A tool for errors detection in printed circuit boards production

A. de Luca Pennacchia*, L. G. de la Fraga, U. Martínez Hernández

Computer Science Department, CINVESTAV-IPN.
Av. IPN 2508, 07360 México, D.F.
*dlap@cs.cinvestav.mx

ABSTRACT
The progressive implementation of software functions in Integrated Circuits (ICs) has considerably increased the number of transistors and pin connections of ICs. For that reason, Printed Circuit Boards (PCBs) are fabricated with the Surface Mount Technology (SMT) nowadays and IC mounting on PCB is a crucial process that requires high precision. An Automatic Mechanical Montage (AMM) system is used to mount ICs on the sockets using a couple of reference points for every IC in order to find the correct positions for mounting the IC. Due to some factors in the process of PCB development, there are differences between designed and manufactured PCBs, which could generate delays in their production. In this work, a software tool which allows to work with digital images of PCBs is described. This tool finds the differences generated in PCB development, especially the differences in IC reference points using Digital Image Processing (DIP) techniques.

RESUMEN
La realización progresiva de funciones de *software* en los Circuitos Integrados (CI) ha incrementado considerablemente el número de transistores así como de pines de conexión de los CI. Por esta razón, hoy en día las Tarjetas de Circuito Impreso (TCI) son fabricadas con la Tecnología de Montaje Superficial y el montaje de los CI sobre la TCI es un proceso crucial que requiere una alta precisión. Un sistema de Montaje Mecánico Automático (MMA) se usa para montar los CI sobre los zócalos usando unos puntos de referencia para cada CI de modo que encuentra los posiciones correctas para el montaje del CI. Dados algunos factores en el proceso de desarrollos de las TCI, existen diferencias entre las TCI diseñadas y las manufacturadas, las cuales generan retardos en su producción. En este trabajo se describe una herramienta de software que permite trabajar con imágenes digitales de una TCI. Esta herramienta encuentra las diferencias generadas en el desarrollo de TCI, especialmente las diferencias en los puntos de referencia para CI usando técnicas de Procesamiento Digital de Imágenes.

Keywords: Printed Circuit Board, Surface Mount Technology, Digital Image Processing, Software Development.

1. Introduction

Nowadays, there are two types of electronic components mounting in Printed Circuit Boards (PCBs): Through Hole Technology (THT) and Surface Mount Technology (SMT). The SMT has a lot of advantages, for example, reduced dimensions, increased functionality, components can be mounted in both sides on a same PCB, and density of mounted electronic components is high.

The electronic components manufactured with SMT have reduced dimensions, for example, pin separation is in range of 0.6 mm to 1.27 mm, which requires high precision in PCB development. Manufactured PCBs and designed PCBs can have differences which are usually generated by the mechanical degradation in the system calibration for PCB production. Those differences generate errors in IC sockets, changing their positions which produce delays in IC mounting by the Automatic Mechanical Montage (AMM) system.

Every IC socket has a pair of reference points which are used by the AMM to mount an IC. And to mount an IC, the AMM is programmed to find the reference points inside determined regions of a PCB first, and then the IC is correctly mounted.

In this work, a software system which allows to recognize the real positions of PCB's reference points, to analyze them and to calculate possible differences between the correct PCB is presented. Also, the knowledge of the positions of reference points can be used to reprogram the AMM system and by this way to eliminate errors and to reduce the IC mounting time.

The developed system works with Digital Image Processing (DIP) techniques. Then, it is necessary to have images of PCBs through scanning different PCBs. The system uses DIP techniques like color model transform, object extraction, pattern recognition and object reconstruction. The system has a Graphic User Interface (GUI) developed with C++ language and Qt library [1].

Figure 1 shows an example of reference points in sockets on a PCB. As it was mentioned before, reference points recognition and characteristics extraction of those points are the main objectives of this work. This paper is organized as follows: in Sec. 2, the description of the problem is given; in Sec. 3, the development tool is described; and finally, in Sec. 4, conclusions are drawn.

2. System Description

The system developed in software works analyzing reference points on images of a PBC and comparing them with the original PCB. Figure 2 shows a general diagram of the system; in this diagram we can see that the system receives an image as input and it is analyzed using color models to get more information about the image.

Once the input image has been analyzed, extraction and object recognition is developed using a vector of characteristics of the object. If the analyzed object has the required conditions, then this object advances to the reconstruction step. The reconstruction step improves positions calculation by fitting a circle. Finally, the object centroid is calculated and saved. This process is developed for every reference point on a PCB.

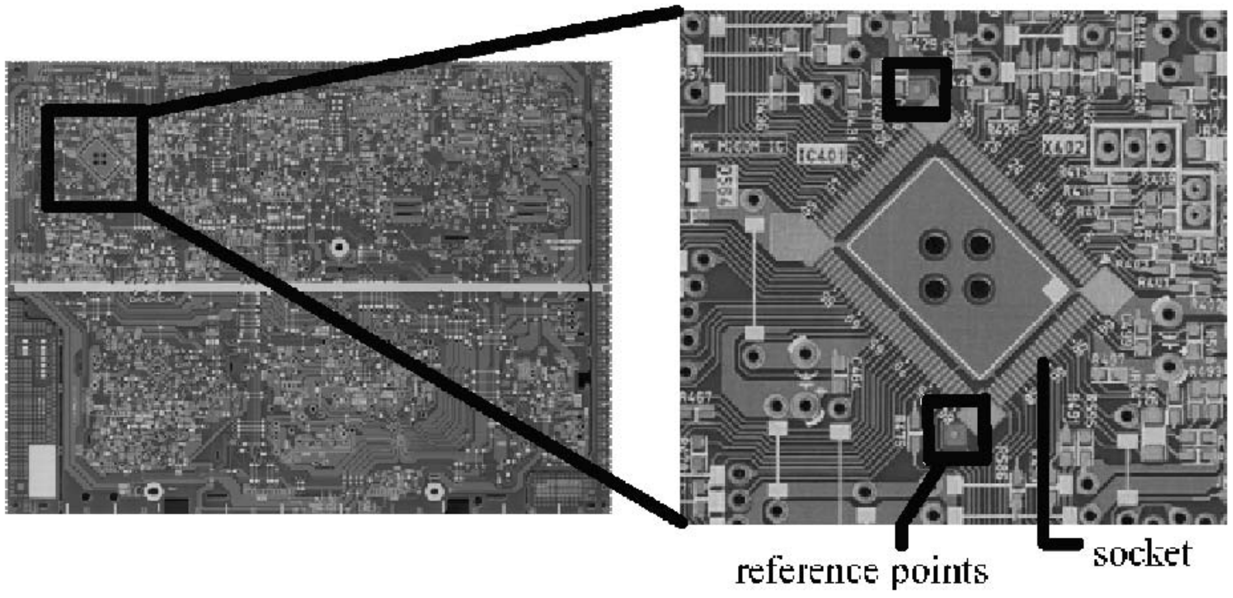


Figure 1. Reference points on a PCB

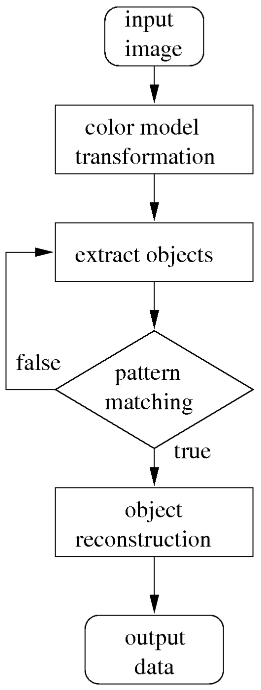


Figure 2. General system diagram

The operations developed by the system will be explained next.

2.1 Color model transformation

The DIP works with both gray scale and color images. Gray scale images have one byte of information per pixel while color images have three bytes per pixel, which give us more information to analyze them. Choosing to work between gray scale or color images depends on the type of objects to be recognized. In this case, we opted to work with color images because of their good characteristics.

A color model is the specification of a three-dimensional coordinated system. Most of the color models are oriented to hardware. Some examples of color models are

- RGB (Red, Green, Blue): used in video cameras and color TV
- CMY (Cyan, Magenta, Yellow): used in printers
- HSI (Hue, Saturation, Intensity): used in image manipulation

The HSI model works in the same way humans perceive color [2]. The HSI model description is as follows:

- Hue: it describes the pure color, for example red, green, blue, yellow, orange
- Saturation: it describes the level in which the pure color is mixed with white light
- Intensity: it describes the level of brightness

In this work, the HSI model is used because it works as humans perceive color, also because we can work with three bytes (Hue, Saturation, Intensity) independently. This is especially important because we can analyze the images taking into account only the pure color of each pixel. The HSI model is obtained from the RGB model using the following expressions:

$$r = \frac{R}{R+G+B},\ s = \frac{G}{R+G+B},\ b = \frac{B}{R+G+B}$$

$$h = \begin{cases} \cos^{-1}\left\{ \dfrac{0.5[(r-g)+(r-b)]}{\left[(r-g)^2+(r-b)(g-b)\right]^{1/2}} \right\} & if\ b \le g,\ h \in [0,\pi] \\ 2\pi - \cos^{-1}\left\{ \dfrac{0.5[(r-g)+(r-b)]}{\left[(r-g)^2+(r-b)(g-b)\right]^{1/2}} \right\} & if\ b > g,\ h \in [\pi,2\pi] \end{cases}$$

$$s = 1 - 3 \cdot min(r,g,b), \qquad s \in [0,1]$$

$$t = \frac{R+G+B}{3 \cdot 255}, \qquad t \in [0,1]$$

Figure 3 shows images with different level of brightness. Objects on an image are extracted using only the Hue parameter as only pure color like red, green, blue, yellow, orange. The extracted objects are in binary format (blank and white), because this is a requirement in the next extraction and recognition steps.

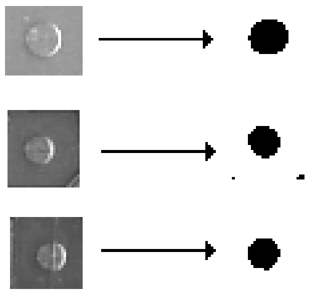


Figure 3. Images using the HSI model

After color model transformation, the next step is the extraction of characteristics of the objects.

2.2 Extraction and object recognition

The objective in the extraction process is to separate different objects inside the input image. The input for the extraction process is a binary image with *N* objects and the output is a set of *N* binary images having every binary image one object only. Thus, the vector of characteristics is extracted from every image which is used to develop the object recognition process.

For extraction and object recognition, the SCimagen library [3] is used, which has a lot of functions for DIP. Figure 4 shows the object extraction step.

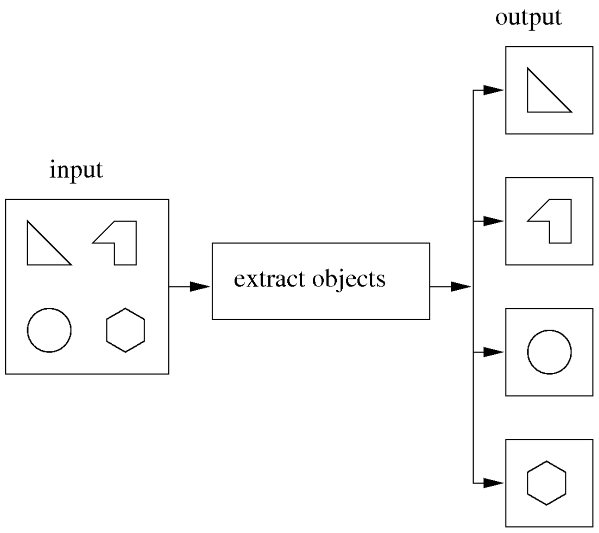


Figure 4. Diagram of the extraction step

Once objects have been extracted, the next step is to get the characteristic vector to develop the object recognition process. Among the huge number of possible methods that can be used for object recognition [4], we chose moment invariants. These moments are invariants to geometrical transformations, thus recognition can be achieved even if objects are rotated, translated and/or scaled.

Geometrical moments $m_{p,q}$ of a gray scale image *f(x,y)* are defined as follows:

$$m_{p,q} = \sum_{x=1}^{N}\sum_{y=1}^{M} x^p y^q f(x,y) \qquad (1)$$

where, *p* and *q* are indexes of a digital image of size *NxM*. The central moments are calculated as

$$m_{p,q} = \sum\sum (x-\bar{x})^p (y-\bar{y})^q f(x,y) \qquad (2)$$

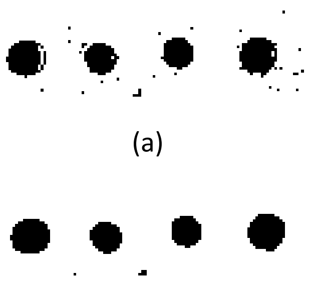
And an object centroid is calculated as

$$x = \frac{m_{1,0}}{m_{0,0}} \qquad (3)$$

$$y = \frac{m_{0,1}}{m_{0,0}} \qquad (4)$$

Applying (1), (2), (3) and (4) we can calculate object centroids, thus one of the objectives of this work is solved. There are seven moment invariants that can give us information about the object. In this work, the first moment invariant and the object's area are used to form the characteristic vector. Because the area is not an invariant feature, we define a range of values to work with it, this is because the size of objects could change.

We developed a lot of recognition examples using images of 300 dpi, 600 dpi and 1200 dpi. In some cases, the object extraction could be affected by factors such as image resolution, movement of the camera, and noise. Figure 5(a) shows an example where extracted circles have been affected by some of the factors mentioned above.


(a)

(b)

Figure 5: Morphological operations: (a) affected circles with noise; (b) morphological operations applied to circles

Quality of the images can be improved by applying morphological operations such as erosion, dilation, opening and closing. Morphological operations work over one pixel taking into account neighbor pixels. Erosion consists in the reduction of size of the objects using a second object called structural element of size *3x3* pixels, normally erosion is used for noise reduction (isolated pixels on the image) and objects separation. The dilation process increases the object size by using also the structural element. Figure 5(b) shows extracted circles after the application of morphological operations. We can observe that the image quality has been improved. Also, we can apply some type of object reconstruction to improve centroids calculation results; in this case, it is necessary to use a circle reconstruction method.

2.3 Circle reconstruction

The acquisition of the images by an electronic device can possibly have a low quality produced by noise in the sensor, low resolution or fuzzy images due to movements of the camera.

There are many algorithms used in object reconstruction to improve image quality. In [5] different algorithms for circle reconstruction are proposed. We selected the algorithm for reconstruction called Modified Least-Squares, which is stable and a closed solution. This algorithm works with pixels positions of the circle perimeter. Thus, we can reconstruct the affected circle and improve the centroid calculation.

The centroid is calculated as follows:

$$a_M = \frac{DC - BE}{AC - B^2} \qquad (5)$$

$$b_M = \frac{AE - BD}{AC - B^2} \qquad (6)$$

where:

$$A = n\sum_{i=1}^{n} x_i^2 - \left(\sum_{i=1}^{n} x_i\right)^2 \qquad (7)$$

$$B = n\sum_{i=1}^{n} x_i y_i - \left(\sum_{i=1}^{n} x_i\right)\left(\sum_{i=1}^{n} y_i\right) \qquad (8)$$

$$C = n\sum_{i=1}^{n} y_i^2 - \left(\sum_{i=1}^{n} y_i\right)^2 \qquad (9)$$

$$D = 0.5\left\{n\sum_{i=1}^{n} x_i y_i^2 - \left(\sum_{i=1}^{n} x_i\right)\left(\sum_{i=1}^{n} y_i^2\right) + n\sum_{i=1}^{n} x_i^3 - \left(\sum_{i=1}^{n} x_i\right)\left(\sum_{i=1}^{n} x_i^2\right)\right\} \qquad (10)$$

$$E = 0.5\left\{n\sum_{i=1}^{n} y_i x_i^2 - \left(\sum_{i=1}^{n} y_i\right)\left(\sum_{i=1}^{n} x_i^2\right) + n\sum_{i=1}^{n} y_i^3 - \left(\sum_{i=1}^{n} y_i\right)\left(\sum_{i=1}^{n} y_i^2\right)\right\} \qquad (11)$$

The centroid of the reconstructed circle is represented by a_M and b_M. A way to obtain the perimeter is applying an erosion technique to the original circle and then calculating the difference between the original and the eroded circle. An example of these perimeters is shown in Figure 6(a). Circles are more affected in the right side due to the light condition, for that reason, reconstruction takes into account only pixels of the left side of the circle. Figure 6(b) shows the left side of three perimeters used in the reconstruction step.

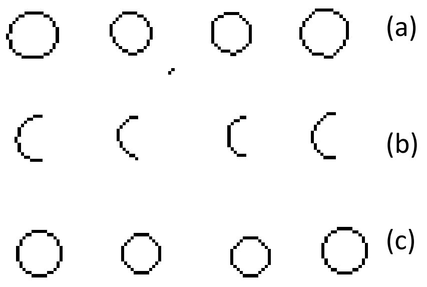


Figure 6: Circles reconstruction stops: (a) circles without reconstruction; (b) points used in the reconstruction; (c) reconstructed circles

Thus, using circle points for the reconstruction, we can obtain circles as the ones shown in Figure 6(c). These circles are used finally to calculate their centroids improving their localization.

3. Description of the designed tool

We developed the proposed tool with C++ language and Qt library [6,7]. In this section, the process to develop the tool and its functionality will be explained. In Figure 7, the designed Graphic User Interface (GUI) is shown.

The GUI components shown in Figure 7 are

- Working area: area used to analyze an image
- Menu
  - New project: to create a new project
  - Open: to open an image
  - Close: to close the current project
  - Settings: to set the PCB's characteristics.
  - Help
- Image analysis
  - Settings: PCB size and offset
  - IC model: model of an IC
  - Points
    - Center: centroid calculation
    - Gerber: centroids from the original design
    - Difference: difference among centroids
- Restart: restart analysis
- Next: save changes

To analyze one image, we need to create a new project, to define the project's name and path in order to save the results. This condition is shown in Figure 8.

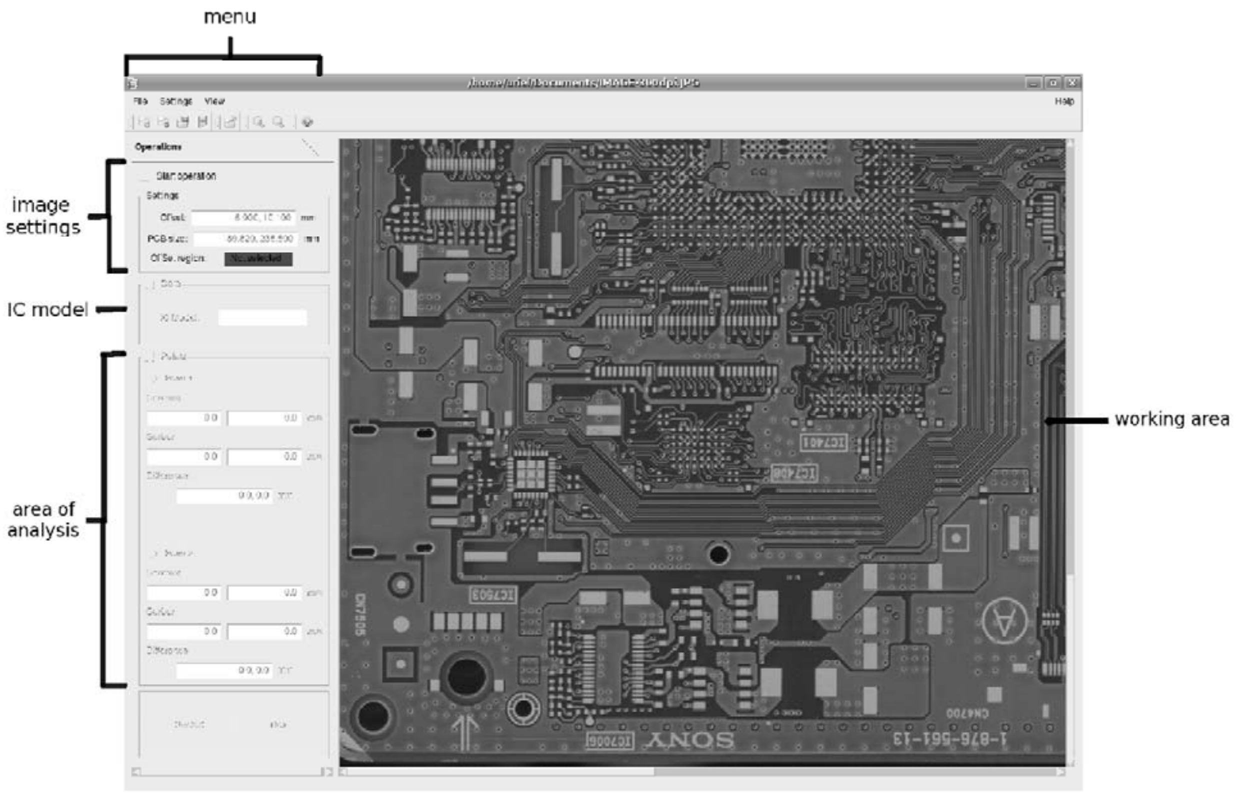
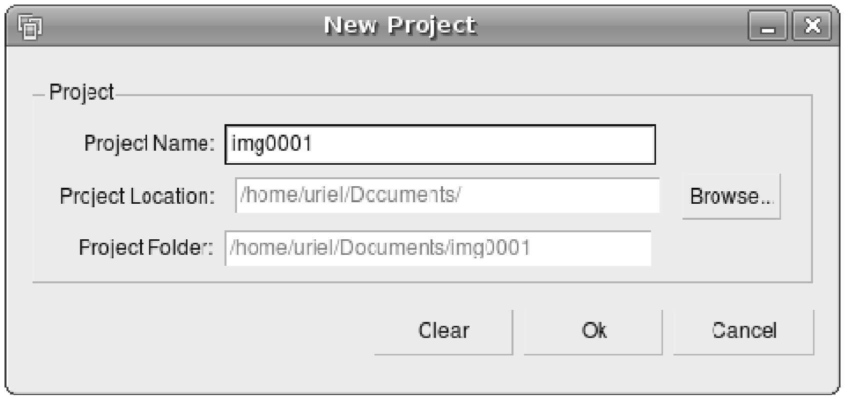


Figure 7. Description of the tool

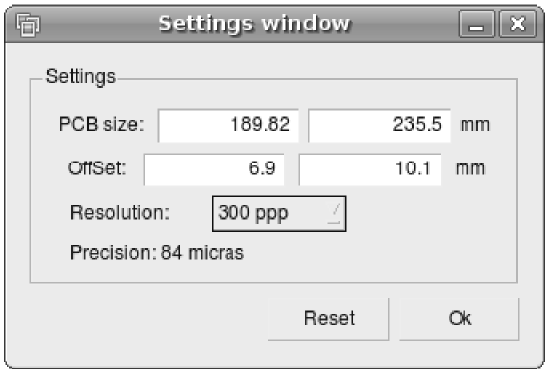


Project Name: img0001
Project Location: /home/uriel/Documents/
Project Folder: /home/uriel/Documents/img0001

Figure 8. New Project function

The next step is to open a PCB image and set its size and offset characteristics. The PCB has a point of reference in one corner which is used to calculate distances to centroids. The offset is taken from the original PCB design. This operation is shown in Figure 9.

Once the image has been opened, the analysis of reference points for every IC can be developed. Figure 10 shows an example of the selection of a region where the search of the circle and its centroid is performed.



Settings

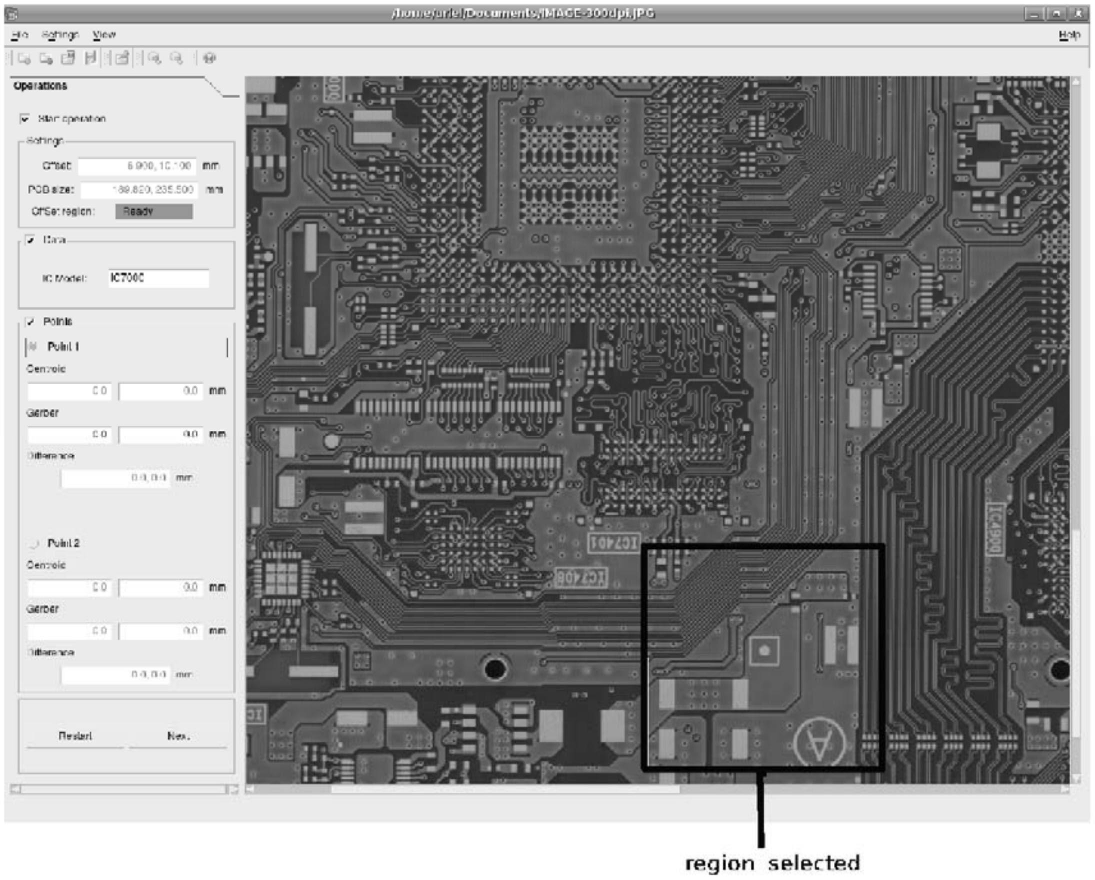
PCB size: 189.82 235.5 mm

OffSet: 6.9 10.1 mm

Resolution: 300 ppp

Precision: 84 micras

Figure 9. PCB characteristics



Start operation

Settings

Offset: 6.900, 10.100 mm

PCB size: 189.820, 235.500 mm

OffSet region: Ready

Data

IC Model: IC7000

Points

Point 1

Centroid 0.0 0.0 mm

Gerber 0.0 0.0 mm

Difference 0.0, 0.0 mm

Point 2

Centroid 0.0 0.0 mm

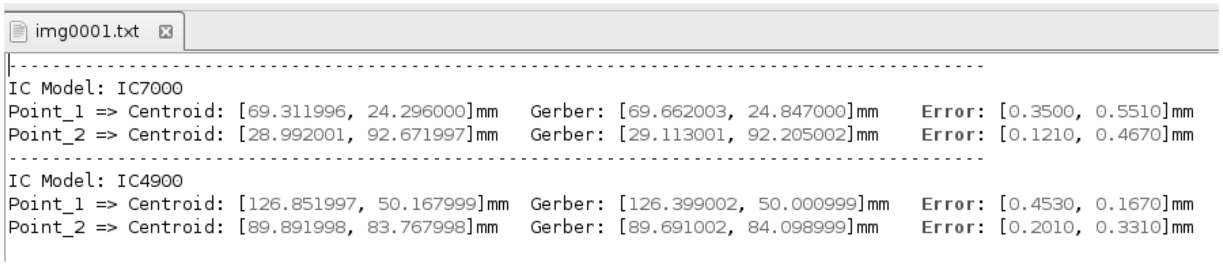
Gerber 0.0 0.0 mm

Difference 0.0, 0.0 mm

Figure 10. Selection of a region

The process is developed with every reference point of the PCB. The results are saved in a text file located in the defined path at the beginning of the project, this text file is created automatically when the project is created. Figure 11 shows an example of the results saved after the analysis of a PCB.

The system has been developed to work with images of 300 dpi, 600 dpi and 1200 dpi. Analyzing



img0001.txt

```
-------------------------------------------------------------------------------------
IC Model: IC7000
Point_1 => Centroid: [69.311996, 24.296000]mm   Gerber: [69.662003, 24.847000]mm    Error: [0.3500, 0.5510]mm
Point_2 => Centroid: [28.992001, 92.671997]mm   Gerber: [29.113001, 92.205002]mm    Error: [0.1210, 0.4670]mm
-------------------------------------------------------------------------------------
IC Model: IC4900
Point_1 => Centroid: [126.851997, 50.167999]mm  Gerber: [126.399002, 50.000999]mm   Error: [0.4530, 0.1670]mm
Point_2 => Centroid: [89.891998, 83.767998]mm   Gerber: [89.691002, 84.098999]mm    Error: [0.2010, 0.3310]mm
```

Figure 11. Results saved

images of 300 dpi is faster than analyzing images of 1200 dpi, but an image of 1200 dpi gives us better results, because we get an accuracy of 21 microns, while accuracy for an image of 300 dpi is of 84 microns. Then, the result depends on the input image resolution level.

4 Conclusions

A tool to analyze ICs location on PCBs has been developed. This tool is programmed in C++ using Qt library.

The tool allows us to analyze PCBs and to know the differences between produced PCBs and original PCBs. Nowadays, with this tool, the analysis can be applied to circles, but the tool can be improved to analyze more types of objects. The results obtained of the analysis can be calculated with different accuracy levels. This accuracy depends on the resolution of the input image. The tool works with resolutions of 300 dpi, 600 dpi and 1200 dpi with accuracies of 84, 42 and 21 microns, respectively. Thus, images with resolution of 1200 dpi give us better results but require more processing time. Input image formats can be BMP, JPEG, PNG and PNM. This type of tool can be used by a PCB producer to determine if her/his final product is within the established quality parameters. The incorporation of this tool to analyze PCBs does not produce delays on production lines because this tool has been designed to work out of line.

References

[1] Blanchette J & Summerfield M, C++ GUI Programming with Qt 3, Prentice Hall. (2004).

[2] Gonzalez R.C. & Woods R.E., Digital Image Processing, (1992), Springer.

[3] De la Fraga L.G., Cornejo Herrera J, Lara López A. & Landa Becerra R, Una biblioteca para procesamiento de imagen: Scimagen, VIII Conferencia de Ingeniería Eléctrica, 4, 5 y 6 de septiembre del 2002. CINVESTAV-IPN. Available at http://cs.cinvestav.mx/~fraga/Publicaciones/bibliotecapdi.pdf.gz

[4] Trier O.D., Jain A.K. & Tast T, Feature extraction methods for character recognition - a survey, Pattern Recognition, Vol. 29, No. 4, pp. 641-662. (1996).

[5] Umbach D. & Jones K.N., A Few Methods for Fitting Circles to Data, IEEE Transaction on Instrumentation and Measurement, Vol. 52, No. 6, Dec., 2003, pp. 1881-(1885).

[6] Eng E, Qt GUI Toolkit, Linux Journal, Nov, 1996, http://www.linuxjournal.com/article/201.

[7] Rempt B. & Laird C, Visual Development with Qt 3.0, Linux Journal, Jun, 2002, http://www.linuxjournal.com/article/4749.

Authors Biography

Adriano DE LUCA PENNACCHIA

He was born in Foggia (Italy) in 1935, graduated in Mathematics and Physics at Sto. Severo (Foggia) and received his doctorate degree in Nucleonics e Automazione in Milan (Italy). Dr. De Lucca has been member of the SNI since 1986. He worked for 10 years at the ININ, Mexico where he founded several research teams and participated actively in the group that won the National Prize of Applied Instrumentation in 1974. From 1982 to 1985, he worked at SGS Thompson in Milan. He obtained the USA patent 07/483915 in 1991: "Digital anode to determine the location of electrons in a given surface". He published a book: "Digital Systems" by the Metropolitan Autonomous University in 1992. Since 1995, he has been full professor at CINVESTAV, Computer Science Department, in Mexico City. He has published more than 92 technical articles.

Luis Gerardo DE LA FRAGA

He received the B.Sc. degree from the Instituto Tecnológico de Veracruz in 1992, the M.Sc. degree in electrical engineering from the Instituto Nacional de Astrofísica Optica y Electrónica in 1994, and the Ph.D. degree in informatics engineering from the Universidad Autónoma de Madrid, Spain, in 1998. From 1998 to 2000, he was assistant professor at the Universidad Autónoma del Estado de Morelos, Cuernavaca, Mexico. Since 2000, he has been research scientist at CINVESTAV in the Computer Science Department in Mexico City. His research interests include computer graphics, computer vision, image processing, deformable models, network security, and he is very enthusiastic of open-source software and GNU/Linux systems. He has authored more than twenty technical articles published in international conference proceedings and journals and has been guest speaker at many national conferences. Dr. De la Fraga has been member of IEEE and ACM since 2005.

Uriel MARTÍNEZ HERNÁNDEZ

He received his B.S. degree in communications and electronics engineering from the Escuela Superior de Ingeniería Mecánica y Eléctrica, Mexico City, in 2004. He obtained his M.Sc. degree in computer science from CINVESTAV, Computer Science Departament, in 2008. His interest working areas are software development, computer networks, computer graphics, image processing, digital and analog electronics, and automatic control using microprocessors.