Correcting Radial Distortion of Cameras with Wide Angle Lens Using Point Correspondences

Leonardo Romero and Cuauhtemoc Gomez

Universidad Michoacana de San Nicolas de Hidalgo
Morelia, Mich., 58000, Mexico
lromero@umich.mx, cgomez@codinet.com.mx

Abstract. Digital cameras with wide angle lens are typically used in mobile robots to detect obstacles near the robot. Unfortunately cameras with wide angle lens capture images with a radial distortion. This paper describe the problems associated with severe radial distortion, review some previous relevant approaches to remove the distortion, and presents a robust method to solve the problem. The main idea is to get feature points, which must be easily and robustly computed, from a pattern image (which is in front of the camera) and from the distorted image (acquired by the camera). An iterative non-linear optimization technique is used to match feature points from one image to the other. Experimental results show the robustness of the new method. A Linux implementation of this approach is available as a GNU public source code.

1 Introduction

Most algorithms in 3-D Computer Vision rely on the pinhole camera model because of its simplicity, whereas video optics, especially wide–angle lens, generate a lot of non–linear distortion. In some applications, for instance in stereo vision systems, this distortion can be critical.

Camera calibration consists of finding the mapping between the 3-D space and the camera plane. This mapping can be separated in two different transformations: first, the relation between the origin of 3–D space (the global coordinate system) and the camera coordinate system, which forms the external calibration parameters (3–D rotation and translation), and second the mapping between 3–D points in space (using the camera coordinate system) and 2–D points on the camera plane, which form the internal calibration parameters [1].

This paper introduces a new method to find the internal calibration parameters of a camera, specifically those parameters related with the radial distortion due to wide–angle lens.

The new method works with two images, one from the camera and one from a calibration pattern (without distortion) and it is based on a non–linear optimization method to match feature points of both images, given a parametric distortion model. The image from the calibration pattern can be a scanned image, an image taken by a high quality digital camera (without lens distortion), or even the binary image of the pattern (which printed becomes the pattern).

First, a set of feature point correspondences between both images are computed automatically. The next step is to find the best distortion model that maps the feature points from the distorted image to the calibration pattern. This search is guided by analytical derivatives with respect to the set of calibration parameters. The final result is the set of parameters of the best distortion model.

The rest of this paper is organized as follows. Section 2 describe the problem to compute transformed images and it presents the Bilinear Interpolation as a solution to that problem. Sections 3 and 4 describe the distortion and projective model that we are using. Section 5 presents the method to match pairs of points. A brief comparison with previous calibration methods is found in section 6. Here we show the problems associated with cameras using wide angle lens and why some previous methods fails or require a human operator. Besides the advantages of the proposed method are highlighted in this section. Experimental results are shown in Section 7. Finally, some conclusions are given in Section 8.

2 Computing Transformed Images

For integer coordinates $\langle i,j \rangle$, let $I(i,j)$ gives the intensity value of the pixel associated to position $\langle i,j \rangle$ in image I. Let I_0 and I_1 be the original and transformed image, respectively. A geometric transformation, considering a set Θ of parameters, computes pixels of the new image, $I_1(i,j)$ in the following way:

$$I_1(i,j) = I_0(x(\Theta,i,j), y(\Theta,i,j)) \tag{1}$$

If $x(\Theta,i,j)$ and $y(\Theta,i,j)$ are outside of the image I_0, a common strategy is to assign zero value which represents a black pixel. But, What happen when $x(\Theta,i,j)$ and $y(\Theta,i,j)$ have real values instead of integer values? Remember that image $I_0(x,y)$ have only valid values when x and y have integer values. An inaccurate method to solve this problem is to use their nearest integer values, but next section presents a much better method to interpolate a pixel of real coordinates (x,y) from an image I.

From other point of view, pixel $I_0(x,y)$ moves to the position $I_1(i,j)$. However most transformations define points of the new image given points of the original image. In that case, to apply the bilinear transformation, we need to compute the inverse transformation that maps new points (or coordinates) to points (or coordinates) of the original image.

2.1 Bilinear Interpolation

If x_i and x_f are the integer and fractional part of x, respectively, and y_i and y_f the integer and fractional part of y, Figure 1 illustrates the bilinear interpolation method [3] to find $I(x_i + x_f, y_i + y_f)$, given the four nearest pixels to position $(x_i + x_f, y_i + y_f)$: $I(x_i, y_i)$, $I(x_i + 1, y_i)$, $I(x_i, y_i + 1)$, $I(x_i + 1, y_i + 1)$ (image values at particular positions are represented by vertical bars in Figure 1). First

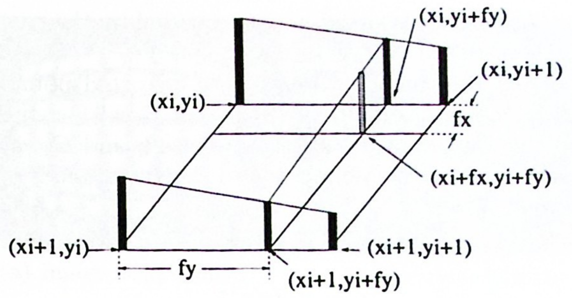
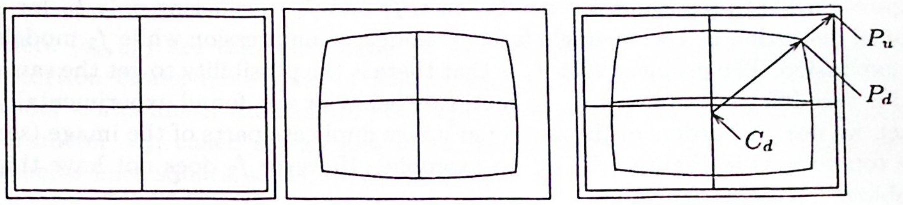


Fig. 1. Using Bilinear Interpolation



(a) Pattern in front of the camera (b) Image captured by the camera (c) Both images

Fig. 2. The distortion process due to lens

two linear interpolations are used to compute two new values ($I(x_i, y_i + y_f)$ and $I(x_i + 1, y_i + y_f)$) and then another linear interpolation is used to compute the desired value ($I(x_i + x_f, y_i + y_f)$) from the new computed values:

$$\begin{aligned} I(x_i, y_i + y_f) &= (1 - y_f)I(x_i, y_i) + y_f I(x_i, y_i + 1) \\ I(x_i + 1, y_i + y_f) &= (1 - y_f)I(x_i + 1, y_i) + y_f I(x_i + 1, y_i + 1) \\ I(x_i + x_f, y_i + y_f) &= (1 - x_f)I(x_i, y_i + y_f) + x_f I(x_i + 1, y_i + y_f) \end{aligned} \tag{2}$$

Using the bilinear interpolation, a smooth transformed image is computed. Now we are able to deal with transformations associated with cameras. In section 5.3 we describe the process to build new images from distorted images and the set of parameters of the distortion and projection model.

3 The Distortion Model

The distortion process due to wide–angle lens is illustrated in Figure 2. Figure 2 (b) shows an image taken from the camera when the pattern shown in Figure 2 (a) is in front of the camera. Note the effect of lens, the image is distorted, specially in those parts far away from the center of the image.

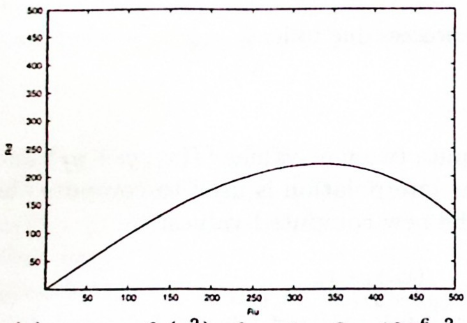
Figure 2 (c) shows the radial distortion in detail, supposing that the center of distortion is the point C_d with coordinates (c_x, c_y) (not necessarily the center of the image). Let I_d be the distorted image captured by the camera and I_u the undistorted image associated to I_d.

In order to correct the distorted image, the distorted point at position P_d with coordinates (x_d, y_d) in I_d should move to point P_u with coordinates (x_u, y_u). Let r_d and r_u be the Euclidian distance between P_d and C_d, and between P_u and C_d, respectively. The relationship between radial distances r_d and r_u can be modeled in two ways:

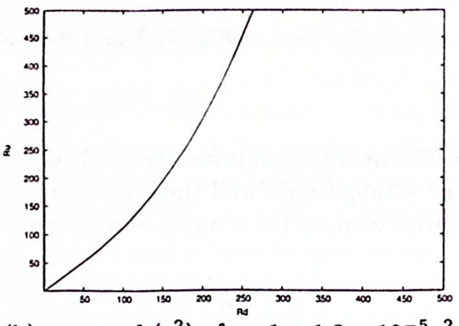
$$r_d = r_u f_1(r_u^2) \tag{3}$$

$$r_u = r_d f_2(r_d^2) \tag{4}$$

Approximations to arbitrary functions f_1 and f_2 may be given by a Taylor expansion: $(f_1(r_u^2) = 1 + k_1 r_u^2 + k_2 r_u^4 + \cdots)$ and $(f_2(r_d^2) = 1 + k_1 r_d^2 + k_2 r_d^4 + \cdots)$. Figure 3 helps to see the difference between f_1 and f_2 considering only k_1 for a typical distortion in a wide–angle lens. f_1 models a compression while f_2 models an expansion. The problem with f_1 is that there is the possibility to get the same r_d for two different values of r_u. In fact, this behavior was found experimentally when we use f_1, borders of the corrected image duplicate parts of the image (see the top corners in Figure 4(b) for an example). However f_2 does not have this problem.



(a) $r_d = r_u f_1(r_u^2)$, $f_1 = 1 - 3 \times 10^{-6} r_u^2$



(b) $r_u = r_d f_2(r_d^2)$, $f_2 = 1 + 1.3 \times 10^{-5} r_d^2$

Fig. 3. Two different functions to model the distortion of images

From now on, we consider only eq. 4. Experimentally we found that we need to consider four terms for f_2, to remove the distortion due to wide–angle lens. Then, the coordinates (x_u, y_u) of P_u can be computed by:

$$\begin{aligned} x_u &= c_x + (x_d - c_x) f_2(r_d^2) \\ &= c_x + (x_d - c_x)(1 + k_1 r_d^2 + k_2 r_d^4 + k_3 r_d^6) \\ y_u &= c_y + (y_d - c_y) f_2(r_d^2) \\ &= c_y + (y_d - c_y)(1 + k_1 r_d^2 + k_2 r_d^4 + k_3 r_d^6) \\ r_d^2 &= (x_d - c_x)^2 + (y_d - c_y)^2 \end{aligned} \tag{5}$$

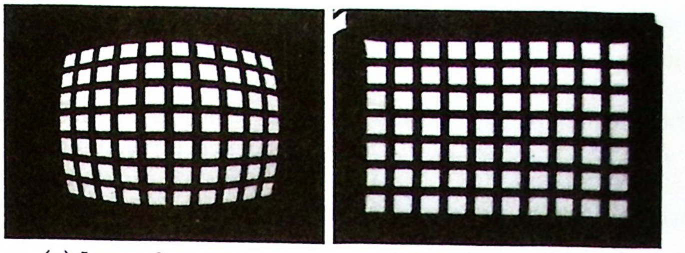

(a) Image from camera (b) Corrected image

Fig. 4. An example of a wrong correction using f_1

where (c_x, c_y) are the coordinates of the center of radial distortion. So, this distortion model have a set of five parameters $\Theta^d = \{c_x, c_y, k_1, k_2, k_3\}$. This model works fine if the camera have square pixel, but if not, we need another parameter, s_x, called aspect ratio that divide the term $(x_d - c_x)$. Since most cameras have square pixels, we consider $s_x = 1$.

4 The Projection Model

Figure 2 shows and ideal case, where the plane of the pattern is parallel to the camera plane and center of the pattern coincides with the optical axis of the camera. Using homogeneous coordinates, the class of 2–D planar projective transformations between the camera plane and the plane of the undistorted image is given by [8] $[x'_p, y'_p, w'_p]^t = M[x'_u, y'_u, w'_u]^t$, where matrix M is called an homography and it has eight independent parameters,

$$M = \begin{bmatrix} m_0 & m_1 & m_2 \\ m_3 & m_4 & m_5 \\ m_6 & m_7 & 1 \end{bmatrix}$$

Plane and homogeneous coordinates are related by $(x_p = x'_p/w'_p,\ y_p = y'_p/w'_p)$ and $(x_u = x'_u/w'_u,\ y_u = y'_u/w'_u)$. So, a point $P_u(x_u, y_u)$ in image I_u moves to $P_p(x_p, y_p)$ in the new projected image I_p. Assigning $w'_u = 1$, the new coordinates of P_p are given by:

$$x_p = \frac{m_0 x_u + m_1 y_u + m_2}{m_6 x_u + m_7 y_u + 1},\ y_p = \frac{m_3 x_u + m_4 y_u + m_5}{m_6 x_u + m_7 y_u + 1} \qquad (6)$$

And now the projection parameters are $\Theta^p = \{m_0, m_1, m_2, m_3, m_4, m_5, m_6, m_7\}$.

5 The Point Correspondences Method

The goal is to find a set of parameters Θ^d and Θ^p that transform the distorted image capture by the camera, I_d into a new projected image, I_p, that match the image, I_r, of the calibration pattern put in front of the camera. To do that, a set of point correspondences are extracted from I_d and I_r (see section 5.2 for details).

Let n be the number of features, (x_{rk}, y_{rk}) be the coordinates of the k–th feature in I_r and (x_{dk}, y_{dk}) be its correspondence in I_d. From (x_{dk}, y_{dk}) and using eq. 5, we can compute (x_{uk}, y_{uk}) and using eq. 6, we can get the coordinates (x_{pk}, y_{pk}) of the projected feature. So we have a set of pairs of points $C = \{< (x_{r1}, y_{r1}), (x_{p1}, y_{p1}) >, \cdots, < (x_{rn}, y_{rn}), (x_{pn}, y_{pn}) >\}$.

We formulate the goal of the calibration as to find a set of parameters $\Theta = \Theta^p \cup \Theta^d$ such the sum, E, of square distances between projected points and reference points is a minimum,

$$\begin{aligned} e_{xk} &= x_p(\Theta, x_{dk}, y_{dk}) - x_{rk} \\ e_{yk} &= y_p(\Theta, x_{dk}, y_{dk}) - y_{rk} \\ E &= \sum_{k=1}^{n} e_{xk}^2 + e_{yk}^2 \\ \Theta &= argmin\ E(\Theta) \end{aligned} \tag{7}$$

5.1 Non-Linear Optimization

The Gauss-Newton-Levenberg-Marquardt method (GNLM) [6] is a non–linear iterative technique specifically designated for minimizing functions which has the form of sum of square functions, like E. At each iteration the increment of parameters, vector $\delta\Theta$, is computed solving the following linear matrix equation:

$$[A + \lambda I]\, \delta\Theta = B \tag{8}$$

If there is n point correspondences and q parameters in Θ, A is a matrix of dimension $q \times q$ and matrix B has dimension $q \times 1$. λ is a parameter which is allowed to vary at each iteration. After a little algebra, the elements of A and B can be computed using the following formulas,

$$a_{i,j} = \textstyle\sum_{k=1}^{n}\left(\frac{\partial x_{pk}}{\partial \theta_i}\frac{\partial x_{pk}}{\partial \theta_j} + \frac{\partial y_{pk}}{\partial \theta_i}\frac{\partial y_{pk}}{\partial \theta_j}\right),\ b_i = -\textstyle\sum_{k=1}^{n}\left(\frac{\partial x_{pk}}{\partial \theta_i}e_{xk} + \frac{\partial y_{pk}}{\partial \theta_i}e_{yk}\right) \tag{9}$$

In order to simplify the notation, we use x_p instead of x_{pk} and y_p instead of y_{pk}. Then, $\frac{\partial x_p}{\partial \theta_i}$ and $\frac{\partial y_p}{\partial \theta_i}$ for $(\theta_i \in \Theta^p)$ can be derived from eq. 6,

$$\begin{array}{ll} \frac{\partial x_p}{\partial m_0} = \frac{x_u}{D} & \frac{\partial y_p}{\partial m_0} = 0 \\ \frac{\partial x_p}{\partial m_1} = \frac{y_u}{D} & \frac{\partial y_p}{\partial m_1} = 0 \\ \frac{\partial x_p}{\partial m_2} = \frac{1}{D} & \frac{\partial y_p}{\partial m_2} = 0 \\ \frac{\partial x_p}{\partial m_3} = 0 & \frac{\partial y_p}{\partial m_3} = \frac{x_u}{D} \\ \frac{\partial x_p}{\partial m_4} = 0 & \frac{\partial y_p}{\partial m_4} = \frac{y_u}{D} \\ \frac{\partial x_p}{\partial m_5} = 0 & \frac{\partial y_p}{\partial m_5} = \frac{1}{D} \\ \frac{\partial x_p}{\partial m_6} = \frac{-x_u x_p}{D} & \frac{\partial y_p}{\partial m_6} = \frac{-x_u y_p}{D} \\ \frac{\partial x_p}{\partial m_7} = \frac{-y_u x_p}{D} & \frac{\partial y_p}{\partial m_7} = \frac{-y_u y_p}{D} \end{array} \tag{10}$$

Where $D = m_6 x_u + m_7 y_u + 1$. Partial derivatives of distortion parameters are derived from eq. 5 and two applications of the chain rule,

$$\frac{\partial x_p}{\partial \theta_i} = \frac{\partial x_p}{\partial x_u}\frac{\partial x_u}{\partial \theta_i} + \frac{\partial x_p}{\partial y_u}\frac{\partial y_u}{\partial \theta_i}, \quad \frac{\partial y_p}{\partial \theta_i} = \frac{\partial y_p}{\partial x_u}\frac{\partial x_u}{\partial \theta_i} + \frac{\partial y_p}{\partial y_u}\frac{\partial y_u}{\partial \theta_i} \tag{11}$$

$$\begin{array}{l} \frac{\partial x_p}{\partial x_u} = (Dm_0 - (m_0 x_u + m_1 * y_u + m_2)m_6)/D^2, \ \frac{\partial x_p}{\partial y_u} = (Dm_1 - (m_0 x_u + m_1 * y_u + m_2)m_7)/D^2 \\ \frac{\partial y_p}{\partial x_u} = (Dm_3 - (m_3 x_u + m_4 * y_u + m_5)m_6)/D^2, \ \frac{\partial y_p}{\partial y_u} = (Dm_4 - (m_3 x_u + m_4 * y_u + m_5)m_7)/D^2 \end{array} \tag{12}$$

Finally, the last set of formulas are derived from eq. 5,

$$\begin{array}{l} \frac{\partial x_u}{\partial k_1} = r_d^2(x_d - c_x) \\ \frac{\partial y_u}{\partial k_1} = r_d^2(y_d - c_y) \\ \frac{\partial x_u}{\partial k_2} = r_d^4(x_d - c_x) \\ \frac{\partial y_u}{\partial k_2} = r_d^4(y_d - c_y) \\ \frac{\partial x_u}{\partial k_3} = r_d^6(x_d - c_x) \\ \frac{\partial y_u}{\partial k_3} = r_d^6(y_d - c_y) \\ \frac{\partial x_u}{\partial c_x} = 1 - (1 + k_1 r_d^2 + k_2 r_d^4 + k_3 r_d^6) - 2(k_1 + 2k_2 r_d^2 + 3k_3 r_d^4)(x_d - c_x)^2 \\ \frac{\partial y_u}{\partial c_x} = -2(k_1 + 2k_2 r_d^2 + 3k_3 r_d^4)(x_d - c_x)(y_d - c_y) \\ \frac{\partial x_u}{\partial c_y} = -2(k_1 + 2k_2 r^2 + 3k_3 r_d^4)(x_d - c_x)(y_d - c_y) \\ \frac{\partial y_u}{\partial c_y} = 1 - (1 + k_1 r_d^2 + k_2 r_d^4 + 3k_3 r_d^6) - 2(y_d - c_y)^2(k_1 + 2k_2 r_d^2 + 3k_3 r_d^4) \end{array} \tag{13}$$

where r was defined previously in eq. 5. Next section describes how to compute feature points from each image, as well as their correspondences automatically.

5.2 Selecting Feature Points

As we can see in Figure 4 (a), the image has white squares over a black background. As robust feature points we select the *center of mass* of each one of the white squares (or distorted white squares) of both images. The mass of each pixel is its gray level in the range [0 – 255] (0 for black pixels and 255 for white pixels).

In the implementation, once a white pixel is found (considering a given threshold), its cluster is identified visiting its neighbors recursively, and the center of mass is computed from all pixels in the cluster.

To compute automatically point correspondences, we assume that the array of white squares in each image is centered, specially in the case of the image from the camera. This is not a problem with the reference pattern, because we use the perfect graphic file.

We also assume that the image from the camera does not have a significant rotation, relative to the reference image. In this way, *bad clusters* (for instance when the camera capture some white areas outside of the calibration pattern) can be eliminated because the *good clusters* are closer to the image center. The correspondences are also computed automatically if the image holds the relative correspondence of clusters with their neighbors. For instance the top–rightmost clusters in the image of the camera and in the reference image, are the closest ones to the top–right corner of the image.

5.3 Computing Corrected Images

If we compute a set of parameters Θ we are able to map a point (x_d, y_d) into a new projected point (x_p, y_p). But to compute a new image we need the inverse mapping: to set the pixel value with integer coordinates (x_p, y_p), get the pixel value with coordinates (x_d, y_d) in the distorted image I_d.

It is easy to compute (x_u, y_u) given (x_p, y_p) and the homography M. In homogeneous coordinates, $[x'_u, y'_u, w'_u]^t = M^{-1}[x'_p, y'_p, w'_p]^t$.

However, it is harder to compute (x_d, y_d) given (x_u, y_u). There is no a direct way to solve this problem considering k_1, k_2 and k_3. To solve it, we use the binary search algorithm.

Our goal is to find $f_2(r_d^2)$ given (x_u, y_u) and Θ^d. Once $f_2(r_d^2)$ has been found, x_d and y_d are easily calculated from eq. 5. From eq. 4, we formulate a new function f:

$$f(r_d) = r_u - r_d f_2(r_d^2) = r_u - r_d(1 + k_1 r_d^2 + k_2 r_d^4 + k_3 r_d^6) \qquad (14)$$

If $x_u = c_x$ and $y_u = c_y$, from eq. 5 we have $x_d = x_u$ and $y_d = y_u$. For other cases, $f(r_d = 0) > 0$ and we need to find a value for r_d such that $f(r_d) < 0$ and then apply the binary search algorithm to find the value of r_d such $f(r_d)$ is close to zero (considering some threshold). In the implementation we iteratively increment r_d until $f(r_d) < 0$.

5.4 The Calibration Process

The calibration process starts with one image from the camera, I_d, another image from the calibration pattern, I_r, and initial values for parameters Θ. In the following algorithm, Θ and $\delta\Theta$ are considered as vectors. We start with (c_x, c_y) at the center of the image, $k_1 = k_2 = k_3 = 0$ and the identity matrix for M. The calibration algorithm is as follows:

1. From the reference image, compute the reference feature points (x_{rk}, y_{rk}), $(k = 1, \cdots n)$.
2. From Θ and the distorted image, compute a corrected image.
3. From the corrected image compute the set of feature points (x_{pk}, y_{pk}), $(k = 1, \cdots n)$.
4. From $(x_{pk}, y_{pk})(k = 1, \cdots n)$ and Θ compute $(x_{dk}, y_{dk})(k = 1, \cdots n)$.
5. Find the best Θ that minimize E using the GNLM algorithm:
   (a) Compute the total error, $E(\Theta)$ (eq. 7).
   (b) Pick a modest value for λ, say $\lambda = 0.001$.
   (c) Solve the linear system of equations (8), and calculate $E(\Theta + \delta\Theta))$.
   (d) if $E(\Theta+\delta\Theta) > E(\Theta)$, increase λ by a factor of 10, and go to the previous step. If λ grows very large, it means that there is no way to improve the solution Θ.
   (e) if $E(\Theta + \delta\Theta), I_r) < E(\Theta)$, decrease λ by a factor of 10, replace Θ by $\Theta + \delta\Theta$, and go to step 5a.
6. Repeat steps 2–5 until $E(\Theta)$ does not decrease.

When $\lambda = 0$, the GNLM method is a Gauss–Newton method, and when λ tends to infinity, $\delta\Theta$ turns to so called steepest descent direction and the size $\delta\theta$ tends to zero.

The calibration algorithm apply several times the GNLM algorithm to get better solutions. At the beginning, the clusters of the distorted image are not perfect squares and so point features can not match exactly the feature points computed using the reference image. Once a corrected image is ready, point features can be better estimated.

6 Related Approaches

There are two kinds of calibration methods. The first kind is the one that uses a calibration pattern or grid with features whose world coordinates are known. The second family of methods is those that use geometric invariants of the image features like parallel lines, spheres, circles, etc. [2].

The Method described in this paper is in the first family of methods. Feature point correspondences with reference points are computed automatically. Some other methods require a human operator (with a lot of patience) to find such correspondences [9]. Some other registration methods uses all points or pixels of images as features, instead of a small set of point correspondences. However these kind of methods need an initial set of parameters close to the right one and also have problems due to non uniform illumination [9].

The main problem when we have a high radial distortion is the accurate detection of features. Detect white clusters of pixels is easier than detect lines or corners. Some other methods apply the function f_1 of eq. (3), computing r_d directly from r_u. But they tend to fail when there is a high radial distortion, as shown in Figure 4. Also, in order to correct images, we have to introduce more terms in the distortion model (k_1, k_2, k_3). Other methods use only k_1 and find a direct solution for r_d. However they also fail to model higher radial distortions. Other methods [4][5] use a Taylor expansion of r_d instead of r_d^2. They report slightly better results than the classical approach using an expansion of r_d^2 with lens of low distortion [5]. In our case, using wide–angle lens with very high distortion, we found slightly better results using the expansion of r_d^2 instead of r_d, considering k_1, k_2 and k_3 in both cases.

Once a set of parameters was found using our method, computing each pixel of the new image is slow (due to the binary search method). However in order to process many images from the same camera, that process of finding correspondences between I_p (the new image) and I_d (the distorted image) should be done only once. Given such correspondences, the bilinear interpolation process is very fast and a new corrected image is computed quickly.

We have described a calibration method based on the Gauss–Newton–Levenberg–Marquardt non–linear optimization method using analytical derivatives. Other approaches compute numerical derivatives [1, 2, 7], so we have faster calculations and better convergence properties.

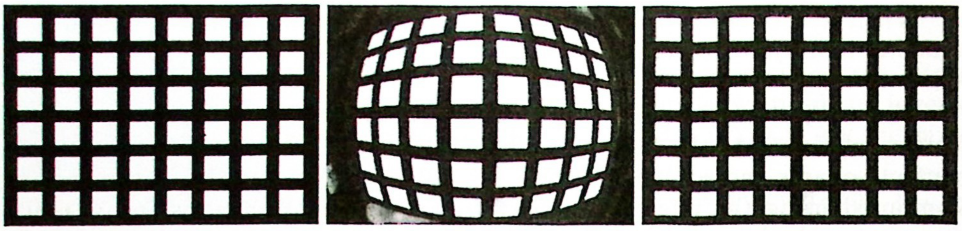

(a) Calibration Pattern (b) Image from camera (c) New image

Fig. 5. The calibration process

7 Experimental Results

Table 1. Final set of parameters

m_0	m_1	m_2	m_3	m_4	m_5	m_6	m_7
.0752	.0146	131.0073	-.0132	.0788	115.4594	-.00002	.000036
m_8	k_1	k_2	k_3	c_x	c_y	s_x	
-.000048	1.2026-06	-4.2812E-13	6.6317E-18	508.936	625.977	1	

We test a MDCS2, $\frac{1}{2}$" format CMOS, firewire color camera from Videre Design with a 3.5mm C-mount lens. This camera acquire 15fps with resolution of 1280×960 pixels.

The pattern calibration (image I_r), showed in Figure 5(a), was made using the program xfig under Linux. The image taken by the camera is shown in Figure 5(b). The corrected and projected image, using our point correspondences method, is shown in Figure 5(c), a very good result. The GNLM process was applied twice, requiring 6 iterations in the first case and 108 iterations in the second case. The error E after the first GNLM search was 1.706×10^5 and at the end the second search it was 1.572×10^5. It is interesting to compute the maximum individual distance between points ($d_i = \sqrt{e_{xi}^2 + e_{yi}^2}$) to see how good was the match. Using this criteria, at the end of the process we got $d_i^{max} = 1.86$ pixels. The final parameters found are listed in Table 1.

Finally, Figure 6 shows an example of removing distortion using an image of our Laboratory.

8 Conclusions

We propose a robust method to remove radial distortion from images using a reference image as a guide. It is based on point correspondences between the

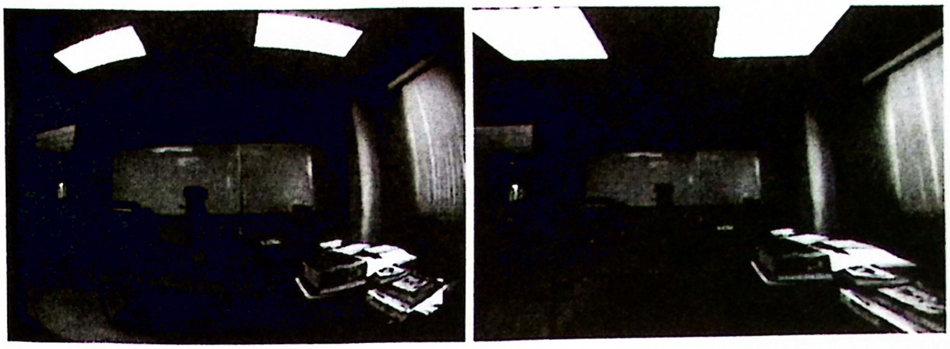


Fig. 6. Original and corrected images

acquired image from the camera (with wide–angle lens) and the reference image. This method is faster than image registration methods and it is able to model high radial distortions. Also the selection of the *center of mass* of clusters of white pixels within images, as point features, are easier to detect than lines or corners. Another advantage of this method is its good convergence properties even starting with a set of parameters that no introduces any distortion.

This method was implemented in Linux and it is available online[1] , using the C language and standard routines from the Free Gnu Scientific library (GSL) to solve the linear system of equations and to find the inverse of matrix M.

References

1. F. Devernay and O.D. Faugeras. Automatic calibration and removal of distortion from scenes of structured environments. *SPIE*, 2567:62–72, July 1995.
2. F. Devernay and O.D. Faugeras. Straight lines have to be straight. *MVA*, 13(1):14–24, 2001.
3. O. Faugeras. *Three-Dimensional Computer Vision.* The MIT Press, 1993.
4. R. I. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision.* Cambridge University Press, ISBN: 0521540518, second edition, 2004.
5. Lili Ma, YangQuan Chen, and Kevin L Moore. A new analytical radial distortion model for camera calibration, 2003.
6. W. Press, B. Flannery, S. Teukolsky, and Vetterling W. *Numerical recipes, the art of scientific computing.* Cambridge University Press, 1986.
7. G. P. Stein. Lens distortion calibration using point correspondences. In *Proc. Conference on Computer Vision and Pattern Recognition (CVPR '97)*, June 1997.
8. R. Szeliski. Video mosaic for virtual environments. *IEICE Computer Graphics and Applications*, 16(2):22–30, March 1996.
9. T. Tamaki, T. Yamamura, and N. Ohnish. A method for compensation of image distortion with image registration technique. *IEICE Trans. Inf. and Sys.*, E84-D(8):990–998, August 2001.

[1] http://faraday.fie.umich.mx/~m_cgomez/calibrate.html