Capítulo 9 Sistema de visión integrado en FPGA para el cálculo de la orientación de objetos usando momentos de inercia de segundo orden

Chapter 9 Vision system FPGA-integrated for object orientation calculation using second order moments of inertia

IBARRA-BONILLA, Mariana Natalia†*, ARAGÓN-MORALES, Jesús Ángel y SÁNCHEZ-TEXIS, Fernando

Instituto Tecnológico Superior de Atlixco. División de Ingeniería Mecatrónica.

ID 1er Autor: *Mariana Natalia, Ibarra-Bonilla* / **ORC ID:** 0000-0001-7123-9105**, CVU CONACYT ID:** 237756

ID 1er Coautor: *Jesús Ángel, Aragón-Morales*

ID 2do Coautor: *Fernando, Sánchez-Texis* / **ORC ID:** 0000-0002-1792-8855, **CVU CONACYT ID:** 95289

DOI: 10.35429/H.2019.2.115.125

M. Ibarra, J. Aragón y F. Sánchez

mariana.ibarra@itsatlixco.edu.mx

A. Marroquín, J. Olivares, P. Diaz y L. Cruz. (Dir.) La invención y las mujeres en Mexico. Handbooks-©ECORFAN-Mexico, Querétaro, 2019.

Resumen

Este capítulo presenta un sistema de visión capaz de calcular el ángulo de rotación de los objetos capturados por una cámara de video. Se propone un algoritmo basado en los principios de los momentos de inercia de segundo orden. La idea principal es incorporar este algoritmo en un sistema de visión con implementación en FPGA que corrija la orientación de circuitos integrados electrónicos, que son manipulados por una máquina *pick & place*, durante el proceso de fabricación de placas de circuito impreso. El sistema de visión funciona en un FPGA Spartan 6 y controla una cámara OV7670, una pantalla TFT, comunicación RS-232 con una PC y una memoria SDRAM. El algoritmo para calcular la rotación, usando Matlab, se ejecuta en la PC. Los resultados preliminares muestran en promedio una precisión del 99.998% en el cálculo del ángulo de rotación.

Visión, FPGA, Ángulo de rotación, Segundos momentos de inercia

Abstract

This chapter presents a vision system capable of calculating the objets rotation angle captured by a video-camera. An algorithm based on the principles of inertia second order moments is proposed. The main idea is to incorporate this algorithm into an FPGA-vision system that will correct the orientation of electronic integrated circuits, which are manipulated by a pick & place machine, during the printed circuit boards manufacturing process. The vision system performs on a FPGA Spartan-6 and controls an OV7670 camera, a TFT display screen, RS-232 communication with a PC and an SDRAM memory. The algorithm for rotation calculating, using Matlab, is executed on the PC. The preliminary results show in average a precision of 99.998% in the rotation angle calculation.

Vision, FPGA, Angle rotation, Second order moments

1. Introduction

In electronics, a printed circuit board (PCB) is a platform on which the components are mounted to provide electrical interconnections between them. A PCB is found in almost all electronic products. PCB manufacturing is very complicated, requiring large equipment investments and over 50 process steps (LaDou, 2006). One of these processes is the components assembly on the PCB. The miniaturization constant of the electronic components causes a greater PCBs manufacture with surface mount technology (SMT). SMT is the process where the components are welded directly on the plate surface. Currently, the SMT has replaced the through hole technique, as there are no holes to drill, and thus the components can be placed closer, and therefore a reduction in the plate size and better electromagnetic compatibility is generated, because the radiation space is reduced then a smaller number of radiated emissions is produced (Prasad, 2013).

The PCBs manufacture using SMT is complicated, because it requires the different processes execution and equipment large investments, for example a pick & place machine. A pick & place has the function of taking electronic components, such as resistors, capacitors and integrated circuits, and mounting them on the PCB in the position indicated by a computer. This work originates from the need of the Mexican company: INTESC Electrónica & Embebidos, dedicated to the development boards design and manufacture based on FPGA and microcontrollers using SMT (INTESC, 2019). INTESC engineers have developed a pick & place machine with the purpose of increasing their production, reducing assembly time and replacing the operator. However, this machine requires perfecting the integrated circuits positioning on the PCB. The problem arises when the integrated circuits are placed on feeder trays that do not provide an exact position and result in an offset in the components position, therefore, it is necessary to incorporate a position correction system and thus eliminate the offset.

As a solution to this problem, it has been proposed to incorporate a vision system that works out as feedback to the control loop and thus reduce the offset to the microns order. Because the SMT assembly process requires working at high speed and precision, the proposed vision system must be implemented on a computational platform capable of processing images with high performance, for this, the digital signal processors (DSP) are being widely used in high performance applications and a large amount data processing (Ganeswara, 2016), (Rao, et al., 2016).

The FPGA (Field Programmable Gate Array) represent a superior alternative to the DSP, since the FPGA does not have a defined hardware architecture and can be configured according to the user requirements using hardware description languages, such as VHDL, Verilog HDL, etc. (Monmasson, et al., 2011). In contrast, DSPs have a fixed architecture with memory, controller and instructions executed sequentially according to software.

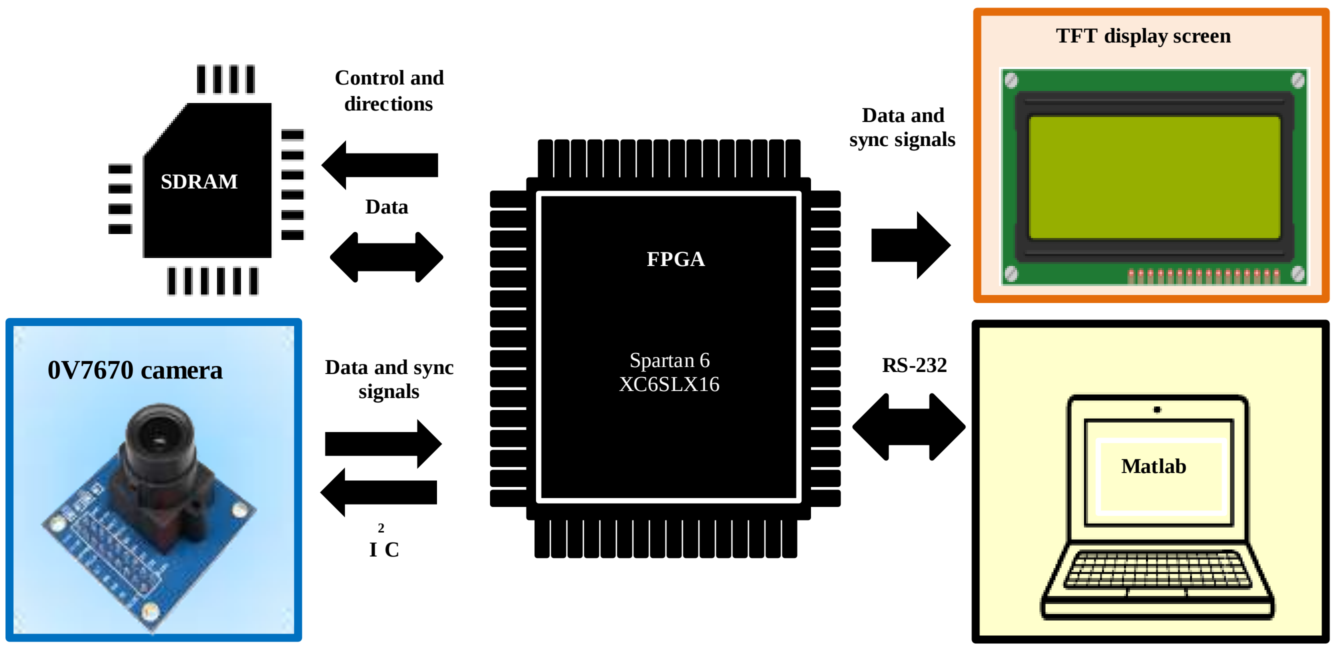
Image processing algorithms must process a large amount of data (pixels) at high speed, and for this type of applications, FPGAs are more suitable due to their parallel processing capacity (Patel, et al., 2017). The above is not possible with DSPs due to their data processing architecture, which is defined as 8-bits, 16-bits, 32-bits, etc. The present project proposes to develop the vision system for the pick & place type manufacturing equipment used an FPGA for high-speed image processing and angle rotation calculation of the components. The FPGA parallel processing capacity makes it more efficient than a DSP.

The chapter organization is as follows: Section 2 describes the FPGA vision system and the proposed methodology. Section 3 presents the description of the algorithm for the rotation angle calculation based on the second order moments of inertia. Section 4 describes the preliminary results. The conclusions and future work are presented in section 5.

2. FPGA vision system

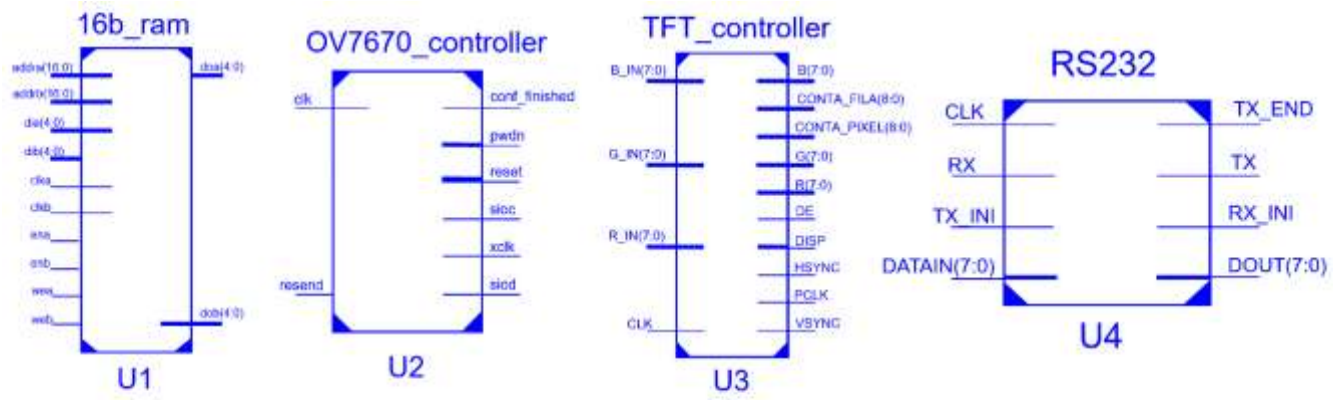
A proposed vision system block diagram is presented in Figure 9.1. The camera incorporates a CMOS VGA OV7670 image sensor, capable of working at a maximum of 30 frames per second at a resolution of 640x480 pixels. The Asserta development board (INTESC, 2019) containing a Spartan-6 XC6SLX16 FPGA and 64 Mb SDRAM memory is used. The FPGA controls the images acquisition and displays the video on a 4.3 inches TFT display screen of 480x272 pixels resolution. The vision system general operation is as follows: the FPGA receives and processes the data from the OV7670 camera, each received frame is saved into the SDRAM memory and sent to the TFT display screen. Additionally, a PC is communicated with the FPGA through the RS-232 serial protocol communication, thus if the FPGA receives an instruction can send the captured image to the PC. For the vision system, the FPGA incorporates 4 IP-cores or controller blocks to operate the OV7670 camera, RS-232 communication, TFT screen and SDRAM memory. The blocks programming was carried out in VHDL language using the Xilinx ISE-Design Suite software. All blocks are interconnected within a general entity for physical implementation in the FPGA Spartan-6. The blocks implemented in the FPGA are presented in Figure 9.2.

Figure 9.1 General block scheme of the proposed system



Reference source: Author

Figure 9.2 VHDL hardware modules

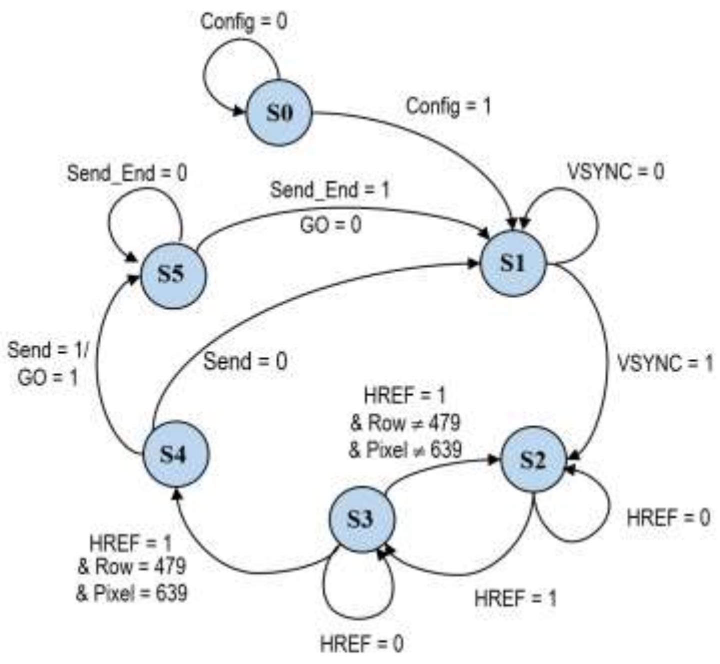


Reference source: Author

2.1. Camera control block

The video-camera control block consists of the 6-state machine presented in Figure 9.3. Each state is responsible for performing a specific task in the camera. The first state, called S0, performs an initial configuration on the camera's internal registers. The appropriate configuration values are loaded by means of an I2C serial communication port; which incorporates the camera for this purpose and that was also included in the camera control block. The S0 state also checks if a correct camera configuration has been made, and the machine does not change to the next state, if the camera does not respond with a positive confirmation. The next state, called S1, monitors whether the camera is ready to start the image frame transmission. The monitoring is done through the camera synchronization signal, VSYNC, which indicates when a new image starts. The third state, called S2, captures the bytes coming from the camera and groups them in a temporary buffer with the capacity to store a complete line of 640 pixels. The pixels are stored in this buffer in an RGB-565 format. Each time a pixel is captured, the machine moves on the next state S3. In state S3, the controller monitors whether the current image line capture has already been completed. This is done by checking the HREF synchronization signal from the camera. If a complete line has been captured, the block goes to state S4, which is responsible for saving the pixels in the SDRAM and goes to the next state S5. In this state, the machine monitors whether a new image line will be sent by the camera or if a new image will be sent by the camera. If a line continues to be sent, the machine returns from state S5 to S2, however, if a new image will be sent, the control will jump to state E1 to begin a new capture cycle.

Figure 9.3 Camera control state machine



Reference source: Author

2.2. SDRAM control block

This control block is responsible for storing orderly the pixels to the SDRAM memory. It also performs a processing of each pixel in the image by threshold binarization. Thus, the original information and the binarized information of the image is stored. This process is performed every time the camera control block sends the order to store an image line. In this way, the waiting time between each line capture is used to store the current line. In addition to the above, this block can read an image stored in the SDRAM and send it to the TFT display screen control block or to the RS-232 communication block. This process is performed between each waiting time for capturing a new image from the camera. Then it can be seen that the block functions as a slave system, which awaits the camera control block order or the RS-232 block order. In addition to these two processes, it autonomously transfers the information from the SDRAM to the TFT display screen control for image display. However, if a transfer order, from the RS-232 control, is received, the block interrupts the image transmission to the screen and redirects it to the RS-232 block; providing priority of the transfer to a user computer.

2.3. Bloque de control de RS-232

This block allows to send an image stored in the SDRAM to a computer for processing. The user can select the image type that is transferred between the original image and the binarized image. To perform this task, the block accepts two 8-bits simple commands; the reception of a number 2 indicates the original image transfer and a number 3 indicates the binarized image transfer. The information is sent using a standard format: 1 start bit, 8 data bits and 1 stop bit. The block also verifies that all pixels in the image are sent by counting them. The transfer rate reached by this block is 115200 baud.

2.4. TFT display screen control block

The principal function is to generate the TFT display screen control signals. The screen works under a VGA scheme. That is, the controller must generate the vertical and horizontal synchronization signals, VSYNC and HSYNC, to indicate to the screen when it will start the new image deployment. Pixel data are sent in a 24-bit configuration (8-bits for each color). The block uses a PCLK clock signal to synchronize the data and display of each image line. The image refresh rate is set at 15 frames per second. In relation to the images display and screen control, this block is autonomous, however, to obtain the pixel information the block depends on the SDRAM memory control block. This is due the SDRAM block sends the information when it is available in the memory. However, when it is required to transfer the image to a computer, the SDRAM control block interrupts the information normal sending to this block and instead sends black pixels. For this reason, when an image is being transferred to the user, the display of the image on the screen is interrupted and turns black.

2.5. Image processing

The processing that the FPGA executes on the images consists of a binarization. Applying the rotation calculation on binary images reduces the processing time, because in this type of image it is easier and faster to obtain the geometric and topological properties of the captured objects. To obtain binarization, the first step is to work with the image in grayscale, thus the pixel data are values between 0 and 255. The threshold value is fixed in the grayscale, this value is used to convert to a binary value 0 all pixels whose gray level is lower than that threshold, and a 1 for all those pixels that exceed the threshold. It is worth mentioning that to apply a fixed threshold implies controlling the environment parameters in which the image will be captured, since any variation in lighting can cause changes in the image gray levels that invalidate the fixed threshold. To execute the binarization process in VHDL code, the matrix of the green color plane, of the RGB format, is acquired. Since being a 565 format, it is the plane that has the most information. The pseudocode present in Table 9.1 is executed in the FPGA.

Table 9.1 Binarization process using a 75-fixed threshold

```
Algorithm. Fixed Binarization.
im : grayscale image of size (rows, columns)
imB : binary image
threshold = 75
  Initialize imB = 0
  While is moving im(x,y)   x = 1… rows,  y = 1… columns
    If im(x, y) > threshold
      imB = 1
    End
  End
  Return imB
```

Reference source: Author

3. Rotation angle calculating algorithm

In this work, the orientation is determined by calculating the axis, with respect to the minimum moment of inertia (De la Fuente and Trespaderne, 2012). The minimum axis of inertia of a region will be the line $Ax + By + C = 0$, such that the sum of the squared distances between the object pixels and the line is minimal (Hibbeler, 2004). As presented in Figure 9.4, the vector (A, B) = (cos θ, sin θ) is a perpendicular vector to the line, being θ the angle that forms the perpendicular from the origin to the line with the axis x. Thus, the angle θ is easily obtained after the second order moments calculation, I_{xx}, I_{yy} and I_{xy}. The axes of inertia pass through the object center of gravity (x_g, y_g). According to De la Funete and Trespaderne, and based on the equations applied in vector mechanics (Hibbeler, 2004), the three second order moments of inertia of a region I_{xx}, I_{yy} and I_{xy} are expressed:

$$I_{xx} = \sum_{x=1}^{r}\sum_{j=1}^{c}(x - x_g)^2 \cdot B(x, y) = \sum_{i=1}^{N}(x_i - x_g)^2 \quad (1)$$

$$I_{xy} = \sum_{x=1}^{r}\sum_{j=1}^{c}(x - x_g) \cdot (y - y_g) \cdot B(x, y) = \sum_{i=1}^{N}(x_i - x_g) \cdot (y_i - y_g) \quad (2)$$

$$I_{yy} = \sum_{x=1}^{r}\sum_{j=1}^{c}(y - x_g)^2 \cdot B(x, y) = \sum_{i=1}^{N}(y_i - x_g)^2 \quad (3)$$

The sum of all squared distances $\sum_{i=1}^{N} d_i{}^2$ is given by:

$$\sum_{i=1}^{N} d_i{}^2 = cos^2\theta \cdot I_{xx} + sin^2\theta \cdot I_{yy} + 2cos\theta sin\theta \cdot I_{xy} \quad (4)$$

To determine the line that minimizes the sum of the squared distances, equation (4) is derived with respect to the parameter θ, and equals 0, which results:

$$-2cos\theta sin\theta \cdot I_{xx} + 2cos\theta sin\theta \cdot I_{yy} + 2(cos^2\theta - sin^2\theta) \cdot I_{xy} = 0 \quad (5)$$

By trigonometry, it is known that $(cos^2\theta - sin^2\theta) = cos(2\theta)$ and $2cos\theta sin\theta = sin(2\theta)$, so that equation (5) is rewritten:

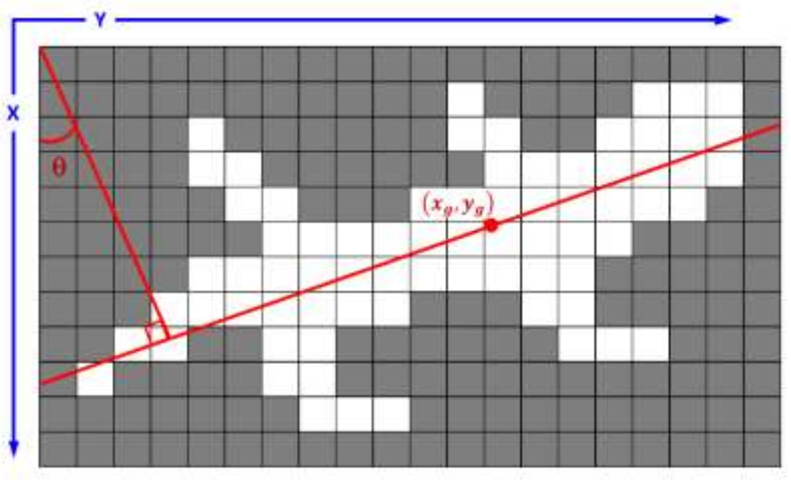
$$sin(2\theta) \cdot (I_{yy} - I_{xx}) + 2cos(2\theta) \cdot I_{xy} = 0 \quad (6)$$

Finally, the angle θ is:

$$\theta = 0.5 \operatorname{arc} \tan \frac{2I_{xy}}{I_{xx} - I_{yy}} \quad (7)$$

The angle of rotation of the object captured by the camera is obtained applying the equation (7), the atan2 function is used to avoid indeterminacies.

Figure 9.4 Representation of the object orientation on an image



Reference source: (De la Fuente and Trespaderne, 2012)

In order to calculate the moments of inertia it is necessary to extract some characteristics of the image: area and center of gravity of the detected object. The area of a region on the image is given by the number of pixels that constitute it, so it is given by the expression:

$$Area = \sum_{x=1}^{rows} \sum_{y=1}^{columns} B(x,y) \tag{8}$$

The area is also called zero order moment. The center of gravity is obtained from the first order moments divided by the area.

$$x_g = \frac{\sum_{x=1}^{r} \sum_{y=1}^{c} x \cdot B(x,y)}{Area} \tag{9}$$

$$y_g = \frac{\sum_{x=1}^{r} \sum_{y=1}^{c} y \cdot B(x,y)}{Area} \tag{10}$$

The algorithms for obtaining the center of gravity and the moments of inertia calculation are presented in Tables 9.2 and 9.3.

Table 9.2 Obtaining the object center of gravity on the image

```
Algorithm. Center of gravity
imB: binary image (0=Background, 1=Objet) of size (rows, columns)
area = 0
Ix = 0
Iy = 0
  While is moving imB(x,y)   x = 1… rows,  y = 1… columns
     If imB(x, y) ≠ 0
        area = area + 1
        Ix = Ix + y
        Iy = Iy + x
     End
  End
Cx = Ix/area
Cy = Iy/area
Center = [Cx   Cy]
```

Reference source: Author

Table 9.3 Obtaining the second order moments

```
Algorithm. Second order moments of inertia
imB: binary image (0=Background, 1=Objet) of size (rows, columns)
Center: vector with the coordinates of the center of gravity (row, column)
Ixx = 0
Iyy = 0
Ixy = 0
  While is moving imB(x,y)   x = 1… rows,  y = 1… columns
    If imB(x, y) $\neq$ 0
      Ixx = Ixx + (y – Center(1)) ^ 2
      Iyy = Iyy + (x – Center(2)) ^ 2
      Ixy = Ixy + (y – Center(1)) * (x – Center(2))
    End
  End
```

Reference source: Author

4. Results

In order to validate the algorithm for calculating the orientation and evaluate its performance, the code was written in the Matlab software. The Matlab preliminary use is due to the difficulty to execute floating point divisions in the VHDL language and to the calculation of trigonometric functions such as arctangent, however the intention is to translate the algorithm into this language, so that it works completely in FPGA. Matlab provides the image processing toolbox, which includes the regionprops command. This command is used to measure the properties of a region in an image, so it is possible to obtain the orientation of the image by directly applying this command. Different test images were generated in the Fireworks CS6 design software, which by rotating an image provides the angle of rotation.

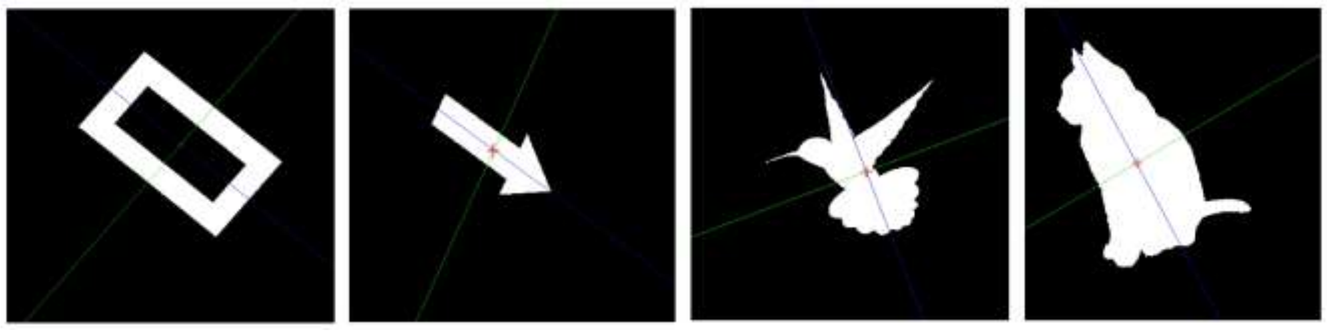
Therefore, the results of the algorithm are compared with the rotation angles obtained with Matlab and Fireworks. The integrated circuits encapsulates that a pick & place machine manipulates, to which this work is directed, generally have a square or rectangular geometric shape, however to evaluate the algorithm, more complex figures were also used, such as those presented in Figure 9.5. Table 9.4 presents the comparison of the results obtained with 10 different images. Considering the Fireworks measure as the reference, the second moment of inertia algorithm delivers an average accuracy on the x-axis of 99.966% and on the y-axis of 99.968%.

Table 9.4 Results comparison of the orientation angle using test images

Image	Fireworks $[\theta_x, \theta_y]$	Regionprops $[\theta_x, \theta_y]$	2nd order moments algorithm $[\theta_x, \theta_y]$
1	$[-80°, -10°]$	$[-80.004°, -9.996°]$	$[-80.004°, -9.995°]$
2	$[-40°, -50°]$	$[-39.995°, -50.009°]$	$[-39.993°, -50.005°]$
3	$[-60°, -30°]$	$[-60.002°, -29.997°]$	$[-60.002°, -29.998°]$
4	$[-70°, -20°]$	$[-70.005°, -19.994°]$	$[-70.005°, -19.995°]$
5	$[-80°, -10°]$	$[-80.005°, -9.996°]$	$[-80.004°, -9.996°]$
6	$[90°, 0°]$	$[90°, 0°]$	$[90°, 0°]$
7	$[10°, 80°]$	$[9.986°, 80.013°]$	$[9.986°, 80.013°]$
8	$[30°, 60°]$	$[29.995°, 60.004°]$	$[9.991°, 60.009°]$
9	$[50°, 40°]$	$[50.013°, 39.986°]$	$[50.014°, 39.986°]$
10	$[70°, 20°]$	$[70.005°, 19.994°]$	$[70.005°, 19.995°]$

Reference source: Author

Figure 9.5 Examples of test images generated in Fireworks

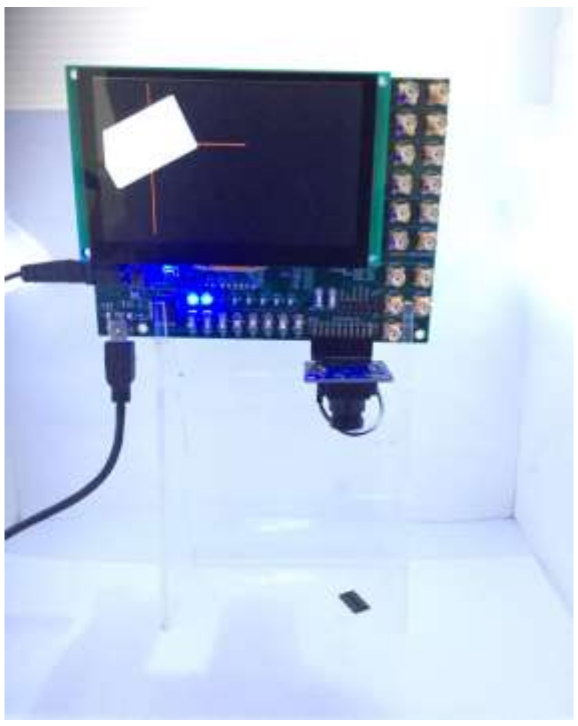


Reference source: Author

To execute the tests with the images captured by the OV7670 camera, the physical assembly of the Figure 9.1 scheme was performed. The camera and the TFT display screen are attached to the Asserta-board, which contains the FPGA Spartan-6, as observed in the Figure 9.6. Because the image binarization process uses the fixed threshold method, it is necessary to keep the lighting conditions controlled, since variations in the scene lighting can cause changes in the image gray levels that invalidates the threshold set. For this reason, a three LED array is incorporated to keep the lighting stable, which is displayed in Figure 9.7.

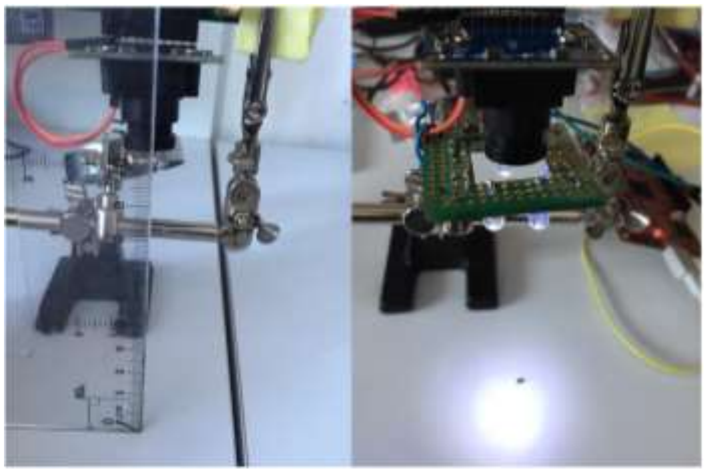
The final tests consisted of the images capture by the FPGA, which were sent by RS232 to a PC, where the algorithm of the second moments was executed in Matlab software. Until now, the captures are made in a controlled environment, because the integrated circuit is placed on a white surface and 10 cm away from the camera. These integrated circuits are of SMT technology, such as those manipulated by the pick & place machine. Figure 9.8 presents the integrated circuits 4-types encapsulation and Table 9.5 the results of these images. For each type of encapsulation, 5 images with different rotations were generated, completing 20 test images. By checking the 20 images in the algorithm and comparing it with the results obtained from the Matlab regionprops command, an average accuracy of 99.998% was obtained. This result is important because many industrial applications require to know the orientation of an object. For example, the robot grip requires the rotation of the object to close properly and thus succeed in capturing (De la Fuente and Trespaderne, 2012). Thus, our algorithm can expand its applications in the industry.

Figure 9.6 Physical assembly of the system



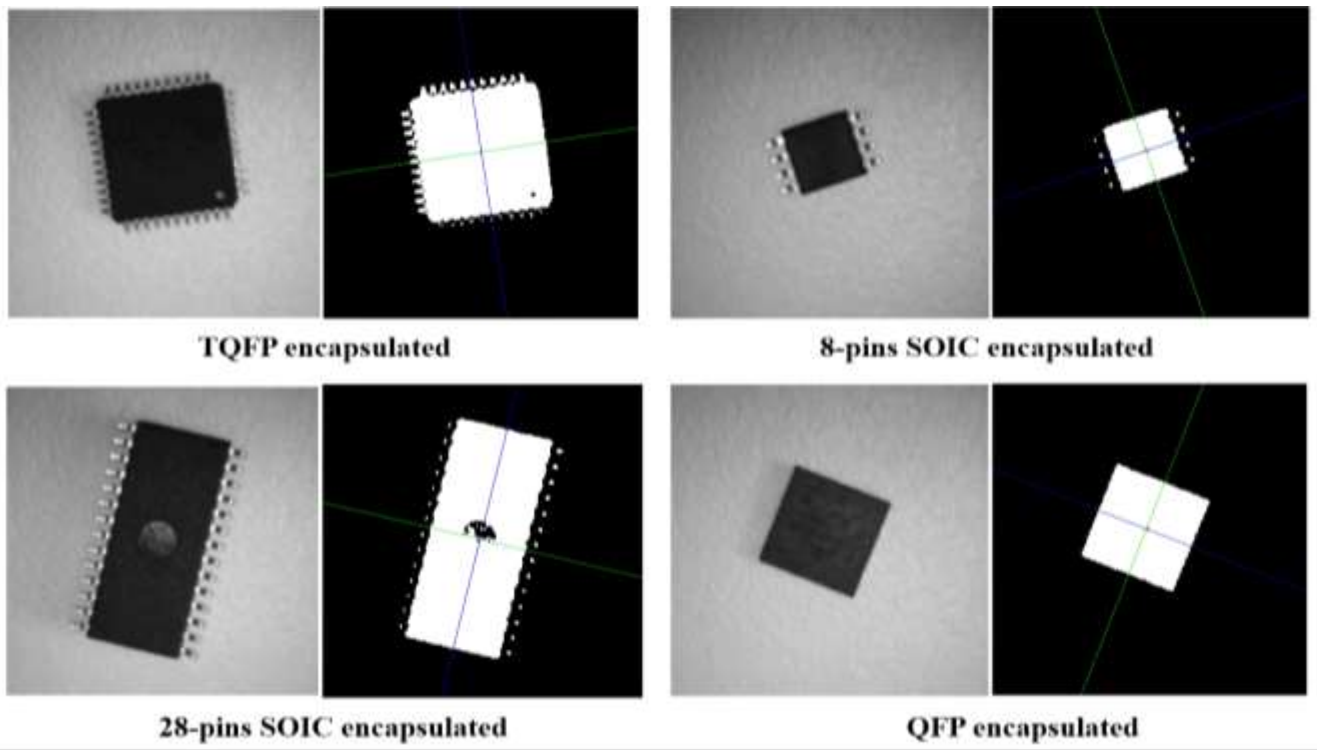
Reference source: Author

Figure 9.7 LED assembly for lighting control



Reference source: Author

Figure 9.8 Images captured from the OVG camera by FPGA and binarization results



Reference source: Author

Table 9.5 Rotation angle results

Encapsulated	Regionprops $[\theta_x, \theta_y]$	2nd order moments algorithm $[\theta_x, \theta_y]$	% Precisión $[x, y]$
TQFP	$[-80.004°, -9.997°]$	$[-80.004°, -9.996°]$	$[100\%, 99.99\%]$
SOIC-8p	$[19.079°, 70.921°]$	$[19.078°, 70.921°]$	$[99.99\%, 100\%]$
SOIC-28p	$[76.546°, 13.455°]$	$[76.544°, 13.456°]$	$[99.73\%, 99.99\%]$
QFP	$[-21.381°, -68.619°]$	$[-21.381°, -68.619°]$	$[100\%, 100\%]$

Reference source: Author

5. Acknowledgments

The first author appreciates the financial support of the Tecnológico Nacional de México for the realization of the project, under the program of Support for Scientific Research and Technological Development of the educational programs of the Decentralized Technological Institutes 2018.

6. Conclusions

A vision system with video capture in FPGA and rotation angle calculation in Matlab was presented. The VHDL-code consisted of the OV7670 video camera controller, a TFT display screen, SDRAM memory and RS-232 communication with a PC. On the PC an algorithm is executed in Matlab to calculate the angle of rotation of the object captured by the camera. This algorithm is based on the principles of the second order moments of static inertia. According to preliminary tests, performed on 20 integrated circuits images, this algorithm has an average accuracy of 99.998%, which is expected to solve the problem of surface mounting that presents a pick & place machine developed by the mexican company INTESC, when assembling components to their development boards. The work in progress consists in translating the Matlab algorithm code into VHDL language, so that the whole system is in a hardware implementation and can be installed in the pick & place machine. Finally, it is suggested that to apply this algorithm in the industry it will be necessary to increase the camera resolution by a minimum of 5 megapixels and thus achieve better vision results, as well as work in a white light environment.

7. References

De la Fuente López, E., & Trespaderne, F. M. (2012). Visión artificial industrial: Procesamiento de imágenes para inspección automática y robótica. Universidad de Valladolid, Secretariado de Publicaciones e Intercambio Editorial.

Ganeswara-Rao M. V., Panakala, R. K. and Mallikarjuna-Prasad A. "Image Processing using FPGAs: a Framework". IJCTA International Science Press, vol. 9, no. 19, pp. 9191-9197, 2016.

Hibbeler, R. C. (2013). Engineering Mechanics: Statics. Pearson Education.

INTESC Electrónica & Embebidos (01-04-2019). Main page. México: Intesc. https://www.intesc.mx/.

LaDou, J. (2006). Printed circuit board industry. International journal of hygiene and environmental health, 209(3), 211-219.

Monmasson, E., Idkhajine, L., Cirstea, M. N., Bahri, I., Tisan, A., & Naouar, M. W. (2011). FPGAs in industrial control applications. IEEE Transactions on Industrial informatics, 7(2), 224-243.

Patel, D., Parmar, R., Desai, A., & Sheth, S. (2017, January). Gesture recognition using FPGA and OV7670 camera. In 2017 International Conference on Inventive Systems and Control (ICISC) (pp. 1-4). IEEE.

Prasad, R. (2013). Surface mount technology: principles and practice. Springer Science & Business Media.

Rao, M. G., Kumar, P. R., & Prasad, A. M. (2016, January). Implementation of real time image processing system with FPGA and DSP. In 2016 International Conference on Microelectronics, Computing and Communications (MicroCom) (pp. 1-4). IEEE.