Camera calibration using adaptive segmentation and ellipse fitting

Raúl Romaní Flores * **Paul Alonzo Quio Añamuro** **
Jose Jaita Aguilar ***

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

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.

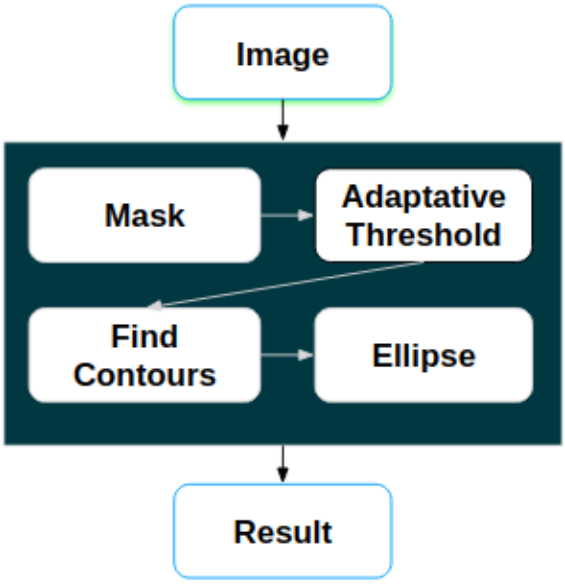

Fig. 1. Pipeline for finding control points

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. At left we can see a good masking after found the 20 pattern points. If we didnt find the 20 pattern points, open the mask(right).

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 doesn't 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.

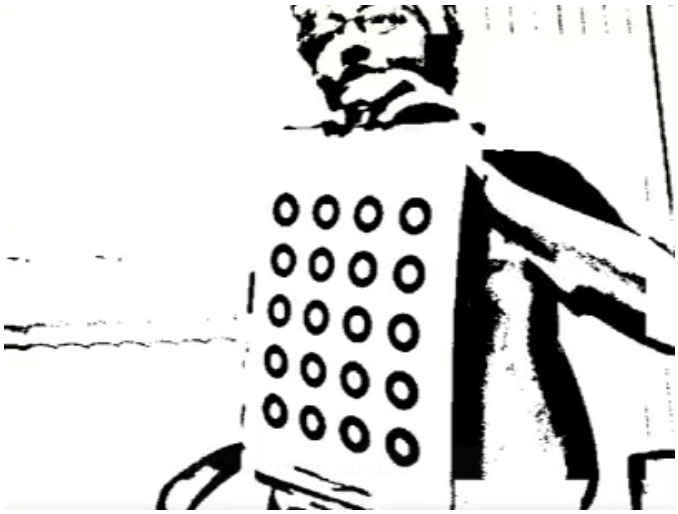

Fig. 3. Thresholding keeping good image information.

We have some troubles when the pattern is moved fast or the camera doesn't focus the pattern correctly.

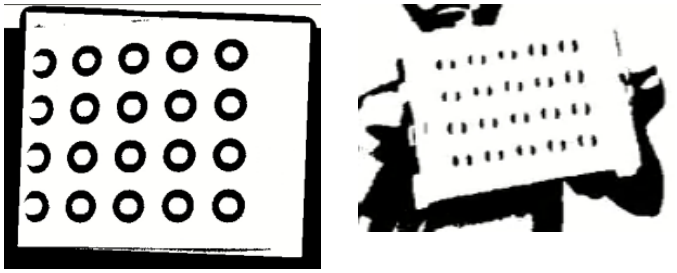

Fig. 4. At left we can see a thresholding error cause by fast movement. At right we can see a thresholding error cause by focus error.

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 don't found all pattern ellipses the next step will fail.

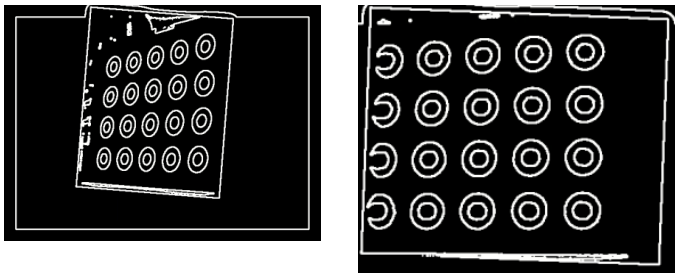

Fig. 5. At left we can see a good contours improved by thresholding. At right we can see a bad contours caused by a fast movement.

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 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. 6. 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.

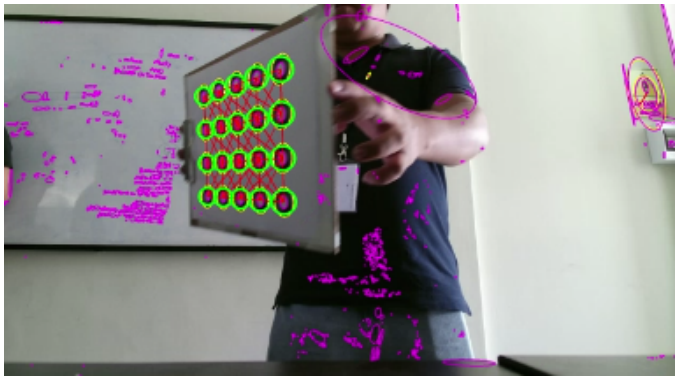

Fig. 7. Discard false positive using the father mode.

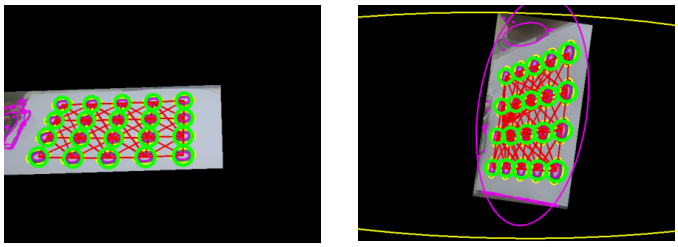

Fig. 8. Good patter points recognition.

Fig. 9. Ellipse not detected correctly cause by a poor image definition.

2.5 Calculate ellipse center

To find the ellipse center (control point) we test some ideas:

- First, we use the method of OpenCV **fitEllipse** to approximate the rings, then our control point comes the average of the father and son's ellipse center.
- Second, the control point comes the average of the father and son's ring center of mass. Definition of moments in image processing is borrowed from physics. Assume that each pixel in image has weight that is equal to its intensity. Then the point you defined is centroid (center of mass) of image. In case of a raster image, the spatial moments $\texttt{Moments::m}_{ji}$ are computed as:

$$\texttt{m}_{ji} = \sum_{x,y} \left(\texttt{array}(x,y) \cdot x^j \cdot y^i \right) \tag{1}$$

The central moments $\texttt{Moments::mu}_{ji}$ are computed as:

$$\texttt{mu}_{ji} = \sum_{x,y} \left(\texttt{array}(x,y) \cdot (x - \bar{x})^j \cdot (y - \bar{y})^i \right) \tag{2}$$

$$\bar{x} = \frac{\texttt{m}_{10}}{\texttt{m}_{00}}, \; \bar{y} = \frac{\texttt{m}_{01}}{\texttt{m}_{00}} \tag{3}$$

Where $(\bar{x}, \bar{y})$ is the mass center.

2.6 Line representation

After we have found the rings of the pattern in the image we start to build the lines representation of the pattern. If we lost the pattern we need to order the points again using the next algorithm and if we doesn't lost the pattern we perform the tracking process described in the next subsection.

- 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, order these points using the x coordinate and check the distance between the first and last elements if the distance is lower than the radio of circles then sort the aux vector again using the y coordinate
- Draw a line between the first and last elements of the aux vector.
- Keep the last 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.

Whit the ordered points we only need to draw lines from the 0 point to 4 point, 5 point to 9 point etc.

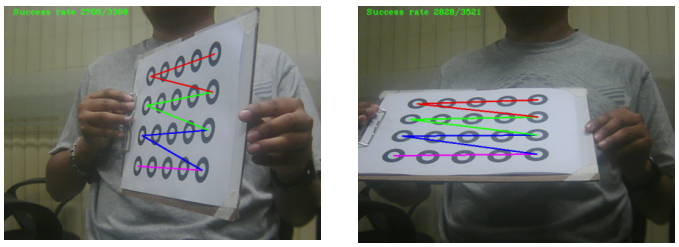

Fig. 10. Results after applying our algorithm on ps3eyecam video.

2.7 Tracking

To keep the order obtained in the previous process we perform a tracking process. From the points found in the previous frame we search in the new points some one that is nearly to these one using the radio as boundary. When the 20 points was found we check if any one has distance bigger than the radio to its near point if this is true we discard old point and consider the pattern was lost.

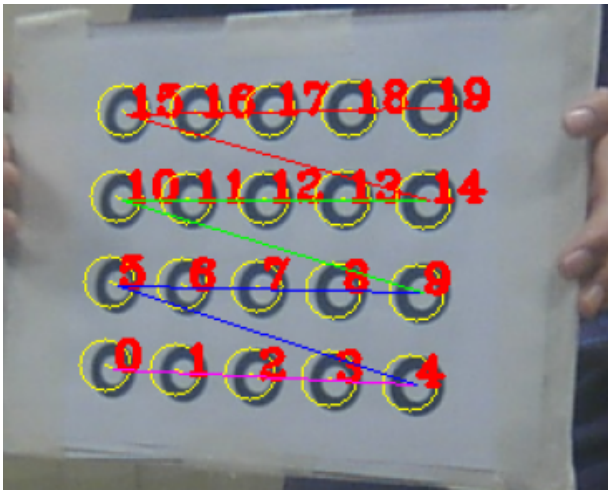

Fig. 11. Tracking process, the yellow circles are the area to search the new position for each pattern point.

3. CAMERA CALIBRATION

To perform the camera calibration we need to select some frames where the pattern was found and use these points to perform the calibration, it's important to choose the set of frames that covers the whole screen to improve the calibration process.

3.1 Initial frame selection

To perform a good calibration we need to choose some well distributed frames, for this task we develop this algorithm:

1 Define a grid over the image with r x c quadrants.
2 Define the acceptance area for every quadrant, that is a circle with radio the width * height / 6
4 Define the number of frames per quadrant as round ((samples + 4)/(r * c))
5 Per each frame we calculate the center point as (point[7] + point[12]) /2
6 Check the quadrant of the calculated center and if the center stay in the acceptance area.
7 Check if there is possible to add the point in these quadrant.
7.1 Add the frame to the calibration frames and skip 30 frames to avoid add nearly frames.
8 If there are no more space in the quadrant we skip 9 frames to speedup the process.

In the Figure 12 we can see the quadrants used to select frames for the PS3 and Lifecam videos, the blue circles are the acceptance area of each quadrant, green circles are the center of the selected frames and the red points are the centers of the evaluated frames.

The Figure 13 show the distribution of points obtained for the algorithm described above.

3.2 Calibration helper

We perform a calibration using the selected frames obtained using the process above, we are going to use this calibration to calculate the rotation in roll, pitch and yaw for each frame.

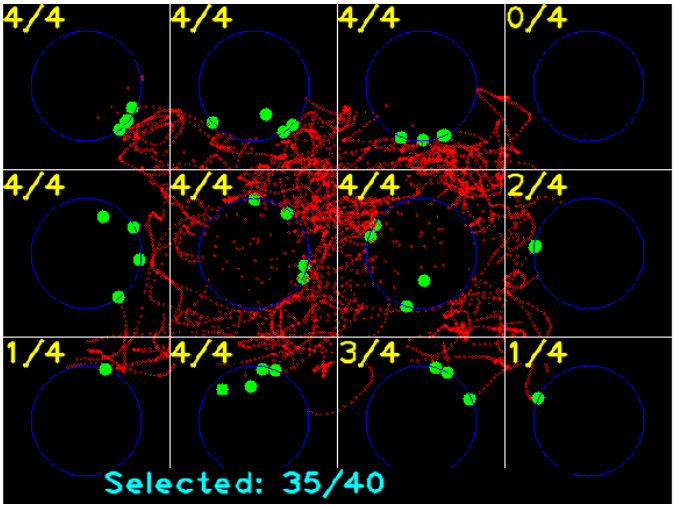

Fig. 12. Visualization of the frame selection using the quadrants distribution.

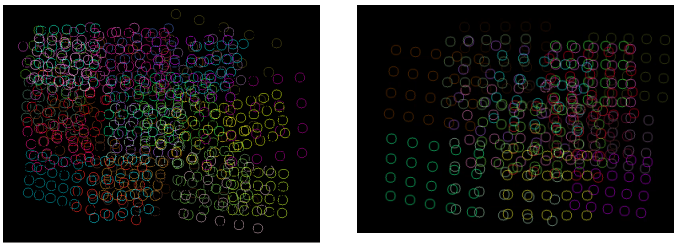

Fig. 13. Points used for the calibration process.

3.3 Frame selection

Using the camera matrix and distortion coefficients found in an initial calibration we run again the process described in section 3.1 adding a filter by angle criteria, after some test we found that filtering roll in range $-30 < r < 30$, yaw in range $-20 < y < 20$ and pitch in range $-180 < p < -150 \vee 150 < p < 180$.

As we can see at Figure 14 frames with high amount of rotation as discarted, this improve the calibration process.

Fig. 14. At left we can see a good masking after found the 20 pattern points. If we didnt find the 20 pattern points, open the mask(right).

The table 1 and 2 show the rms and collinearity obtained for the first calibration using the filtering by angle or not.

Initial calibration					
RMS/Samples	20	30	40	50	60
Without filtering	0.2590	0.2776	0.2709	0.2898	0.2890
With filtering by angle	0.2020	0.2333	0.2421	0.2439	0.2684

Table 1. RMS obtained for different quantity of frames using filtering by angle or not.

Initial calibration					
Collinearity/Samples	20	30	40	50	60
Without filtering	0.2004	0.1886	0.1919	0.2021	0.2136
With filtering by angle	0.1969	0.1883	0.2011	0.1997	0.2191

Table 2. Collinearity obtained for different quantity of frames using filtering by angle or not.

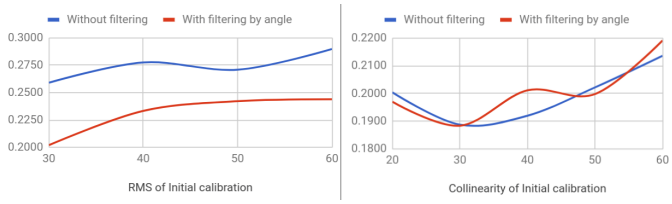

Fig. 15. RMS and collinearity comparison using filtering by angle or not.

Average of collinearity: In addition to the reprojection error we evaluate the distance from every inner line point to its main line to get the average of collinearity value.

4. REFINEMENT PROCESS

4.1 Undistort

Using the camera matrix and the distortion coefficients we can transform the image input to correct the distortion. In this new image we run the pattern detection algorithm to obtain the pattern points.

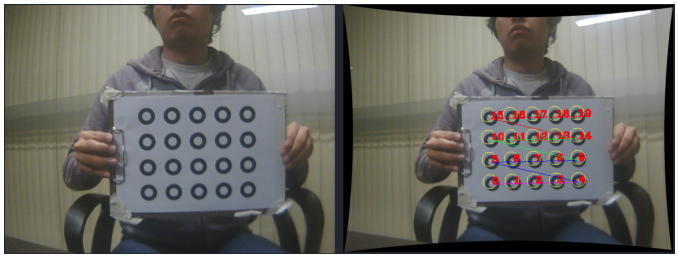

Fig. 16. Image with no distortion using the camera parameters and the distortion coefficients

4.2 Fronto parallel transformation

We use homography transformation from the detected points to an ideal set points to get a fronto parallel view of the image. Then we run the pattern detection algorithm again to find the new centers.

4.3 Reprojection

Using the new points detected in the fronto parallel image we need to reproject these points to the undistorted image,

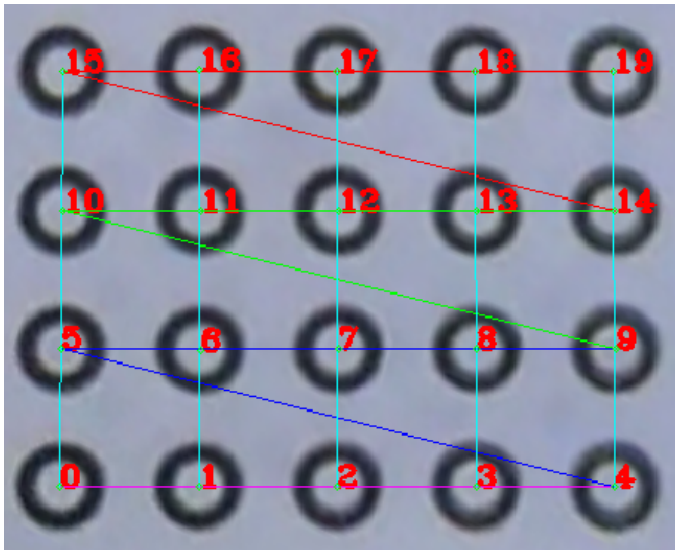

Fig. 17. Image with no distortion using the camera parameters and the distortion coefficients

for this task we use the inverse of the homography matrix and the perspectiveTransform function of opencv.

Points refinement To improve the calibration process and the collinearity we use the undistorted set of points(old points) and the reprojected set of points(new points) the following algorithm describe the process to update the new set of points in the undistorted space.

1 Per each row we fit a line using the 5 points of the new set of points.
2 Per each point in the row
2.1 Calculate the distance from the old point to the line.
2.2 Calculate the distance from the new point to the line.
2.3 Calculate the blend factor as the proportion of each distance to the line giving a priority to the closest one
2.4 Update the new point using the blend factor as newPoint = newPoint * factor + oldPoint * (1-factor)

In the Figure 18 we can see in green the original points and in red the new set of point after the refination process.

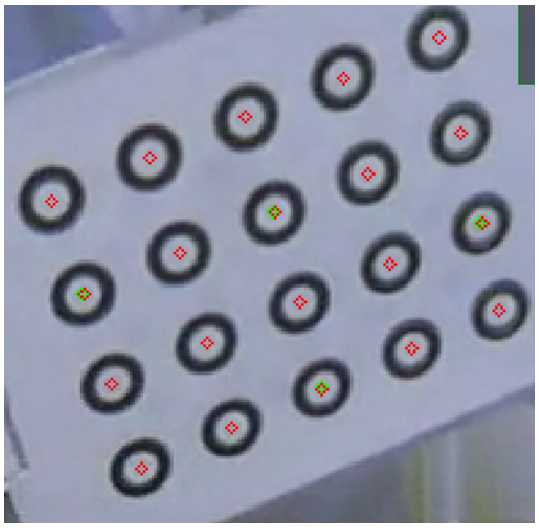

Fig. 18. Original points(green) and refinated points(red).

We compare the refinement using the average of old and new point, and the blending process described above, the comparison results are shown in the Figures 19 and 20

	Rings - Refinement - Avg					Rings - Refinement - Blend				
Samples	**20**	**30**	**40**	**50**	**60**	**20**	**30**	**40**	**50**	**60**
Rms	0.195	0.192	0.235	0.220	0.251	0.194	0.190	0.234	0.219	0.251
fx	845.748	829.826	823.726	819.288	812.192	844.143	832.426	823.290	819.639	811.885
fy	840.082	824.040	819.035	813.653	807.466	838.616	826.569	818.583	813.977	807.060
cx	329.338	312.016	304.019	299.733	298.165	328.923	311.706	304.667	299.790	298.915
cy	241.683	238.191	244.391	259.171	259.310	241.277	237.382	244.639	259.618	259.824
Collinearity	0.197	0.188	0.201	0.200	0.219	0.197	0.188	0.201	0.200	0.219
Collinearity undistorted	0.075	0.077	0.078	0.086	0.089	0.073	0.078	0.079	0.088	0.089

Fig. 19. Results comparison using average vs a blending factor to calculate the new point.

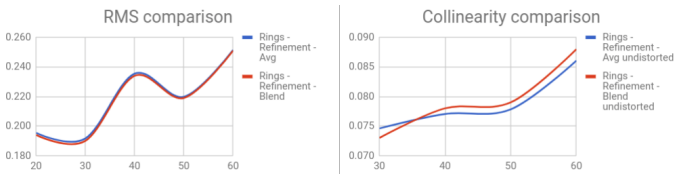

Fig. 20. Results comparison using average vs a blending factor to calculate the new point.

4.4 Distort

To perform the calibration process using the new points we first need to distort the found points. The algorithm to distort the points is shown in the Figure 21 we can see the result of the distort over the found points, in red we can see the original centers, green are the centers with no distortion and blue the points with distortion.

We use the new set of distorted points to perform the camera calibration process.

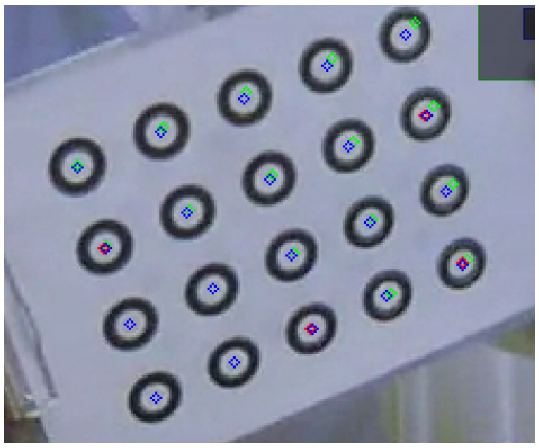

Fig. 21. Result of distort the found points

5. RESULTS

We used the camera calibration algorithm provided by the OpenCV library to perform a comparison with the method implemented in this paper. We disabled the flags FixAspectRatio, AssumeZeroTangentialDistortion and FixPrincipalPointAtTheCenter. See Tables 6.4 and 6.4.

5.1 PS3 calibration

For the PS3 camera we now present the comparison between the chessboard, circles and rings patterns.

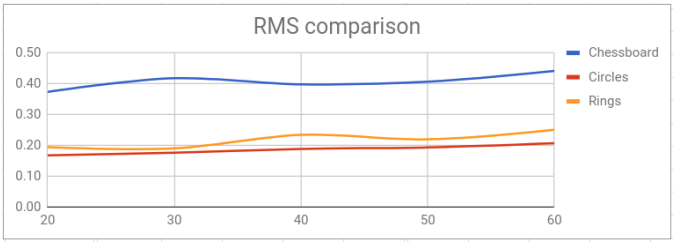

Fig. 22. RMS results using three differents kinds of patterns and differents numbers of samples.

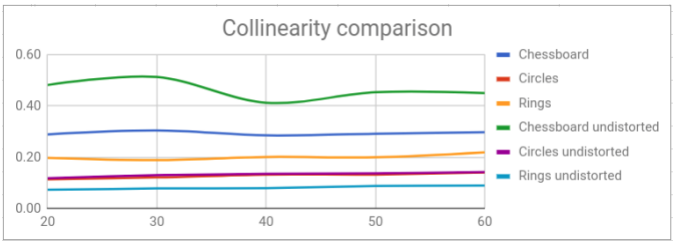

Fig. 23. Collinearity results using three differents kinds of patterns and differents numbers of samples.

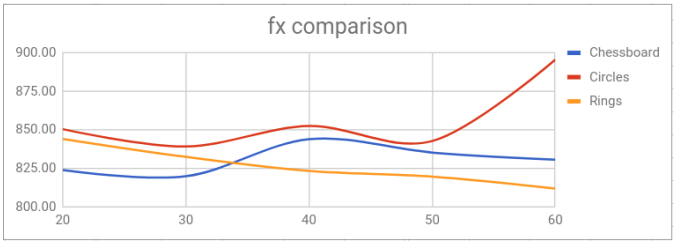

Fig. 24. Focal length X results using three differents kinds of patterns and differents numbers of samples.

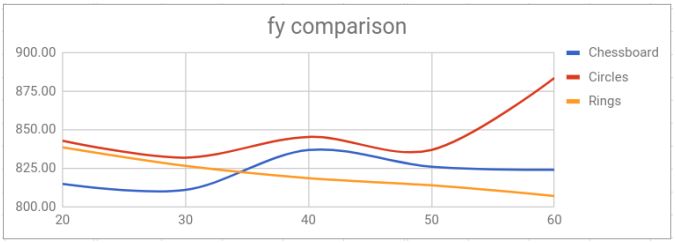

Fig. 25. Focal length Y results using three differents kinds of patterns and differents numbers of samples.

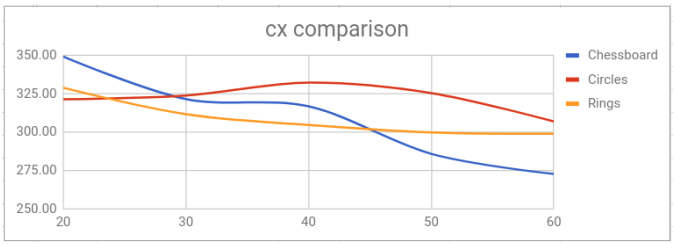

Fig. 26. Optic center X results using three differents kinds of patterns and differents numbers of samples.

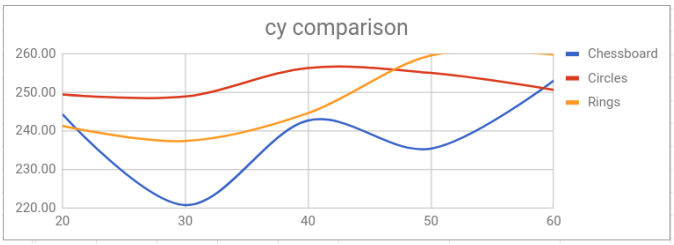

Fig. 27. Optic center Y results using three differents kinds of patterns and differents numbers of samples.

5.2 Lifecam calibration

For the Lifecam camera we now present the comparison between the chessboard, circles and rings patterns.

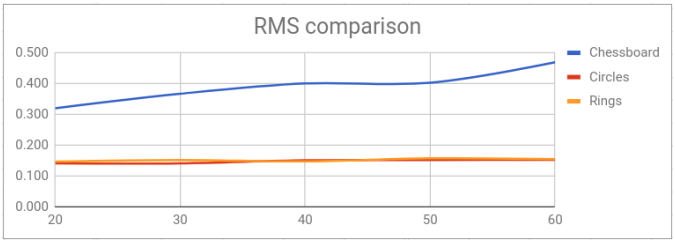

Fig. 28. RMS results using three differents kinds of patterns and differents numbers of samples.

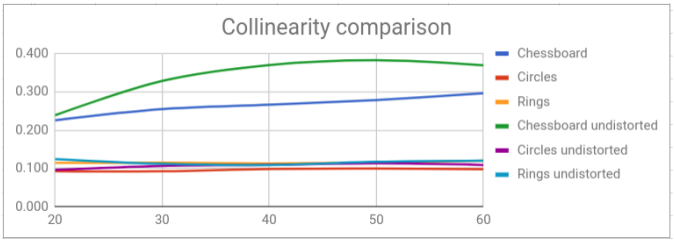

Fig. 29. Collinearity results using three differents kinds of patterns and differents numbers of samples.

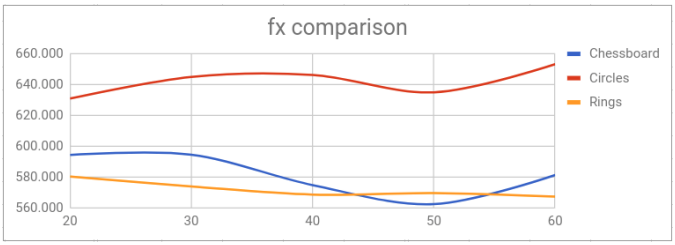

Fig. 30. Focal length X results using three differents kinds of patterns and differents numbers of samples.

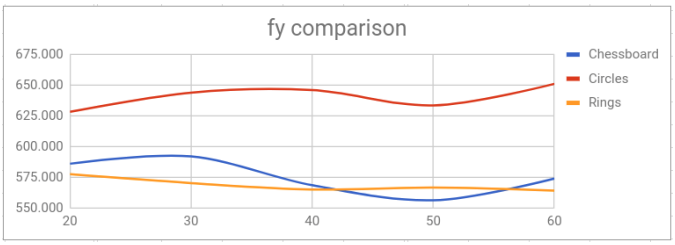

Fig. 31. Focal length Y results using three differents kinds of patterns and differents numbers of samples.

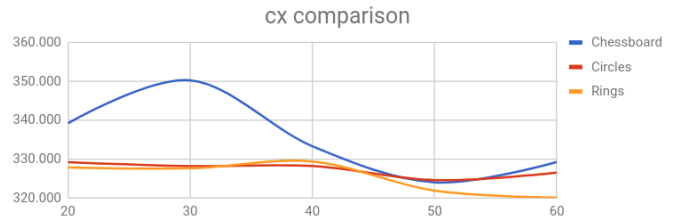

Fig. 32. Optic center X results using three differents kinds of patterns and differents numbers of samples.

Fig. 33. Optic center Y results using three differents kinds of patterns and differents numbers of samples.

6. EXPERIMENTS

6.1 FitEllipse vs Center of Mass

In this experiment we make the comparison between fitEllipse and centers of mass. PS3 camera was used, the results are show in the Figs.(34, 35, 36).

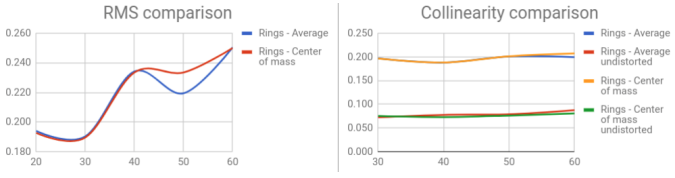

Fig. 34. RMS and collinearity results using fitEllipse and center of mass.

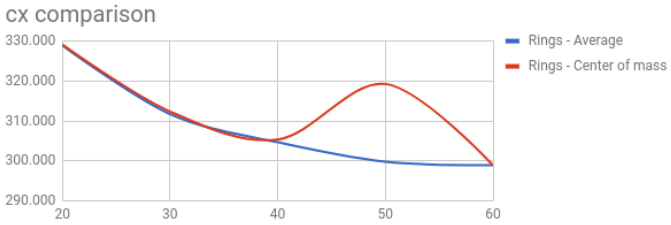

Fig. 35. Optic center X results results using fitEllipse and center of mass.

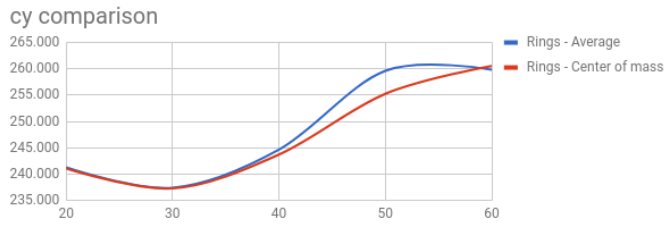

Fig. 36. Optic center Y results results using fitEllipse and center of mass.

6.2 Ideal Point

In this experiment, we add an ideal point in fronto-parallel and the point control is the average between ideal point and the center computed by our algorithm, the results are show in the Figs.(37);

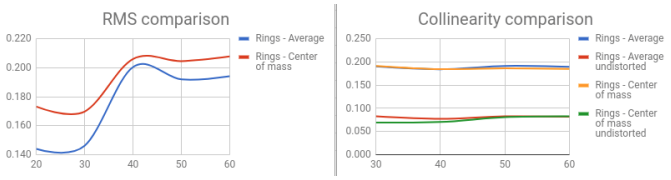

Fig. 37. RMS and Collinearity for ellipse and center of mass using ideal point.

6.3 OpenGL pattern 3d visualization

We use the camera matrix obtained in the calibration process to implement an Opengl application to show the pattern movement in the 3D space.

To achieve this, we use the solvePnP opencv's function to calculate the rotation and translation vectors of a frame using the camera matrix and distortion coefficients found in the calibration process, then we get the euler angles from the rotation vector to rotate a plane with the pattern texture.

In the Figures 38 and 39 we can see the results obtained using this idea.

6.4 Simple augmented reality application

Another experiment that we perform was insert a 3d model in the video using the rotation calculated using the solvePnP Opencv's function.

The Figure 40 shows the execution of our simple augmented reality app.

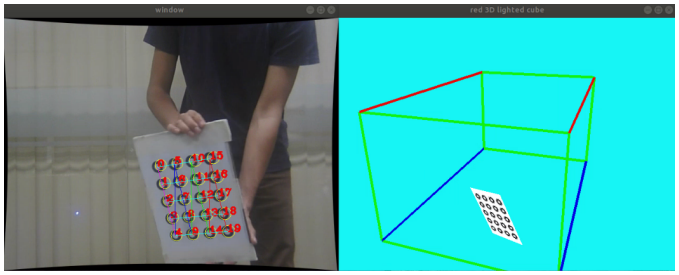

Fig. 38. Experiments using Opencv + Opengl

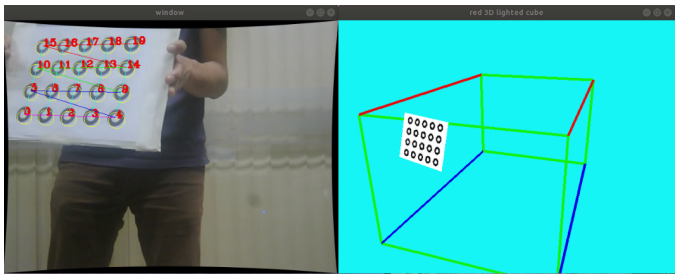

Fig. 39. Experiments using Opencv + Opengl

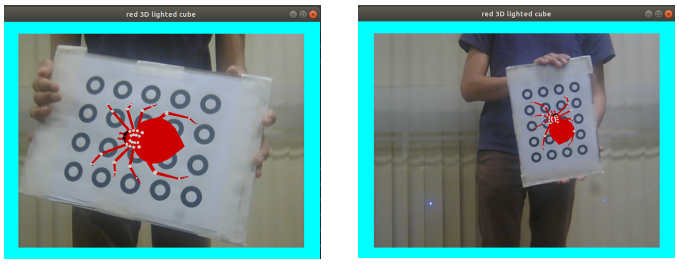

Fig. 40. Simple augmented reality application

7. 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.
- To calculate the control points, center of mass has a slight improvement on fitEllipse, although in our experiment for 50 frames had a loss in performance reflected in the rms.
- For the camera calibration process is important to perform a good tracking algorithm to keep the order of the points.
- The frames selected for the calibration process should cover the four quadrants of the image plane in similar proportions and have good quality to improve the results.
- Using a refinement process based on the collinearity improve the results obtained in the calibration process.

	PS3 calibration														
	Chessboard					Circles					Rings				
Samples	**20**	**30**	**40**	**50**	**60**	**20**	**30**	**40**	**50**	**60**	**20**	**30**	**40**	**50**	**60**
Rms	0.37	0.42	0.40	0.41	0.44	0.17	0.18	0.19	0.19	0.21	0.19	0.19	0.23	0.22	0.25
fx	823.89	819.86	843.96	835.28	830.64	850.45	839.22	852.50	842.75	895.62	844.14	832.43	823.29	819.64	811.89
fy	814.89	810.98	836.91	825.98	824.02	842.98	831.96	845.42	836.96	883.76	838.62	826.57	818.58	813.98	807.06
cx	349.30	321.55	316.79	285.87	272.75	321.46	323.86	332.26	325.48	306.94	328.92	311.71	304.67	299.79	298.92
cy	244.39	220.75	242.71	235.41	253.10	249.47	248.94	256.37	255.07	250.69	241.28	237.38	244.64	259.62	259.82
Collinearity	0.29	0.30	0.29	0.29	0.30	0.11	0.12	0.13	0.13	0.14	0.20	0.19	0.20	0.20	0.22
Collinearity undistorted	0.48	0.51	0.41	0.45	0.45	0.12	0.13	0.13	0.14	0.14	0.07	0.08	0.08	0.09	0.09

Fig. 41. PS3 camera calibration results using three different kinds of patterns and different numbers of samples.

	Lifecam calibration														
	Chessboard					Circles					Rings				
Samples	**20**	**30**	**40**	**50**	**60**	**20**	**30**	**40**	**50**	**60**	**20**	**30**	**40**	**50**	**60**
Rms	0.320	0.367	0.401	0.403	0.469	0.141	0.141	0.151	0.152	0.153	0.147	0.152	0.147	0.157	0.155
fx	594.367	594.552	574.776	562.427	581.313	630.945	645.074	646.279	635.057	653.293	580.284	573.903	568.614	569.633	567.322
fy	586.049	591.965	568.541	556.259	573.979	628.307	643.997	645.990	633.565	651.163	577.496	570.289	565.060	566.595	564.123
cx	339.208	350.287	333.303	324.017	329.273	329.215	328.201	328.227	324.586	326.534	327.856	327.654	329.371	321.865	320.117
cy	225.481	235.792	241.965	245.283	240.238	226.911	217.472	219.844	228.095	229.871	218.551	215.812	216.151	220.605	220.237
Collinearity	0.226	0.255	0.266	0.279	0.296	0.092	0.092	0.098	0.099	0.098	0.114	0.115	0.113	0.115	0.120
Collinearity undistorted	0.238	0.328	0.370	0.383	0.369	0.096	0.106	0.109	0.113	0.109	0.124	0.112	0.109	0.117	0.120

Fig. 42. Lifecam camera calibration results using three different kinds of patterns and different numbers of samples.

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.