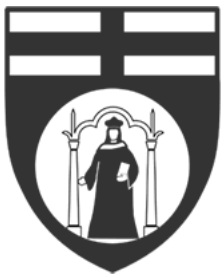


Università degli studi di Genova

# Computer Vision

Assignment 1: Traslation, Rotation, Warping and Bilinear interpolation.

Aurora Bertino, Chiara Saporetti

March 20,2020

# Contents

# 1. Introduction

*The object of the Lab is to compute a transformed image g(x,y) = f(h(x',y')), given a coordinate transform (x,y)=h(x',y') and a source image f(x',y'). Image transformations are performed as rotation, traslation and backward warping with bilinear interpolation for two different source images: 'boccadasse.jpg ' and 'flower.jpg'. A file data.mat is given in order to compute the warping transformation.*

## 1.1 Digital Image

*Digital Image Processing: An image can be represented as a two-dimensional function f(x, y), where x and y are spatial (plane) coordinates and the third axis z is the intensity of the image at that point. We can obtain a digital image through the conversion of data (from sensors) into digital form and this involves two processes: sampling of the domain and quantization of the codomain. The pixel (element of the matrix) is the basic unit of a digital image which is composed of a finite number of elements with a particular location and value. This kind of representation is quite useful to develop algorithms because components of the images are printed and analyzed as numerical values. This means we can simply display the numerical values of f(x, y) as an array (matrix) M x N x I, where M x N is the number of pixels and I is equal to one for an indexed image and is equal to three for a true color image. In addition, an image is represented in the Cartesian coordinate system with the origin at the top left: the positive x-axis extending downward and the positive y-axis extending to the right.*

$$f(x,y) = \begin{bmatrix} f(0,0) & f(0,1) & \dots & f(0,N-1) \\ f(1,0) & f(1,1) & \dots & f(1,N-1) \\ \vdots & \vdots & \vdots & \vdots \\ f(M-1,0) & f(M-1,1) & \dots & f(M-1,N-1) \end{bmatrix} \tag{1.1}$$

## 1.2 Backward Warping and Bilinear Interpolation

*A geometric transformation modifies the spatial relationship between pixels in an image. It consists of two operations: spatial transformation and intensity interpolation. To implement the first step we used image warping. Let us consider the original images f(x',y') and the transformed (warped) images g(x,y) and let us apply the Forward Warping. It consists in scanning the pixels of the input image and, at each location (x',y'), in computing the spatial location (x,y) of the corresponding pixel in the output image using:*

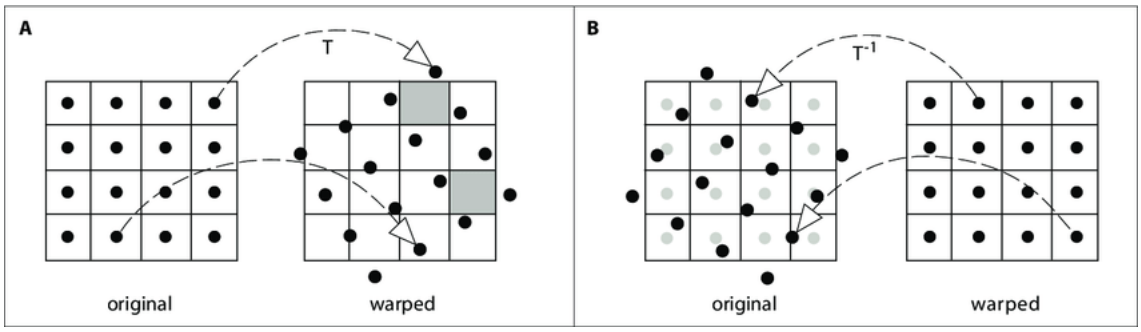
$$(x,y) = T(x',y')$$

*A problem with this approach is that two or more pixels in the input image may be transformed to the same location in the output image. In addition, it is possible that some output locations may not be assigned a pixel at all. A solution to this may be the inverse problem which is called Backward Warping. This algorithm, that*

*is more efficient to implement than the previously described one, starts from the output pixel locations and, by using the following formula:*

$$(x', y') = T^{-1}(x, y)$$

*it computes the corresponding location in the imput image. The second step is the interpolation method, which is the process of using known data to estimate values at unknown locations in order to perform intensity-level assignment. We have different types of interpolation methods (linear, bilinear, spline). In our work we applied the Bilinear Interpolation that uses the four-nearest-neighbors to estimate the intensity at a given location. .*




## 1.3 Rotation

*In general, the transformation of coordinates may be expressed as: (x, y) = T (x', y'). If we only consider the rotation, the transformation matrix is:*

$$(x, y) = \begin{bmatrix} \cos\alpha & -\sin\alpha & 0 \\ \sin\alpha & \cos\alpha & 0 \\ 0 & 0 & 1 \end{bmatrix} (x', y') \tag{1.2}$$

## 1.4 Traslation

*In general, the transformation of coordinates may be expressed as: (x, y) = T (x', y'). If we only consider the traslation, the transformation matrix is:*

$$(x, y) = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ tx & ty & 1 \end{bmatrix} (x', y') \tag{1.3}$$

# 2. Matlab Resolution

*As previously stated, with our code it is possible to apply, to an input image, different kinds of transformations such as translation, rotation and warping by using data from an external file. The image after the acquisition is stored as an m-by-n-by-3 data array that defines red, green, and blue color components for each individual pixel (RGB image). We tested the code on two different RGB images: In the first case the input image (boccadasse.jpg) is translated, rotated and finally warped using data from the given data.mat file. The resulting modified image is then transformed, via the matlab function rgb2gray, from RGB to grayscale in order to plot it in black and white. In the second case the input image (flower.jpg) is just translated and rotated.*

## 2.1 Backward Warping and Bilinear Interpolation

$$transfImg = warping(inputImg, Xwarp, Ywarp) \tag{2.1}$$

This function applies a warping to the image ('boccadasse.jpg') by summing given values to the coordinates of the same. It receives as input the image that the user wants to modify (namely, inputImg), and the values to sum to the coordinates (Xwarp, Ywarp). In the following steps, the transformation is applied:

- Create a grid from the input image with meshgrid(x,y).
- Shift the center of the axis in the middle of the image.
- Apply the transformation by directly summing the values of Xwarp and Ywarp to the coordinates obtained in the previous step.
- Shift back the center of the axis.
- Apply the Bilinear Interpolation with function griddata(x,y,v,xn,yn).
- Unify the results obtained for the three layers in a unique matrix, which represents the transformed image.

After these computations, the result is the modified image.

## 2.2 Rotation

*We use the generic function "transformation", applies a translation and a rotation to an image, to simplify the syntax. It receives as input the image that the user wants to modify (namely, inputImg), the values to use for the shifting along x and y (Xtrans, Ytrans) and the angle about which to rotate the function(theta).*

$$transfImg = transformation(inputImg, Xtrans, Ytrans, theta) \tag{2.2}$$

By using these values it computes the transformation matrix, which is only composed of a rotation about theta (as previously described). Then it is subsequently stored in an object (affine2d) for applying the inverse transformation later in the code. The structure of the function is very similar to the step followed for the warping:

- Create a grid from the input image with meshgrid(x,y).
- Shift the center of the axis in the middle of the image.
- Apply the transformation with transformPointsInverse(T,x,y).
- Shift back the center of the axis.
- Apply the Bilinear Interpolation with function griddata(x,y,v,xn,yn).
- Unify the results obtained for the three layers in a unique matrix, which represents the transformed image.

We apply it separately to the three layers of the matrix and as result, we finally have the modified image.

## 2.3 Traslation

*We use the same functions described in the previous paragraph without considering rotating components and adding the traslationing (Xtrans, Ytrans). By using these values it computes the transformation matrix, which is only composed of a traslation along x-axis and y-axis.*

# 3. Results

## 3.1 Test 1


*This is the result starting from an RGB image and applying the transformations: translation, rotation and backward warping. The last step as we see is to bring the image in a grey-scale colour.*

### 3.1.1 Test 2


*The second result is the comparison between the original image and the modified one. We can see rotation and traslation for all channel of the image.*

[5] [4] [3] [1] [2]

# Bibliography

[1] *Mathwork documentation on cat.*

[2] *Mathwork documentation on griddata.*

[3] *Mathwork documentation on meshgrid.*

[4] *Mathwork documentation on transformPointInverse.*

[5] Richard Szeliski. *Computer Vision: Algorithms and Applications.* University of Whashington, 2010.