Journal of Theoretical and Applied Information Technology
10th February 2014. Vol. 60 No.1

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

GEOMETRIC TRANSFORMATIONS AND ITS APPLICATION IN DIGITAL IMAGES

[1]SILVESTRE ASCENCIÓN GARCÍA SÁNCHEZ, [2]CARLOS AQUINO RUIZ, [3]CELEDONIO ENRIQUE AGUILAR MEZA
Escuela Superior de Ingeniería Mecánica y Eléctrica Unidad Culhuacán IPN, Av. Santa Ana 1000, Col. San Francisco Culhuacán, Deleg. Coyacán C.P. 04430, México D.F.
[1] Prof., Escuela Superior de Ingeniería Mecánica y Eléctrica, del IPN
[2]Prof., Escuela Superior de Ingeniería Mecánica y Eléctrica, del IPN
[3] Prof., Escuela Superior de Ingeniería Mecánica y Eléctrica, del IPN
E-mail: [1]silvestregarcia@hotmail.com , [2]caquino@ipn.mx , [3]cele_ag@hotmail.com

ABSTRACT

Digital images usually represent a wide range of phenomena. The area of image processing has been developed through the theoretical study of the different transformations manifested in the creation of algorithms that cast real-life problems. This paper establishes the theoretical aspects of linear algebra: linear transformations and related. We present some of the most commonly used transformations on both digital images and their pixel intensity values which are implemented by the use of Matlab software. Finally, we study some aspects of numerical interpolation on images.

Keywords*: Linear Transformation, Affine Transformation, Processing Spatial Interpolation.*

1. INTRODUCTION

In signal and image processing some techniques from knowledge and experience of linear and nonlinear operators are used. The advancement of communication technologies and information now allow imaging application (matrices) and transformations of linear algebra to various areas of pure and applied sciences and engineering

Given that a digital image is a matrix representation of vector space concepts and linear algebra turn out to be natural in processing. Transformations are applied to various types of images with different purposes (eg, correction of distortions due to optics, sensor type, camera-view scene, introduction of distortion to register pictures, motion estimation and creating panoramic images. Shape recognition invariant to certain transformations).

This paper is organized as follows: First, it defines commonly used linear transformations in homogeneous coordinates, and matrix representation. Then the methods for transforming digital images (spatial transformations) and the most common methods of interpolation and finally the results and conclusions.

2. METHODOLOGY

Linear Transformations

The geometric transformations modify the spatial relationship between pixels. This consists of two basic operations:

1. A **spatial transformation** defines the relocation of the pixels in the image plane.
2. **Interpolation** of the gray levels, ie mapping intensity levels of the pixels of the transformed image.

A particular case of geometric transformation are linear transformations. For the definition of these transformations, every point is represented (x, y) of the 2D image in homogeneous coordinates. By definition the point (x, y) in homogeneous coordinates is given by (ax, ay, a) where a is a constant. If a = 1 (which is the most widely used convention) it will be given by (x, y, 1).

1. Translation (T)

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} 1 & 0 & t_x \\ 0 & 1 & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x + t_x \\ y + t_y \\ 1 \end{bmatrix} \quad (1)$$

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

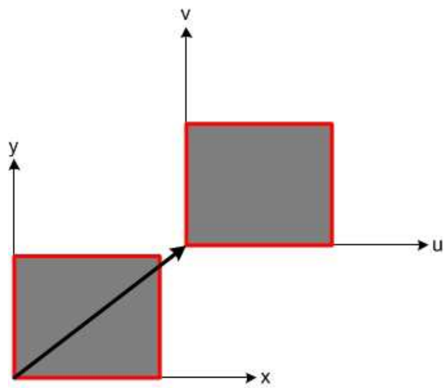
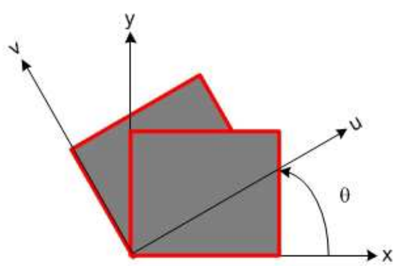


Figure 1. Blink

2. Rotation (R)

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} \cos\theta & -\mathrm{sen}\theta & 0 \\ \mathrm{sen}\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} x\cos\theta - y\,\mathrm{sen}\theta \\ x\,\mathrm{sen}\theta + y\cos\theta \\ 1 \end{bmatrix}$$

(2)



3. Scaling (S)

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & 0 & 0 \\ 0 & s_y & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} s_x x \\ s_y y \\ 1 \end{bmatrix} \quad (3)$$

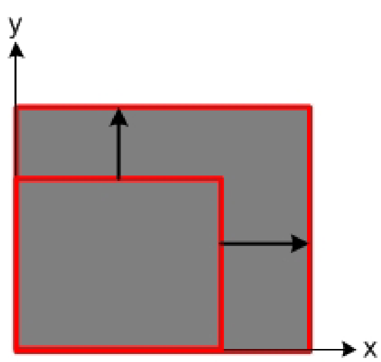


Figure 3. Escalating

5. Affine. An affine transformation is a combination of the above (translation, rotation, scaling and slope)

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} s_x & a_x & t_x \\ a_y & s_y & t_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} s_x x + a_x y + t_x \\ a_x x + s_y y + t_y \\ 1 \end{bmatrix} \quad (5)$$

In the above equation, ax represents the inclination in the horizontal direction and ay tilt vertically. An affine transformation can also be defined as the composition of the following transformations: Similarity (T + R + S isotropic) + S + I.

Affine transformations have the property of preserving straight lines as shown below:

$$Au + Bv + C = 0$$

$$A(s_x x + a_x y + t_x) + B(a_x x + s_y y + t_y) + C = 0$$

$$(As_x + Ba_x)x + (Bs_y + Aa_y)y + (At_x + Bt_y + C) = 0$$

$$A'x + B'x + C' = 0$$

So a grid (horizontal and vertical straight lines) by an affine transformation is transformed to another grid.

Spatial Transformations

There are two methods to relocate transform digital images:

1. Direct transformation (forward mapping). This method requires high computational complexity for implementation. The main disadvantage is that the pixels that fall outside the grid are transformed. For example, consider a 90 ° rotation on the image shown in Figure 5.

Figure 5. Note that by transforming there are some pixels that remain outside of the grid in the output image

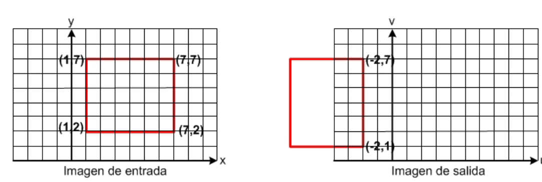


Figure 4 & 5

2. Inverse transformation (inverse mapping). This transformation is easy to implement, and involves taking the domain of the position of the pixels in the output image and determine the position of where they come in the input image. The main disadvantage is that there are pixels that are taken on more than one occasion

Journal of Theoretical and Applied Information Technology
10th February 2014. Vol. 60 No.1

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

as it will be discussed in the interpolation methods.

Interpolation

Once through a linear transformation the position of the pixels is determined in the output image, the next step is to assign a level of intensity. The methods used most are defined below:

1. The nearest neighbor. Consists in assigning to the level of intensity of a pixel of the output image the one of the closest pixel to the input image once the transformation is applied inversely. For example, an isotropic scaling with sx = sy = 3. To show how this method functions, the reverse transformation is obtained from the equation (4) as:

$$\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{bmatrix} 0.33 & 0 & 0 \\ 0 & 0.33 & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} 0.33u \\ 0.33v \\ 1 \end{bmatrix} \quad (6)$$

By applying this transformation to the coordinates (u, v) = {(0, 0), (1, 1), (2, 2)}, which is obtained from the coordinates (x ', y') = {(0, 0), (0.33, 0.33), (0.67, 0.67)}. Thus, by virtue of which the coordina tes in a 2D digital image only have integer values, 0.33 and 0.67 are rounded to 0 and 1 respectively, whereby (x, y) = {(0, 0), (0, 0), (1, 1)}. That is, the pixel in the input image with coordinates (0, 0) is taken twice. Figure 6 illustrates this procedure for a 3x3 grid.

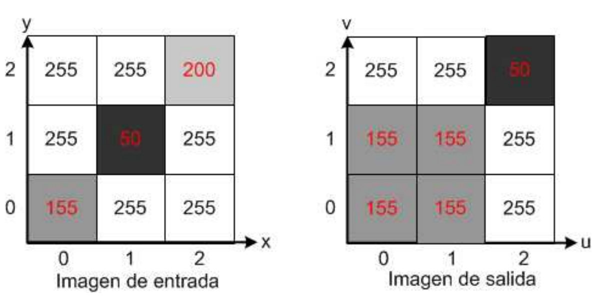


Figure 6. The figure on the left represents the input image. To the right the output image after applying a reverse scaled isotropic with sx = sy = 3 followed with the nearest neighbor interpolation.

Bilinear transformation. In this type of interpolation, linear interpolation along each row and the result afterwards along columns (it is considered for the four nearest neighbors, as shown in Figure 7). Using the linear interpolation function (see figure 8):

$$h(x) = \begin{cases} 1 - |x| \text{ si } 0 < |x| < 1 \\ 0 \text{ cualquier otro caso} \end{cases} \quad (7)$$

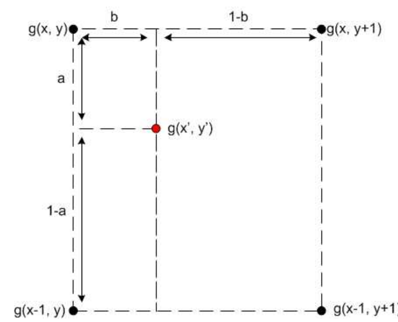


Figure 7. Neighborhood in the bilinear interpolation process g (x, y) indicates the intensity level assigned to the coordinate (x, y) in the input image

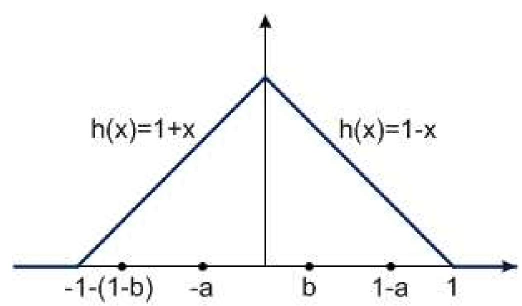


Fig.8 Interpolation Function

Applying the horizontal linear interpolation g (x, y) g (x, y +1) and g (x +1, y) g (x +1, y +1) we have h (y'-y) g (x, y) + h (y'-(y +1)) g (x, y +1) h (y'-y) g (x-1, y) + h (y'-(y +1)) g (x-1, y +1). Then, realizing the vertical interpolation on the previous values it leads us to the expression:

$$g(x',y') = h(x'-x)\left[h(y'-y)g(x,y) + h(y'-(y+1))g(x,y+1)\right] + h(x'-(x-1))\left[h(y'-y)g(x-1,y) + h(y'-(y+1))g(x-1,y+1)\right] \quad (8)$$

Considering that b = y'-y and that a = x-x ', we have that h (y'-y) = h (b), h (y'-(y+1)) = h (- (1-b)), h (x'-x) = h (-a) and h (x'-(x-1)) = h (1-a) Then,

Journal of Theoretical and Applied Information Technology
10th February 2014. Vol. 60 No.1

ISSN: 1992-8645 www.jatit.org E-ISSN: 1817-3195

substituting into the equation (7), we have h (b) = 1-b, h (- (1-b)) = 1 - (1-b) = b, h (-a) = 1-a and (1-a) = 1 - (1-a) = a.

Therefore,
g (x ^ 'y ^') = (1-a) [(1-b) g (x, y) + bg (x, y +1)] + a [(1-b) g (x-1, y) + bg (x-1, y +1)] (9)

3. RESULTS

Figure 9 shows the action of a rotation of 45 ° followed by a translation and scaling isotropic. Subsequently are applied both interpolation methods. Note that the bilinear method softens the resulting image (that is, it removes distortions).

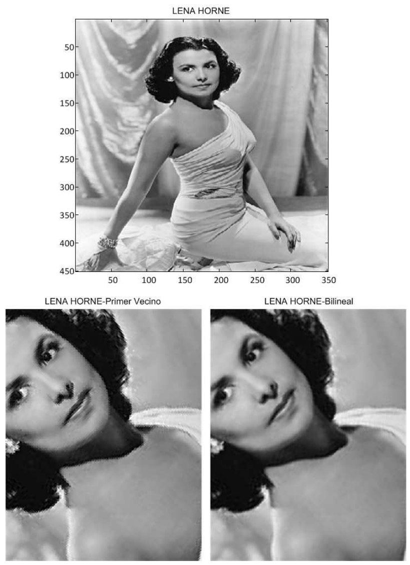


Figure 9. Image LENA HORNE. Note The Difference Between The Interpolation Methods

REFRENCES:

[1]. Deransart, P., AbdelAli, E. & Laurent C. (1991). Prolog: *The standard reference manual.* Springer-Verlag Berlin Heildelberg 1996. Iranzo, P.J. & María, A.F. (2007).
[2]. *Programación Lógica Teoría y Práctica*.
[3]. [Johnsonbaugh, R. (2005). Matemáticas Discretas Sexta Edición. *Trayectorias y ciclos* (pp. 329-336). Pearson Educación de México,
[4]. S.A. de C.V. Armenta, R.A. (2010). Matemáticas Discretas *Permutaciones y combinaciones* (pp. 306-310). Alfaomega Grupo Editor, S.A. de C.V., México.
[5]. Bratko, I. (1986). Prolog programming for artificial intelligence. *Graphs* (pp. 224-232) Addison – Wesley Publishing Company