Heidi Gebauer (geba), Markus Roos (roor)
School of Engineering T

Zürcher Hochschule für Angewandte Wissenschaften

Image Processing via Filtering: Exercise Set

Please hand in the solutions of the below tasks until the next class.

Remark 1: To load the images depicted on this exercise sheet, use `images = getImages()` defined in `load_images`. The images are then accessible form a Python dictionary, i.e., images['Monroe'] corresponds to the picture 'monroe.png'.

Remark 2: Throughout this exercise set, we use the convention that pixel intensities are *doubles*.

Remark 3: In `load_images` some images are converted to grayscale images via `rgb2gray`, which is given in the same module.

Task 1 The goal is to implement a function `[R] = myfilter(Im, F)` that takes as input

- a grayscale image `Im`, and
- a 3×3 matrix `F`

and applies the filter `F` to `Im`. (If the filter operation refers to pixels outside the bounds of `Im` then just assume that these pixels have intensity 0.)

Hint: First write an auxiliary function `get_intensity(Im,i,j)` that returns $\mathrm{Im}(i, j)$, if the point (i, j) lies inside of the boundaries and 0, otherwise.

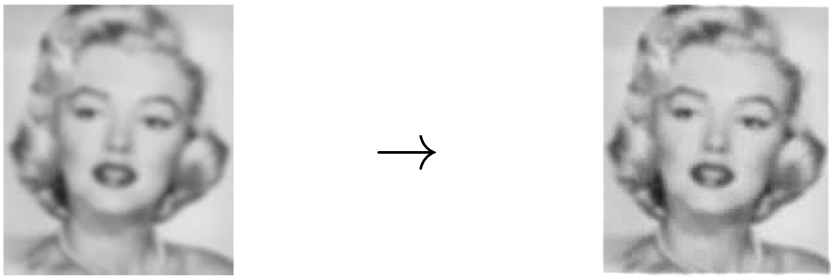
Remark: Do not use built-in commands like `imfilter` for this task.

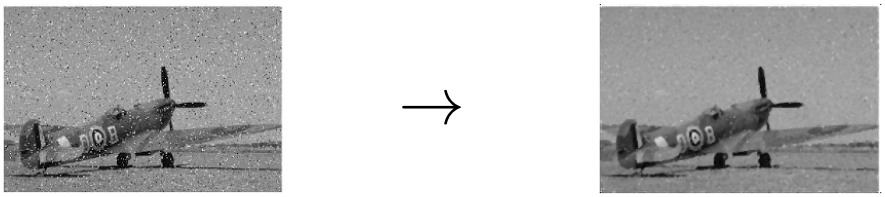
Test Case: If you run your program with $F = \frac{1}{9} \cdot \begin{pmatrix} 1 & 1 & 1 \\ 1 & 1 & 1 \\ 1 & 1 & 1 \end{pmatrix}$ for the grayscale sunflower image you should get the following output:

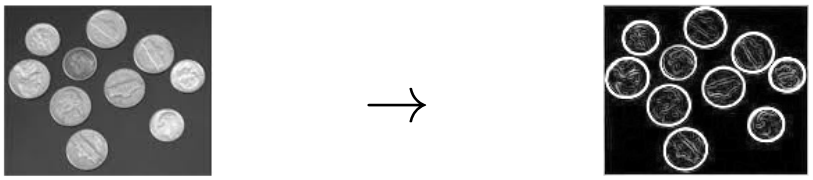
For the following tasks you can use the functions `imfilter` and `medfilt2`. (A description is given on `https://stackoverflow.com/questions/22142369/the-equivalent-function-of-matlab-i` and `https://docs.scipy.org/doc/scipy/reference/generated/scipy.signal.medfilt2d.html`

Task 2 The illustration below depicts three image transformations. The goal is to reproduce the corresponding target image on the right (approximately) using appropriate filters. Use `imshowRgb` and `imshowGray` defined in `load_images`.

Hint: The transformations below have the property that each pixel in the target image depends only on the corresponding source-pixel and its 8 neighbors (vertically, horizontally and diagonally).

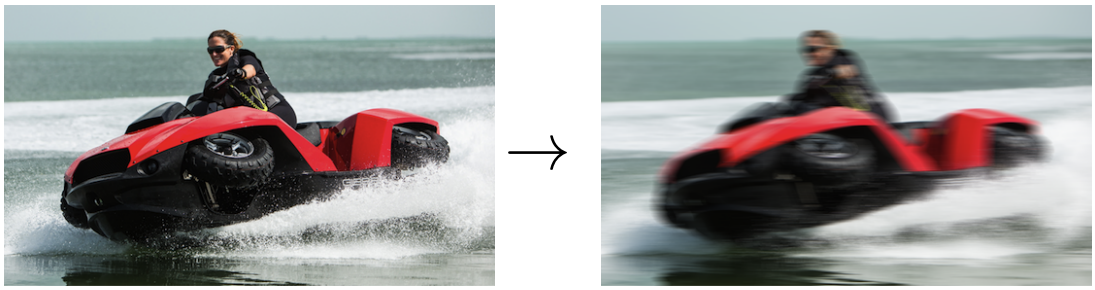
(a) 

(b) 

(c) 

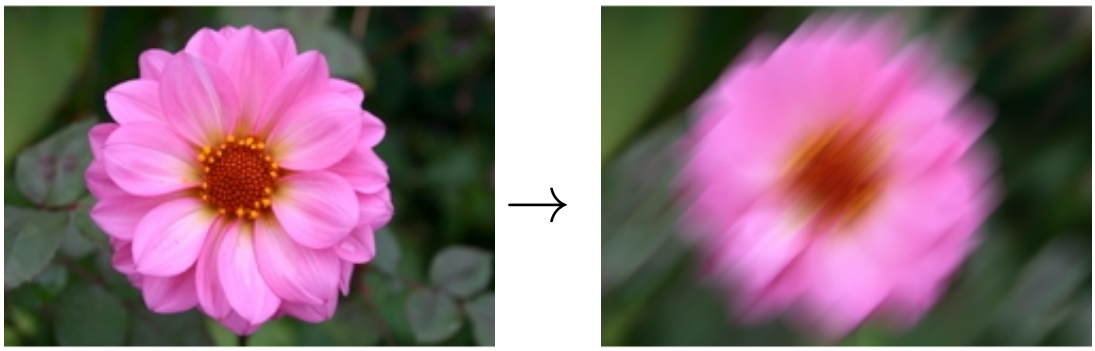
Task 3 The goal of this task is to add motion blur (i.e. blur in one direction only).

(a) In the transformation depicted below, the right image is blurred horizontally. Sketch a suitable filter-matrix and implement the corresponding operation using `imfilter`. **Remark:** There are several plausible solutions.



Source: Prof. Noah Snavely, Cornell University

(b) In the transformation depicted below, the right image is blurred diagonally.
Sketch a suitable filter-matrix and implement the corresponding operation using `imfilter`.
Remark: There are several plausible solutions.



Source: Jerry Huxtable, `http://www.jhlabs.com/ip/blurring.html`

Task 4 In lecture *Basic Transformations* we discussed rotations and translations etc. of objects in space. Now we apply these transformations to pictures. The final goal of this task is function that takes a picture (matrix) and an angle (in radians) as input and provides the picture rotated by the angle as output.

(a) Develop a function to provide pixel_out_ij = f(picture_in, i_out, j_out, angle), which is rotated by the angle about the center of the picture.

Remark: We specify the *output* pixel position to avoid "empty" pixels in the output, which could happen if each input position would be transformed instead (Why?).

During this subtask, several non-trivial issues must be addressed:

- What to do, when the source pixel is outside the input picture?
- In general the exact position of the source pixel will not consist of integer values, i.e., a position *between* the given pixel. Hence, one needs some sort of interpolation to find the corresponding greyscale value.
- For the latter task, you might consider "bi-linear" interpolation, see, e.g., `https://en.wikipedia.org/wiki/Bilinear_interpolation`

Find plausible and simple solutions!

(b) Implement a PYTHON function taking an angle and picture as input that provides the picture rotated by the angle around the center as output. Test the performance of your algorithm with the input picture "testshapes" and a choice of the other examples.

Show the source and the rotated pictures and discuss the observed phenomena of your solution.

(c) (optional) Test filters applied to the picture *after* rotation to improve the picture's quality!