# 1 Geometrische Operatoren

Edit the Jupyter Notebook `Geometric_Transformation.ipynb`. In this exercise, we will use and understand geometric operations to correct geometric distortions on images.

## 1.1 Bilineare Interpolation

In order to apply geometric operations in a meaningful way, we need the possibility to interpolate. The function `interpolate_bilinear` accepts a touple of four rgb colors ($P_{0,0}$, $P_{0,1}$, $P_{1,0}$, and $P_{1,1}$) and a set of coordinates $x, y$. It assumes that the points are located at 0/0, 0/1, 1/0, and 1/1 respectively and computes a bilinear interpolated value between them.

a) Argue, how this function could be tested.

b) Setup Unittests for the function.

b) Implement the function.

I needed the following time to complete the task:

## 1.2 Geometric Transformation on Images

Using the bilinear interpolation, we can now transform images. The function `transform_image` accepts an image as numpy array and a geometric transformation $f$, which is also a function. Remember, that in Python, functions are just values. In order to compute the new image, which has the same resolution as the original image, `transform_image` transforms each pixel in the output image to the input image. It then performs an interpolated lookup.

a) Describe, in your own word, how you are going to deal with pixels outside the input image (you can choose a method).

b) Describe, how the function could be tested.

b) Setup Unittests for the function.

b) Implement `transform_image` function.

I needed the following time to complete the task:

## 1.3 Affine Transformations

Now, we will implement a number of geometric transformations. Each of the transformations will be realized by a function that accepts parameters depending on the transformation, and returns a $3 \times 3$ matrix to realize the operation.

1. First, we need a function that applies an affine transformation (multiplication with matrix) to pixel coordinates. This generic function can conveniently be realized using the partial function from functools. The mechanism is demonstrated in the notebook.
2. Translation accepts two parameters $x$ and $y$ and realizes a translation operation.
3. Scale accepts two parameters $sx$ and $sy$ and realizes a non-uniform scaling operation.
4. Rotation accepts a single parameter *alpha* (in radians) and realizes a rotation operation around the coordinate origin.

For each of the transformations:

a) Describe, how the function could be tested. Consider using the inverse operation.

b) Setup Unittests for the function.

c) Implement the transformation and its inverse.

Image translated by -50/+22.5 Pixels:

Image mirrored vertically:

Image translated by 45 degree around the origin:

d) In order to implement rotations around arbitrary points, we need to concatenate transformations. Implement a rotation functions that accepts three parameters: *origin_x*, *origin_y* and *alpha*. It rotates by *alpha* (in radians) and implements a rotations around the specified origin. Remember how to implement arbitrary rotations: translate the rotation origin to 0/0, rotate, translate back.

I needed the following time to complete the task:

## 1.4 Perspective Correction

Now for something a little more fancy. The image `source_code/perspective.png` was fotographed from a non-optimal angle. We want to transform it in such a way, that the corners of the poster become the new corners of the image (0/0), (0/*width*) and so on. Finding good transformation parameters by trial and error might be time-consuming, though. In order to solve this problem, we consider the problem as an optimization problem: we use the *argmin* function to find the suitable transformation matrix, then project the image as required.

The function `find_transformation` accepts to sets of four points each ($S = [S_{00}, S_{10}, S_{01}, S_{11}]$ and $D = [D_{00}, D_{10}, D_{01}, D_{11}]$), and returns a $3 \times 3$ transformation matrix $M$ that minimizes the $L_2$ distance between $S$ and $M \cdot D$.

a) Describe, how the function could be tested. Consider using the inverse operation.

b) Setup Unittests for the function.

c) Implement the transformation and its inverse.