Camera calibration using adaptive segmentation and ellipse fitting

Raúl Romaní Flores * **Paul Alonzo, Quio Añamuro** **

* *Universidad Católica San Pablo (e-mail: raul.romani@ucsp.edu.pe).*
** *Universidad Católica San Pablo (e-mail: paul.quio@ucsp.edu.pe).*

Abstract: We describe an algorithm that Detects the control points in the raw images for camera calibration for ring calibration patterns. The algorithm was implemented in c++ 11 and using opencv library. The proposed algorithm for finding the control points consist of four parts, Apply a mask that encloses the control points in order to reduce the work area, Apply adaptive threshold in order segment the image, find contours and find ellipses. We also implemented an algorithm for drawing lines(independent of rotation), in a specific order, starting from the top-left to the top-right and going down. We have conducted an set of experiments with recorded videos and with two cameras in real time.

Keywords: Camera calibration, threshold, integral image, contours, ellipse.

1. INTRODUCTION

In many machine vision applications, a crucial step is to accurately determine the relation between the image of the object and its physical dimension by performing a calibration process. Over time, various calibration techniques have been developed. Nevertheless, the existing methods cannot satisfy the ever-increasing demands for higher accuracy performance. In this paper, we propose an algorithm for control points detection that is used in the first stages of camera calibration techniques. We use a ring calibration patterns (Fig. 1). The proposed algorithm for finding the control points consist of four parts, Apply a mask that encloses the control points in order to reduce the work area, Apply adaptive threshold in order segment the image, find contours and find ellipses. We also implemented an algorithm for drawing lines(independent of rotation), in a specific order, starting from the top-left to the top-right and going down.

We run our algorithm in 4 videos recorded with 5 different cameras (kinnect v2, mslifecam, ps3eyecam and real sense) and two of them was used to test in real time. In order to implement this algorithm we read some theory about Zhang [2000], Datta et al. [2009], and taken Prakash and Karam [2012] as a base for the creating the algorithm. Additionally we made some test of real time pattern detection using the Ps3EyeCam and MsLifeCam getting an average of 48 pfs.

2. ALGORITHM

The camera calibration algorithm for ring calibration patterns implemented in c++ 11 and opencv library. The proposed algorithm for finding the control points consist of four parts, Apply a mask that encloses the control points in order to reduce the work area, Apply adaptive threshold in order segment the image, find contours and find ellipses. We also implemented an algorithm for drawing lines(independent of rotation), in a specific order, starting from the top-left to the top-right and going down. Those ordered control points will be used later to calibrate the camera. We will describe both algorithms in detail in the next two subsections.

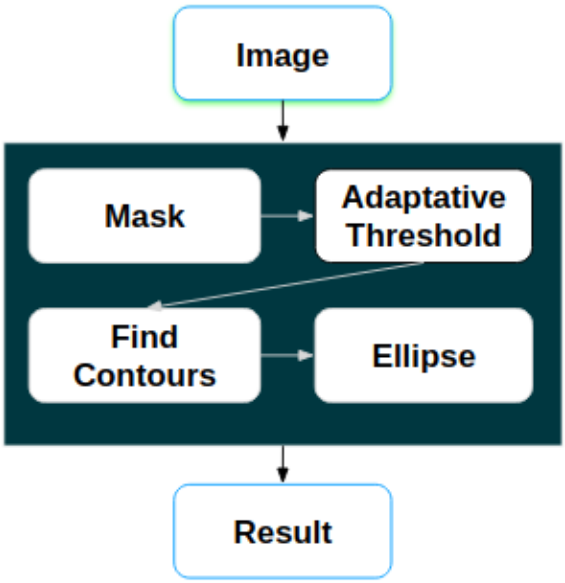

Fig. 1. Workflow for finding control points

2.1 Mask

Apply a mask that encloses the control points in order to reduce the work area. This mask is computed and applied to the input image after we have found 20 control points, we computed this mask using minAreaRect(opencv function), this function return a rotated rectangle, we use this a mask.

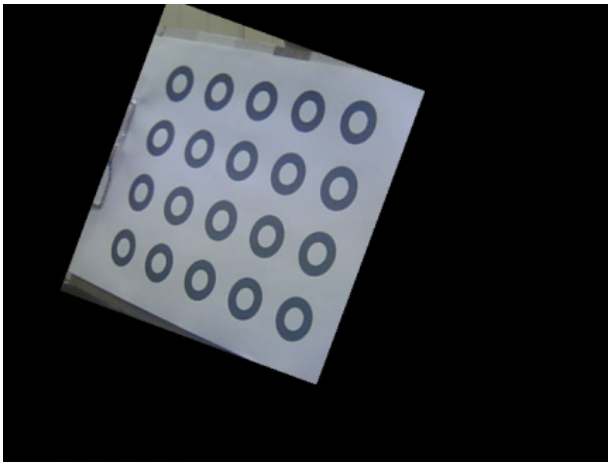

Fig. 2. Good masking after found the 20 pattern points

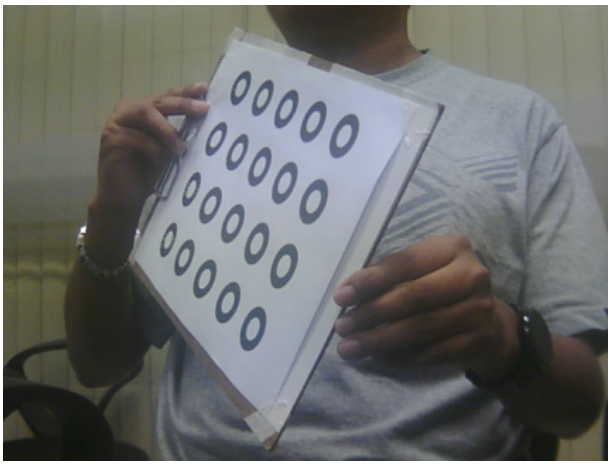

Fig. 3. If we didnt find the 20 pattern points, open the mask.

2.2 Adaptative Threshold

Preprocess the input image/video by usign adaptive thresholding using the integral image based on Bradley and Roth [2007].

In the paper cited above doenst say what we should do with the image borders so we usse the opencv threshold to complete that information.

Additionally we calculate the adaptive threshold using opencv to complement the information of the edges with the obtained by the threshold implemented previously.

We have some troubles when the pattern is moved fast or the camera doenst focus the pattern correctly.

2.3 Find contours

We use the function of openCV findContours to get all the contours from the segmented image in a hierarchy of contours, which will allow us to filter the ellipses in the next step.

Its an important step because if we dont found all pattern ellipses the next step will fail.

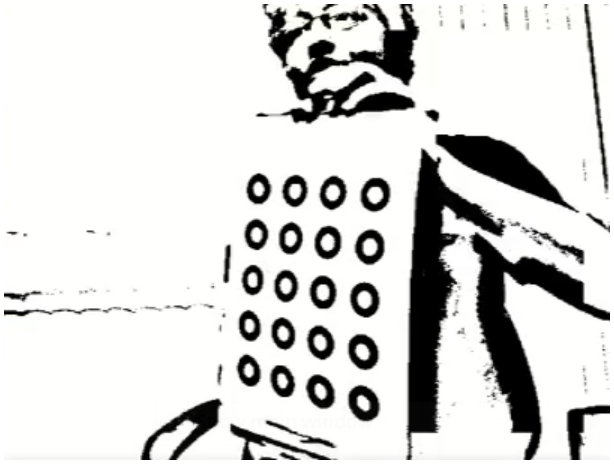

Fig. 4. Thresholding keeping good image information.

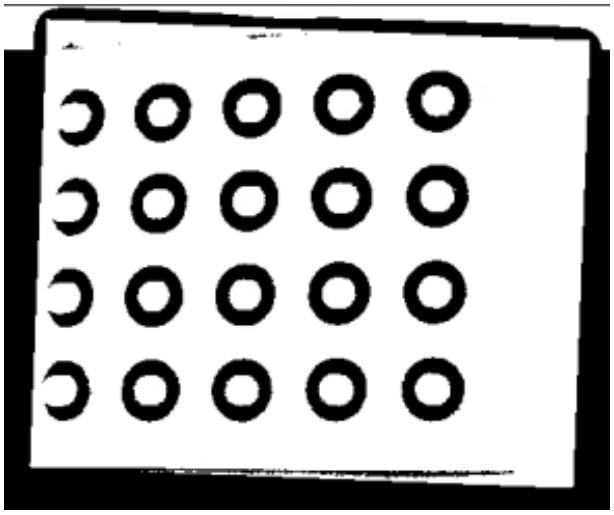

Fig. 5. Thresholding error cause by fast movement

Fig. 6. Thresholding error cause by focus error

2.4 Find ellipse

- From the found contours we look for all the ellipses that have a father and a son, besides that the center of the son is very close to his own.
- From the set of ellipses found we verified that we have at least 2 ellipses near us at a distance no greater than

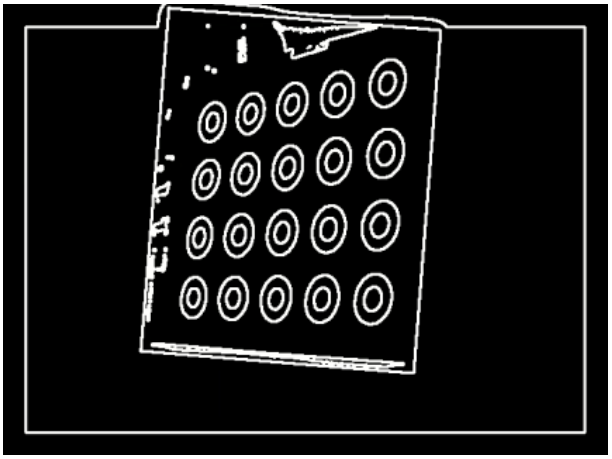

Fig. 7. Good contours improved by thresholding.

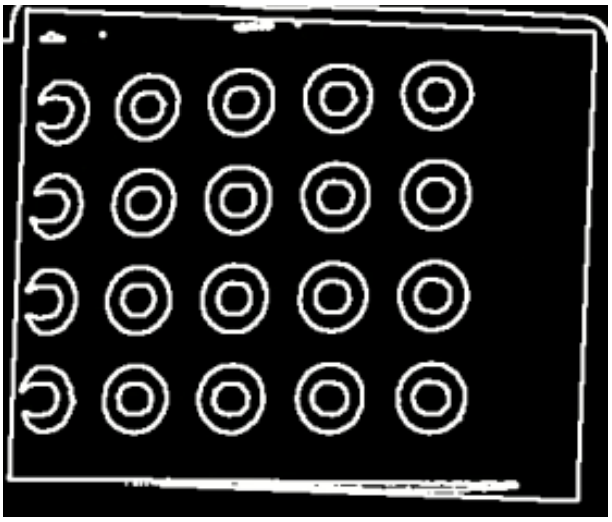

Fig. 8. Bad contours caused by a fast movement.

5 times the radius of the ellipse with which we make the comparison.

- If we have more than 20 ellipses we look for mode based on the hierarchy of the father of each ellipse and we eliminate all those who have a different father.

Fig. 9. Comparison between two ellipses to check if the ellipse c1 belong to the control point. the small Yellow line is the radio of the ellipse c2, the large Yellow line is five times the radio of the ellipse c2. We can see that condition distance(c1, c2) < 5 * radio C2, does not hold, therefore the ellipse c1 does not belong to the control points.

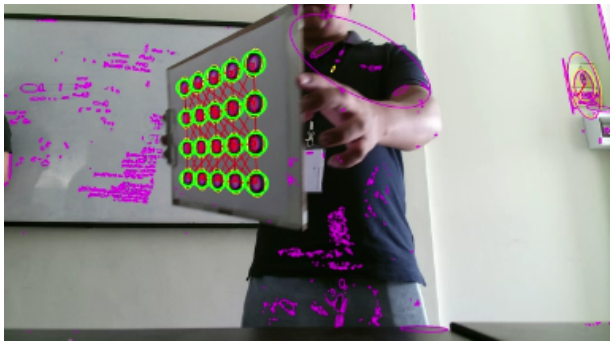

Fig. 10. Discard false positive using the father mode.

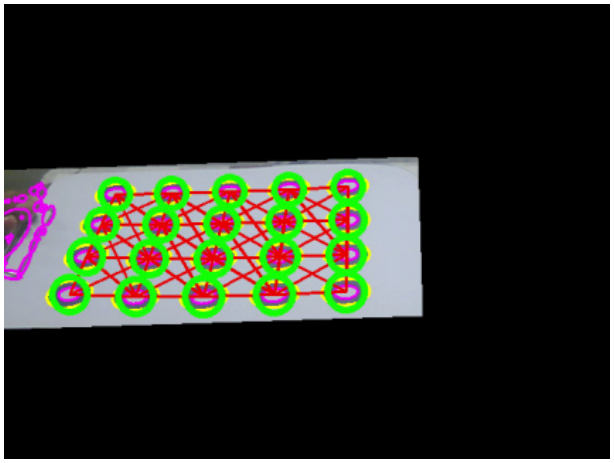

Fig. 11. Good patter points recognition.

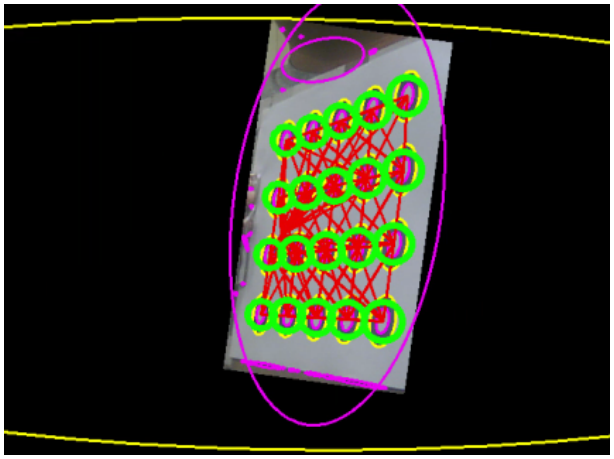

Fig. 12. Good patter points recognition.

2.5 Line representation

After we have found the rings of the pattern in the image we start to build the lines representation of the pattern.

- Using each pair of points we make a line and determine the line equation.
- Using the line equation count the number of points near to the line and store these points in a aux vector.
- If there are 5 points in the aux vector, determine which two points are most furthest to take them as limits of the line and order these points using their x coordinate
- Draw a line between these two points.

Fig. 13. Ellipse not detected correctly cause by a poor image definition.

- Keep the second point got from the process described before.
- Remove the points and continue searching process to find the next line until we found the 4 lines.
- Use the kept point and the first point of the new line detected and draw a line between these two points.

3. RESULTS

We perform some tests of the algorithm in real time using the Ps3 eye camera and the Microsoft life cam getting an average of 48 fps.

As we can see in the following figures, we obtain a good performance in images with high rotation.

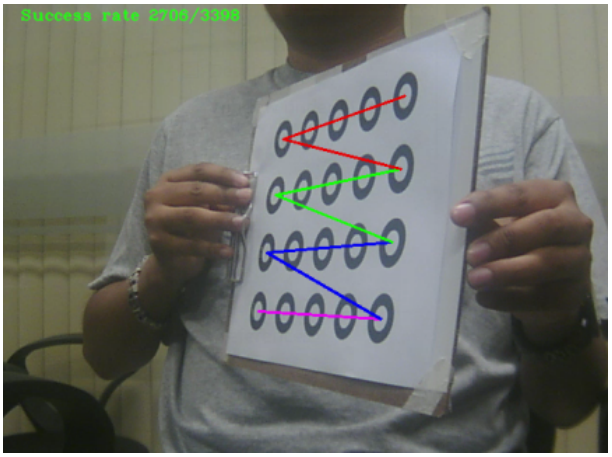

Fig. 14. Results after applying our algorithm on ps3eyecam video.

4. CONCLUSIONS

- It is important to clean irrelevant information in the videos in order to capture relevant characteristics.
- The algorithm needs to be further improved to get better performance on other videos.
- The algorithm depends on how well the ellipses of the calibration pattern are detect. Depends basically on fitEllipse method of opencv.

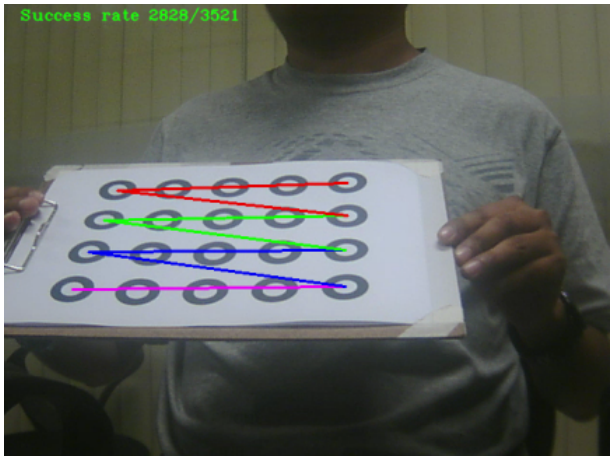

Fig. 15. Results after applying our algorithm on ps3eyecam video.

REFERENCES

Bradley, D. and Roth, G. (2007). Adaptive thresholding using the integral image. *Journal of graphics tools*, 12(2), 13–21.

Datta, A., Kim, J.S., and Kanade, T. (2009). Accurate camera calibration using iterative refinement of control points. In *Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on*, 1201–1208. IEEE.

Prakash, C.D. and Karam, L.J. (2012). Camera calibration using adaptive segmentation and ellipse fitting for localizing control points. In *Image Processing (ICIP), 2012 19th IEEE International Conference on*, 341–344. IEEE.

Zhang, Z. (2000). A flexible new technique for camera calibration. *IEEE Transactions on pattern analysis and machine intelligence*, 22(11), 1330–1334.