A Novel Algorithm to Perform Precalculated Tables for the Real-Time Image Processing: Illustration with Image Rotation

Zohir IRKI[1], Michel DEVY[2]
[1]*Numerical Systems Laboratory/EMP, BP17 Bordj El Bahri, Algeria*
[2]*Laboratoire d'Analyse et d'Architecture des Systèmes, LAAS/CNRS, Toulouse, France*
zohir_irki@yahoo.fr, zirki@laas.fr

***Abstract*—In this paper, we present an approach which makes uses of precalculated tables in order to carry out a real time image processing task. This approach is very useful when the relation between the input image and the output image (after treatment) is given in the backward direction (the input image is expressed using the output image). We illustrate this approach in the case of the rotation of an image around its center. We illustrate how theses precalculated tables should be built and the influence of some parameters on the size of theme. By the end, we discuss how this approach could be implemented on a reconfigurable hardware as an FPGA.**

***Index Terms*—algorithm, field programmable gate arrays, image processing, precalculated table, real-time.**

I. Introduction

Image processing is often used to make measurements for real-time applications. In such applications, it is important to ensure that accurate measurements can be made from captured images [1]. This task is sometimes very difficult to carry out especially when the original image is expressed using the result image (after treatment). This is typically the case of image rotation and correction of radial distortion.

In these two cases, the coordinates in the untreated image are given as function of those in the treated image. This form is unsuitable for the real-time treatment because it is necessary to know the untreated output pixel before displaying in the treated pixel. Often, the treated image is bigger than the captured image and since the displaying screen has a limited size, the dimensions of the original image should be recalculated in order to adapt the size of the treated image with the size of the displaying screen.

Image rotation is one of the geometric operations which change the spatial relationship between objects in an image [2, 3]. This operation turns or rotates an input image by any angle θ. In this paper, we are interested by a rotation around the image center. In general, the rotated image is bigger than the input one. But according to the desired application, the size of the rotated image is computed starting from the size of the input image if there are no constraints on the size of the displaying screen. If such constraints exist, the size of the input image should be limited in order to display the whole rotated image.

The use of pre-calculated look-up tables to perform image transformations has been worked on previously. They have been used for image warping. When the geometric transformation does not change from one image to the next, or a more generalized warp is desired, a spatial lookup table (LUT) composed of precalculated inverse (or forward) mapping coordinates has been used instead of performing the same geometric transformation on every image. In [4] is presented an algorithm that takes the LUT approach one step further. It uses an enhanced LUT (ELUT) that incorporates precomputed data transfer and interpolation information. Once computed, it speeds up processing by making data transfers more efficient and eliminating the need for the pixel address and interpolation coefficient calculations. [5] is another work which used LUT to perform image warping. The aim of [5] is to extend the 2-pass approach to handle arbitrary spatial mapping functions.

The aim of using precalculated tables in our work is to implement the desired application in a reconfigurable component such as FPGA [6]. This implementation appears to be difficult in the cited examples when it is based on the resolution of mathematical models. The problem will be summarized to memories manipulation. This avoids us to make complex calculus and this will have a direct influence on the processing time [7].

The purpose of this work is not to study image rotation; many works have treated this problem but it will be helpful to describe the mathematical model used for rotation. The aim of this work is to show how to built look up tables for a real time treatment. In this paper, we proposed an improvement of the approach used in [7] so that memory requirement will be optimized. We start by describe tables creation for the problem of image rotation. After that we describe the optimization of these tables. By the end, we propose an architecture which allows the implementation of the method on a reconfigurable hardware (FPGA).

II. Mathematical Model

Geometric image operations are, in a sense, the opposite of point operations: they modify the spatial positions and spatial relationships of pixels, but they do not modify gray level values. Generally, these operations can be quite complex and computationally intensive, especially when applied to video sequences [8, 9]. A geometric image operation generally requires two steps. The first is a spatial mapping of the coordinates of an original image f to define a new image g: $g(n) = f(n') = f[a(n)]$.

Thus, geometric image operations are defined as functions of position rather than intensity. The two-

Digital Object Identifier 10.4316/AECE.2011.04011

dimensional, two-valued mapping function, $a(n)=[a1(n1,n2),a2(n1,n2)]$, is usually defined to be continuous and smoothly changing, but the coordinates a(n) that are delivered are not generally integers. The question then is, which value(s) of f are used to define $g(n)$, when the mapping does not fall on the standard discrete lattice?

Thus implies the need for the second operation: interpolation of noninteger coordinates $a1(n1,n2)$ and $a2(n1,n2)$ to integer values, so that g can be expressed in a standard row-column format.

Among the geometric image operations we quote image rotation which allows any point (i,j) to be the center of rotation. In order to rotate the original image and to display the entire rotated image, we compute the size of the rotated image which is in general bigger than the size of the original one. Figure 1 illustrates how the input image revolves about the image center.

The input image is composed from x rows and y columns. Let X and Y be the numbers of rows and columns of the rotated image and θ the angle with which we want to rotate the original image.

As illustrated in figure 1, when we rotate the original image, the rotated image corners belong to a circle of radius:

$$R=\frac{\sqrt{x^2+y^2}}{2} \tag{1}$$

The size of the rotated image is computed using these corners positions [10].

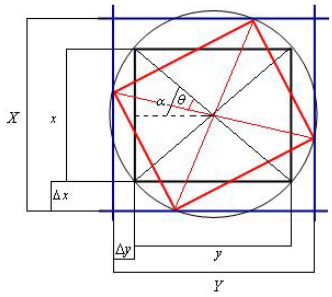

Figure 1. Computation of the rotated image size

The maxima values of X and Y are reached when the rotation angle is equal to:

$$\theta=\alpha=tg^{-1}\left(\frac{x}{y}\right) \tag{2}$$

In general, we have

$$\begin{cases} X=x+2*\Delta x \\ Y=y+2*\Delta y \end{cases} \tag{3}$$

With:

$$\begin{cases} \Delta x=\left|0.5*x-R*\cos\left(\frac{\pi}{2}-\alpha-\theta\right)\right| \\ \Delta y=\left|0.5*y-R*\cos(\alpha-\theta)\right| \end{cases} \tag{4}$$

After the computation of (X,Y), we proceed to the gray level attribution. To each pixel (i,j) belonging to the rotated image, we compute its corresponding pixel (u,v) in the original image. This is done by applying the following equation to each (i,j) pixel:

$$\begin{bmatrix} u \\ v \end{bmatrix}=M*\begin{bmatrix} i \\ j \end{bmatrix}+\frac{1}{2}\begin{bmatrix} X \\ Y \end{bmatrix}-\frac{1}{2}M*\begin{bmatrix} X \\ Y \end{bmatrix}-\begin{bmatrix} \Delta x \\ \Delta y \end{bmatrix} \tag{5}$$

With:

$$M=\begin{bmatrix} \cos(\theta) & \sin(\theta) \\ -\sin(\theta) & \cos(\theta) \end{bmatrix}$$

The displacement $[\Delta x \quad \Delta y]$ is due to the fact that the original and rotated images have not the same origin. When u and v are calculated, we distinguish 2 cases:

- The original coordinates are out of original image limits: this happens when $u<0$, $u>x$, $v<0$ or $v>y$. In this case, the considered rotated pixel will be considered as black.
- The original coordinates are in the limits of the original image: this happens when $0<u<x$ and $0<v<y$. In this case, we attribute to the considered rotated pixel the gray level of the (u,v) pixel.

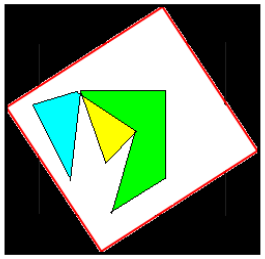

Figure 2. Image after rotation

The size of the rotated image is a function of the rotation angle θ. It is also function of input image dimensions and the proportion between them. In this paper, we define image size as the number of its pixels.

Figure 3 represents the evolution of the rotated image size when θ changes. This evolution is represented for different values of the ratio x/y. This evolution is π periodic. For different input images having approximately the same size (im1[100x100] ; im2 [80x125]; im3[70x150]), we observe that in the case of the third image, the rotated image size can reach a value of 35000 pixels which represents 3 times the original image size. This is due essentially to the fact that the radiuses of the circles, to which belong the corners of the rotated images, are different.

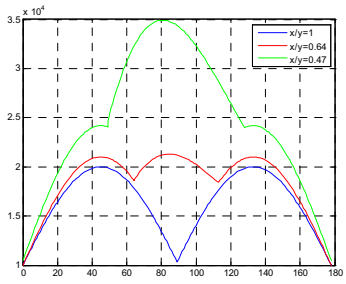

Figure 3. Evolution of the rotated image size

The problem of rotation can be seen differently when the size of the displaying screen if fixed. In this case, we can not change the size of rotated image. If we want to display the whole input image, we must choose an input image which can be rotated and displayed in the possible limits.

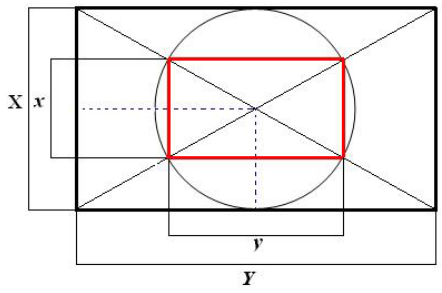

Figure 4. Inverse case

The computation of the input image size represents the inverse problem of the computation of the rotated image size but it is simpler. Input and rotated images have the same center. Rotated image corners belong always to a circle. In order to compute the input image size, we must first, determine the radius of this circle.

Let X and Y be the number of rows and columns of the displaying screen and let x, y be the maxima numbers of line and columns of an input image.

The radius of the circle which must contain rotated image corners is given by the equation (1). Its value must verify the following relation:

$$R = \frac{\sqrt{x^2 + y^2}}{2} = \min\left(\frac{X}{2}, \frac{Y}{2}\right) \tag{6}$$

Let take the case illustrated in figure 4 in which $R = \frac{X}{2}$. Using trigonometric relationships, we can demonstrate that:

$$x = \frac{X^2}{\sqrt{X^2 + Y^2}}$$
$$y = \frac{X * Y}{\sqrt{X^2 + Y^2}} \tag{7}$$

If $R = \frac{Y}{2}$, we will have

$$x = \frac{X * Y}{\sqrt{X^2 + Y^2}}, \; y = \frac{Y^2}{\sqrt{X^2 + Y^2}}. \tag{8}$$

In this case:

$$\Delta x = \frac{X - x}{2}; \; \Delta y = \frac{Y - y}{2} \tag{9}$$

In the real time processing, the limitation of the size of the input image can be done using a CMOS progressive scan camera which allows selecting a window of interest [11] or by ignoring rows and columns which do not belong to the computed zone if the input image is acquired using a CCD camera. In this paper, we will consider the general case in which the size of the rotated image is computed. Here the application of image rotation is taken just as example. The main purpose of this paper is how to use precalculated tables in order to resolve a similar problem.

As cited before, the fact that u and v are no integer implies them interpolation to integer values. There are many possible approaches for accomplishing interpolation. In this work, we used the simplest of them: the nearest neighbor interpolation [3]. When using this interpolation, an integer value can be used as correspondent more than once. The influence of this aspect will be discussed later.

III. Precalculated Tables

As mentioned in equation (5), in order to attribute the gray level of the (i, j) pixel (belonging to the rotated image), we must compute its correspondent (u, v) belonging to the original image and this implies the memorization of a certain number of input image rows. This is unsuitable for real-time processing in which data is used immediately after its acquisition. In order to carry out a task in the real-time, an approach based on the use of precalculated tables is proposed [5].

Precalculated tables were used in [12, 13] in order to avoid complex computation of correspondents. In this work, author was interested to the rectification of stereoscopic images. The problem is similar to image rotation because the attribution of a rectified image pixel gray level implies to compute its correspondent in the non rectified image using a complex equation. Thus the author memorized, for each rectified image pixel, its correspondent in the input image. This operation will be done once and memorized data will be used each time. This memorization avoids complex computation but do not resolve problem of real-time treatment. This problem was resolved in [5] by modifying precalculated tables nature. In this work, author was interested to the real-time rectification of stereoscopic images. For this, the nature of the precalculated tables used in [8] has been modified. Instead using, one table, used two tables were used. In the first table, is memorized for each original pixel, a unique characteristic number. For a rectified pixel, this number, which can 0 or different, represent the number of correspondents combined with an address (in the second table) where the first of these correspondents is memorized. According to this parameter, original pixels are classified into *passive* and *active* pixels. In the second table, correspondents are memorized only for active pixels. This second table is subdivided into areas. The size of each area is equal to the number of correspondents of the considered pixel. We note that in this table, active pixels and areas are numbered so that user can avoid any data confusion.

This work appears to be suitable for a real-time processing. But because of the serial aspect (due to the areas in the second table), the real-time processing could not be carry out. A parallel processing was proposed in [14]. The second table is subdivided into tables having the same size. The number of these tables is the maxima number of correspondents. Some correspondents will be duplicated so that all active pixels will have the same number of correspondents.

In this paper, we have used precalculated tables in order to carry out a real-time image rotation. For this, we have been inspired of [6]. We have used a first table which allows classifying input pixels into active and passive, but in this table we have memorized only the number of correspondents in the rotated image. This represents a first improvement. As used in [6]; we used also tables for the memorization of correspondents (a table for rows and a table for columns). Each table was subdivided into tables having the same size with duplication of some correspondents. In this paper, correspondents tables are subdivided into tables which have not the same size, thus no correspondent should

be duplicated. This contributes in the minimization of the used memory resources. The approach of precalculated tables was also used in [15] to carry out several image processing tasks.

IV. STEPS OF PRECALCULATED TABLES CREATION FOR IMAGE ROTATION

Used tables are calculated, off-line, once for each rotation angle. For this, we must first compute the size of the rotated image (X,Y) as described before. After that, we must proceed on four steps before that final tables will be ready to be used. These steps are:

- Creation of intermediate tables;
- Creation of the characteristic table;
- Creation of correspondent's tables;
- Decomposition of correspondent's tables on N tables.

Figure 5 illustrates steps 2, 3 and 4. For a better explanation, let us take for example the rotation of an 8X8 image with 45°. This example illustrates how to built rotation tables and how to use them later.

A. Creation of the intermediate tables

We consider two intermediate tables $T1$ and $T2$, each table contains $X*Y$ elements. If we note u_{ij} and v_{ij} the nearest integer to the result of the application of the equation (5) on the pixel (i,j), we will have:

$$\begin{cases} T1\left(j+(i-1)*Y\right)=u_{ij} \\ T2\left(j+(i-1)*Y\right)=v_{ij} \end{cases} \quad (10)$$

For an (i,j) pixel having no correspondents in the input image (The calculated u_{ij}, v_{ij} are out of original image limits), we will have:

$$\begin{cases} T1\left(j+(i-1)*Y\right)=0 \\ T2\left(j+(i-1)*Y\right)=0 \end{cases} \quad (11)$$

At the end of this step, we will have two tables that we will use in order to create the real-time useful precalculated tables. The size of the intermediate tables is function of the rotation angle. These intermediate tables will be used just off-line and they are relative to the rotation angle. If we want to make two successive rotations, the association of tables is not to use because we must take care of some parameters, in particular, influence of the rotation angle on the size f the rotated image. In the case of the considered example, the size of the rotated image is calculated and it is equal to 11. For symmetry reasons, they are equal and given by the following table.

TABLE I. INTERMEDIATE TABLE FOR A SIMPLE EXAMPLE

0	0	0	0	0	1	0	0	0	0	0
0	0	0	0	1	2	2	0	0	0	0
0	0	0	1	2	2	3	4	0	0	0
0	0	1	2	2	3	4	5	5	0	0
0	1	2	2	3	4	5	5	6	7	0
0	2	2	3	4	5	5	6	7	7	0
0	0	3	4	5	5	6	7	7	0	0
0	0	0	5	5	6	7	7	0	0	0
0	0	0	0	6	7	7		0	0	0
0	0	0	0	0	7	0	0	0	0	0
0	0	0	0	0	0	0	0	0	0	0

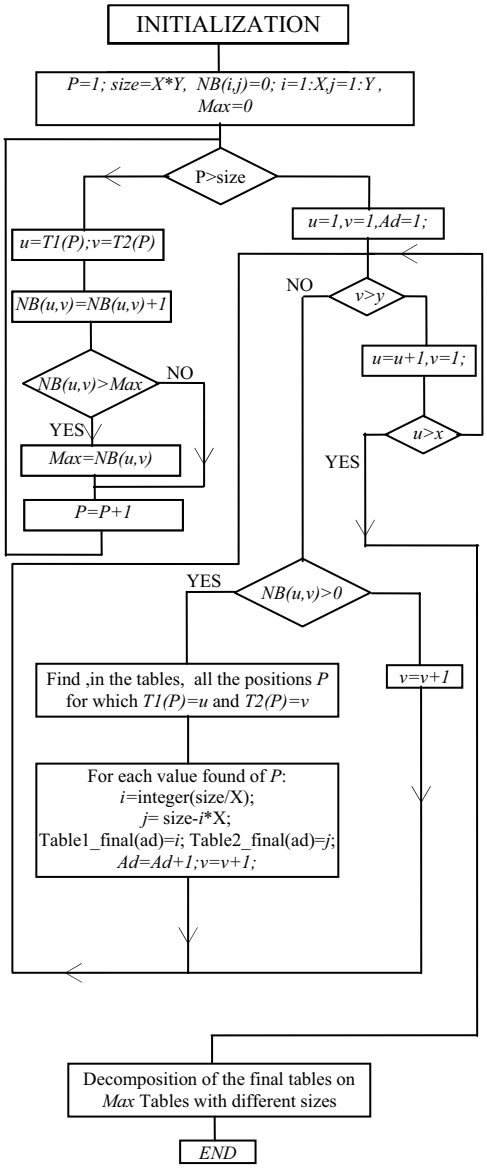

Figure 5. Steps of precalculated tables creation

B. Creation of the intermediate tables

By characteristic table we mean a table which allows classifying input image pixels into active and passive pixels. An active pixel is a pixel having at least one correspondent pixel in the rotated image. Its gray level will be attributed to a set a correspondents. A passive pixel is a pixel which has not correspondents in the rotated image. Thus, the factor will can be used in order to classify input pixels is the number of correspondents.

The characteristic table can be a matrix of x rows and y columns as it can be considered as an array of $x*y$ elements. The representation as an array is more suitable for implementation because the table will be assimilated to a ROM. The characteristic factor of the (u,v) pixel will be stored in the row $v+u*y$.

As illustrated in figure 5, the characteristic table is created after a first sweeping of the intermediate tables. For each (u,v) memorized in these tables, we compute how many times this value has appeared. This is done by incrementing the value memorized in the $v+u*y$ row each time we found (u,v) in the intermediate tables.

At the end of this step, we will define for each input pixel its characteristic number. Also we need to know a maxima number of correspondents; this is easily done during this step. In the case of rotation with 30°, 45° or 60° the maxima number is 2. This number will be used in the last step. After the creation of correspondent's table, it will be subdivided into 2 separate tables. In the case of the considered example, the characteristic table is given by the following table.

TABLE II. CHARACTERISTIC TABLE FOR A SIMPLE EXAMPLE

0	1	1	0	1	1	1	0
1	2	1	1	2	1	2	0
1	1	0	1	1	0	1	0
0	1	1	0	1	1	1	0
1	2	1	1	2	1	2	0
1	1	1	1	1	0	1	0
1	2	1	1	2	1	2	0
0	0	0	0	0	0	0	0

After the creation of the characteristic table, we can compute the number of elements in each separate table by a simple counting operation.

C. Creation of the corresponden's tables

By correspondent's table we mean, the table in which are memorized the correspondents of the active pixels. This table is subdivided into areas so that the correspondents of the n^{th} active pixel will be memorized in the n^{th} area.

In order to create this table, we must use the intermediate tables built in step 1 and the characteristic table built in step 2. For each pixel belonging to the input image, we must determine its nature (active or passive from the characteristic table). For an active pixel, we must sweep the intermediate table in order to determine all its correspondents and to memorize each correspondent in an adequate emplacement. We must take care of not forgetting to increment the address after each memorization.

At the end of this step, we will have a table for rows and another for columns. These two tables can be supposed as only one by making a data fusion. For reasons of symmetry, these tables are equals for the given example.

D. Decomposition of the corresponden's tables

The table of step 3 will not be used in a real-time processing because of the serial aspect of memorization. This aspect makes that in order to determine the correspondents of an active pixel having more than one correspondent; we need more than a single access to the correspondent's table. This is physically difficult because the access to a memory is synchronized by the system's clock. At each top of clock, a new pixel must be processed. This fact obliges us to decompose the correspondent's table into N tables so that we pass to a parallel processing.

The number of tables, into how the correspondent's table must be decomposed, is the maxima number of the correspondents of an active pixel. Final table will not have the same number of elements. The correspondent of an active pixel having N correspondents are memorized in N separate tables in different emplacements. When passing to a parallel processing, a counting system allows specifying the exact correspondents using information memorized in the characteristic table.

At the end of this step, we will have all the correspondents which are memorized in N separate tables. This manner is suitable for a real-time processing. And here all the necessary tables are ready to be used. In the considered example, first tables contain 41 elements, second tables contain 9 elements.

V. USE OF THE BUILT TABLES

For a later use (an eventual implementation), necessary tables are only the characteristic table and the separate tables. For each pixel, belonging to the input image, we must read its characteristic parameter from the characteristic table. And then, if the considered pixel is an active one, we must determine exactly its correspondents from the separate tables. For a better explanation, let N be the number of the separate tables and let $M < N$ be the number of the correspondents of the **first** founded active pixel. The correspondents of the first active pixel are memorized in the first boxes of the M first separate tables. Now let suppose that the second active pixel has N correspondents. Its M first correspondents are memorized in the second boxes of the M first separate tables while its other correspondents are memorized in the first boxes of the $N - M$ latest separate tables. An associated counting system allows to correctly managing this task.In order to validate this approach, we have built precalculated tables for the rotation of an input image of size [256x256]. These tables ensure the rotation of any input image of 256 rows and 256 columns with an angle of 45°. Figure 6 illustrates the result of this application on two different images.

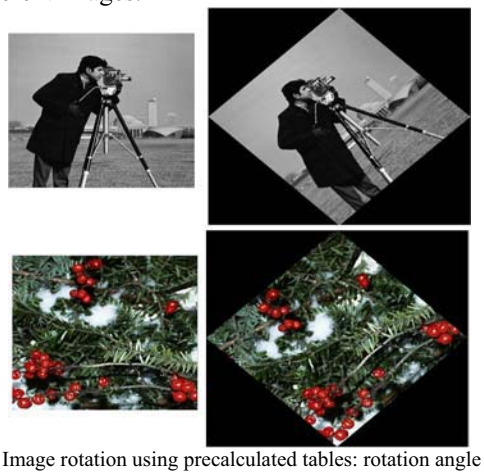

Figure 6. Image rotation using precalculated tables: rotation angle is 45°

In the rotated image, the number of the non black pixels is 65161. This means that 375 (which represents 0.57% of pixels number) pixels have not been rotated. This is essentially due to the approximation made when resolving equation (5). This fact does not affect the quality of the rotated image because pixels which are ignored are those of the image border.

VI. ARCHITECTURE FOR FPGA IMPLEMENTATION

The implementation of this approach on an FPGA component is based on the use of ROMs and counting systems. For an image of size 256X256, The total required memory is evaluated to 1,30397 Mbits. A camera provides different signals which are a gray level signal and synchronization signals.

From the clock signals we can compute the column and

the row of the acquired pixel and then we compute the address where the characteristic parameter is memorized in a ROM (this first ROM represents the characteristic table). In order to determine the column, we propose to use a first counter that we call ''columns counter''. The clock of this counter is a signal combined from the process clock and other synchronization signals which are relative to the used camera. The columns counter has an output which indicates the end of a considered row. This output will be used as a clock signal to a second counter used as "rows counter". The outputs of the used counters will be used in order to compute the address where the DATA corresponding to the current pixel is memorized. Let R, C be respectively the row and the column, the address is calculated like following:

$$Ad = (R-1) * Nb_C + C - 1 \quad (12)$$

Where Nb_c represents the columns number. The output of the compute address bloc will be ready 4.5 ns after the clock rising edge, this justifies the introduction of a shift_clock bloc. This bloc ensures that data memorized in the ROM will not be reed before the computation of the exactly address. The data which represents the number of the correspondents of a considered pixel will be ready to be used in approximately 6 ns after the clock rising edge. In order to correctly use this data, we must add a bloc which allows to reproduce the original clock signal but in 6 ns later. According to the memorized characteristic parameter, three cases can be distinguished:

- The characteristic parameter is null: we wait the next acquisition;
- The characteristic parameter is equal to 1: the system will authorize reading data only from the ROM which represents the first separate table. This ROM is addressed by a counter which will be automatically incremented after the end of data reading;
- The characteristic number is equal to 2: the system will authorize reading from the two memories which represent the separate tables. The used counters for addressing these memories will be automatically incremented after the end of data reading.

The result can be memorized in a RAM for a later use as it can be displayed on a screen.

VII. Conclusion

In the real-time image processing applications, it is a vital to ensure that accurate measurements can be made directly from the acquired image. This can be guaranteed by the conversion from a software algorithm to one that can run in hardware in real-time. But this conversion presents some difficulties such as the inability to do offline processing and the need to minimize logic gates count.

The use of the precalculated tables can reduce the complexity of some image processing application especially when the input image is expressed as a function of the result image. Image rotation is one of these applications.

From the mathematical model, we memorize the inverse relationship giving correspondence between input and output pixels. This correspondence is in the geometric aspect.

The approach of precalculated tables can be generalized to others applications. For this, it is enough to correctly create these tables. The number, the size and natures of the used tables depend on the application. But the common point is the necessity of existence of a characteristic table which allows classifying input images.

The approach of the precalculated tables can easily be implemented on a reconfigurable component as an FPGA using only memories and counters. All calculus will be avoided.

References

[1] K.T.Gribbon, C.T. Johnston, and D.G. Bailey, "A Real-time FPGA Implementation of a Barrel Distortion Correction algorithm with Bilinear Interpolation," in I*mage and Vision Computing NZ*, Palmerston North, November 2003 , pp. 408–413.

[2] D.Phillips, "Image Processing in C, Second Edition", Electronic Edition April 2000;

[3] A. Jain, "Fundamentals of Digital Image Processing", Prentice-Hall, 1986, p 321.

[4] P. Mattson, D. Kim, Y. Kim "Generalized image warping using enhanced lookup tables", International Journal of Imaging Systems and Technology, Volume 9, Issue 6, Pages 475 - 483 Special Issue: Advanced Imaging Chip Architectures and Applications, 12 Dec 1998.

[5] G Wolberg, T. E. Bolt, "Separable image warping with spatial lookup tables", Computer Graphics, Volume 23, Issue3 (July 1989), pp 369 – 378.

[6] Z.Irki, M.Devy, P.Fillatreaud and J.L.Boizard, "An Approach for the Real-time Correction of Stereoscopic images", in ECMS2007 and Doctoral School (EDSYS,GEET); Librec, Czech Republic, May 2007, ISBN 978-80-7372-202-9.

[7] Z.Irki, M.Devy, K.Achour and M.S.Azzaz "The Real-time Correction of Stereoscopic Images: From The serial to a Parallel Treatment", in The INTELLIGENT SYSTEMS AND AUTOMATION: 1st Mediterranean Conference on Intelligent Systems and Automation (CISA 08), AIP Conf. Proc. June 12, 2008 -- Volume 1019, pp. 3-8.

[8] Editor Al Bovik, "Handbook of Image and Video Processing", a volume in The Academic Press Series in Communications, Networking, and Multimedia, Series Editor in chief J.D. Gibson, ISBN 0-12-119790-5.

[9] A. Zisserman," Notes on Geometric and Invariance in Vision", British Machine Vision Association and Society for Pattern Recognition, 1992, Chap. 2.aa

[10] B. Horn, "Robot Vision", MIT Press, 1986, pp 314 - 315.

[11] Operation manual of '' Digital Monochrome Quad Speed CMOS Scan Progressive Camera: CV-A33CL'', manual version 1.0.

[12] A.Naoulou: ''FPGA Based Architecture for Real Time Computation of the Census Transform and Correlation in Various Stereovision Contexts'', LAAS report December 2004.

[13] M.DEVY & al, "Stereovision Algorithm to be Executed at 100Hz on a FPGA Based Architecture" in Advanced in Theory and Applications of Stereo Vision, Intech 2011,

[14] Z.Irki and M.Devy, '' A Parallel Architecture for The real Time Correction of Stereoscopic Images'', in PWASET Volume 30 July 2008 ISSN 1307-6884,pp 960-966.

[15] B.Cherrad & al, "A software hardware mixed design for the FPGA implementation of a real time edge detection", SMC 2010, 10-13 Octobre 2010, Istanbul, Turquie.