Improved Camera Calibration with a planar pattern using OpenCV

Solange Ramos
*Computer Science San Pablo Catholic University

***Abstract*—Camera calibration is still a challenging problem in which even small improvements will be favorable for tasks such as reconstruct a world model or interact with the world as in case of a robot. In this paper, we present a complete process for an accurate camera calibration combining many techniques from the actual state of art. The improvement relies on a fast pattern detection procedure for a concentric circles planar pattern, that makes it possible to be performed in real time within scenes that have a lot of noise and different illumination intensity. The results show a comparison for pattern detection, between our proposed and OpenCV methods, and for camera calibration process, between our proposed and other techniques.**

***Index Terms*—Camera calibration, self-calibration, OpenCV, machine vision, concentric circles planar pattern.**

I. Introduction

Camera calibration is an important process in computer vision that consists in extracting metric information from 2D images. The overall goal of camera calibration is to find out the parameters of the transformation between an object in 3D space and 2D image observed by the camera. [3]
The transformation parameters include:
Extrinsic parameters: The location and orientation of the camera in the world which is denoted by the rotation and translation of the camera.
Intrinsic parameters: The relationship between pixel coordinates and camera coordinates.
Algorithms for camera calibration can be classify into two categories: [1]
Photogrammetric calibration: This category requires objects with known 3D geometry with very good precision. Calibrations can be done efficiently.
Self-calibrations: Techniques in this category do not use any calibration object. Just by moving a camera in a static scene, the rigidity of the scene provides in general two constraints [8] [9] on the cameras internal parameters from one camera displacement by using image information alone. Even though, this approach is very flexible and easy to use, it is not yet mature due to there are many parameters to estimate, we cannot always obtain reliable results.
The calibration procedure typically consists of either localizing the calibration pattern control points and then solving for the camera parameters, or using some geometry property of the pattern itself to solve for the camera parameters itself.[2]
In this paper we proposed a method to perform a real time detection pattern for a concentric circles planar pattern and a calibration camera process that combine many techniques of the actual state of art that leads a low re-projection error and a close approximation to the real camera parameters. The structure of this article is as follows. In section 2 we summarize the related work in the area of camera calibration. Next, in Section 3 we describe the proposed approach for calibrating the camera. Section 4 covers the experimental results which are followed by conclusion and future work covered in the last Section.

II. Related Work

The work in the camera calibration field was first done using squares as control points by Tsai [4]. 60 control points were used to find out the camera position and orientation parameters along with focal length and radial lens distortion. In order to solve the calibration problem it requires N control points per image to solve a set of N linear equations which are based on radial alignment restrain. This method have shown very good results, however they need high accurate information from input data.
Different patterns were later used such us Circular and Rings patterns. Researchers in [5] [6] [7] have been used this kind of markers following the method presented by Tsai [4].
In 2000, Zhang [1] presented a flexible method using the point position coordinates on the 2D planar pattern. Thus, it is combined methods based on the measurements of the pattern coordinates and self-calibration where plannar patterns are not needed. This leads this method being flexible where either the camera or the planar pattern can be freely move.
Later in 2009 Datta [2] presented an iterative algorithm to refine control point detection and get an more accurate calibration camera parameters. They get parameters from traditional calibration algorithms as initialization to perform undistortion and unprojection of calibrations images to a canonical fronto-parallel plane, that is later used to recompute the camera parameters in an iterative refinement until converge.
The techniques employed by Tsai, Zhang and Datta all use the pinhole projective model to map three dimensional scenes to the two dimensional camera image plane.

III. Pinhole camera model

The pinhole camera model describes homogeneous coordinates between a 2D point in an image plane denoted by $p = [xy1]^T$ and the corresponding 3D point denoted by $P = [XYZ1]^T$, the model relates them as follows:

$$sp = A[Rt]P$$

where s is the unknown scalar factor, R is a $3x3$ rotation matrix containing the three rotation angles ω, ϕ and ψ respect to x, y and z axes, t is a $3x1$ translation vector containing the x, y and z displacement directions, both called extrinsic parameters. A is $3x3$ matrix containing the intrinsic parameters.

$$A = \begin{bmatrix} f_x & \gamma & u_0 \\ 0 & f_y & v_0 \\ 0 & 0 & 1 \end{bmatrix}$$

where (f_x, f_y) are the focal lengths along the x and y axes of the image, γ is the skew between the two image axes, (u_0, v_0) is the central point. Modern cameras are well manufactured such the skew can be assumed to be zero. Figure 1 shows the model.

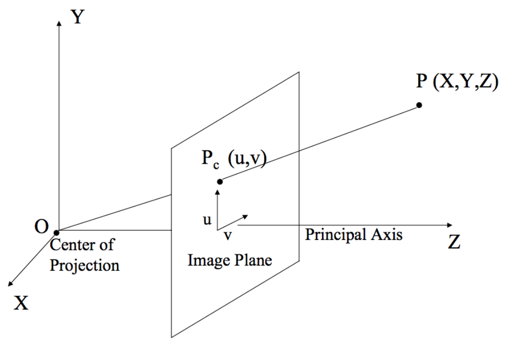

Fig. 1: Pinhole camera model

Using a 2D planar pattern for calibrating a camera, it can be assumed that the pattern lies in the plane $Z = 0$ without losing generality. ...

IV. Proposed approach

Figure 2 (a) show the flowchart of the proposed approach for calibrate a camera. There are 5 main steps, capture frames, get control points, perform camera calibration, undistort and unproject and reproject images. The improvement presented in this paper relies on a fast pattern detection procedure for a concentric circles planar pattern, that makes it possible to be performed in real time within scenes that have a lot of noise and different illumination intensity, letting us get accurate control points positions in real scenarios. Figure 2 (b) shows this sub process.
After getting an initial camera calibration parameters, an iterative process is performed from the undistortion and unprojection step to the performing for a new camera calibration step until get the more accurate camera extrinsic parameters.

A. Capture Frame

The first step was easily performed by using OpenCV library. This library let us read images in very wide group of formats or read real time video captures. The functions were used were *cv::imread* and *cv::VideoCapture*.

B. Get control points positions

The control points in every frame were localized following a series of steps that let, first pre-process the frame and the apply some heuristics to get the correct arranged control points in every frame.

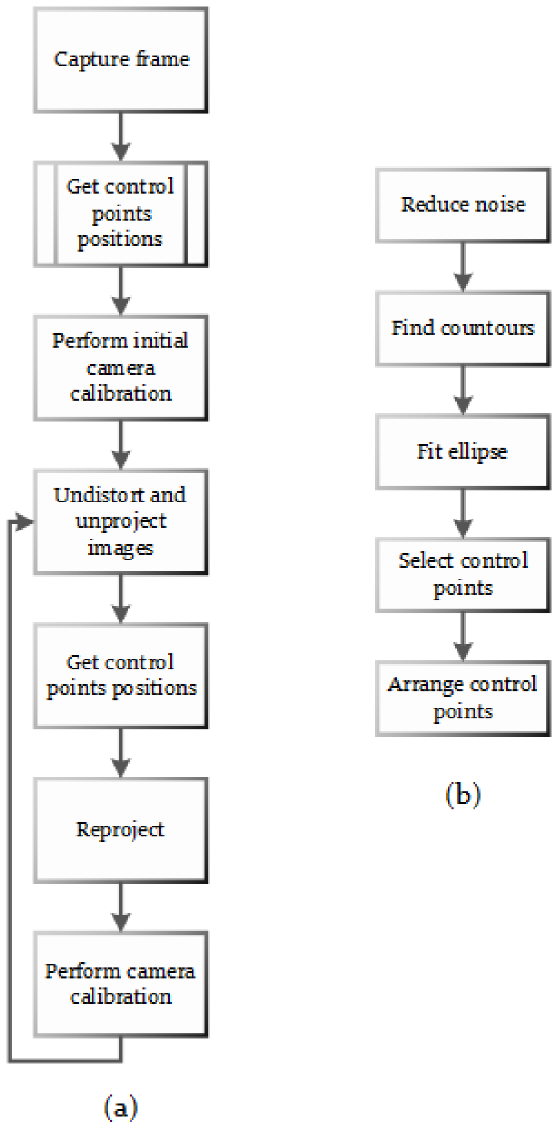
Capture frame
Get control points positions
Perform initial camera calibration
Undistort and unproject images
Get control points positions
Reproject
Perform camera calibration
(a)
Reduce noise
Find countours
Fit ellipse
Select control points
Arrange control points
(b)

Fig. 2: Proposed approach flowchart. (a) Proposed camera calibration algorithm. (b) Proposed control points localization algorithm.

1) Reduce noise: In order to detect the concentric circles in the raw images, these need to go trough a pre-process to reduce information and noise. First, we convert the raw image to grayscale, using the *cv::cvtColor* function and applying a Gaussian Blur filter, using *cv::GaussianBlur* function, this filter smooths the raw image that will help to identify the contours in the image. Then, we convert the raw image to binary image, in this case we use the *cv::adaptiveThreshold* to deal with brightness of the input images.

2) Find contours: In order to identify the contours using the *cv::findContours* function. We get all the contours in a hierarchical manner in order to identify which contours are inside others, considering that our pattern presents concentric circles, and we get the information as a tree, so we will find out parent-child relationships among the contours. Figure 3 shows one of the frames captured in real time, where this step was performed using an OpenCV function.

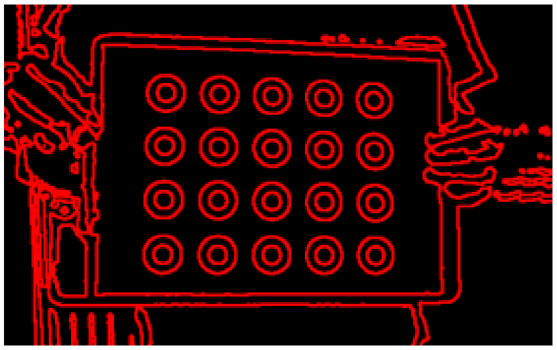

Fig. 3: Find contours in frame

3) Fit ellipse: The information of the previous step was used to find the minimum bounding ellipse that fits every contour. The centers of theses ellipses are considered as possible control points and will be filtered in the next process. OpenCV offer a function called *cv::fitEllipse* that can perfom this step.

4) Select control points: To discard the ellipses that do not belong to the grid pattern, ellipses relationships are analyzed looking for parent-child relations and compute the distance between their centers. A couple of ellipses are chosen if they are related as parent-child ellipses and the child does not have another child, as well as the center points of both, parent and child, are close located. The distance between the center points must be less than a threshold value.
As a second filter, the average point of all points that were selected in the previous process is calculated, which in most of the input images is located in the middle of the grid pattern, and then it is compared to the rest of the central points. The aim of this second filter is to choose the central points that are closer to the average point and discard the rest.

5) Arrange control points: The last step of getting control points aims to put them in order and conserve that order for all input image. In this part it is considered two cases, the first is when the pattern and the correct order were found in the previous input image, and the second is when the pattern or the order were not found in the previous input image.
As the first time there is no previous already processed image, the second case is chosen. To assert that exists a proper order between all control points, a set of combinations is generated from the input list of control points. Then, using the *cv::fitline* function to get a line where all points of the current combination should fit, however a minimum distance of displacement from the is permitted, the filter is done via a threshold. From all the combinations that were generated, the amount of selected lines is according to the pattern size height. After getting the correct lines, they are order by their position in Y-axes and then the points that belong to each line are order according to X-axes. When the pattern and the correct order are found, next input image is evaluated with the first case. In this case, the order of the pattern found in the previous image is verify by matching the previous points with the current points. If this order can not be verified in the current image, the second case will be evaluated in the next input image.
Finally, the correct order is draw in the frame that is being processed using the *cv::drawChessboardCorners* function and displayed. Figure 4 the result.

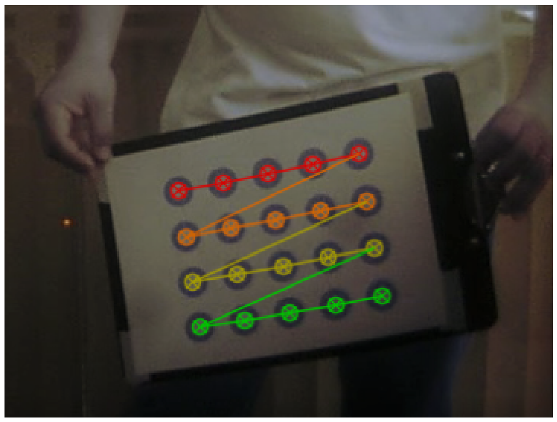

Fig. 4: Real Time Detection of Rings Pattern

Before performing an initial camera calibration, a group of frames were selected by an algorithm that get the most appropriate frames for the iterative process refinement. This frames are under the following features. A group of 25 frames were selected and stored to use in the next step.

C. Perfom the initial camera calibration

In order to perform a first calibration that will be the input camera parameter for the iterative refinement process, a set of control points per frame were stored and then used to perform camera calibration. First, frames that were selected in the previous step are loaded. Then, since finding the pattern in every selected frame is a very important and crucial task to get a correct first parameter estimation, this task was done again while collecting the control points of every frame. Using the OpenCV function *calibrateCamera*, the RMS (Root Squared Mean) was obtained for this first

D. Undistort and unproject

After one initial calibration, the process of iterative refinement is performed as an intent to reduce the reprojection error and approximate the camera parameters to the real values. This step was develop following the the process proposed by [2]. Using the initial camera calibration parameters, each input image is undistorted using the distortion coefficients and then projected into a frontal pattern. This task was implemented using an Homography matrix calculated between the model plane and its image. This matrix can be calculated following the proccess show in [1]. We assume the model plane is on

Z = 0 of the world coordinate system. Lets denote the i^th column of the rotation matrix R by r_i. Then we have:

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = A \begin{bmatrix} r_1 & r_2 & r_3 & t \end{bmatrix} \begin{bmatrix} X \\ Y \\ 0 \\ 1 \end{bmatrix}$$

$$s \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = A \begin{bmatrix} r_1 & r_2 & t \end{bmatrix} \begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}$$

We denote the model plane as M, and M = [X,Y]T since Z is always equal to 0. In turn $M' = \begin{bmatrix} X & Y & 1 \end{bmatrix}^T$. Therefore, a model point M and its image m is related by a homography H:

$$sm' = HM'$$

donde:

$$H = A \begin{bmatrix} r_1 & r_2 & t \end{bmatrix}$$

As is clear, the 33 matrix H is dened up to a scale factor. In the implementation, this step was performed using the functions $cv :: undistort$ and $cv :: undistortPoints$ to undistort the image and control points respectively. To compute the Homography matrix, the function $cv :: findHomography$ was used using the image pattern control points and a set of points calculated and arrange using an scalar of the real distance between the control points in the real image plane pattern. With all this information, the pattern region was extracted and projected onto a fronto parallel image using the function $cv :: warpPerspective$, where the control points were identify again, but this time using the function $cv :: findCirclesGrid$. With this function the localization of the control points should be more accurate since we are eliminating distortion.

E. *Reproject*

Taking a set of control points that were identify in the fronto parallel image, we reproject them onto the input image.

V. Experimental Results

Two different cameras were used to perform camera calibration and get their intrinsic parameters. We used three different videos recorded by both cameras of three different patterns: chessboard, asymmetric circles and concentric rings.

A. *Pattern Detection*

Time taken by the algorithm proposed in this paper for rings pattern detection was compared to two OpenCV functions that perform pattern detection in chessboard and asymmetric circles patterns. In addition, the amount of frames where the pattern was detected correctly were count and presented as a percentage of the total processed frames. Tables II and I shows the results for each pattern from the two different cameras. Recorded videos and real time capturing frames were tested in this phase.

Pattern	Average time (ms)	Total Detected (%)
Chessboard	87.73	25%
Asymmetric circles	91.25	31%
Rings	21.87	93%

TABLE I: *Camera 1 - Pattern Detection time and accuracy.*

Pattern	Average time (ms)	Total Detected (%)
Chessboard	89.43	92%
Asymmetric circles	94.44	95%
Rings	27.70	88%

TABLE II: *Camera 2 - Pattern Detection time and accuracy.*

B. *Camera calibration parameters and average errors*

Tables III and IV show the obtained camera parameters using the proposed approach for the two different cameras respectively. This parameters include focal lengths f_x and f_y, focal center u_0 and v_0, and lens distortion parameters k_1 and k_2. the average reprojection errors are also shown in these tables.

Parameters	Ground Truth	Initial calibration	Iterative refinement
f_x	630	679.45	629.56
f_y	630	674.62	625.79
u_0	320	321.23	324.13
v_0	240	281.69	282.81
RMS	0	0.55	0.51

TABLE III: Camera 1 - Iterative refinement of camera calibration parameters.

Parameters	Ground Truth	Initial calibration	Last Iteration
f_x	500	495.72	492.73
f_y	500	497.05	494.67
u_0	640	313.62	313.15
v_0	360	190.09	193.40
RMS	0	0.46	0.45

TABLE IV: Camera 2 - Iterative refinement of camera calibration parameters.

References

[1] Z. Zhang. A exible new technique for camera calibration. IEEE Trans. Pattern Anal. Mach. Intell., 22(11):13301334, 2000

[2] A. Datta, J. Kim, and T. Kanade. Accurate camera calibration using iterative renementof control points. In Workshop on Visual Surveillance (VS), 2009

[3] S.Asthana, Enhanced Camera Calibration for Machine Vision using OpenCV.IAES International Journal of Artificial Intelligence (IJ-AI), 2014

[4] R. Tsai. A versatile camera calibration technique for high-accuracy 3d machine vision metrology using off-the-shelf TV cameras and lenses. IEEE JRA, 3(4), 1987.

[5] Q. Chen, H. Wu, and T. Wada. Camera calibration with two arbitrary coplanar circles. In ECCV, 2004.

[6] J. Heikkila. Geometric camera calibration using circular control points. PAMI, 22(10), 2000.

[7] G. Jiang and L. Quan. Detection of concentric circles for camera calibration. In ICCV, 2005.

[8] Q.-T. Luong. Matrice Fondamentale et Calibration Visuelle sur lEnvironnement-Vers une plus grandeautonomiedessyst‘emesrobotiques. PhDthesis,UniversitedeParis-Sud,CentredOrsay, Dec. 1992

[9] S. J. Maybank and O. D. Faugeras. A theory of self-calibration of a moving camera. The International Journal of Computer Vision, 8(2):123152, Aug. 1992.