4

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

Leonardo Romero and Cuauhtemoc Gomez
División de Estudios de Postgrado, Facultad de Ingeniería Eléctrica
Universidad Michoacana de San Nicolás de Hidalgo
Morelia, Michoacán, México

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, generates a lot of non-linear distortion. In some applications, for instance in stereo vision systems and robotic 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 forms the internal calibration parameters (Devernay & Faugeras, 1995).

Fig. 1 shows two types of distortion due to lens: barrel and pincushion distortions and a rectangle without any distortion like reference (e.g. the image taken by an ideal pinhole camera) (Weng et al. 1992). The pincushion distortion is due to zoom lens and the barrel distortion is due to wide angle lens. In commercial cameras with wide angle lens the most important component of the barrel distortion is the radial distortion and this chapter introduces a method to find the internal calibration parameters of a camera, specifically those parameters required to correct the radial distortion due to wide-angle lens.

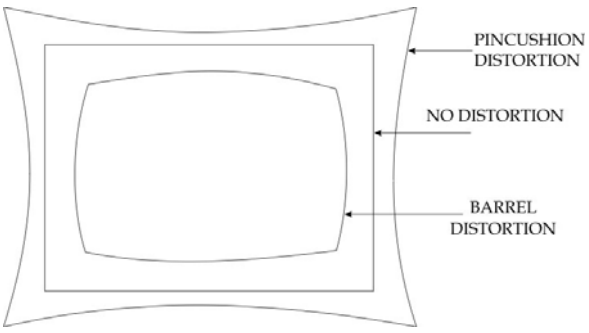


Figure 1. Types of distortion

Source: Scene Reconstruction, Pose Estimation and Tracking, Book edited by: Rustam Stolkin,
ISBN 978-3-902613-06-6, pp.530, I-Tech, Vienna, Austria, June 2007

The 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 chapter is organized as follows. Section 2 describes 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 fail or require a human operator. Experimental results are shown in Section 7. Finally, some conclusions are given in Section 8.

2. Computing Transformed Images

For integer coordinates (i,j), let $I(i,j)$ gives the intensity value of the pixel associated to position (i,j) in image I. Let I_d and I_t be the original (distorted image taken from the camera) and the transformed image, respectively. A geometric transformation, considering a set Θ of parameters, computes pixels of the new image, $I_t(i,j)$ in the following way:

$$I_t(i,j) = I_d(x(\Theta,i,j),y(\Theta,i,j)) \quad (1)$$

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 $I_d(x,y)$ 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 the bilinear interpolation, a much better method to interpolate a pixel with real coordinates (x,y) in an image.
From other point of view, pixel $I_d(x,y)$ moves to the position $I_t(i,j)$. However, most transformations define points in the new image given points in 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) in 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 2 illustrates the bilinear interpolation method (Faugeras, 1993) 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 2). First 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:

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

Using the bilinear interpolation, a smooth transformed image is computed. Now we are able to deal with the transformation 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.

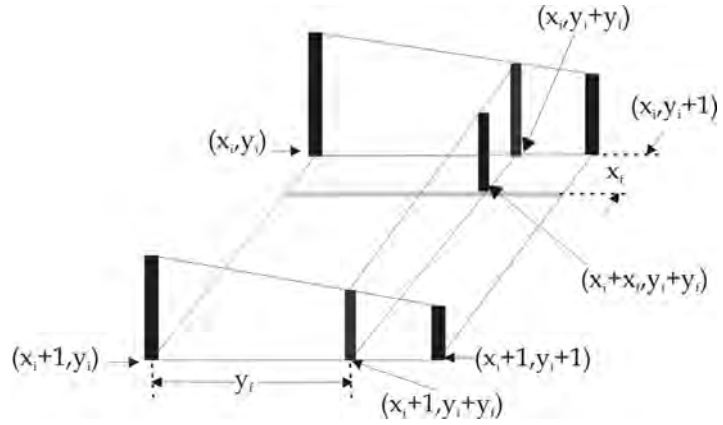


Figure 2. Using the bilinear interpolation

3. The Distortion Model

The radial distortion process due to wide-angle lens is illustrated in Figure 3. Figure 3 (b) shows an image taken from the camera when the pattern shown in Figure 3 (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.

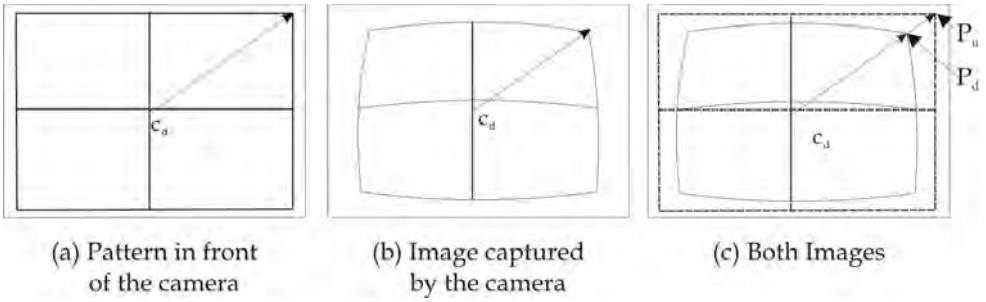


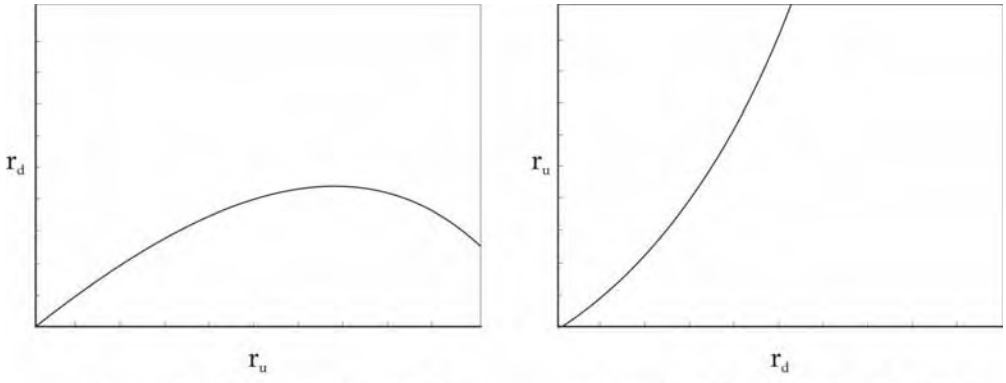
Figure 3. The distortion process due to wide angle lens

Figure 3 (c) shows the radial distortion in detail, supposing that the center of distortion is the point C_d with coordinates (c_x,c_y) (no 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 function f_1 and f_2 may be given by a Taylor expansion: $(f_1(r_u^2) = 1+k_1r_u^2+k_2r_u^4+...)$ and $(f_2(r_d^2) = 1+k_1r_d^2+k_2r_d^4+...)$. Figure 4 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.



(a) $r_d = r_u f_1(r_u^2)$, $f_1 = 1 - 3 \times 10^{-6} r_u^2$ (b) $r_u = r_d f_2(r_d^2)$, $f_2 = 1 + 3 \times 1.3x10^{-5} r_d^2$

Figure 4. Two different functions to model the distortion of images

The problem with f_1 is that there is the possibility to get the same r_d for two different values of r_u (see Fig. 5).

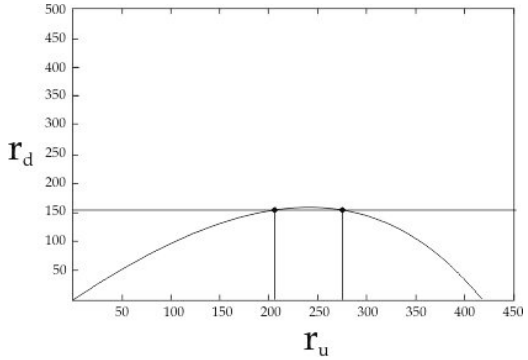
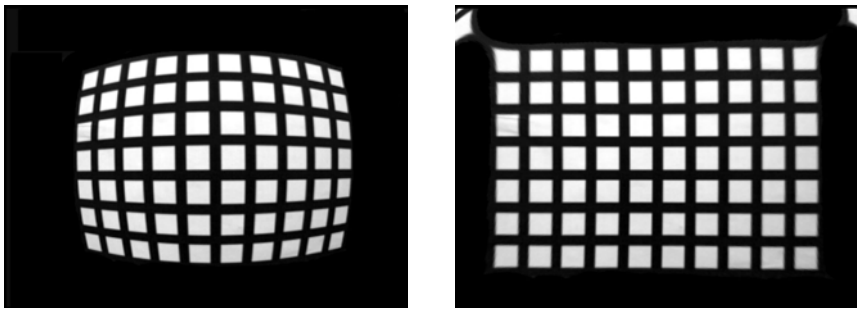


Figure 5. Problem using f_1. A single r_d is related with two different r_u

In fact, this behavior was found experimentally when we use f_1. Figure 6 shows an example. Borders of the corrected image duplicate parts of the image (see the top corners in Figure 6(b)). However f_2 does not have this problem.



(a) Original image (b) Corrected image

Figure 6. An example of a wrong correction using f_1

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:

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

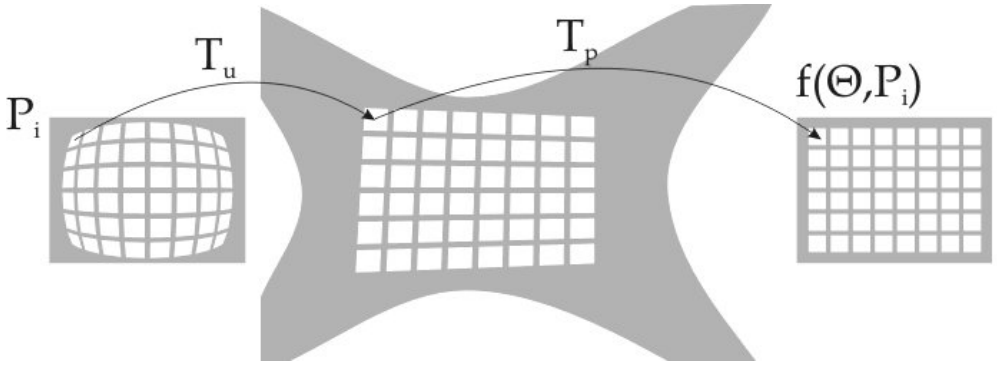
where (c_x,c_y) are the coordinates of the center of radial distortion. So, this distortion model have a set of five parameters $\Theta^d = \{c_x,c_y,k_1,k_2,k_3\}$. This model works fine if the camera have square pixel, but if not, we need another parameter, s_x, called aspect ratio that divide the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

Figure 7 helps to understand the process to transform the image taken from the camera, I_d, to a new image, I_d, similar to the reference image. A point P_i in image I_d with coordinates (x_d, y_d) maps to an undistorted point (x_u, y_u). Figure 7(b) illustrates the image without radial distortion and the transformation T_u that maps the point with coordinates (x_d, y_d) to new coordinates (x_u, y_u). This new image is bigger than I_d because the compression due to the wide angle lens has been removed, lines in the environment maps to lines in this image.

In a second step, the point with coordinates (x_u, y_u) is projected to a new point (x_p, y_p) in image I_t. Next section focuses in this projection step.

4. The Projection Model

Figure 3 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. A more realistic case is illustrated in Figure 7. Conceptually we can assume that a pinhole camera captures the undistorted image (a planar image) into the new image I_t. Since cameras are projective devices, a projective transformation is involved. A projective transformation is the most general transformation that maps lines into lines (Hartley & Zisserman, 2004).



(a) Input image I_d b) Image I_u without radial distortion (c) New image I_t

Figure 7. Transforming the input image

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 (Szeliski, 1996) (Hartley & Zisserman, 2004) $[x_p', y_p', w_p']^t = M[x_u', y_u', w_u']^t$, where matrix M is called an homography and it has eight degrees of freedom. For our calibration application, M has the form:

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

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_t. Assigning $w_u'=1$, the new coordinates of P_p are given by:

$$x_p = \frac{m_0 x_u + m_1 y_u + m_2}{m_6 x_u + m_7 y_u + 1}$$

(6)

$$y_p = \frac{m_3 x_u + m_4 y_u + m_5}{m_6 x_u + m_7 y_u + 1}$$

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 captured by the camera, I_d, into a new projected image, I_t, 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 $(k=1,...,n)$ in I_r and (x_{dk},y_{dk}) be its correspondence point 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 $C = \{<(x_{r1},y_{r1}),(x_{p1},y_{p1})>,...,<(x_{rn},y_{rn}),(x_{pn},y_{pn})>\}$.

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

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

5.1 Non-Linear Optimization

The Gauss-Newton-Levenberg-Marquardt method (GNLM) (Press et al., 1986) 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 qxq and matrix B has dimension $qx1$ and $\delta\Theta = [\,\delta\theta_1, \delta\theta_1, ..., \delta\theta_q\,]^t$. 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,

$$a_{i,j} = \sum_{k=1}^{n}\left(\frac{\partial x_{pk}}{\partial\theta_i}\frac{\partial x_{pk}}{\partial\theta_j} + \frac{\partial y_{pk}}{\partial\theta_i}\frac{\partial y_{pk}}{\partial\theta_j}\right)$$

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

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

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

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,

$$\frac{\partial x_p}{\partial \theta_i} = \frac{\partial x_p}{\partial x_u}\frac{\partial x_u}{\partial \theta_i} + \frac{\partial x_p}{\partial y_u}\frac{\partial y_u}{\partial \theta_i} \qquad \theta_i \in \{c_x, c_y, k_1, k_2, k_3\} \tag{11}$$

$$\frac{\partial y_p}{\partial \theta_i} = \frac{\partial y_p}{\partial x_u}\frac{\partial x_u}{\partial \theta_i} + \frac{\partial y_p}{\partial y_u}\frac{\partial y_u}{\partial \theta_i} \qquad \theta_i \in \{c_x, c_y, k_1, k_2, k_3\}$$

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

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

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

Where r_d 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 6(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 neighbours 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. 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. This is not a problem with the reference pattern, because we use the perfect graphic file of the image and there are no bad clusters of white pixels.

We also assume that the image from the camera does not have a significant rotation, relative to the reference image, so relative positions of white squares hold in both images. For instance, the top left-most white square is the closest square to the top-left 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 I_t we need the inverse mapping: to set the pixel value with integer coordinates (x_p,y_p) in I_t, we need to compute 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. To solve it, we use the binary search algorithm. Our goal is to find r_d given $r_u^2 = (x_u-c_x)^2+(y_u-c_y)^2$, k_1, k_2 and k_3. Once r_d has been found, x_d and y_d are easily computed using eq. 5. ($x_d = (x_u-c_x) / f_2(r_d^{\,2}) + c_x$ and $y_d = (y_u-c_y) / f_2(r_d^{\,2}) + c_y$). From eq. 4, we formulate a new function f:

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

If $x_u=c_x$ and $y_u=c_y$, from eq. 5 we have $x_d = x_u$ and $y_d=y_u$. If $x_u\neq c_x$ and $y_u\neq c_y$, we need to find a low limit, r_{d0}, and high limit, r_{d1}, such that $f(r_{d0}) > 0$ and $f(r_{d1}) < 0$. With these limits, the binary search algorithm is able to find the right r_d such that $f(r_d)=0$ (or very close to zero) and then $r_u = r_d\, f_2(r_d^{\,2})$.

If $x_u\neq c_x$ and $y_u\neq c_y$ then $r_u>0$ and $f(0) > 0$, so we have the low limit $r_{d1}= 0$. To find the high limit r_{d1} 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}), $(k=1,...n)$.
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}), $(k=1,...n)$.
4. From $(x_{pk},y_{pk})(k=1,...n)$ and Θ compute $(x_{dk},y_{dk})(k=1,...n)$.
5. Find the best Θ that minimize E using the GNLM algorithm:
   (a) Compute the total error, E(Θ) (eq. 7).
   (b) Pick a modest value for λ, say λ=0.001.
   (c) Solve the linear system of equations (8), and calculate E($\Theta+\delta\Theta$).
   (d) If E($\Theta+\delta\Theta$) >= E(Θ), 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($\Theta+\delta\Theta$) < E(Θ), 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(Θ) does not decrease.

When $\lambda=0$, the GNLM method is a Gauss-Newton method, and when λ tends to infinity, δΘ turns to so called steepest descent direction and the size of $\delta\theta_i$ 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 use geometric invariants of the image features like parallel lines, spheres, circles, etc. (Devernay & Faugeras, 2001).

The method described in this paper is in the first family of methods. Feature point correspondences are computed automatically. Some other methods require a human operator (with a lot of patience) to find such correspondences (Tamaki et al., 2001). Some other registration methods use all pixels of images as features, instead of a small set of point correspondences. However these methods need an initial set of parameters close enough to the right one and also have problems due to non uniform illumination (Tamaki et al., 2001).

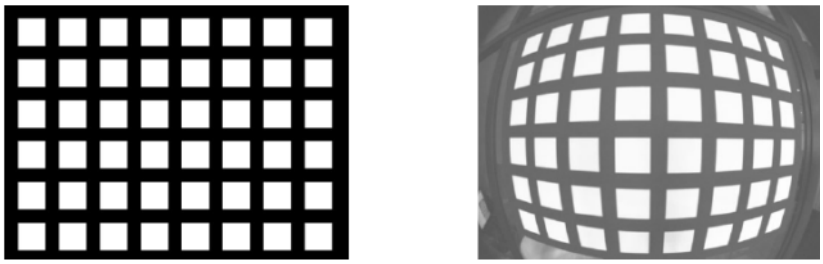
The main problem when we have a high radial distortion is the accurate detection of features. Detect white clusters of pixels is easier than detect lines or corners. Some other methods apply the function f_1 of eq. (3), computing r_d directly from r_u. But they tend to fail when there is a high radial distortion, as shown in Figure 6. 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 (Ma et al. 2003) use a Taylor expansion of r_d instead of r_d^2. Experimentally we found better results using r_d^2 instead of r_d for wide angle lens.

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

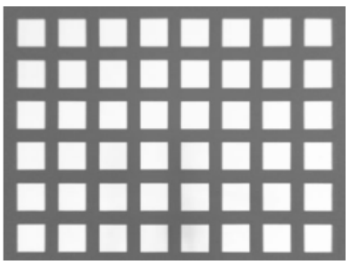
from the same camera, that process of finding correspondences between I_t (the new image) and I_d (the distorted image) should be done only once. Given such correspondences, the bilinear interpolation process is very fast and a new corrected image is computed quickly.
We have described a calibration method based on the Gauss-Newton-Levenberg-Marquardt non-linear optimization method using analytical derivatives. Other approaches compute numerical derivatives (Devernay, 1995; Sten, 1997; Devernay 2001), so we have faster calculations and better convergence properties.

7. Experimental results

We test a MDCS2, ½" format CMOS, Firewire color camera from Videre Design with a 3.5mm C-mount lens. This camera acquire 15fps with resolution of 1280 x 960 pixels.
The pattern calibration (image I_r), showed in Figure 8(a), was made using the program xfig under Linux. The image taken by the camera is shown in Figure 8(b). The corrected and projected image, using our point correspondences method, is shown in Figure 8(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 $1.706x10^5$ and at the end of the second search it was $1.572x10^5$. It is interesting to compute the maximum individual distance between points $\left(d_i = \sqrt{e_{x_i}^2 + e_{y_i}^2}\right)$ to see the maximum individual error. 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.



(a) Calibration Pattern (b) Image from camera



(c) New image

Figure 8. The calibration process

Finally, Figure 9 shows an example of removing distortion using an image of our Laboratory.

Figure 9. Original and corrected images

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	1.2026E-6	-4.2812E-13	6.6317E-18	508.936	625.977	1	

Table 1. Final set of parameters

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 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*.

9. References

Devernay, F & Faugeras, O. (1995). Automatic calibration and removal of distortion from scenes of structured environments, *SPIE*, 2567;62-72

Devernay, F & Faugeras, O.D. (2001), *Straight lines have to be straight*. *MVA*, 13(1);14-24

Faugeras, O. (1993). *Three-Dimensional Computer Vision*, The MIT Press

Hartley, R. I. & Zisserman A. (2004) *Multiple View Geometry in Computer Vision*, .Camdbrige University Press, ISBN: 0521540518, second edition

Ma, Lili; Chen, YangQuan & Moore, Kevin L. (2003), A new analytical radial distortion model for camera calibration. http://www.citebase.org/abstract?id=oai:arXiv.org:cs/0307046.

Press, W.; Flannery, S.; Teukolsky & Vetterling, W. (1986), *Numerical Recipes, the art of scientific computing,* Cambridge University Press

Stein, G. P., (1996) Lens distortion calibration using point correspondences. In Proc. *Conference on Computer Vision and Pattern Recognition (CVPR '97)*

Szeliski, R. (1996), Video mosaic for virtual environments. *IEICE Computer Graphics and Applications*, 16(2):22-30

Tamaki, T.; Yamamura, T. & Ohnish, N. (2001), A method for compensation of image distortion with image registration technique. *IEICE Trans. Inf. and Sys*, E84-D(8):990-998

Weng, J.; Cohen, P. & Herniou, M. (1992), Camera calibration with distortion models and accuracy evaluation. *IEEE Trans. Pattern Analysis and Machine Intelligence.*, 14(10):965-980. ISSN 0162-8828.

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