Computer Vision Laboratory Report n.1

IMAGE WARPING AND BILINEAR INTERPOLATION

March 19, 2019

Claudio Curti ID: 4216203
Nicola De Carli ID: 4198668
Alberto Ghiotto ID: 4225586
University of Genoa

Contents

1 Introduction **2**
1.1 Defining the objective 2
1.2 Implemented transformation 2

2 Theoretical Background **4**
2.1 Backward vs Forward warping 4
2.2 Bilinear interpolation 4

3 Implementation and Results **6**
3.1 Translation 8
3.2 Rotation 9
3.3 Shear 10
3.4 Rototranslation 12
3.5 Colored rotation 13

Chapter 1

Introduction

1.1 Defining the objective

This project aims to implement in a MATLAB environment the first concepts of image manipulation, in order to get a broad understanding of the possible interpretations of the perceived visual world.
These manipulations are achieved exploiting the procedures of backward warping and bilinear interpolation.

1.2 Implemented transformation

By applying the above mentioned procedures it is possible to implement a variety of transformations on the provided images, such as:

- Rotation
- Translation
- Horizontal shear
- Vertical shear

All these manipulations are based upon the coordinate transformations seen in figure 1.1 and figure 1.2 in the next page, which emphasize the **forward** transformation matrices with the relative coordinates. In this particular case, in order to implement backward warping, the transformation matrices had to be inverted.

The actual implementation in MATLAB of these coordinate transformations will be covered in the next chapter.

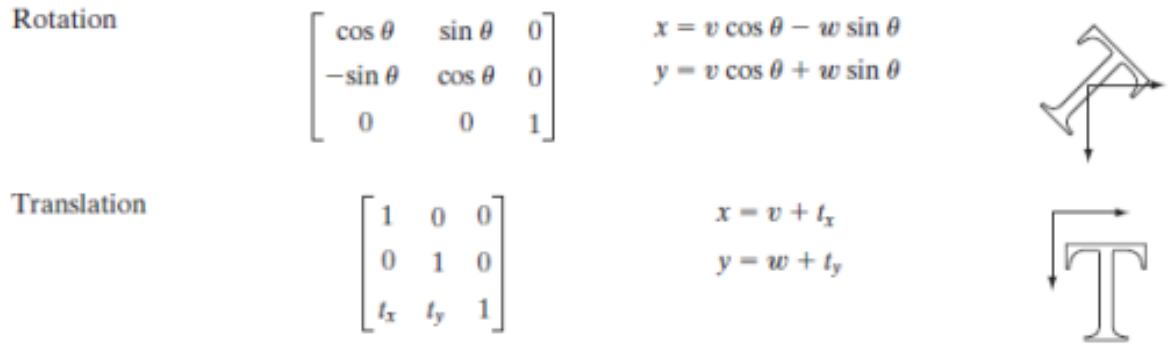


Figure 1.1: Rotation and Translation.

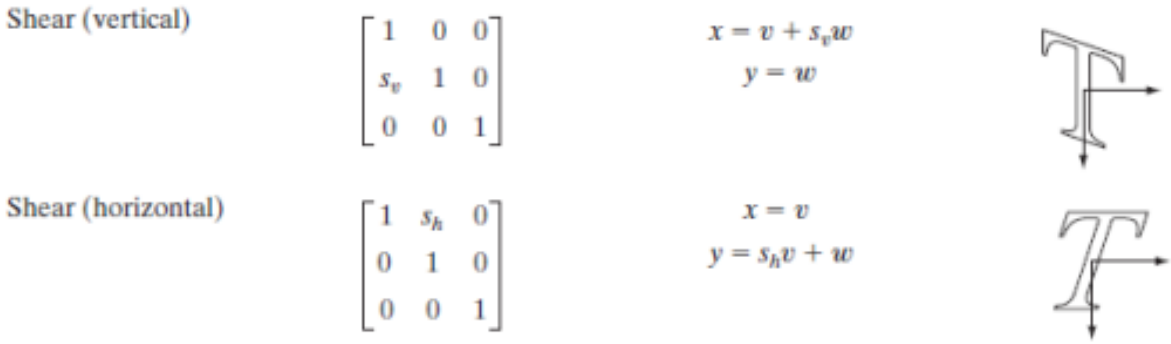


Figure 1.2: Horizontal and vertical shear.

All these transformations can be applied on any given RGB image, however, for the sake of simplicity, in this laboratory they were applied on RGB images which were beforehand transformed into grayscale ones. Thanks to this adjustment it will be possible to focus entirely on the image manipulation rather than worrying about all the 3 RGB channels.
However, to prove the feasibility of the RGB image manipulation, the rotation has been implemented in both grayscale and RGB modality.

Chapter 2

Theoretical Background

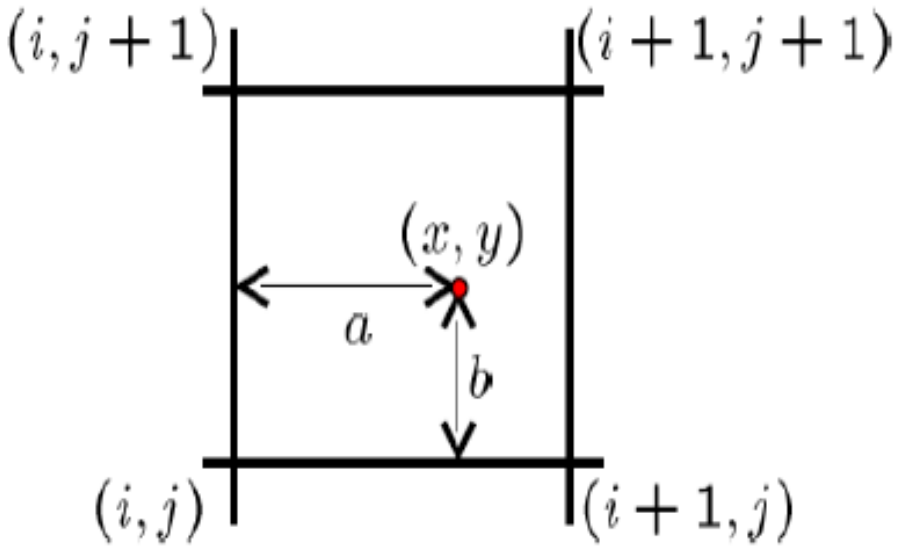
2.1 Backward vs Forward warping

To achieve all the mentioned geometric transformations, which basically correspond to changes in the domain of the function (the image itself), it is necessary to associate each pixel of the source image to those of the destination image (forward warping) or viceversa (backward warping). Since it is much more effective, in this case the backward warping was required by the instructions. In fact, the forward warping shows some problems due to the fact that in this way two or more pixel of the source image could correspond to the same pixel in the destination image after the transformation and it is likely that holes will appear in the transformed image.
The transformation matrices and equations showed in the previous section are relative to the forward warping, in fact in the backward warping, in order to achieve the correspondence between the pixels of the destination image with respect to those of the source image, it is necessary to invert these matrices and therefore the corresponding equations.

$(x, y) = T(v, w)$ (Forward warping)
$(v, w) = T^{-1}(x, y)$ (Backward warping)

2.2 Bilinear interpolation

It is likely that a pixel of the destination image could corresponds to a pixel in the source image that has no integer coordinates, in such case the bilinear interpolation can be used to find the value of that pixel by computing a weighted average among the values of the pixels surrounding the point. The weights are inversely proportional to the distance of the non-integer destination point(the red dot in figure 2.1) with respect to the neighbour pixels, in MATLAB this is done with the function "griddata". This procedure is shown in the following page in figure 2.1.



$$
\begin{aligned}
f(x,y) = \quad & (1-a)(1-b) \quad f[i,j] \\
& +a(1-b) \quad f[i+1,j] \\
& +ab \quad f[i+1,j+1] \\
& +(1-a)b \quad f[i,j+1]
\end{aligned}
$$

Figure 2.1: Interpolation procedure

Chapter 3

Implementation and Results

The code is composed by a main script, in which the image is loaded and converted in gray scale, and different functions which are deputed to the image's manipulation which are called by the script. The functions that have been developed are the following:

- **translate**: which executes a translation of the image given the x-y translation coefficients in pixel.
- **rotate**: which rotates the image of a given angle around its center.
- **horizShear** and **verShear**: which shear the image (horizontally or vertically).
- **rotoTransl**: which applies a translation and a rotation in sequence.
- **coloredRotation**: which highlights the possibility of doing these manipulations even with colored images.

Within these methods some predefined MATLAB functions are repeatedly used and therefore their behavior is hereby outlined.

- The **meshgrid** function returns 2-D grid coordinates based on the coordinates contained in the input vectors. This grid is used to interpolate the image after having computed the desired geometric coordinate transformations.
- The **griddata** function is needed to interpolate scattered data and to produce gridded data. It takes as inputs the coordinates of the data points, their values and the query points of interpolation. The linear interpolation is used by default.

Starting from one of the provided images, which can be seen in figure 3.1, the first step is to render it in gray scale. The converted image used for all the following manipulation can be seen in figure 3.2.

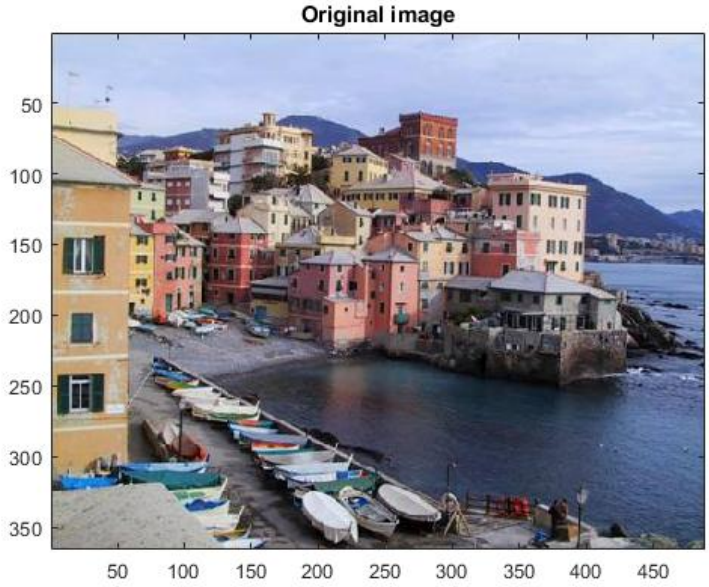


Figure 3.1: Original image acquired by MATLAB

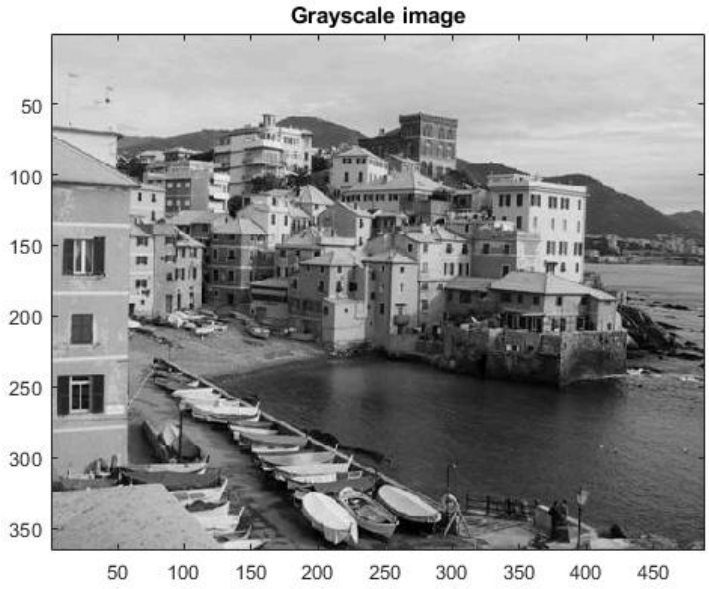


Figure 3.2: Grayscaled image

3.1 Translation

This function takes in input the desired translation along x-axis, the translation along y-axis, the grayscale image and a boolean variable which must be set to true if it is desired to have the figure showed.
After having used the "meshgrid" function, the obtained grid is translated of the desired amount using the opposite sign for the relative translation value, this is due to the backward warping, which requires the inverse of the transformation matrix. The image is eventually interpolated with the "griddata" function.

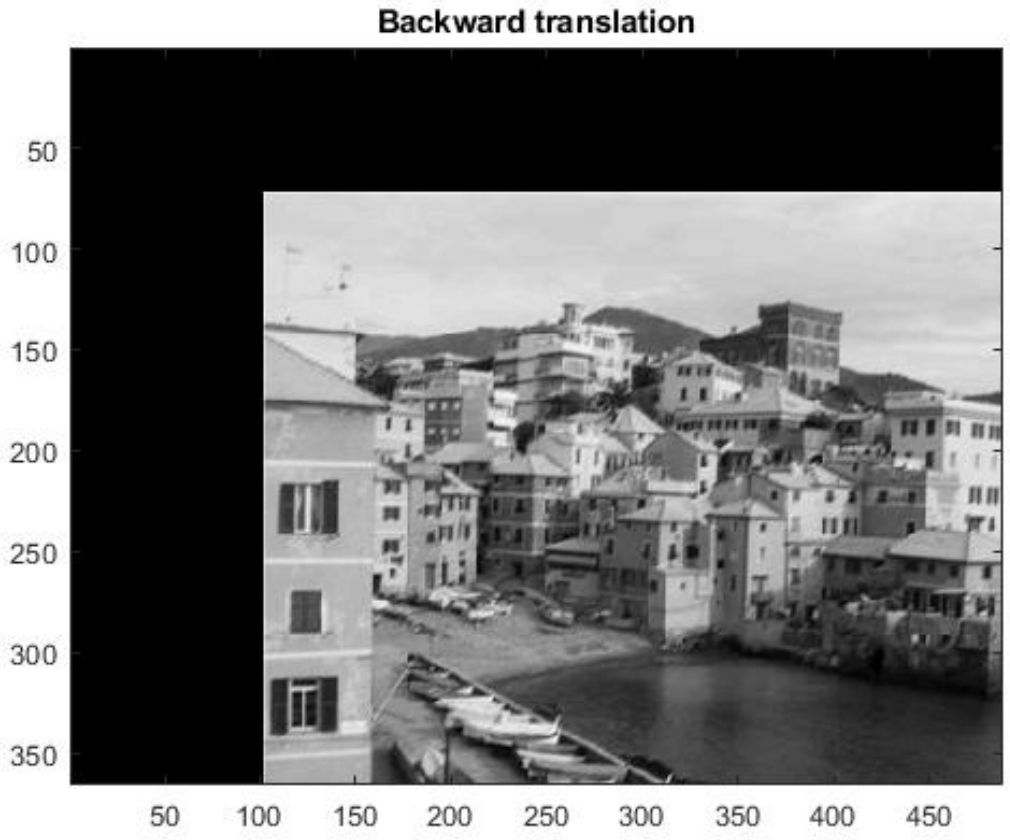


Figure 3.3: Backward Translation

3.2 Rotation

This function takes in input the desired angle for the rotation, the grayscale image and a boolean variable which must be set to true if it is desired to have the figure showed. After having used the "meshgrid" function, the image is rotated taking into account the center of the image as (0,0) coordinates, by subtracting half of the columns (midX) from X and half of the rows from Y (midY), and it is rotated using the inverse of the rotation matrix which corresponds to its transposed since the matrix is orthonormal. Then, the image is translated of midX and midY to bring again the (0,0) coordinate at the top left of the image. The image is eventually interpolated with the "griddata" function.

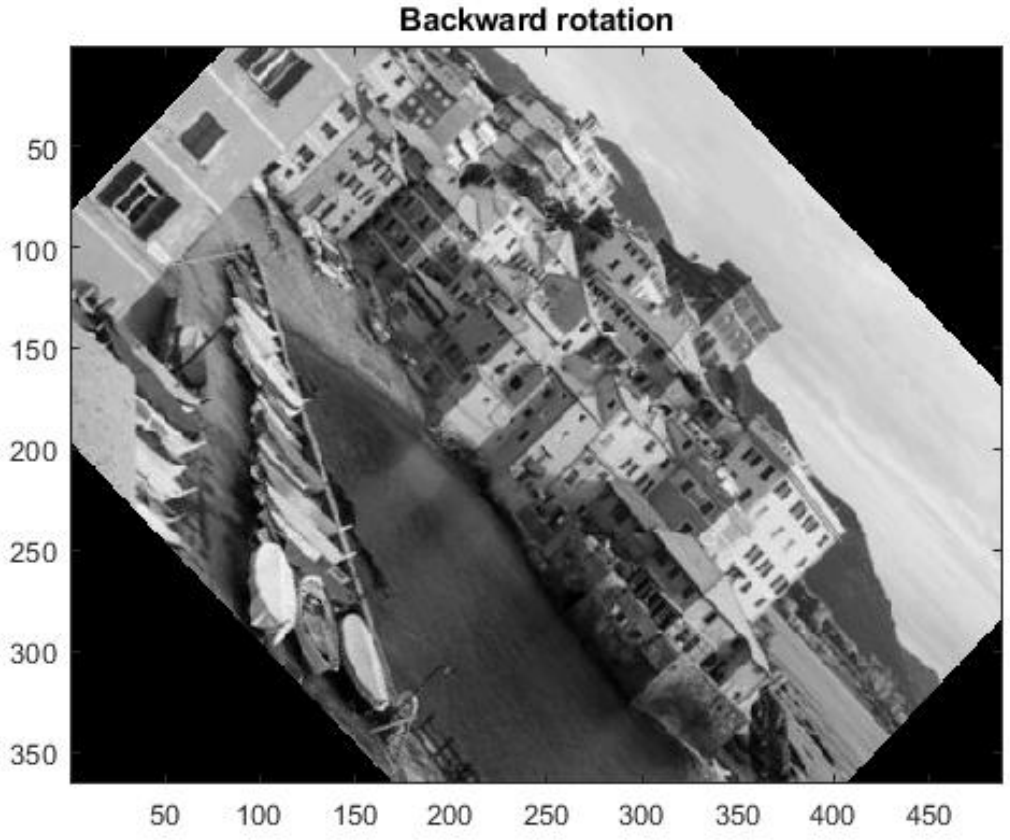


Figure 3.4: Backward 45 degrees Rotation

In figure 3.5, in the next page, it can be seen the image rotated by a different angle.

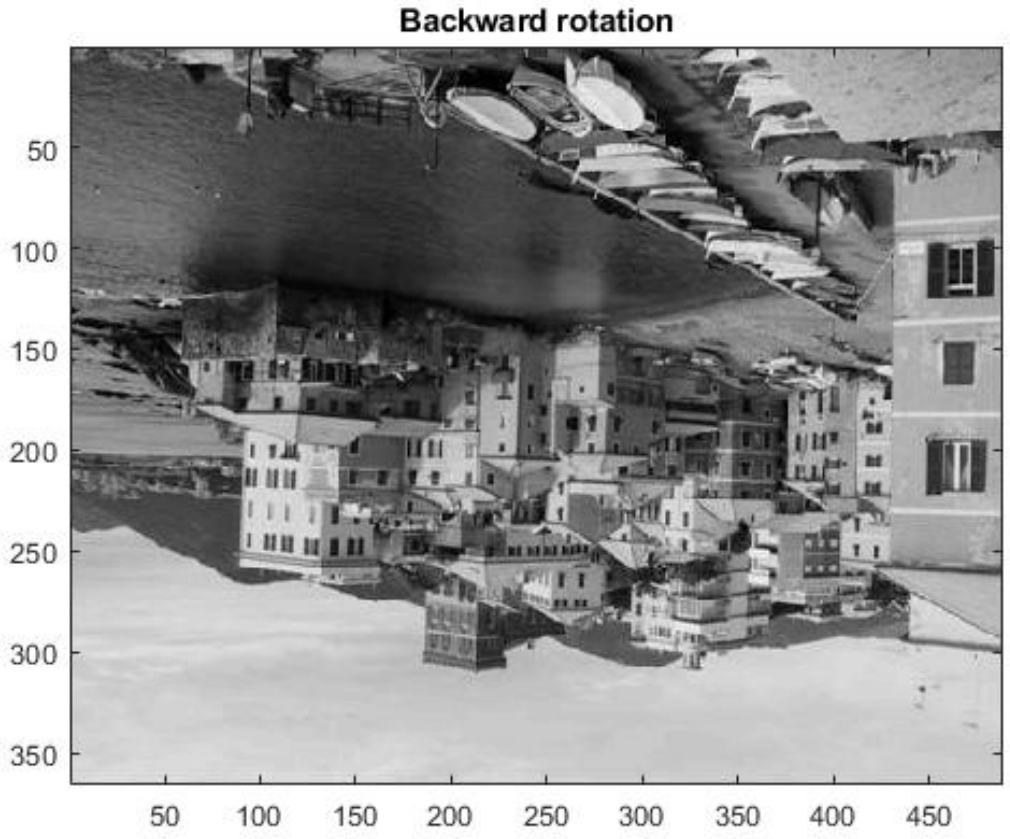


Figure 3.5: Backward 180 degrees Rotation

3.3 Shear

This function takes in input the desired shearing coefficient, the grayscale image and a boolean variable which must be set to true if it is desired to have the figure showed.
After having used the "meshgrid" function, the obtained grid is transformed by subtracting from one of the coordinates, X or Y (whether if it is the horizontal or vertical shear), the value of the other coordinate multiplied by the coefficient given in input. Then, the image is recentered by translating it along the the distorted axis of a value which equally depends from both the dimension of the image and the shearing coefficient. The image is eventually interpolated with the "griddata" function.

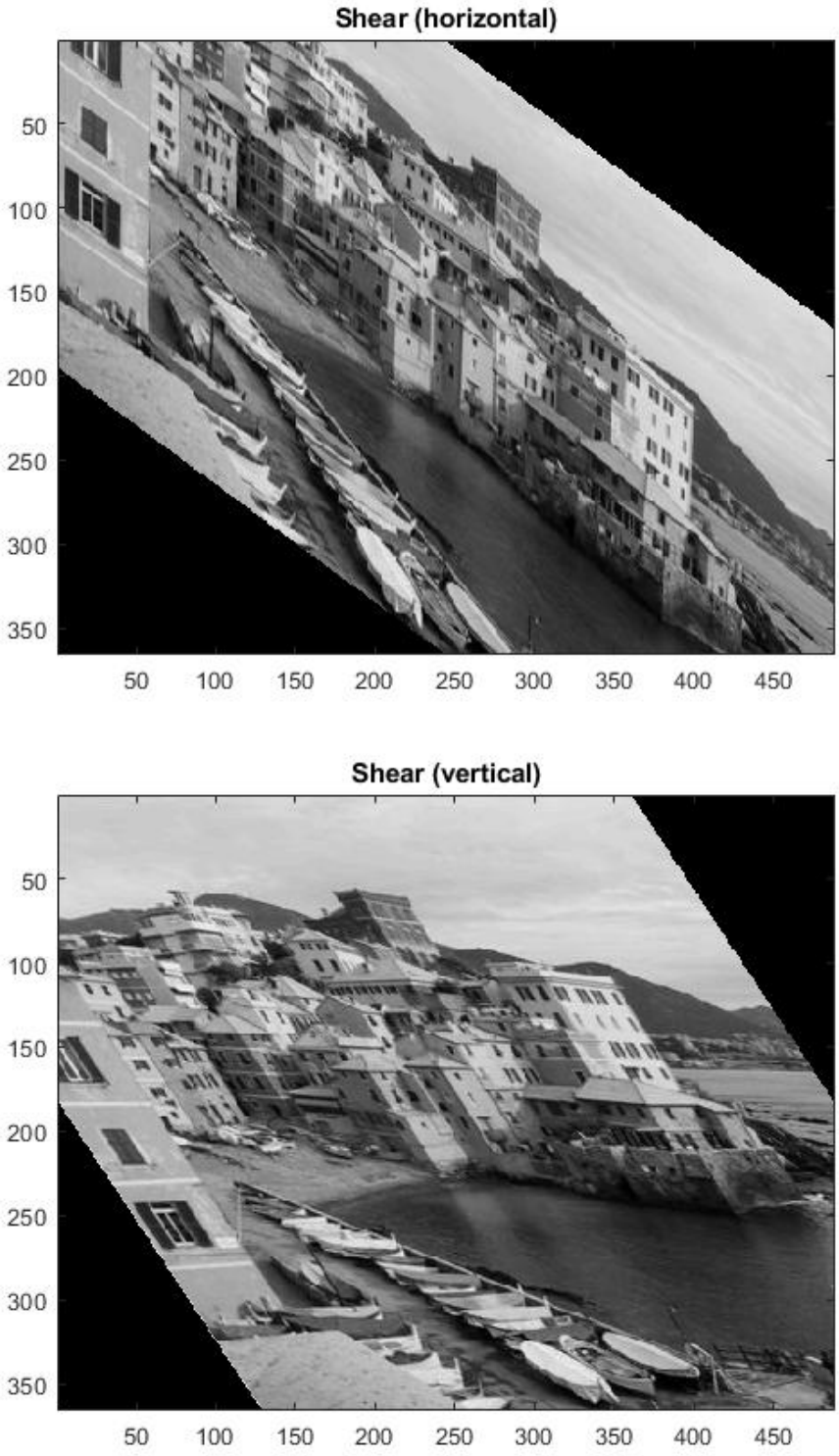


Figure 3.6: Horizontal Shear

3.4 Rototranslation

This function simply combines a rotation and a translation to generate a rototranslation of the image. It was not possible to directly call the two functions "translate" and "rotate" one after the other since the resultant image would not have been rectangular. Therefore the implementation is very similar to the "rotate" function with the addition of the offsets "tx" and "ty" in the computation of the query points of interpolation that correspond to the translation which occurs along with the rotation.

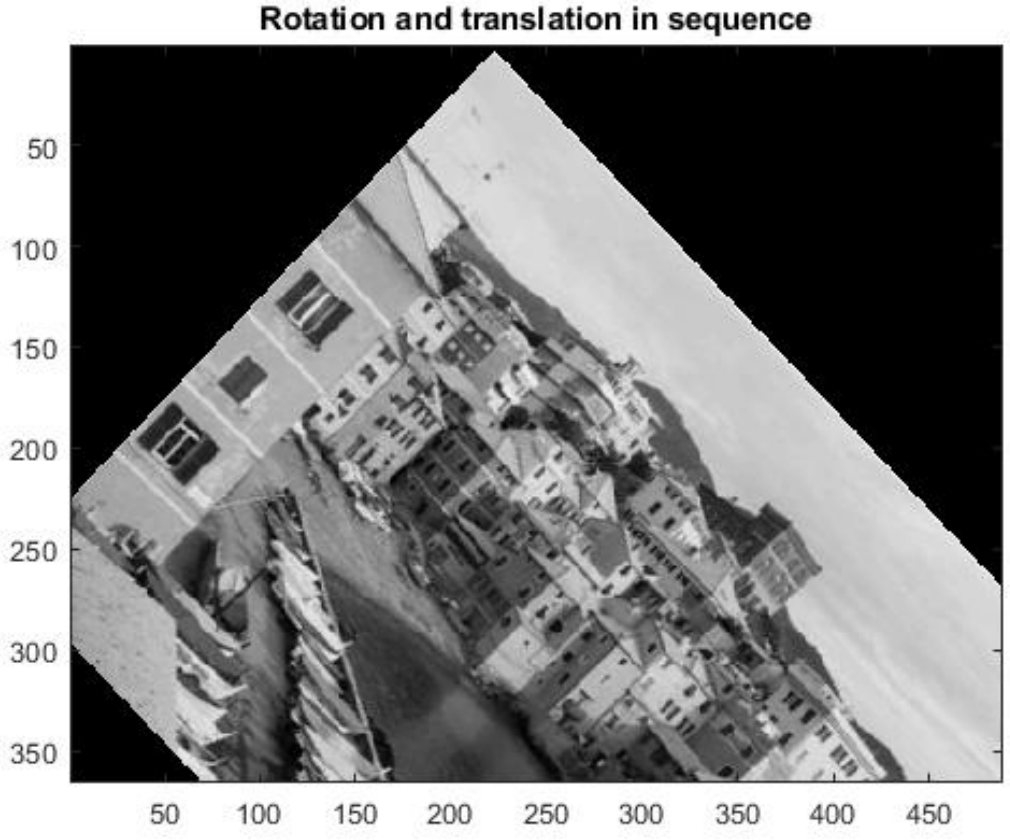


Figure 3.7: Rotation and translation in sequence

3.5 Colored rotation

As it was said before, it has been also realized a version of the rotation function that rotates an RGB image. The only difference in this case is the fact that the input image (clearly RGB in this case) is manipulated separately in the three channels R, G and B. Each channel is rotated separately in the same way as before, then the three channels are recomposed simply by using the function "cat", which concatenates the matrices. Eventually, the values of the final image has to be remapped in the range between 0 and 1, by dividing the image itself by 255 in order to properly display the RGB image.

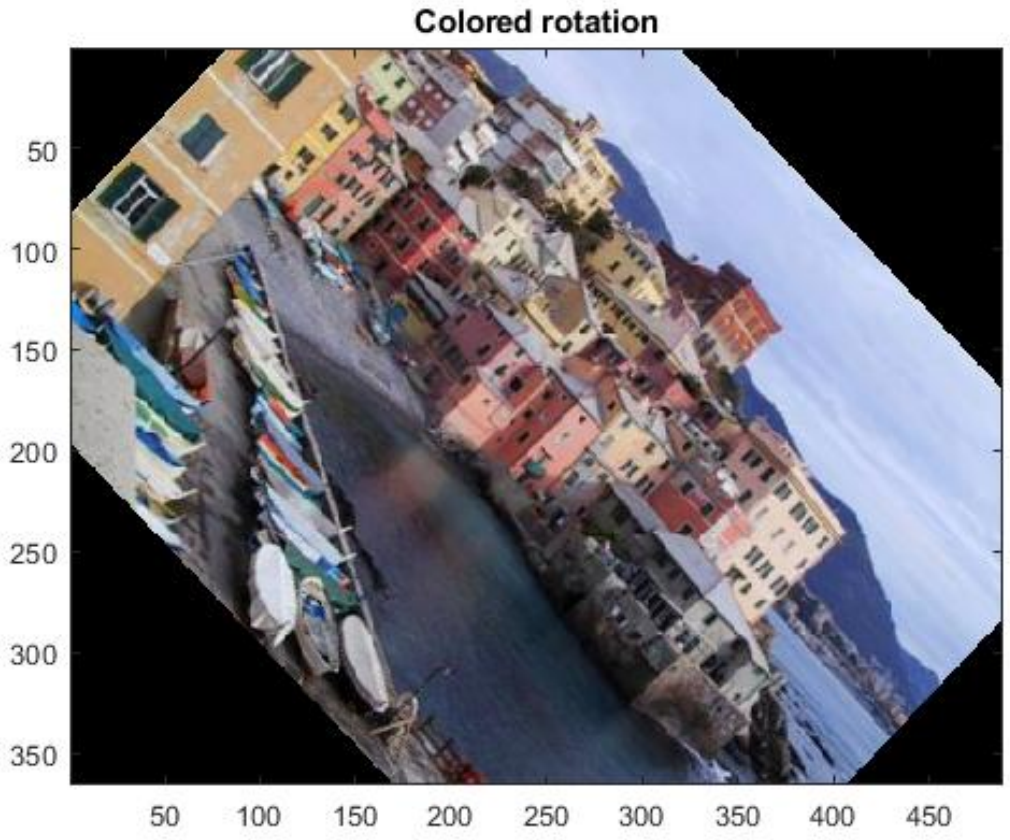


Figure 3.8: Colored Backward Rotation