University of Ljubljana, Faculty of Electrical Engineering
Robot Vision (RV)

April 5/6, 2016

Exercise 4: Basic Image Processing

Created by: Žiga Špiclin | http://lit.fe.uni-lj.si/RV | Homework deadline: April 12/13, 2016

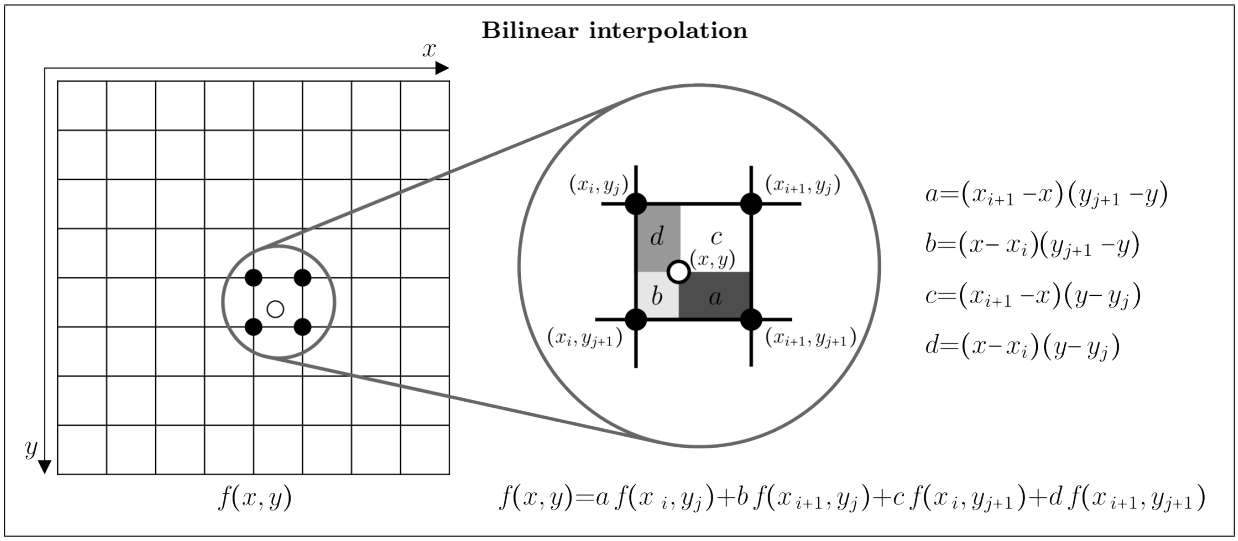
Instructions

During this exercise you will get familiar with basic image processing method like 2D filtering for the purpose of image smoothing and sharpening, and image interpolation and decimation.

Image filtering can be performed using **2D discrete convolution** between an image $g(x, y)$ of size $X \times Y$ and convolution kernel $k(u, v)$ of size $U \times V$:

$$f(x,y) = \sum_{u=-U/2}^{U/2} \sum_{v=-V/2}^{V/2} k(u,v)\, g(x-u,\, y-v)\,,$$

where $f(x, y)$ is the output image. According to the definition of convolution the coordinates (x, y) that extend out of the image $g(x, y)$ have a grayscale value of 0. In Python, the 2D discrete convolution can be easily implemented using for nested `for` loops. The outer two loops are used to address the elements (x, y) of image $f(x, y)$, while the inner two loops are used to address the elements (u, v) of the convolution kernel $k(u, v)$. The convolution kernel $k(u, v)$ can be defined as a 2D array, whereas the center of the kernel $k(0, 0) \leftrightarrow$ `K` is equal to Python indices `floor(K.shape/2)`, which needs to be taken into account where addressing the values in the 2D convolution kernel.



With **image interpolation** a grayscale value can be assigned to any coordinate location in the image. In this way, the sampling frequency and the image size can be increased. The order of interpolation methods is based on the number of neighboring pixel and their grayscale values, which are taken into account to compute the grayscale value at the new coordinate location. For instance:

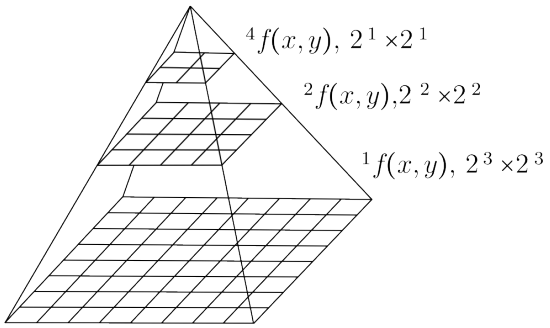
1. **zero order or nearest neighbor** interpolation – only the closest pixel and its grayscale value are taken into account,
2. **first order or (bi)linear** interpolation – four closest pixels and their grayscale values are taken into account,
3. **higher order**, e.g. (bi)cubic interpolation (second order), which takes into account 16 neighboring pixels and their grayscale values.

Computational complexity of 2D image interpolation is approximately proportional to the square of the order of interpolation. For instance, bicubic interpolation (second order) is approximately four times

more demanding (or slower) than the bilinear (first order) interpolation. The 2D image interpolation can be easily extended to three or even higher dimensional images.

With **image decimation** the maximal sampling frequency and the image size are decreased. According to the Nyquist sampling theorem the image needs to be filtered with a lowpass filter prior to decimation so as to reduce the maximal frequency of the image intensity signals such that their maximal frequency after filtering is equal or less than half the sampling frequency of the original image. Pyramid schemes are often used for decimation such that the sampling frequency is decreased consecutive steps by an integer value, typically by two.

Pyramid decimation scheme



Examples of lowpass filter kernels

$$\begin{array}{|c|c|c|}\hline \frac{1}{16} & \frac{1}{8} & \frac{1}{16} \\ \hline \frac{1}{8} & \frac{1}{4} & \frac{1}{8} \\ \hline \frac{1}{16} & \frac{1}{8} & \frac{1}{16} \\ \hline \end{array}$$

$$\begin{array}{|c|c|c|c|c|}\hline \frac{1}{400} & \frac{1}{80} & \frac{1}{50} & \frac{1}{80} & \frac{1}{400} \\ \hline \frac{1}{80} & \frac{1}{16} & \frac{1}{10} & \frac{1}{16} & \frac{1}{80} \\ \hline \frac{1}{50} & \frac{1}{10} & \frac{1}{4} & \frac{1}{10} & \frac{1}{50} \\ \hline \frac{1}{80} & \frac{1}{16} & \frac{1}{10} & \frac{1}{16} & \frac{1}{80} \\ \hline \frac{1}{400} & \frac{1}{80} & \frac{1}{50} & \frac{1}{80} & \frac{1}{400} \\ \hline \end{array}$$

During this exercise you will write functions for image filtering based on 2D discrete convolution for the purpose of smoothing and sharpening the grayscale and color images and functions for image interpolation and decimation for the purpose of magnification and minimization of grayscale and color images. Load the RGB color image `slika.jpg` into Spyder–Python environment and convert to grayscale image using $S = 0,299R + 0,587G + 0,114B$. You will use the color and grayscale image to verify the functions in the lab and homework assignments.

1. Write a function that computes 2D discrete convolution between the input grayscale image `iImage` and the convolution kernel `iKernel`:

```
def discreteConvolution2D( iImage, iKernel ):
        return oImage
```

   where `oImage` is the output grayscale image. Verify the function for 2D discrete convolution and its influence on the output grayscale image by using convolution kernels `iKernel = numpy.ones([k,k])/k**2` for different values of $k = 2n + 1, n = 1, 2, 3, \ldots$.

2. Write a function for zero order image interpolation, which sample the input 2D grayscale image `iImage`:

```
def interpolate0Image2D( iImage, iCoorX, iCoorY ):
        return oImage
```

   where `iCoorX` and `iCoorY` are sampling coordinates in the space of the input image `iImage`, in which we want to compute the grayscale value. The computed grayscale values represent the grayscale values of the output image `oImage`. In the zero order interpolation, an arbitrary point with coordinates (x, y) is assigned a grayscale value that is equal to the grayscale value of the nearest-neighbor point on the discrete sampling grid of the input image. To seek for the closest point in the discrete sampling grid you can use the Python function `numpy.round()`, while the discrete sampling grid of the input image can be obtained by using function `numpy.meshgrid(..., indexing='xy')`. Verify the function by interpolating a subregion between corners $(260, 210)$ and $(360, 280)$ of the grayscale image such that the subregion is interpolated with a discrete grid three times denser than the grid of the input grayscale image.

3. Write a function for first order image interpolation, which sample the input 2D grayscale image `iImage`:

```
def interpolate1Image2D( iImage, iCoorX, iCoorY ):
        return oImage
```

where `iCoorX` and `iCoorY` are sampling coordinates in the space of the input image `iImage`, in which we want to compute the grayscale value. The computed grayscale values represent the grayscale values of the output image `oImage`. In the first order interpolation, an arbitrary point with coordinates (x, y) is assigned a grayscale value that is equal to a weighted sum of the grayscale value of the four nearest-neighbor points on the discrete sampling grid of the input image (*see description of bilinear interpolation*). To seek for upper left nearest-neighbor point in the discrete sampling grid of the input image you can use the function `numpy.floor()`. Verify the function by interpolating a subregion between corners $(260, 210)$ and $(360, 280)$ of the grayscale image such that the subregion is interpolated with a discrete grid three times denser than the grid of the input grayscale image.

4. Write a function for pyramid decimation of a grayscale image `iImage`:

```
def decimateImage2D( iImage, iLevel ):
        return oImage
```

   where `iLevel` is the number of consecutive decimations with a factor of two. Apply a lowpass filter of size 3×3 (*see description of pyramid decimation*) prior to each image decimation. The function should return a decimated grayscale image `oImage`. Verify the function on the grayscale image for values of `iLevel = 1, 2, 3`.

Homework Assignments

Homework report in the form of a Python script entitled `NameSurname_Exercise4.py` should execute the requested computations and function calls and display requested figures and/or graphs. It is your responsibility to load library packages and provide supporting scripts such that the script is fully functional and that your results are reproducible. The code should execute in a block-wise manner (e.g. `#%% Assignment 1`), one block per each assignment, while the answers to questions should be written in the corresponding block in the form of a comment (e.g. `# Answer: ...`).

1. In function for 2D discrete convolution we have assumed that the grayscale value of points lying outside the image are equal to zero. Because of this assumption, the image obtained from the convolution function exhibits edge artifacts in the form of decreasing grayscale values when closer to the image edges. This arfifacts is more apparent when using larger kernels for convolution. A way to mitigate this artifact is to enlarge the input image by half the kernel size $(U/2, V/2)$ in all directions, before computing the convolution, and set the values in the enlarged part to nearest-neighbor grayscale value in the image.

   Modify the function `discreteConvolution2D()` by introducing a new input parameter `iPadImage` set to either `'zeros' or \sverb`'nearest'+. Let parameter `iPadImage` define the way the input 2D image is modified prior to computing the 2D discrete convolution, for instance, when `iPadImage = 'zeros'` assume that the grayscale values lying outside the image are zero, while for `iPadImage = 'nearest'` they are the nearest neighbor values in the original input image. Verify the modified function on the grayscale image using kernel `iKernel = numpy.ones([11,11])/121`.

2. Use the function `convolve()` in the Python library package `scipy.ndimage` to perform a 2D discrete convolution similarly to the previous assignment. Compare the obtained convolved images and the execution times.

3. Modify the function for 2D discrete convolution such that it enables the convolution of a color image with an arbitrary kernel. Verify the function on the given color image using the kernel `iKernel = numpy.ones( [11,11])/121` and display the obtained image.

4. Write a function that computes a convolution kernel if the form of 2D symmetric Gaussian function:

```
def discreteGaussian2D( iSigma ):
        return oKernel
```

   where `iSigma` is the standard deviation σ in the 2D symmetric Gaussian kernel, which is defined as:

$$k(u, v) = (2\pi)^{-1}\sigma^{-2}\exp^{-(u^2+v^2)/2\sigma^2} .$$

The size of convolution kernel $U \times V$ should be set according to the value of `iSigma` through expression $U, V = 2 \cdot \lceil 3 * \sigma \rceil + 1$, whereas the value $k(0, 0)$ lies in the central element of the output 2D array `oKernel`. Ensure that the sum of all elements in the 2D array `oKernel` is equal to 1. Compute the convolution kernels for $\sigma = 0.1$ and 3 and use the kernels to perfom a 2D discrete convolution on the given color image. Display the obtained images and discuss the influence of the value of σ on these images.

5. Image sharpening is similar to spatial derivation of grayscale values. For the purpose of image sharpening, a second order derivative is often applied, because it is easily computed through convolution of the image with the Laplace kernel. Then, the input image $g(x, y)$ can be sharpened by subtracting a weighted image of the second order derivative $\Delta g(x, y)$ as follows:

$$f(x, y) = g(x, y) - c\,\Delta g(x, y)\,,$$

Laplace kernel

0	1	0
1	−4	1
0	1	0

where c is a scalar parameter, which determines the degree of image sharpening. Ensure that after the subtraction the obtained grayscale image has unsigned 8-bit values in range [0, 255]. Verify the function by using $c = 2$ and display the resulting image. Furthermore, verify the function with different values of parameter c and discuss its influence on the resulting grayscale image.

6. Modify the functions for zero and first order image interpolation such that they can be applied to color images. Verify the modified functions by interpolating the color image using a sampling grid that is three times denser than the original. Display the resulting interpolated color images.

7. Image interpolation can be used to zoom-in or magnify regions of an image. Perform a five-fold (5×) magnification of a rectangular subregion of the color image between (x, y) corners $(100, 50)$ and $(250, 200)$ by using the function for first order image interpolation. Display the original and five-fold (5×) magnified subregion of the color image.

8. Modify the function for pyramid decimation such that it enables the decimation of color images. Verify the modified function on the given color image.

Gaussian function for various values of parameter σ

$\sigma = 3.0$ $\sigma = 6.0$ $\sigma = 9.0$

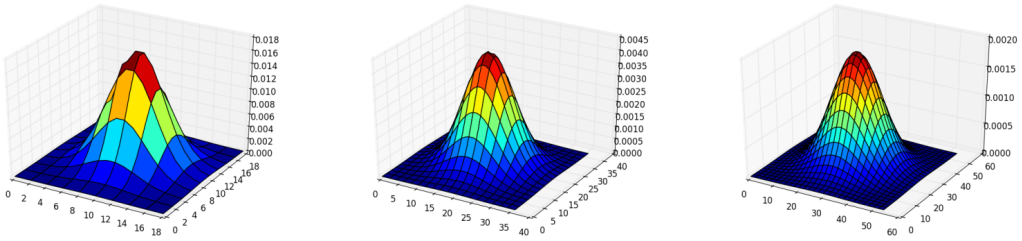


Image sharpening using the Laplace kernel

$c = 0.0$ $c = 1.0$ $c = 2.0$

Interpolation and decimation of color images

5×zoom-in→ zero order first order zoom-out→ 2× 4×