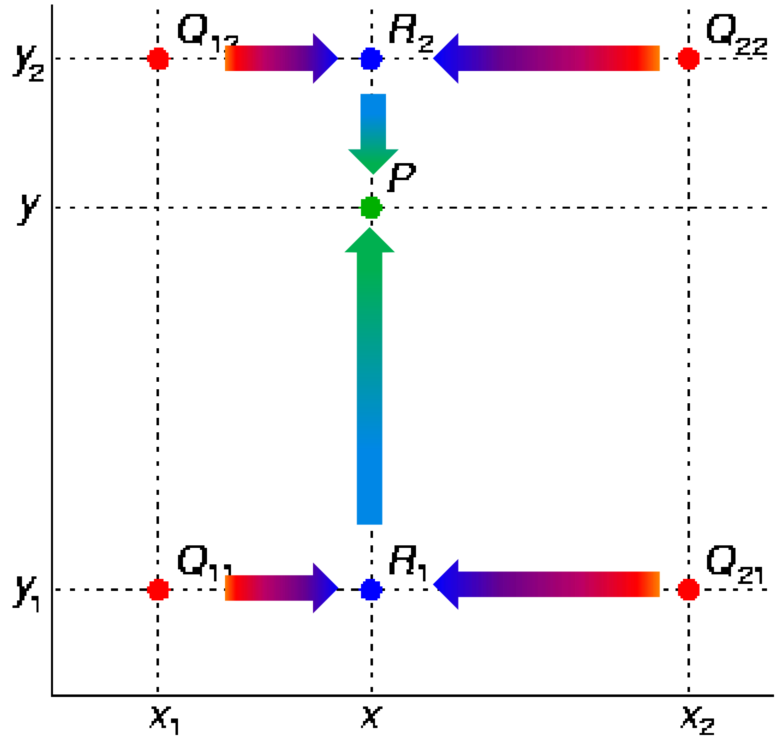
1. Horizontally:

$$f(R_1) = \frac{x_2 - x}{x_2 - x_1} f(Q_{11}) + \frac{x - x_1}{x_2 - x_1} f(Q_{21})$$

$$f(R_2) = \frac{x_2 - x}{x_2 - x_1} f(Q_{12}) + \frac{x - x_1}{x_2 - x_1} f(Q_{22})$$

2. Vertically:

$$f(P) = \frac{y_2 - y}{y_2 - y_1} f(R_1) + \frac{y - y_1}{y_2 - y_1} f(R_2)$$



`bilinear()`

1. Implement the function `bilinear()` based on the function `upscale()` to fill the empty pixels with interpolated values. The easiest way is to pass the image twice, horizontally and vertically to calculate the missing values (use the formerly indicated formula).
2. First pass horizontally: process only the rows which contain nonzero pixels.
3. Second pass vertically: process all columns

```
function im_bil = bilinear(image, fh, fv)
```

Input:

- `image`: grey scale image of type `uint8` or `double`
- `fh`: horizontal amplification factor
- `fv`: vertical amplification factor

Output:

- `im_bil`: matrix containing resulting image (`double`)

`bilinear ()`

Test program:

- Amplify the image **Zoneplate_200_20.bmp** to double size. Compare the resulting image with **Zoneplate_400_5.bmp** and with the one previously obtained with `replica().`
- Repeat the test with Lena

`quantizing()`

Implement the function `quantizing()` which works with quantizing steps of height e:

```
function im_quant = quantizing(image, e)
```

Input:

- `image`: grey scale image of type `uint8` or `double`
- `e`: quantizer step size

Output:

- `im_quant`: matrix containing quantized pixels (`double`)

Test program:

- Quantize the image **Head_XRay.bmp** with step size e from 7 bits reducing to 1 bit
- When do you observe “false borders”?

quantizing()

Notes on implementation:

- Concept: the quantizing process is based on
  - **division** by the quantizer step (quantizing) :

    `quant_val = round(orig_val / e)`

  - and **multiplication** as the inverse process:

    `rec_val = quant_val * e`

  where `rec_val` (recovered value) is an approximation of `orig_val`.
- Quantizer step `e = (max_val - min_val)/(num_levels-1).`
- The implemented function should quantize the image with the given step and afterwards recover it with the same step such that the range of output values would be the same (for visualization) and the quantization errors could be observed.

For testing

psnr()

Implement the function **psnr()**, which calculates the PSNR (*peak signal to noise ratio*) of a deteriorated image with respect to the original one.

```
function psnr = psnr(image_det, image_orig)
```

Input:

`image_orig`: grey scale image of type `uint8` or `double`

`image_det`: grey scale image of type `uint8` or `double`

Output:

`Psnr`

- Calculate first the MSE = (*mean square error*), then the PSNR based on it.

Test program:

- Calculate the PSNR values resulting from the quantizing of **Head_XRay.bmp**

For testing

psnr_rgb(), psnr_ycbcr

Later on, the PSNR values for colored images are needed. Implement the functions `psnr_rgb()`,and `psnr_ycbcr()`to obtain an average PSNR over the three components.

a) `psnr_rgb()`: average over RGB

b) `psnr_ycbcr()`: PSNR for luma + average over CbCr:

```
function [psnr_luma , psnr_croma] =
psnr_ycbcr(image_det, image_orig)
```

REPORT

Write a short report about your tests. **The report should contain**:

1. Your Name
2. An appropriate title
3. Describe in different sections:
    1. The downsampling process
    2. The upsampling process applying either replication or bilinear interpolation
    3. The quantizing process

    Present your results obtained with the mentioned images, present PSNR values where requested.

    Note: each figure should have a descriptive title below
4. Some conclusion about the findings

EVALUATION

Compress the **required functions** and the **report** in a zip-file and send per **e-mail** to the professor (martina.aux@gmail.com). Do not add your personal test programs and no images, only:

- **`downsample.m`** *
- **`upsample.m`** *
- **`replicate.m`** *
- **`bilinear.m`** *
- **`quantizing.m`**
- **`psnr.m`**
- **`psnr_rgb.m`**
- **`psnr_ycbcr.m`**

Until Monday 6th of May

The zip-file should be carry your named or ID.