Maestría en Ciencias de la Computación
Camera Calibration with Rings Pattern and OpenCV

Diego Alonso Javier Quispe , David Choqueluque Roman
Universidad Católica San Pablo
{ diego.javier, david.choqueluque}@ucsp.edu.pe
Arequipa,Perú

***Abstract*— The are many applications to machine vision, and is important to know the relation between the image of some object and its physical dimension in the space. The main idea to this work is obtain a good camera calibration procedure using a special rings pattern. We developed the algorithm in c++ using OpenCV. In the next sections we explain our general pipeline with different stages for camera calibration and show the results for each stage.**

Key words: Canny algorithm, Pattern tracking, Camera Calibration ,OpenCV

I. INTRODUCTION

An important step in the Camera Calibration pipeline is the detection and tracking of the control points of the pattern in real time. The objective of this step is to maintain a order of the control points detected through modifications in its position in space. In the present document we first explain the general pipeline that we follow, then we develop each stage of it.

II. GENERAL PIPELINE

The pipeline that we follow is composed of the following stages (Figure 1). First we read a video frame by frame, then we preprocess the frame, detect the control points and finally we follow up on them.

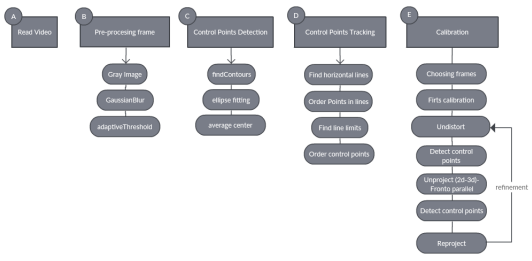

Fig. 1. General Pipeline.

III. STAGES

III-A. Read Video

In this stage we simply read frame-by-frame input video supported by Opencv reading functions.

III-B. Pre-processing Frame

In this stage we convert the input to gray scale, apply the Gaussian filter and finally perform the thresholding of the frame. The following images show the result of each operation.

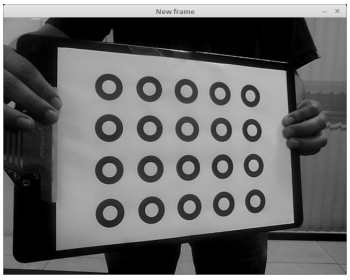

Fig. 2. Original Frame to GrayScale.

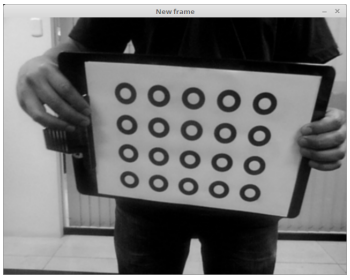

Fig. 3. Gaussian Filter.

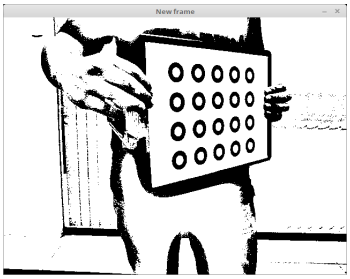

Fig. 4. Adaptative Threshold.

III-C. Control Point Detection

In this stage we takes as input the preprocesed image and look for contours, then we calculate the ellipse of these.

Whit the ellipses calculated above we select those that have a child and calculate the ellipse of this. To rule out false positive ellipses, we found that the centers of the parent-child ellipses do not exceed a maximum distance, this distance is calculated as half the radius of the ellipse son. The following image show the result of this stage. Finally for the ellipses that fulfill the condition we calculate the average of its two centers and keep it as a point of control of the pattern.

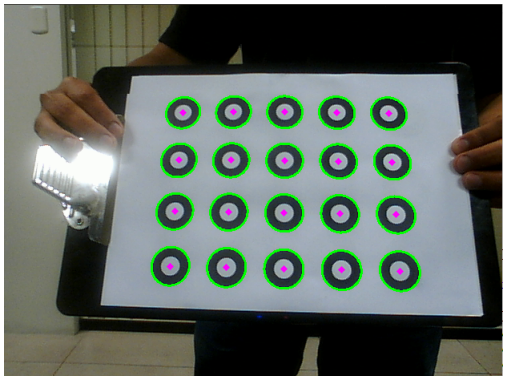

Fig. 5. Control Points Detection

III-D. Control Point Tracking

In this stage we take as input the control points calculated in the previous stage (Centers) and we proceed to find the horizontal lines (lines) that contain the centers of a row in the pattern. For that we take centers in pairs, calculate the equation of the line they define and for the remaining centers we calculate the distance from a point (center) to the previously calculated line, this is done until we obtain a line containing the number of ellipses in a row of the pattern (number of columns).

After calculating all the lines that the pattern contains, we proceed to enumerate each control point (center).

The following image shows the result of this stage.

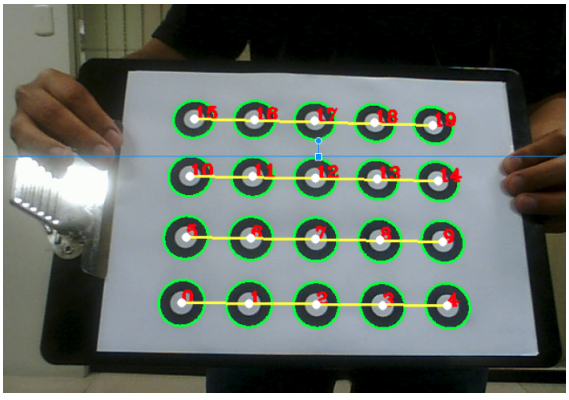

Fig. 6. Control Points Tracking.

III-E. Choosing Frames

For this stage prior to the calibration algorithm we select "good"frames. First we define circular regions in the image plane (Fig. 7). The selection algorithm is the following:

- Read one frame each 50 ms.
- Detect control points in frame and verify their correctness (order).
- Calculate the center of the control points.
- Check if the center of the control points is contained in one region defined above. If so, compare the regions of the current frame with the previous one.
- If the current frame and the previous one do not belong to the same region then check the region capacity.
- If the region has space then save the frame.

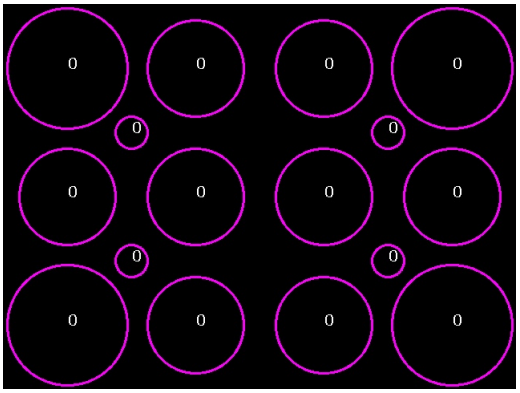

Fig. 7. Defined regions for frame selection.

The Figure 7 show the initial circular regions defined. Each region has a capacity as limit for frames in the region. This regions are build with the following steps.

- Divide the image plane into 12 rectangular regions (3 rows and 4 columns) with a height h and width w .
- For each rectangular region build a circular region with radius r equal to $w/2-20$ for median circles, $w/2-5$ for corners(big) circles and 5 for small circles.
- Set a capacity of 2 for the small circles and $desired_frames/12$ for the median and big circles.

The Figure 8 show a example for choosing frames in a given time. The left image show the tracking of the center of control points, the right image shows the capacity of the regions in that time.

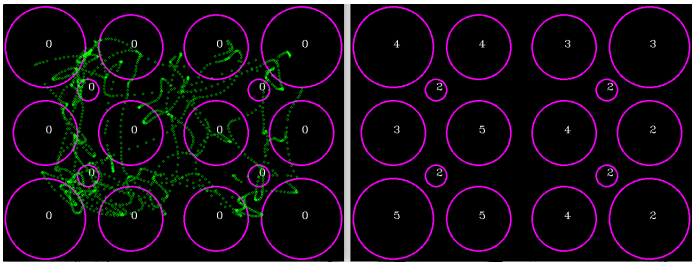

Fig. 8. Choosing frames example.

III-F. Calibration

For the calibration we calculate the matrix calibration and the average re-projection error (rms) for two different

cameras. We use the calibrateCamera Opencv function for this purpose. To choose the images for the calibration we use time heuristics for the video.

IV. Experimental result

We have a general view of the stages described in the pipeline, this window is shown below.

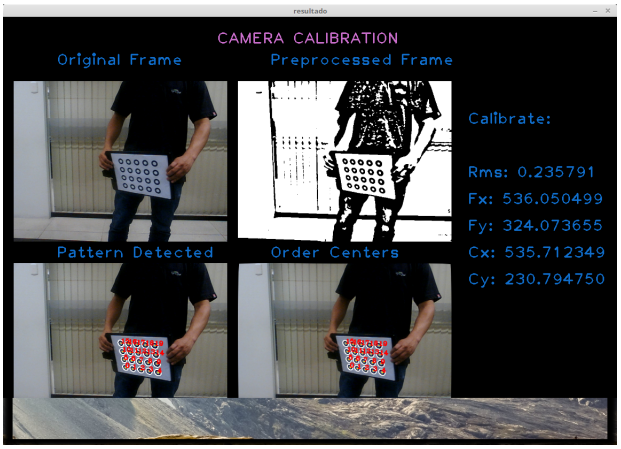

Fig. 9. General window to Camera Calibration.

For the pattern tracking we use two types of rings pattern, pattern 1 with 12 rings and pattern 2 with 20 rings. The next table show the time and accuracy.

	Patter 1	**Pattern 2**
Time (ms)	5.01	5.06
Frames	Fail 467 of 5972	Fail 413 of 5144
Accuracy (%)	92.18	91.97

For the calibration. In the next tables we show the results of calculating the intrinsic parameters of the two cameras using three different types of pattern: chessboard, asymmetrical circles and rings.

Camera 1			
	ChessBoard	**Assym. Circles**	**Our Impl.**
rms	0.5972	0.7987	0.2918
fx	674.10767	751.7648	705.0014
fy	673.45705	755.7854	701.6671
cx	309.26380	348.8548	351.4085
cy	262.5772	263.02961	258.9697

For the camera 2 we obtain the next results.

Camera 2			
	ChessBoard	**Assym. Circles**	**Our Impl.**
rms	0.5308	0.2783	0.3781
fx	514.5797	492.4746	508.94199
fy	514.8973	493.9645	509.63553
cx	335.1768	328.70143	300.25258
cy	181.4782	175.3383	184.21757

The next images show the original image and the corrected image after applying the distortion removal using cv::initUndistortRectifyMap and the cv::remap function to remove distortion.

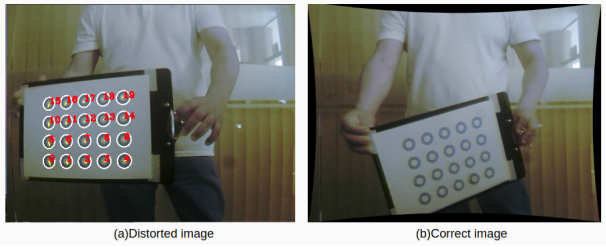

Fig. 10. Distorted and Undistorted images (Camera 1).

For the second camera we show the result in the next image.

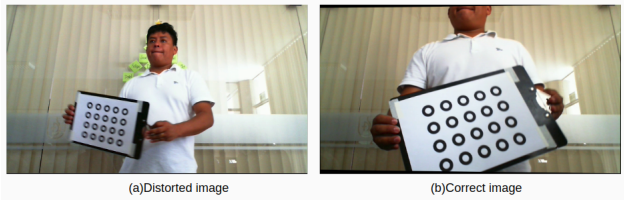

Fig. 11. Distorted and Undistorted images (Camera 2).

IV-A. Control Point Refinement

There are four steps to refinement control points in the camera calibration [2].

1. First Calibration: For this step we calculate initial parameters described in $III F$.
2. Undistort: With the initial camera parameters we undistort the image.
3. Localize control points: We localize the control points in the undistort image.
4. Unproject (Fronto-parallel): We use the function **cvFindHomography** to find the transform between the **control points** and the **real points**. We use the method **cvWarpPerspective** to generate the frontal image.
5. Localize control points: In this step we localize the control points in the fronto-parallel image, then we calculate intersections of row and column for each control point (Fig. 12).
6. Reproject: In this step we reproject (3d to 2d) the control points using the reverse homography of 4.
7. Refinement: In this step we calculate the average or the baricenter of the reprojected points.

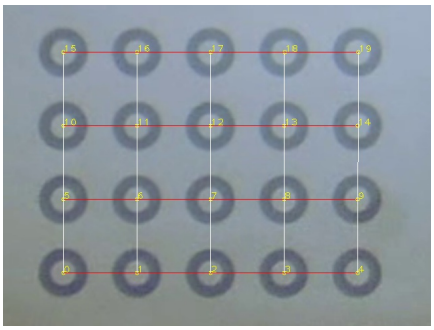

Fig. 12. Intersections.

REFERENCES

[1] Zhengyou Zhang,"A Flexible New Technique for Camera Calibration", 2000.

[2] Ankur Datta, "Accurate Camera Calibration using Iterative Refinement of Control Points", 2009.

[3] Prakash, Çamera Calibration using Adaptive Segmentation and Ellipse Fitting for Localizing Control Points", 2012.