An Algorithm for Detection of Calibration Points for Photogrammetric Camera Calibrations Independent of Illumination Variations

Otniel Portillo, Luciano Chirinos, Carlos Vázquez
División Ingeniería y Arquitectura
Instituto Tecnológico y de Estudios Superiores de Monterrey. Campus Toluca.
Eduardo Monroy Cárdenas No. 2000. Toluca, Estado de México. C.P. 50110.
México

Abstract: - Getting 3D information from 2D images depends on the appropriate camera calibration of the internal and external CCD (Charge Couple Device) camera parameters. The photogrammetric camera calibration technique needs a set of world coordinates and their corresponding calibration object image coordinates as an input. During this work we developed an algorithm to detect the image coordinates of a calibration object for different illumination conditions with subpixel accuracy. In other words, the proposed algorithm allows that any photogrammetric camera calibration technique that needs image coordinates of the calibration object will not be sensitive to illumination changes within a 200 to 6000 lux interval. The algorithm was tested in a vision system composed by a P.C., a robotic arm and a CCD camera. The system automatically controls the position of the camera within the work space and it captures and processes the images using Matlab®.

Key-Words: - Machine Vision, Image Processing and Applications.

1 Introduction

The camera calibration is the process of finding the internal geometry and optic characteristics of the device (internal parameters), as well as the position and orientation of the camera with respect to an external reference point (external parameters). The purpose of the calibration is to establish the relationships between the 3D coordinates of the external reference system and the corresponding 2D coordinates of the image. Once the relationships are established it is possible to get 3D information from 2D images and vice versa.

The camera calibration techniques are classified within two categories: photogrammetric calibration and auto-calibration. The camera calibration under photogrammetric technique is done using a calibration object whose geometry in the 3D space is well known [1], [2]. The calibration object consists mainly of two or three orthogonal planes. The photogrammetric techniques were the first to be developed and they have reached enough maturity [3]. For this reason the camera calibration can be done efficiently.

Some work has been done to determine the calibration points of the calibration objects with controlled illumination conditions: in [4] calibration points are detected using binarization and object labeling procedures, in [5] Hough Transform is used and in [6] calibration point coordinates are given manually. In [5] and [6] the images to be processed are sent to the Matlab® work space using graphic files of previously taken images, making the coordinate procuring process slow and boring.

Using the proposed algorithm it is possible to strengthen any photogrammetric camera calibration algorithms like [1] [2], because we have minimized illumination effects to detect the calibration image's coordinates.

The main contributions of this work are:

- A strengthened algorithm to processes an image of a calibration object to find the calibration points in different illumination conditions with subpixel accuracy.
- The implementation of the algorithm in an automatic vision system developed with Matlab®, electronically available for those people that are interested in calibrating their own cameras[1].

2 Methodology

In this section we are describing the vision system in which the image processing algorithm was implemented. Such a system consists of a 700 MHz Pentium III personal computer, a CCD Electrim

[1] The implementation in Matlab of this algorithm can be obtained in: http://paginasweb.tol.itesm.mx/Campus/otnielp/calibra.

EDC-1000U black and white camera, a CRS Robotics A465 robot arm and its controller. The algorithm was implemented in Matlab® 6.0. The program controls the robot using the serial port, and an image is acquired with the camera (using dynamic link library -DLLs- to communicate Matlab® with the camera driver), and the calibration object image is processed. The illumination intensity was measured using the exposition measure device Sekonic L-308.

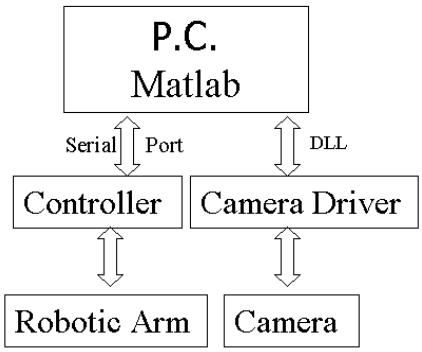

Figure 1. Block diagram of vision system

There are many types of possible calibration objects: single plane or multiple plane [1], with squares or circles [7]. To develop our algorithm we used a white stepped object with black squares. There are four different planes with a distance of ten centimeters between each one. There are three squares of 10x10 cm in each plane and all its corners are taken as calibration points to get 48 points as total. We have given a number to each corner to identify them (see Figure 2). In Figure 3 we can see the robot arm holding the CCD camera. The camera is focusing on the calibration plate. The robot can move the camera to every position alloweable inside the work space making our system very flexible.

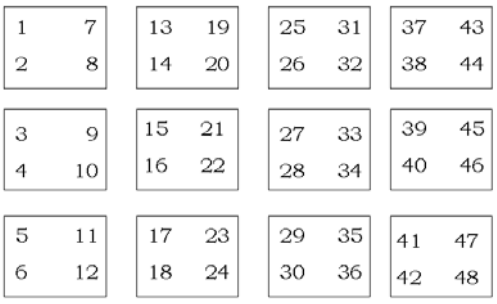

Figure 2. Numeration of calibrations points.

It is necessary to make sure that all the squares of the calibration object are fully visible to assure proper performance of our algorithm, otherwise the program will indicate an improper illumination condition.

Figure 3. Elements of the vision system (robotic arm CRS A465, CCD camera and calibration object).

3 The Algorithm

The aim of the algorithm is to aquire the coordinates (row and column) of the calibration points under different illumination conditions. The core of the algorithm is what we call "binarization threshold adjust" using image histogram equalization.

Figure 4 shows the flow diagram of the algorithm. Next, every single step is clearly explained using our vision system as an implementation example.

3.1 Image Capture

The first step is the image aquisition. In our vision system we developed a dynamic link library that allows communication between Matlab® and the camera driver, such that the image is acquired and sent automatically to the Matlab® work space. Figure 6 shows an image that was acquired from the vision system. The resolution of the image is 486 rows and 1134 columns.

3.2 Filtering

The noise present in the image results in highlighted pixels. If such pixels aren't eliminated, they will produce mistakes in the following stages of the image processing. To clean the image we use a median filter. Figure 7 is the result of applying a median filter to the Figure 6 image.

3.3 Image Equalization

Scene illumination is a prime factor in the calibration object image processing. For example, if the illumination is low the image will present a small histogram in the dark side of the gray scale. Otherwise, if the illumination is too high the histogram will be in the brighter side of the gray scale [8]. To distribute the histogram completely over the gray scale, we use a technique called "Histogram Equalization".

To standardize our concept of illumination we must define what low and high illumination is. The recommended illumination to use in laboratories by the CIE (Comisión Internationale de L'eclairage) is

500 lux [9]. Then, for us, a low illumination is 200 to 499 lux and high illumination is 500 to 6000 lux.

In our vision system, if the illumination fluctuates between 200 and 6000 lux the system will work properly, if not, faulty results may be obtained. Figure 8 shows the resulting image after applying histogram equalization in image 7.

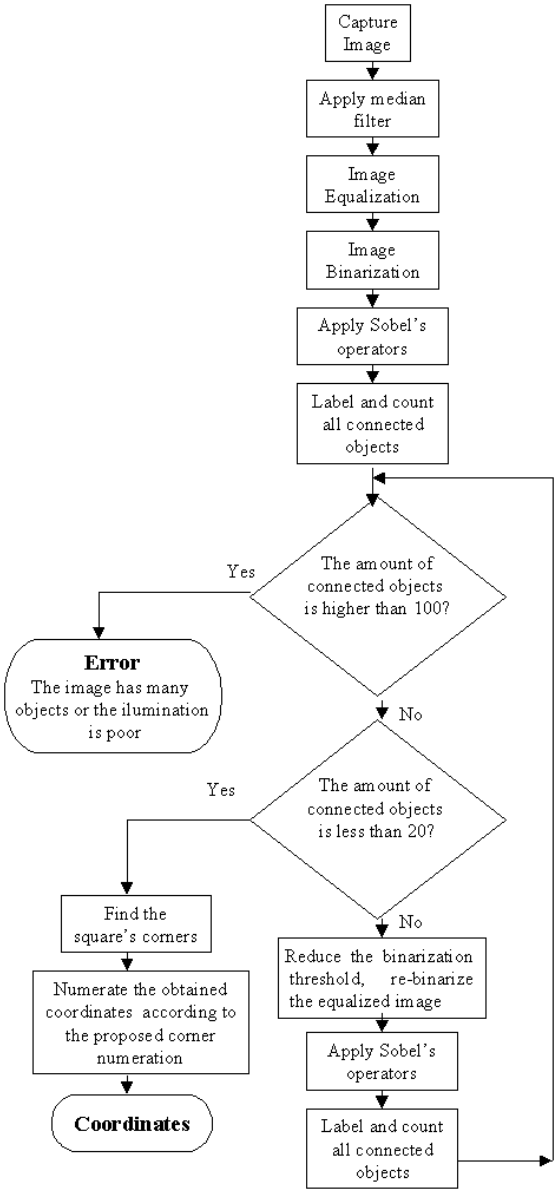

Figure 4. Algorithm's flow diagram

3.4 Binarization

Once the image is equalized, it is necessary to identify the squares and to separate them from the other objects in the scene. The first process used to separate them is to binarize the image [8]. Before doing this, we need to obtained the optimal threshold value. That is a parameter that must be calculated before running the algorithm.

To get the optimal threshold value, previously we must acquire several images come from 200 to 6000 lux illumination scenes. Each image must be equalized and binarized modifying the threshold value. If the number of the connected objects from each image varies between 20 and 80, then we have found the optimal threshold value. In our vision system the threshold value is 30%.

Once we get the optimal threshold value we can start the thresholding process. That means all the image pixels must be compared to the threshold value. If the compared pixel value is less than the threshold value then it is part of the background and it is assigned with the white attribute. If the compared pixel value is more then the threshold value then it could be part of a square and it's assigned with the black attribute. That is how we can slice our image to get a binary one, so that, at the end of the process we will have a black and white image.

Figure 9 shows the resulting image after applying the binarization process to a equalized image. We can see that all squares and some other objects (such as shadows) were removed from the scene because they were dark enough to be changed to black after the binarization process.

If the threshold level of Figure 9 is modified to 28%, some of the equalized image pixels could be higher in value than the threshold value and after applying the thresholding process those pixels will be changed to white and they will disappear from the scene (Figure 12).

The basis in our image processing procedure is to find a threshold level from the optimal one to minimize the amount of undesirable objects. Since those undesirable objects won't be processed later, the processing time to find the image calibration coordinates is reduced.

3.5 Edge Finding

The resulting image from the binarization process always contains undesirable objects (shadows, lost pixels, etc). For this reason it is necessary to find the square's edges and finally their corners. In our vision system we used Sobel's operators [8]. In the Figure 10 the edge detection of a binarized image is shown. We did tests with other edge detection algorithms such as Canny and Prewitt [8]. We decided to use Sobel's operators because the time of detection was the lowest of the three.

3.6 Connected objects labeling and counting

Once the edges are found, it is necessary to label and count the connected objects using an eight connected neighborhood method [8] (see Figure 11). Depending on the amount of connected objects of the image you will either get the square's corners or reduce the binarization threshold.

3.7 Logic Conditions

In this section the logic conditions of the algorithm flow diagram are shown (Figure 4).

1. If the amount of connected objects is higher than 100, the algorithm will produce an error condition because the light in the camera's diaphragm isn't enough or because there are many objects in the scene. To solve this problem, the scene must be illuminated with more light intensity within the 200 to 6000 lux interval. Otherwise, proceed to step two.
2. If the amount of connected objects is less than 20 go to step 4.Otherwise, go on to step three.
3. Reduce the binarization threshold, re-binarize the equalized image, find the edges and label the connected objects (Figures 12, 13, 14). Go to step two.
4. Find the square's corners (section 3.8).
5. Enumerate the obtained coordinates according to the proposed corner numeration (see Figure 2).

3.8 Getting the Corners

The most frequently used procedure to get the corners consists of using masks according to the corners we want to find. To find a corner in the processed image, a mask sweep over the image must be done. If a set of pixels is equal to the mask the desired corner has been found [8].

The process described is very useful when the object images are well delimited. Such images are acquired under controlled illumination conditions , in our case, we have illumination variations. In spite of the edge processing, sometimes no corner can be found using any known mask.

To solve the afore mentioned problem we devised a different process. First, we must know the number of pixels and coordinates for each connected object. With this information we can get the external coordinates of every connected object using it's maximum and minimum row and column values (Figure 5). In our vision system, in order to consider the connected object as a square, it must have the following conditions:

1. The connected object must be greater than 350 pixels.
2. It must be at least 50 pixels minimum but not more than 230 pixels height as maximum.
3. It must have a width of at least 100 pixels but not more than 210 pixels.

It is possible to adapt these conditions to some other vision system by changing the height and width values.

After detecting all the squares, its sides are segmented. Using the coordinates of every pixel of each segment, a linear regression is performed to get a straight line. Then every corner is configured by finding the intersection between two perpendicular lines. This way we get the corners with subpixel accuracy.

Figures 15 and 16 shows the result of the previously described procedure. You can see the square's center, the side segments and the corners. Figure 17 shows the detected corners with the algorithm overlapping the original image.

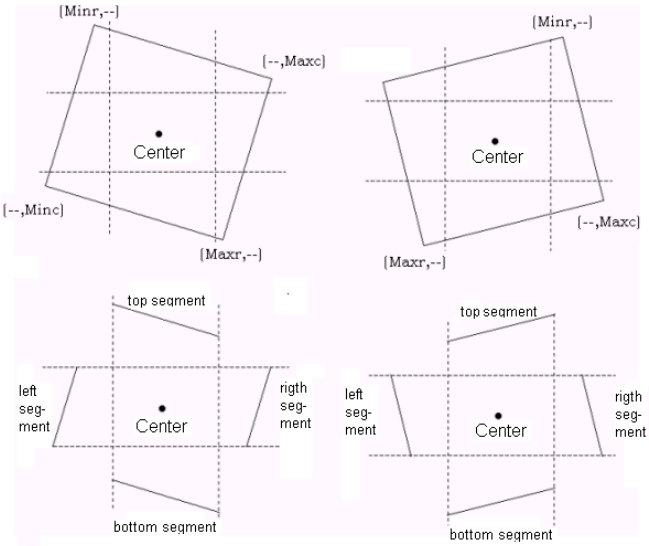

Figure 5. Method to getting the corners

4 Results

An algorithm to detect the calibration points was successfully implemented. The algorithm will compute the image calibration coordinates under the appropriate illumination interval (200 to 6000 lux).

Figure 6 shows an image captured with our vision system under a 600 lux. In Figure 17, we can see the corners (calibration points) detected by our proposed algorithm. In the Figure 18, can be seen a second image captured with a 400 lux illumination, you can also see how the algorithm detects the corners in a proper manner (see Figure 19). In Figure 20 you can see that, when the correct threshold binarization adjust is not used the detected corners are wrong.

5 Conclusion

The heart of our proposed algorithm is called "Binarization Threshold Adjust". It basically consists of adjusting the binarization threshold value from the number of connected objects in the image to process the next stages faster and in a more efficient way.

Matlab® was useful tool to make the proposed algorithm automatically work and to control the

elements of the vision system (camera and robotic arm). We developed a dynamic link library to allow communication between Matlab® and the camera drive. This performings only memory operations, increasing the transference speed of the image and avoiding the use of graphic files.

The results of this work are better than [4] and [5], because we are considering the illumination variations effect, in order to detect the calibration coordinates. An improvement over against Bennett's work [5] is that our work automatically acquires, processes and detects the calibration points of the image.

With the proposed algorithm it is possible to strengthen any available photogrammetric camera calibration technique that needs image coordinates for calibrations points, for this reason in a future work we will strengthen the Tsai camera calibration technique [1] and detect 3D world coordinates of any object using stereo vision [10].

References:

[1] Tsai R. Y., A versatile camera calibration technique for high-accuracy 3D machine vision metrology using off the shelf TV cameras and lenses, *IEEE Journal of Robotics and Automation*, Vol. 3 No. 4, August 1987, pp. 323-344.

[2] K. W. Wong, Mathematical formulation and digital analysis in close-range photogrammetry, *Photogrammetric Engineering Remote Sensing*. No. 41, 1975, pp.1355-1373.

[3] Faugeras Oliver, Three-Dimensional Computer Vision: a Geometric Viewpoint, MIT Press, 1993.

[4] Kampel M., *Calibration of the Acquisition System*. Pattern Recognition and Image Processing Group Institute of Computer Aided Automation, Computer Science Department, Vienna University of Technology. http://www.prip.tuwien.ac.at/Research/3DVision/ calib. html.

[5] Bennett W., *Automatic Feature Point Extraction for Light Field Camera Calibration***.** Stanford University's VLSI Research Group. http://www-vlsi.stanford.edu/~wilburn/Calibration/report.html

[6] Bouguet Jean-Yves, *Camera Calibration Toolbox for Matlab*®. Computer Vision Research Group, California Institute of Technology. http://www.vision.caltech.edu/bouguetj/calib_doc.

[7] Heikkilä, J., Geometric Camera Calibration Using Circular Control Points. *IEEE Transactions on Pattern Analysis and Machine Inteligence.* Vol. 22 No. 10, October 2000, pp.1066-1077.

[8] Gonzalez R. C. and Woods R. E., *Digital Image Processing*, Addison-Wesley, 1992.

[9] CIE Website. Commission Internationale de L'eclairage. http://members.eunet.at/cie/.

[10] Tebourbi Riad, 3D reconstruction of natural targets by stereovision. *Proceedings of the IEEE 1999 International Symposium on Geoscience and Remote Sensing,* No.2, 1999, pp. 1128 –1130.

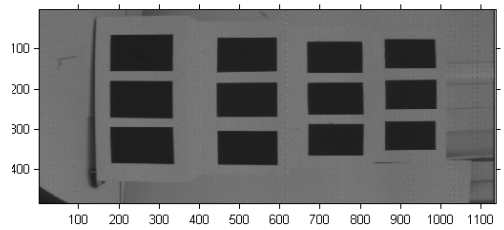

Figure 6. Original Image with noise.

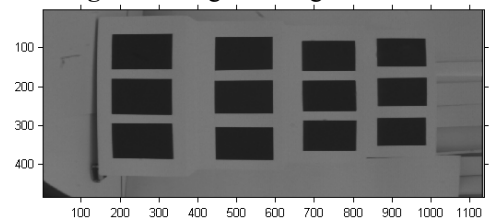

Figure 7. Resulting Image after applying median filter.

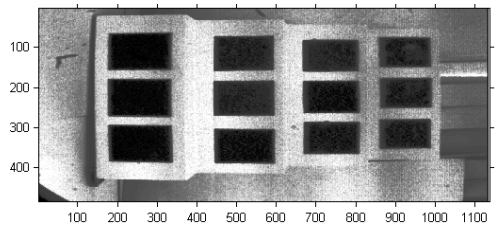

Figure 8. Resulting Image after applying histogram equalization.

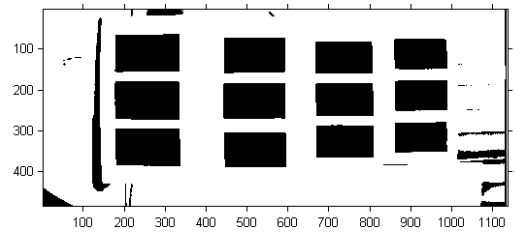

Figure 9. Binarization of the equalized image.

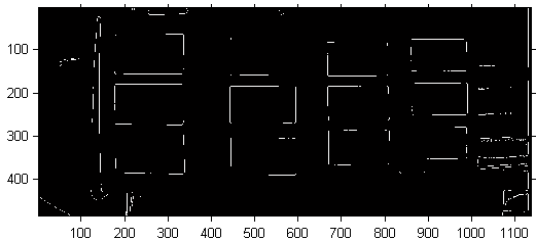

Figure 10. Resulting image after applying Sobel's operators.

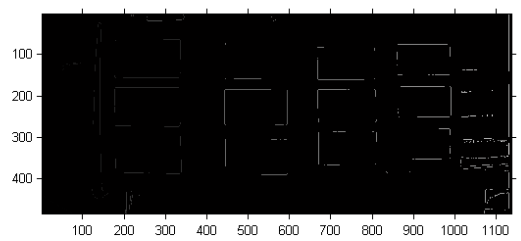

Figure 11. Resulting image after labelling all connected objects.

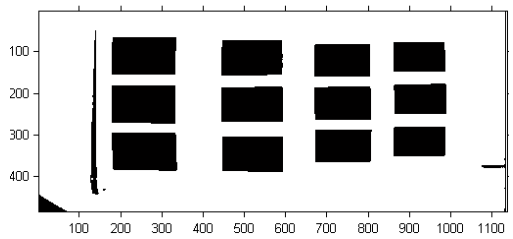

Figure 12. Binary image resulting after reducing threshold value to 28%.

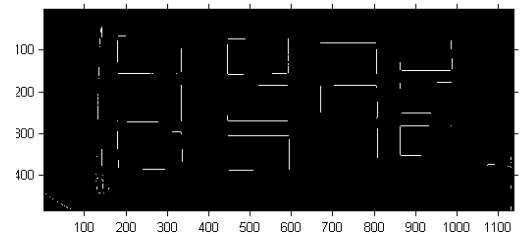

Figure 13. Resulting image after applying Sobel's operators to the last image.

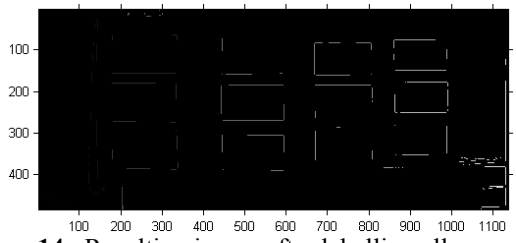

Figure 14. Resulting image after labelling all connected objects in the last image.

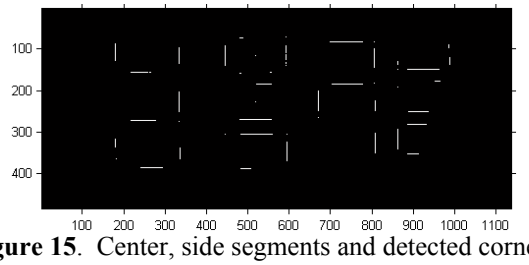

Figure 15. Center, side segments and detected corners of each square.

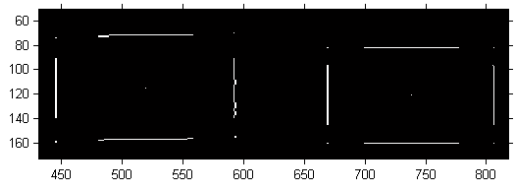

Figure 16. Center, side segments and detected corners of each square. (Amplified image to see details)

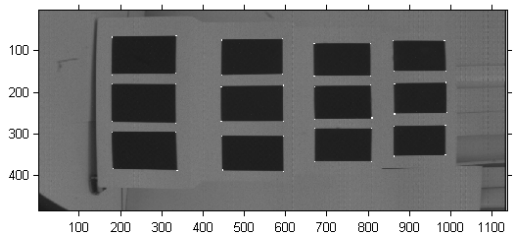

Figure 17. Detected corners overlapping the original image.

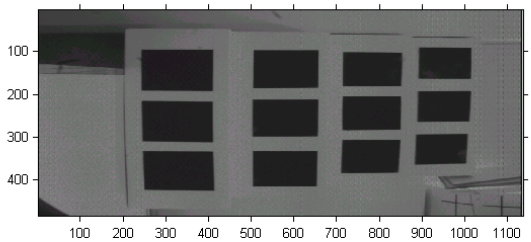

Figure 18. An second probe image with noise.

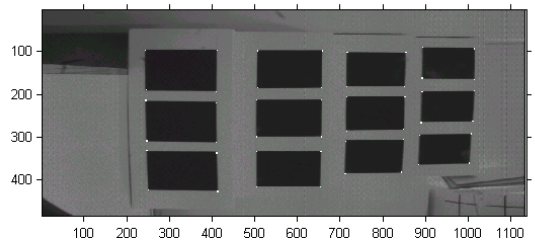

Figure 19. Detected corners using the proposed algorithm.

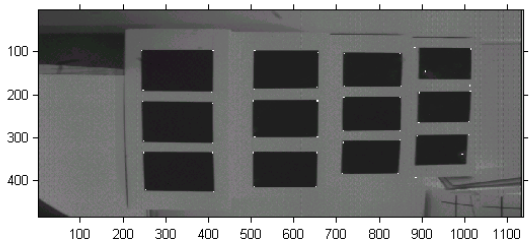

Figure 20. Detected corners without used the thresholding adjust.