Detection of concentric circular patterns through filters, oval detection and metaheuristics.

Antoni Mauricio
Research and Innovation Center in Computer Science
Universidad Católica San Pablo
manasses.mauricio@ucsp.edu.pe

Jorge López
Research and Innovation Center in Computer Science
Universidad Católica San Pablo
jorge.lopez.caceres@usp.pe

***Abstract*—Camera calibration is one of the primary processes in computer vision, its correct calibration significantly defines efficiencies in more complex methods such as augmented reality, 3D reconstruction or application as SLAM (all cases require to obtain 3D spatial information). The calibration process is necessary to obtain 3D information from 2D images. There are different techniques based on photogrammetry and self-calibration. As a result, the intrinsic and extrinsic parameters of the camera are obtained. Much work has been done in the calibration and also in data pre-pos-processing (metaheuristics). Most authors work over methods based on two-dimensional template as the easiest path to perform and obtain the best results. However, the improvements realized with different metaheuristics can contribute to calibration process, even if these are normally not considered in calibration pipeline.**

Many metaheuristics which improve calibration process are presented and evaluated in Zhang [1]. These techniques optimally determine the calculation processes, eliminate noise in points coordinates and perform a non-linear search appropriate in a set of camera parameters. This paper aims to define the complete procedure to calibrate a camera using a flat template of concentric circular patterns and achieve optimal results with the process. By other hand, calibration process (and also segmentation for our case) can be improve in probabilistic model introduction in order to define a robust algorithm to detect the flat template in video with different sources of noise.

Keywords: Camera calibration, filters, metaheuristics.

1. Introduction

The calibration accuracy determine the accuracy of the measures that are carried out from the images. It is for this reason that it is essential to perform the camera calibration with full guarantees that the parameters obtained are like the real ones. This commitment implies both: the right choice of calibration method as well as the correct use of it. So, the calibration process should start by making a exhaustive review of the state of art over different calibration methods to choose the one that could get better results under defined conditions. Due to the large amount of work done in calibration field, it is an arduous and uncomplicated task choice of method and conditions for develop it.

The first step in calibration process is achieve to identify the flat template of concentric circular patterns, to do so, a segmentation algorithm is needed, most known segmentation algorithms and techniques are described in [2] and [3]. Segmentation process use features in common inside images to define an object, which even in human vision and perception also causes many confusions.

The second step, according to [4] is object tracking, this process allow us to follow an object of interest over a video, instead of segmenting per frame the whole video, which has highly-cost. The methods that have presented better performances in tracking are based on Monte Carlo and probabilities, considering that tracking could be described as Markov Chain [5]. We opted for Particle Filter over Extended Kalman Filter, which is the other most used method, not just due points described before but also for cost-benefit between the implementation, efficiency of the method and its computational cost.

The third step, is solves calibration equation, this approach runs from images distortions introduced by pinhole cameras (very common). Two major distortions are radial distortion and tangential distortion. Due to radial distortion, straight lines will appear curved. Its effect is more as we move away from the center of image.

The fourth step, implys to obtain fronto parallel projection using calibration camera parameters (to get undistort projected image), obteined from step third. To do so, first we have to project corrected and undistorted images to our full windows in order to calculate collinearity (if points supposedly corrected look actually as original/ printed pattern), then using others parameters like homography matrix and perspective parameters as stop-criteria we can repeat calibration process until a convergence.

Finally, to finish our ideal pipeline, first a good flat template tracking has done, many position estimations should be performed (in better way) and finally a propper calibration process have to be performed, this part gonna be explored in future presentations.

2. Methods

This section presents the mathematical and theoretical definitions of points described before.

2.1. Detection Algorithm

The algorithm works each frame in gray scales. First we use median filter to reduce impurities. Then an elliptical kernell is used for dilatation process in order to remark black contours (or white contours expanding) to highlight the rings. We calculate an adaptive threshold to turn the image into black-white scale (binarization) without affecting the brightness. we invert the result and dilated them because it must give us a very reduced result. Median-term result is used as a mask to filter the noise in edges when applying canny filter to the original image with a Gaussian filter. to this result we scan for elliptic edges and keep them limited by size. From candidate objects we choose the ones that are their centers are concentric or close, the form to look for them is by means in a histogram (every center is stored). We only choose those that are concentric with at least one more. Pipeline as is presented in figure 1, runs as follow:

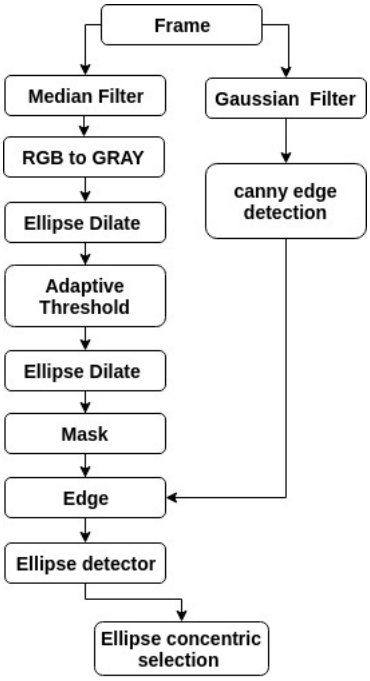

Figure 1. General pipeline for detection

2.2. Tracking Algorithm

We are implementing a particle filter (PF) version, using the basic idea of image odometry (obteined from previous frames), doing so, we could get a naive pose estimation, but good enough to implement any error correction method.

The objective of a particle filter is to estimate the posterior density of the state variables given the observation variables. The particle filter is designed for a Hidden Markov Model (HMM), where the system consists of hidden and observable variables. The observable variables (observation process) are related to the hidden variables (state-process) by some functional form that is known. Similarly the dynamical system describing the evolution of the state variables is also known probabilistically.

For our work we are implementing a generic particle filter, which estimates the posterior distribution of the hidden states (pose estimation - X_k) using the observation measurement process (object detection - z_k) and the discretization of the movement respect to the previous state. Considering a state-space shown in figure 2.

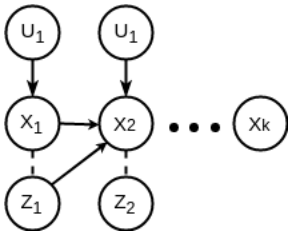

Figure 2. Markov chain used for particle filter.

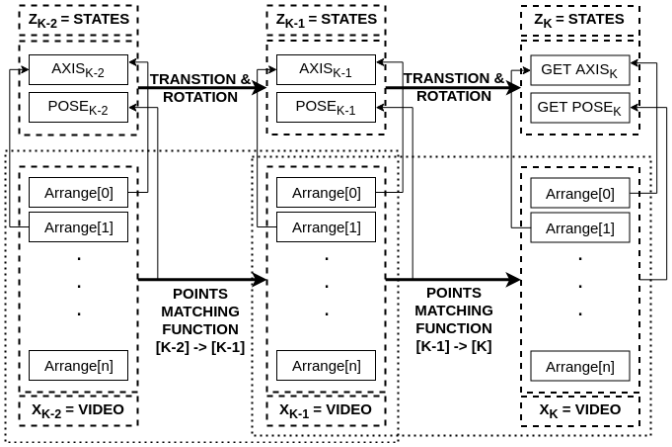

Figure 3. Particle filter representation using video/patters states.

Figure 3 presents the same idea from 2, but replacing the video states istead of Markov chain representation. First to reduce estimation error in rotation (which is normally the harder state to estimate), a rotation-translation process works from last state to new one, then a point-match function run over last and new points obteined from pattern in $k-1$ and k times, to match the last order to the new one (this assume that last order is right), finally some security parameters are verified otherwise the algorithm reboots the order.

When the pattern is lost, the algorithm tries to estimate the elements positions of the pattern using the previous frame completely detected. To do so, a cost function is evaluated between the possible elements positions of the

previous state (x_{k-1}) with the elements of the new state (x_k) using a MSE (mean squar error) of possible poses.

The Particle Filter pseudocode is presented below for same nomenclature used in figure 2:

Algorithm Particle Filter (X_{t-1}, u_t, z_t):
$\hat{X}_t = X_t = \emptyset$
for $M = 1$ **to** M **do**
  sample $x_t^{[m]} \sim (x_t)$
  $w_t^{[m]} = \frac{p(x_t^{[m]})}{\pi(x_t^{[m]})}$
  $\hat{X}_t = \hat{X}_t + \langle x_t^{[m]}, w_t^{[m]} \rangle$
end for
for $M = 1$ **to** M **do**
  draw i with probability $w_t^{[i]}$
  $x_t^{[i]}$ to X_t
end for
return X_t

2.3. Frame Distribution

For the correct calibration of the camera (and in order to avoid of parameters overfitting over an exclusive zone of the images), it is necessary to have a quasi-equitable distribution of the patterns over the total of the scene, for which a density-based pattern distribution algorithm is applied, that is, the frames are selected for calibration based on how his spatial arrangement impacts on the average density and by sectors in the scene.

The sector density method include quad segmentation over scene, in our case we define 4x5 sectors due the window size and proportion. To set the maximum number of points per zone or density (p), we establish a maximun number of frames to use in calibration process (N_{frames}), then based on number of elements per arrange ($N_{arrange}$) and total number of sections ($N_{sections}$), we define equation 1:

$$p = \frac{N_{frames} * N_{arrange}}{N_{sections}} \tag{1}$$

The following algorithm presents the pseudocode of the density selection algorithm.

Algorithm Density Distribution (Vector of IFrames):
$p = N_{frames} * N_{arrange}/N_{sections}$
for $S = 1$ **to** $N_{sections}$ **do**
  for $F = 1$ **to** N_{frames} **do**
    Evaluate density with Iframes[F]
    if Pass evaluation **then**
      Add frame to OFrames
    end if
  end for
end for
return Vector of OFrames
.

Another point to be considered is that this algorithm depends on effectiveness of detection and tracking, due if in any section of image, some frames are never captured then, this could fall. In order to prevent it, we developed an algorithm for limiting edges on the patterns captured in detection step and scrubbed by tracking algorithm. To do so, first we get min-max limits in each axis - doing this we define a new filled rectagle, easier to divide in sections above defined.

2.4. Camera Calibration

According to [6], camera distortion is solved using five camera parameters, known as distortion coefficients (DC):

$$DC = \{k_1, k_2, p_1, p_2, k_3\}$$

Where $K_n = n^{th}$ are the radial distortion coefficients and $P_n = n^{th}$, the tangential distortion coefficients.

The radial distortion is irregular, the most commonly encountered distortions are radially symmetric, or approximately so, arising from the symmetry of a photographic lens.

$$\begin{bmatrix} X_{corrected} = x(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \\ y_{corrected} = y(1 + k_1 r^2 + k_2 r^4 + k_3 r^6) \end{bmatrix} \tag{2}$$

Similarly, another distortion is the tangential distortion which occurs because image taking lense is not aligned perfectly parallel to the imaging plane. So some areas in image may look nearer than expected. It is solved in 3:

$$\begin{bmatrix} X_{corrected} = x + [2p_1 xy + p_2(r^2 + 2x^2)] \\ y_{corrected} = y + [p_1(r^2 + 2y^2) + 2p_2 xy] \end{bmatrix} \tag{3}$$

In addition to this, we need to find a few more information, like intrinsic and extrinsic parameters of the camera. Intrinsic parameters are specific to a camera. It includes information like focal length (f_x, f_y), optical centers (c_x, c_y) and others. It is also called camera matrix (CM). It depends on the camera only, so once calculated, it can be stored for future purposes. Extrinsic parameters corresponds to rotation and translation vectors which translates a coordinates of a 3D point to a coordinate system. It is expressed as a $3x3$ matrix:

$$CM = \begin{bmatrix} f_x & 0 & c_x \\ 0 & f_y & c_y \\ 0 & 0 & 1 \end{bmatrix} \tag{4}$$

In order to reduce camera calibration process we decided to use **calibrateCamera** function from *OpenCV*. **calibrateCamera** function uses ordered arrays obteined from detection/tracking algorithm and windows size, and returns camera calibration matrix, distortion coefficients ($k_1, k_2, p_1, p_2[, k_3[, k_4, k_5, k_6]]$) of 8 elements, rotation and translation vectors.

2.5. Fronto Parallel projection

Is the view obtained by eliminating the perspective in an image. First a homography matrix is defined as the transformation between two planes inclusing scale factor (S), as can be seen in equation 5:

$$S \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = H \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} \quad (5)$$

The homography matrix is a $3x3$ matrix but with 8 degrees of freedom as it is estimated up to a scale. It is generally normalized considering $h_{33} = 1$ or $h_{11}^2 + h_{12}^2 + h_{13}^2 + h_{21}^2 + h_{22}^2 + h_{23}^2 + h_{31}^2 + h_{32}^2 + h_{33}^2 = 1$.

To get fronto parallel projection from each patter we first use corners in arrange corresponding to extreme elements (4 points) in **getPerspectiveTransform** function, this function allow us to obtain homography matrix from normal video to fronto parallel projection, then using **warpPerspective** we finally get fronto parallel view (with or without distotions depending on camera calibration process) then to get the perspective matrix to turn back to normal view we just have to apply **getPerspectiveTransform** but now considering the normal view as a destination and fronto parallel view as source. This process became very usefull in order to calculate collinearity and as stop-criteria in the iterative process to refine calibration.

2.6. Iterative Calibration Refinement

The first calibration considerably reduces the distortion error, this can be observed visually in the corrected image (with corrected distortion) or by means of descriptors obtained from the corrected image, among the most important characteristics is the collinearity of the elements of each pattern (that the line formed by the elements stays in the direction). In order to improve the calibration, it is necessary to perform an iterative refinement of the calibration, to do so the parallel fronto projection of the corrected image is used (so the collinearity is made in the parallel frontal plane).

Then, when performing the inverse projection of the fronto plane parallel to the normal view with distortion, new points are obtained on which the calibration calculations can be made. From these 3 planes (parallel fronto, original/distorted and corrected), new control points can be estimated for next calibration process. A new calibration process which use these control points, improves image distortion and collinearity errors.

When this process is performed iteratively, a greater gradient in descending order is expected in the error decay curve, this obserbations are exposed in the results sections of this paper. For our case, the barycenter was calculated between the control points of the 3 planes for the next step in the calibration.

The iterative calibration refinement process described above is presented in figure 4.

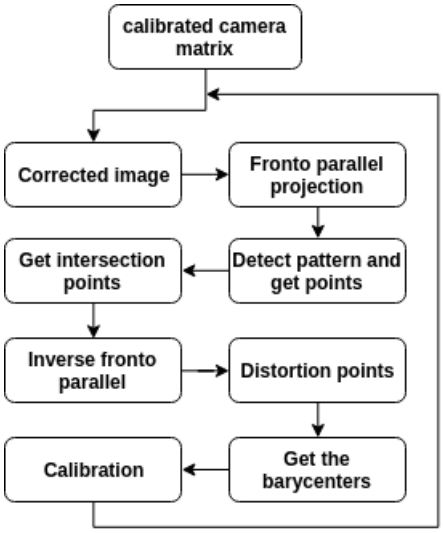

Figure 4. General pipeline in iterative calibration refinement

3. Results

This section presents results from all algorithms an modifications, errors included. So, each subsection shows results step by steps focusing on camera calibration and fronto-parallel parameters. Due most codes were developed for concentric ring pattern, most of our results are oriented to that particular patter (except for calibration and fronto-parallel) because chessboard and asymmetrical circles have a their own detection/tracking function in *OpenCV*.

3.1. Pattern Recognition

Based on Gaussian Filter and Canny Edge Detection, we perform new results for feature recognition over elliptical patterns, those results are presented on figure **??**. As can be seen, the added features turn our pipeline in a quite robust system, with less deficiencies for situations that in the previous samples were adverse.

3.2. Tracking

For tracking, we use the first two frames in order to estimate initial conditions (needed for future estimations), this first initial variables are: angle and pose (usings points in arrange obtained from previous steps). Figure 6 shows average results from tracking.

The main goal was solved, the method understand the pattern as a full element, which is composed by elements (rings/chessboard/circles centroids). But a mal-detection of elements in rotation could cause a total estimation confution, and the main consecuence to fail in this part are future mal-estimation of states. To avoid it, we define a flag for special cases like: mal-detection in rotation or total missing of points. When those situations happens then all systems is rebooted matching last states calculated with new states estimated.

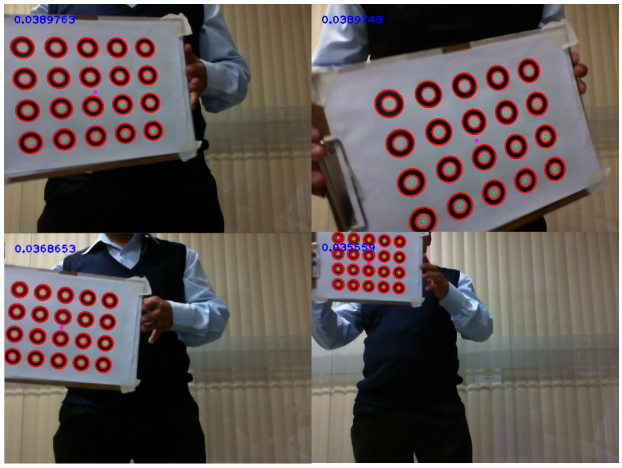

Figure 5. Concentric rings detection using pipeline in section 2.1

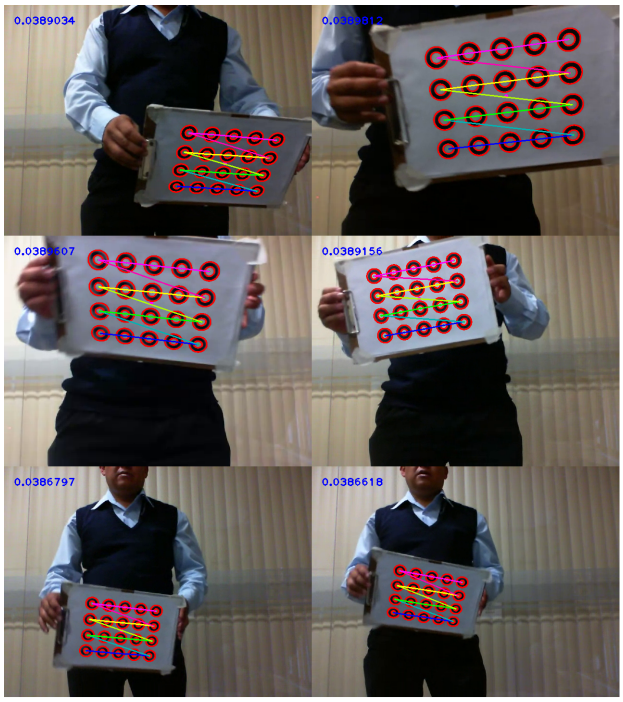

Figure 6. Concentric rings tracking results, the frames shows the results of Particle Filter Tracking in video, the video is presented by frames from top to down

As can be seen in figure 6, even in rotation the pattern are tracked. In some frames, a partial or total mal-detection happens (in middle of frames showed) then the cost function machting starts to work.

In case of partial lose or total lose, the algorithm try to match elements from previous frame detected and new one. This is show in figure 7.

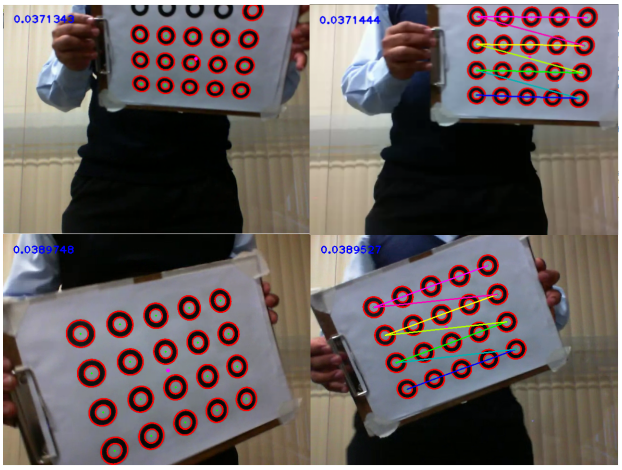

Figure 7. Matching function working over a pattern with partial or total mal-detection. In this case

3.3. Density Distribution

As was described before, density distribution could be very useful for calibration, due parameter obteined from a well-distributed vector of arranges should ensure a good distortion correction over the whole image. Image **??** shows a good distribution over scene using our algorithm, then **??** shows the same number of frames per pattern but using a frame capture by time, then in tables the collinearity error could be compared regarding good/ bad frames distribution.

Figure 8 shows distribution results for concentric ring pattern in both videos, for 75 frames in 20 (4x5) sections.

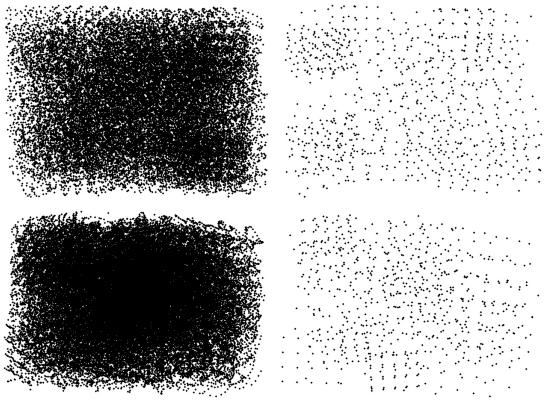

Figure 8. Patterns distribution over the image for calibration. Left: Non sampled distribution. Right: Sampled distribution. In descending order: LifeCam and PS3 videos

Another improvement added in 8 was a shuffle of possible points to be consider. Normally the distribution using density by sections had a segmented appearance than the dispersion presented in the figure 8. But using a random shuffle before choose final frames, we get a more uniform distribution.

3.4. Calibration

After tracking, and using *OpenCV* calibration function for 3 patters (P): Concentric rings (2), Assimetric disks (1) and Chessboard (0), to calibrate 2 cameras: PS3 and LifeCam. (c_x, c_y) is a principal point that is usually at the image center (f_x, f_y) are the focal lengths expressed in pixel units, $Coll$ is collinearity calculated using average deviation of points from vectors formed with arrange corners.

P	F_x	F_y	C_x	C_y	$Coll$	**RMS**
N_F = 25						
0	598.095	593.730	348.270	235.708	0.0056899	0.4408
1	619.877	615.677	359.883	217.080	0.0041629	0.2495
2	587.567	582.977	310.336	222.569	0.0052875	0.2555
N_F = 35						
0	588.038	589.47	315.49	228.460	0.0054542	0.7608
1	622.340	625.00	340.08	205.049	0.0048213	0.1734
2	615.420	617.19	325.00	223.590	0.0054572	0.1817

TABLE 1. LIFECAM CALIBRATION RESULTS

P	F_x	F_y	C_x	C_y	$Coll$	**RMS**
N_F = 25						
0	850.705	840.663	270.959	289.313	0.0067586	0.3721
1	859.836	858.969	367.794	259.614	0.0043134	0.2918
2	842.759	836.334	340.519	249.733	0.0022995	0.2956
N_F = 35						
0	854.051	856.016	348.220	250.09	0.0068142	0.2254
1	845.390	846.885	338.313	248.68	0.0044587	0.2104
2	844.195	847.380	340.606	260.84	0.0024955	0.2012

TABLE 2. PS3 CALIBRATION RESULTS

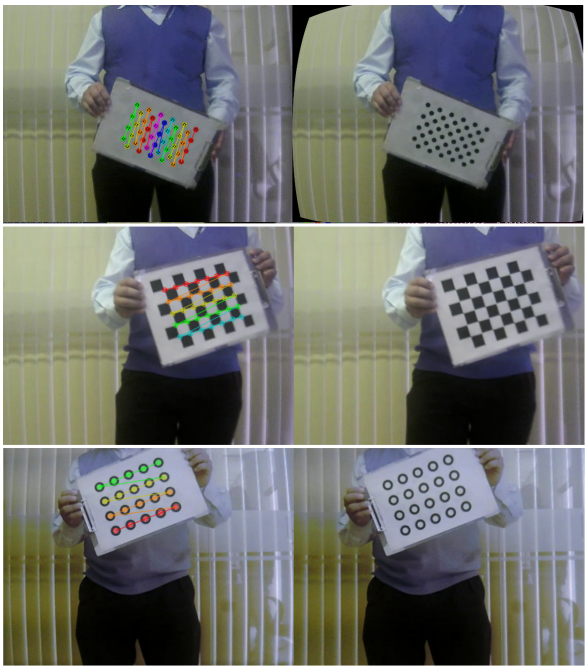

Figure 9. Image correction over pattern in PS3 camera. Left: original frames. Right: undistorted frames

The Table 1 shows full results considering number of frames (N_F) used for calibration in LifeCam camera, while 2 shows results in PS3 camera.

Corrected frames using calibration parameters are shown in figure 9 for concentric rings, chessboard and assimetric disks patterns in PS3 camera (due the correction is visually more remarkable in this camera), for this purpose we use **undistort** function from *OpenCV*, passing output parameters obteined from **calibrateCamera** function.

3.5. Calibration using density distribution

To improve camera callibration parameters and in order to reduce RMS and collinearity, we use density function described in sections 2.3 and 3.3. Visual results are similar like 9, even in PS3 camera. So table 3 show results using 25 frames in calibration for both cameras (C_T).

P	F_x	F_y	C_x	C_y	Cl	**RMS**
C_T = LifeCam						
0	607.510	601.122	355.519	238.528	0.0066566	0.4881
1	598.650	599.681	361.555	216.388	0.0046648	0.2450
2	586.771	581.560	311.889	226.199	0.0049612	0.2291
C_T = PS3						
0	808.407	798.441	283.756	248.286	0.0050957	0.4330
1	878.781	883.128	395.578	319.782	0.0043304	0.3084
2	844.247	837.769	344.901	253.990	0.0023091	0.2982

TABLE 3. CALIBRATION RESULTS USING DENSITY DISTRIBUTION CAPTION OF FRAMES FOR BOTH CAMERAS FOR 25 FRAMES

3.6. Fronto Parallel projection

Following the pipeline presented on figure 1, we proceed to obtain the front-parallel view over the patterns. For this purpose as explained in section 2.5, we define the limit points (corners) of each pattern as the space to be projected (S_o) and the limits of the frontal screen as the space where the pattern have to be projected (S_f). Results in LifeCam camera are plotted in figure **??**, while figure **??** present results in PS3 camera.

3.7. Iterative refinement

As explained in section 2.6, it is likely that the results using an iterative refinement method for calibration (generating new control points over the calibration functions of the camera and parallel fronto projection). For this reason and according to the results of table 3 (better results using density for the concentric ring pattern, as proposed in [7]) the pipeline presented in figure 1 was proposed, also using the frame selection function by density. Results are presented in tables 4 and 5 for LifeCam and PS3 cameras respectively with 6 iterations (It).

As can be seen, mostly results from concentric ring pattern and assymetric disks are affected in iterations, but in general after effects produced by the first iteration, RMS and collinearity are not reduced much more in that magnitude.

	Chessboard		Assim. disks		Concc. rings	
I	**RMS**	$Coll$	**RMS**	$Coll$	**RMS**	$Coll$
1	0.4408	0.0056899	0.2495	0.0041629	0.2554	0.0052875
2	0.4740	0.0057268	0.2178	0.0040713	0.2240	0.0049633
3	0.5177	0.0074094	0.2392	0.0045387	0.2305	0.0049960
4	0.5126	0.0081552	0.2190	0.0050206	0.2318	0.0047513
5	0.5276	0.0067847	0.2351	0.0038565	0.2260	0.0049553
6	0.4881	0.0066566	0.2448	0.0046648	0.2291	0.0049612

TABLE 4. ITERATIVE REFINEMENT CALIBRATION RESULTS USING DENSITY DISTRIBUTION CAPTION FOR LIFECAM CAMERA

	Chessboard		Assim. disks		Concc. rings	
I	**RMS**	$Coll$	**RMS**	$Coll$	**RMS**	$Coll$
1	0.3721	0.0067586	0.2918	0.0043134	0.2956	0.0022995
2	0.4424	0.0055703	0.3284	0.0044416	0.2958	0.0022504
3	0.4137	0.0047987	0.3249	0.0046165	0.3006	0.0023319
4	0.3735	0.0058645	0.3136	0.0046498	0.2975	0.0023014
5	0.4806	0.0060481	0.2894	0.0047778	0.2967	0.0023171
6	0.4329	0.0050957	0.3084	0.0043304	0.2982	0.0023091

TABLE 5. ITERATIVE REFINEMENT CALIBRATION RESULTS USING DENSITY DISTRIBUTION CAPTION FOR PS3 CAMERA

4. Conclusions

As seen in the results, the asymmetric circles and rings patterns are better than the chessboard because they reduce the error by detecting their centers instead of edges and vertices, which ussually tend to fail. This generates an error in the calibration calculus obtaining very variable and distant results. This phenomenon is corrected using the iterative method but it does not apply to the rest of patterns.

Between the assymetric circles and concentric rings, the last one have several concentric circles that help reduce the error when calculating the centers of the pattern.

The iterative method does not imply a great advantage (in comparison with results obtained selection density function), it helps to reduce the error but it is not very significant, only in cases of cameras with greater distortion it is possible to appreciate like in ps3 case and for patterns that do not have much noise like concentric rings.

Acknowledgments

This work was supported by grant 234-2015-FONDECYT (Master Program) from Cienciactiva of the National Council for Science,Technology and Technological Innovation (CONCYTEC-PERU).

References

[1] Zhang Z. (1998) A flexible new technique for camera calibration. Technical Report MSR-TR-98-71, Microsoft Research.

[2] Meier, Thomas, and King Ngi Ngan. "Automatic segmentation of moving objects for video object plane generation." IEEE Transactions on Circuits and Systems for Video Technology 8.5 (1998): 525-538.

[3] Cucchiara, Rita, et al. "Detecting moving objects, ghosts, and shadows in video streams." IEEE transactions on pattern analysis and machine intelligence 25.10 (2003): 1337-1342.

[4] Yilmaz, Alper, Omar Javed, and Mubarak Shah. "Object tracking: A survey." Acm computing surveys (CSUR) 38.4 (2006): 13.

[5] Yang, Changjiang, Ramani Duraiswami, and Larry Davis. "Fast multiple object tracking via a hierarchical particle filter." Computer Vision, 2005. ICCV 2005. Tenth IEEE International Conference on. Vol. 1. IEEE, 2005.

[6] Szeliski, Richard. Computer vision: algorithms and applications. Springer Science and Business Media, 2010.

[7] Datta, Ankur, Jun-Sik Kim, and Takeo Kanade. "Accurate camera calibration using iterative refinement of control points." Computer Vision Workshops (ICCV Workshops), 2009 IEEE 12th International Conference on. IEEE, 2009.