sensors

MDPI

Article

An Exact Formula for Calculating Inverse Radial Lens Distortions

Pierre Drap *,† **and Julien Lefèvre** †

Aix-Marseille Université, CNRS, ENSAM, Université De Toulon, LSIS UMR 7296, Domaine Universitaire de Saint-Jérôme, Bâtiment Polytech, Avenue Escadrille Normandie-Niemen, Marseille 13397, France; Julien.Lefevre@univ-amu.fr
* Correspondence: Pierre.Drap@univ-amu.fr; Tel.: +33-4-91-82-85-20
† These authors contributed equally to this work.

Academic Editor: Manuela Vieira
Received: 4 March 2016; Accepted: 24 May 2016; Published: 1 June 2016

Abstract: This article presents a new approach to calculating the inverse of radial distortions. The method presented here provides a model of reverse radial distortion, currently modeled by a polynomial expression, that proposes another polynomial expression where the new coefficients are a function of the original ones. After describing the state of the art, the proposed method is developed. It is based on a formal calculus involving a power series used to deduce a recursive formula for the new coefficients. We present several implementations of this method and describe the experiments conducted to assess the validity of the new approach. Such an approach, non-iterative, using another polynomial expression, able to be deduced from the first one, can actually be interesting in terms of performance, reuse of existing software, or bridging between different existing software tools that do not consider distortion from the same point of view.

Keywords: radial distortion; distortion correction; power series

1. Introduction

Distortion is a physical phenomenon that in certain situations may greatly impact an image's geometry without impairing quality nor reducing the information present in the image. Applying the projective pinhole camera model is often not possible without taking into account the distortion caused by the camera lens. This phenomenon can be modelled by a radial distortion, the most prominent component, and a second, with a lesser effect, a decentering distortion which has both a radial and a tangential components. Radial distortion is caused by the spherical shape of the lens, whereas tangential distortion is caused by the decentering and non-orthogonality of the lens components with respect to the optical axis ([1,2]). It is important to note that radial distortion is highly correlated with focal length [3] even if in literature it is not modelled within the intrinsic parameters of the camera [4]. This is due to the fact that the radial distortion model is not linear, in contrary to other intrinsic parameters. We can see in Figure 1 the displacement applied to a point caused by both radial and tangential distortion.

Decentering distortion was modelled by Conrady in 1919 [5] then remodelled by Brown in 1971 [6] and a radial distortion model was proposed by Brown in 1966 [7]. These distortion models have been adopted by the Photogrammetry as well as the Computer Vision communities for several decades. Most photogrammetric software such as PhotoModeler (EOS) uses these models (see Equations (1) and (2)) to correct observations visible on the images and provide ideal observations.

Roughly, radial distortion can be classified in two families, barrel distortion and pincushion radial distortion. Regarding the k_1 coefficient in Formula (1), barrel distortion corresponds to a negative value of k_1 and pincushion distortion to a positive value of k_1, for an application of the distortion and not a compensation. As shown in Figure 2, barrel and pincushion distortions have an inverse effect and an image affected by a pincushion distortion can be corrected by a barrel distortion (and vice-versa) [8–11].

Barrel distortion can be physically present in small focal length systems, while larger focal lengths can result in pincushion distortion [8,10]. These radial distortion effects can be very important, especially in inexpensive wide-angle lenses which are often used today.

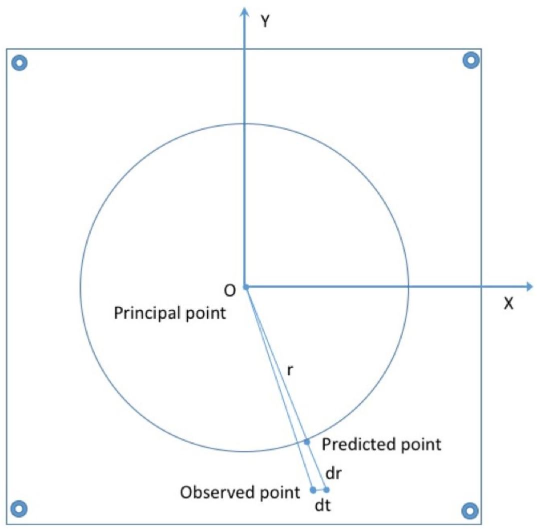


Figure 1. Point shifted by distortion.

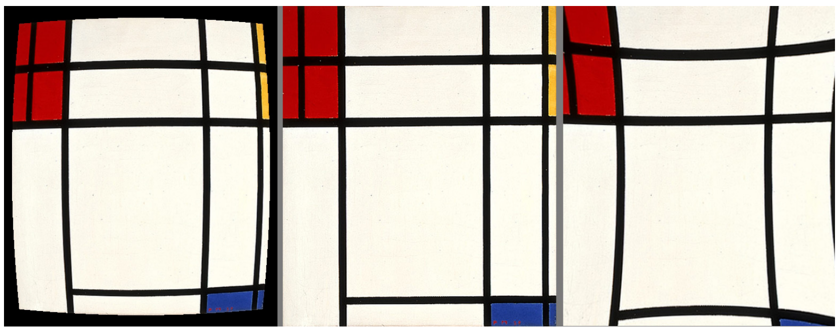


Figure 2. In the center, a painting from Piet Mondrian [12] (which is now in the public domain since 1 January 2016); on the left, the painting with a barrel effect; and on the right, the same image with pincushion distortion.

Using these models to compensate the observations is now well known and many software dealing with images or panoramas propose plugins dedicated to distortion correction (mainly only radial distortion) [13] . However, although we have the equations to compensate the distortion, how to compute the inverse function in order to apply such a distortion is not obvious. For example, when an image of a known 3D point is computed using a calibrated camera, the 2D projected point can be easily computed, but we need then to apply the distortion to the image point in order to obtain an accurate projection of the original 3D point. This first application justifies the present work. How to determine the inverse of a closed form solution for distortion model equations? The second reason is the merging the work of the two communities involved, photogrammetry and computer vision. While having worked separately for years, the situation between the two communities has drastically changed for almost a decade [14]. This is also visible in the form of new commercial or open-source software dealing with photogrammetry or computer vision. For example, PhotoScan (from Agisoft) or MATLAB that comes with the camera calibration toolbox or, for the open-source side, OpenCV toolbox which provides a solution for multiview adjustment. These three software use the same Equations (1) and (2) to manage distortion, but the mathematical model here is used to apply distortion and not to compensate it. Determining an exact formula to calculate inverse lens distortion, which allows using the same software to apply and compensate distortion with two set of k_n parameters can be very useful and, in fact, is the purpose of this work.

This paper is organized as follows: in the next section, a demonstration on computing an exact formula for inverse radial distortion is presented. This approach gives a set of $k_1 ... k_n$ coefficients computed from the original polynomial distortion model. In Section 3, several applications of this formula are presented. First of all an experiment on this inversion formula is done using only the $k_1 ... k_n$ coefficient. Then, an application used to compute a formula for an inverse radial lens distortion is applied to an image coming from a metric camera (Wild P32) with a 3-micron distortion in the edge of the frame. The next experiment is done on a calibration grid built with black disk. Finally, a discussion on converting the distortion model between several photogrammetric software, PhotoModeler (EOS [15]), PhotoScan (Agisoft [16]) and OpenCV [17] is proposed.

2. Previous Work

2.1. Calibration Approach

Radial distortion is mainly considered in the camera calibration process. Since Duane Brown's first publication, a large quantity of work was performed in the field of camera calibration, opening the way for new methods. Several techniques were proposed using orthogonal planes, 2D objects with plannars patterns up to self-calibration with unknown 3D points. Interesting reviews were published on both the photogrammetry and computer vision sides by Fraser [18], Zhang [19] and more recently by Shortis [20].

When Brown [6] proposed a radial distortion model in 1971, he also proposed a way to calibrate cameras using a set of plumb lines. The idea of using a set of straight wires to compute a distortion model in a camera calibration process remains in use 45 years later in the fields of photogrammetry and computer vision; Hartley in 1993 [21,22] then Faugeras and Devernay [23] and recently Nomura [24], Clauss [25], Tardif [26], and Rosten [27].

2.2. Inverse Radial Distortion

After Conrady and Brown a lot of work was done to deal with removing distortion from images. As the problem is shared by photogrammetry community as well as computer vision we can refer to many books and papers on this topic. Including the famous Manual of photogrammetry [28] and Atkinson gives an overview of these problems for the two communities [29].

Nevertheless, the problem of reverse distortion is somewhat the poor relation of the problems of distortion. As mentioned by Heikkilä and Silvén [30], "only a few solutions to the back-projection problem can be found in literature, although the problem is evident in many applications." And in the same paper, "we can notice that there is no analytic solution to the inverse mapping".

In the particular case of high distortion as in wide-angle and fish-eye lenses, some non polynomial (and invertible) models have been proposed; for example Basu and Licardie introduced the Fish-Eye Transform (FET) in [31]. Also Faugeras and Devernay [23] propose another invertible model based on the Field-of-View. A complete description of these models can be found in the review written by Hugues [32] and also in [33].

Regarding the polynomial model, several solutions have been tested to perform inverse radial distortion and the solution can be classified in three main classes (even if other approaches can be found such as the use of a neural network [34]):

- Approximation. Mallon [35], Heikkilä [30,36] and then Wei and Ma [37] proposed inverse approximations of a Taylor expansion including first order derivatives. According to Mallon and Welhan, "This is sometimes assumed to be the actual model and indeed suffices for small distortion levels." [35]. A global approach, inverse distortion plus image interpolation, is presented in a patent held by Adobe Systems Incorporated [38].

- Iterative. Starting from an initial guess, the distorted position is iteratively refined until a convenient convergence is determined [39–41];
- Look-up table. All the pixels are previously computed and a look-up table is generated to store the mapping (as for example in OpenCV).

All these methods involve restrictions and constraints on accuracy, time processing or memory usage.

Nevertheless, some very good results can be obtained. For example, implementing the iterative approach gives excellent results, however the processing time is drastically increased. Given in Peter Abeles' blog [40], the method is easy to implement. Results are shown in Figure 3.

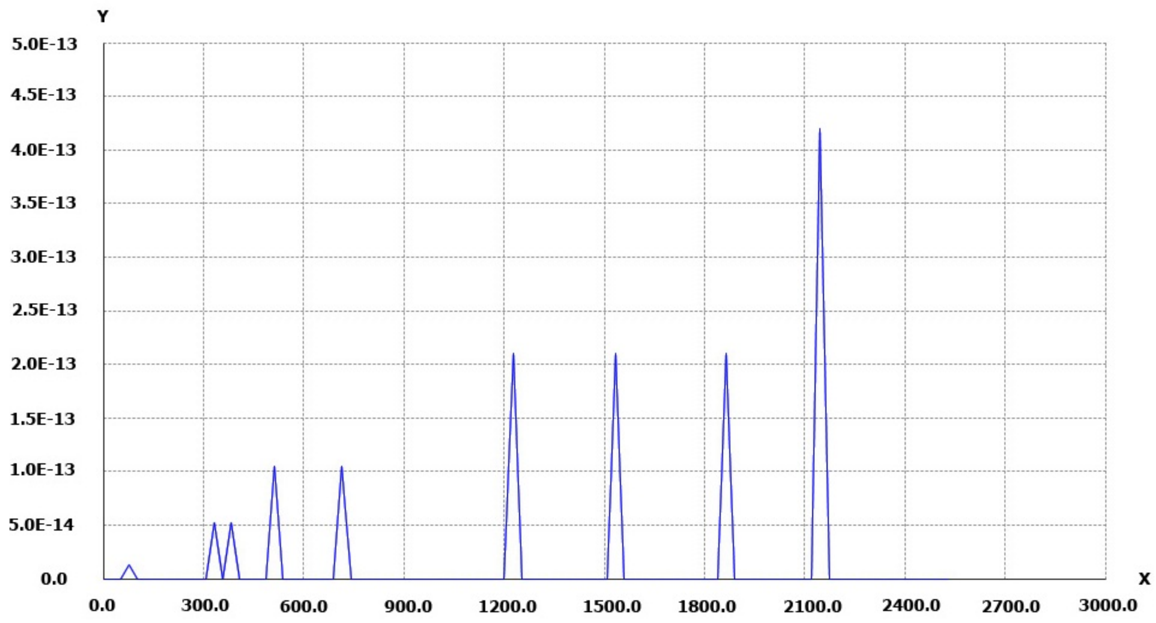


Figure 3. Iterative method applied to a Nikon D700 camera with a 14mm lens. Along the frame diagonal the points are first compensated by what is a normal process of the radial distortion using Equation (1) and then the distortion is applied with the iterative process [40] and the result compared to the original point. On Y-axis, the distance between the original point and the computed reverse point.

The iterative solution works by first estimating the radial distortion magnitude at the distorted point and then refining the estimate until it converges.

Algorithm 1 shows an implementation of this approach.

Algorithm 1 Iterative algorithm to compute the inverse distortion

```
Require: point $P_n$
  $P_c = P_n$
  repeat
      $r = ||P_c||$
      $dr = 1 + k_2 r^2 + k_4 r^4 + ...$
      $P_c = P_n / dr$
  until Convergence of $P_c$
  return $P_c$
```

Only a few iterations are necessary.

The results presented in Figure 3 are in pixels. The used camera here is a Nikon D700 with a 14 mm lens. The calibration was done with PhotoModeler EOS and the results are $k_1 = 1.532 \times 10^{-4}$, $k_2 = -9.656 \times 10^{-8}$, $k_3 = 7.245 \times 10^{-11}$. The coefficients are expressed in millimeters. The center of autocollimation and the center of distortion are close to the image center. Only a few iterations are necessary to compute the inverse distortion. In this case, with a calibration made using PhotoModeler [15], the inverse of distortion represents the application of the distortion to a point projected from the 3D space onto the image.

This iterative approach is very interesting when the processing time is not an issue, as for example in the generation of a look-up table. Note, however, that a good initial value is needed.

According to these existing methods, we now want to obtain a formula for the inverse radial distortion when modelled by a polynomial form as described by Brown. The inverse polynomial form will be an expression of the original $k_1...k_4$ coefficients of the original distortion.

3. Exact Formula for Inverse Radial Distortion: An Original Approach

3.1. Lens Distortion Models

We consider the general model of distortion correction or distortion removal that can be written in the following form by separating radial and tangential/decentering components:

$$x' = x + \bar{x}\left(k_1 r^2 + k_2 r^4 + k_3 r^6 + ...\right) + \left[p_1\left(r^2 + 2\bar{x}^2\right) + 2p_2\bar{x}\bar{y}\right]\left(1 + p_3 r^2 + ...\right) \tag{1}$$

$$y' = y + \underbrace{\bar{y}\left(k_1 r^2 + k_2 r^4 + k_3 r^6 + ...\right)}_{\text{Radial Distorsion}} + \underbrace{\left[p_2\left(r^2 + 2\bar{y}^2\right) + 2p_1\bar{x}\bar{y}\right]\left(1 + p_3 r^2 + ...\right)}_{\text{Tangential Distorsion}} \tag{2}$$

where $\bar{x} = x - x_0, \bar{y} = y - y_0, r = \sqrt{\bar{x}^2 + \bar{y}^2}$.

In the following we will consider only radial distortion.

3.2. General Framework

Given a model of distortion or correction with parameters $(k_1, k_2, k_3, ...)$, our general objective is to find the inverse transformation. A natural assumption is to express the inverse transformation on the same form of the direct transformation, *i.e.*, with parameters $(k'_1, k'_2, k'_3, ...)$. Therefore we want to express each k'_i as a function of all the k_j .

Radial distortion

Let us assume that there exists two transformations T_1 and T_2 :

$$T_1 : \begin{pmatrix} x \\ y \end{pmatrix} \longrightarrow \begin{pmatrix} x' \\ y' \end{pmatrix} = P(r) \begin{pmatrix} x \\ y \end{pmatrix} \tag{3}$$

$$T_2 : \begin{pmatrix} x' \\ y' \end{pmatrix} \longrightarrow \begin{pmatrix} x \\ y \end{pmatrix} = Q(r') \begin{pmatrix} x' \\ y' \end{pmatrix} \tag{4}$$

where $r = \sqrt{x^2 + y^2}$, $r' = \sqrt{x'^2 + y'^2}$, and P and Q are power series:

$$P(r) := \sum_{n=0}^{+\infty} a_n r^{2n} \tag{5}$$

$$Q(r') := \sum_{n=0}^{+\infty} b_n r'^{2n} \tag{6}$$

with $a_0 = 1$, $a_1 = k_1$,..., $a_n = k_n$ (in order to use k as an index in the calculus of the Appendix). In addition to starting at $n = 0$ we facilitate those calculi. We can scale r' as $r' = \alpha r$ in order to have the same domain of definition for P and Q. So Q reads:

$$Q(r') = \sum_{n=0}^{+\infty} b'_n r^{2n} \tag{7}$$

with $b'_n = b_n \alpha^{2n}$. In the following b_n is used instead of b'_n but we keep in mind this change of variable.

Given the definition of r and by using transformation T_2 in Equation (4) we obtain:

$$r = r'|Q(r')|$$

and similarly with Equation (3):

$$r' = r|P(r)|$$

P and Q are positive which allows removing the absolute value. Hence by injecting the last equation in the first we get:

$$r = rP(r)Q\big(rP(r)\big)$$

and at the end:

$$1 = P(r)Q\big(rP(r)\tag{8}$$

It is possible to derive a very general relation between coefficients a_n and b_n but it is not exactly adapted to real situations where P is a polynomial of finite order. Therefore we can derive a slightly simpler relationship in the case where only $a_1, ..., a_4$ are given. It is summarized in the following result:

Proposition 1. *Given the sequence $a_1, ..., a_4$ it is possible to obtain the recursive relation:*

$$b_0 = 1 \textit{ and for } n \geq 0 \;\; b_n = -\sum_{k=1}^{4} a_k q(n-k) - \sum_{\substack{j+k=n \\ 0\leq k \\ 1\leq j\leq 8k}} b_k p(j, 2k) \tag{9}$$

where we use the following intermediate coefficients:

$$p(j,k) = \sum_{\substack{n_1+...+n_k=j \\ 0\leq n_i\leq 4}} a_{n_1}...a_{n_k}$$

$$q(k) = -\sum_{j=1}^{4} a_j q(k-j)$$

We will derive this expression in Appendix A and show how the coefficients $b_1, ..., b_n$ can be computed both with symbolic and numeric algorithms in Appendix B.

Several remarks can be made about this result:

Remark 1

The problem is symmetric in terms of P and Q, so the relations found for a_n can of course be applied in the reverse order.

Remark 2

For any n the coefficient b_n can be computed recursively. In Equation (9), the first summation is obtained thanks to $a_0, ..., a_4$ and $q(n-1), ..., q(n-4)$ that only depends on the sequence a_n. Similarly the second summation involves $b_0, ..., b_{n-1}$ and values $p(j, 2k)$ which both depend only the given sequence a_n.

Therefore the recursive formula for b_n can be implemented at any order n. We provide the 4 first terms:

$$b_1 = -a_1 \tag{10}$$

$$b_2 = 3a_1^2 - a_2 \tag{11}$$

$$b_3 = 8a_1a_2 - 12a_1^3 - a_3 \tag{12}$$

$$b_4 = 55a_1^4 + 10a_1a_3 - 55a_1^2a_2 + 5a_2^2 - a_4 \tag{13}$$

All formula till b_9 are summarized in Appendix C.

4. Results and Experimental Section

In this section we propose three experiments to test this inverse formula for radial distortion in order to evaluate the relevance of such approach.

- First, we begin by testing the accuracy of the inverse formula by applying the forward/inverse formula recursively within a loop. Hence, the inverse of the inverse radial distortion is computed 10,000 times and compared to the original distortion coefficients.
- Then, for a given calibrated camera, we compute the residual after applying and compensating the distortion along the camera frame. A residual curve shows the results of the inverse camera in all the frame.
- The last experiment is the use of inverse distortion model on a image made with a metric camera built with a large eccentricity (a film-based Wild P32 camera), without distortion. We apply a strong distortion and then compensate it and finally compare it to the original image.

4.1. Inverse Distortion Loop

In this experiment, as the formula gives an inverse formula for radial distortion we do it twice and compare the final result with the original one. In a second step, we iterate this process 10,000 times and compare the final result with the original distortion.

The Table 1 shows the original radial distortion and the computed inverse parameters. The original distortion is obtained by using PhotoModeler to calibrate a Nikon D700 camera with a 14 mm lens from Sigma.

Table 1. Radial distortion calibration in Column two. Column three, inverse of radial distortion with coefficient from $k_1 ... k_9$.

Radial Distortion Coefficients	**Original Value**	**Computed Inverse Value**
k_1	1.532×10^{-4}	1.532×10^{-4}
k_2	-9.656×10^{-8}	1.6697072×10^{-7}
k_3	7.245×10^{-11}	$-2.33941625216 \times 10^{-10}$
k_4	0.0	$3.1255518770316804 \times 10^{-13}$
k_5	0.0	$-4.774156462972984 \times 10^{-16}$
k_6	0.0	$7.680785197322419 \times 10^{-19}$
k_7	0.0	$-1.1582853960835112 \times 10^{-21}$
k_8	0.0	$2.1694555835054252 \times 10^{-24}$
k_9	0.0	$-3.779164309884112 \times 10^{-27}$

The results for this step are presented in Table 2. In the columns 'Delta Loop 1' and 'Delta Loop 10000' we can see that k_1 and k_2 did not change and the delta on k_3 and k_4 are small with respect to the corresponding coefficients: the error is close to 1E-10 smaller than the corresponding coefficient. Note that k_4 was not present in the original distortion and as the inverse formula is in function of only $k_1 ... k_4$, the loop is computed without the coefficients $k_5 ... k_9$ which influences the results, visible from k_4.

Table 2. Radial distortion inverse loop and residual between coefficients of the orignal distortion and after n inversions (n = 2 and n = 10,000).

Coefficient	Original	Inverse 1	Delta Loop 1	Delta Loop 10,000
k_1	1.532×10^{-4}	-1.532×10^{-4}	0.0	0.0
k_2	-9.656×10^{-8}	1.6697072×10^{-7}	0.0	0.0
k_3	7.245×10^{-11}	$-2.33941625216 \times 10^{-10}$	$1.292469707114 \times 10^{-26}$	$1.292469707 \times 10^{-26}$
k_4	0.0	$3.1255518770316804 \times 10^{-13}$	$1.009741958682 \times 10^{-28}$	$1.009842932 \times 10^{-24}$

This first experiments shows the inverse property of the formula and of course not the relevance of an inverse distortion model. But this experiment shows also the high stability of the inversion process. However, even if coefficients $k_1 ... k_4$ are sufficient in order to compensate distortion, the use of coefficients $k_1 ... k_9$ are important for the inversion stability.

The next two experiments show the relevance of this formula for the inverse radial distortion model.

4.2. Inverse Distortion Computation onto a Frame

This second experiment uses a Nikon D700 equipped with a 14 mm lens from Sigma. This camera is a full frame format, *i.e.*, a 24 mm × 36 mm frame size. The camera was calibrated using PhotoModeler and the inverse distortion coefficients are presented in Table 1, where Column 1 gives the calibration result on the radial distortion, and Column 2 the computed inverse radial distortion.

Note that the distortion model provided by the calibration using PhotoModeler gives as a result a compensation of the radial distortion, in millimeters, limited to the frame.

The way to use this coefficient is to first express a 2D point on the image in the camera reference system, in millimeters, with the origin on the CoD (Center of Distortion), close to the center of the image. Then the polynomial model is applied from this point.

The inverse of this distortion is the application of such a radial distortion to a point theoretically projected onto the frame.

In all following experiments, the residuals are computed as follows:

A 2D point p, is chosen inside the frame, its coordinate are previously computed in millimeters in the camera reference system with the origin on the CoD. Then p_1 is p compensated by the inverse of distortion. Finally p_2 is p_1 compensated by the original distortion.

The residual is the value $dist(p, p_2)$.

The following results show the 2D distortion residual curve. For a set of points on the segment $[O, \max X/2]$ the residuals are computed and presented in Figure 4 as Y-axis. The X-axis represents the distance from the CoD. These data comes from the calibration process and are presented in Table 1.

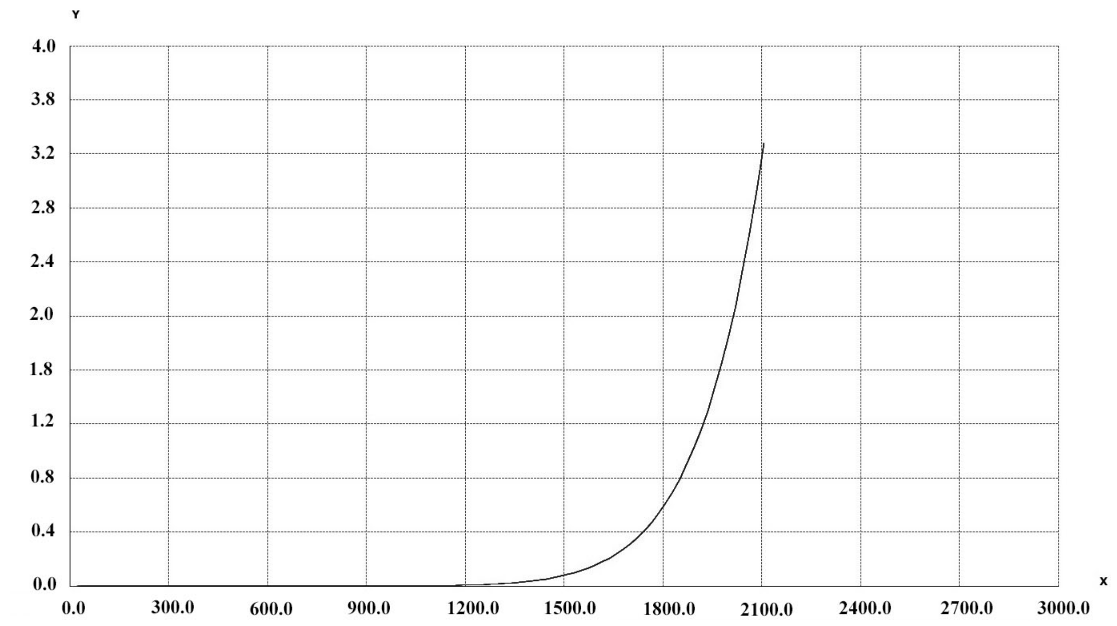
The results shown in Figures 4 and 5 are given in pixels.

In Figure 4a, below, we present the residuals using only coefficients $k_1 .. k_4$ of the inverse distortion. The maximum residual is close to 4 pixels, but residuals are less than one pixel until close to the frame border. This can be used when using non configurable software where it is not possible to use more than 4 coefficients for radial distortion modeling.

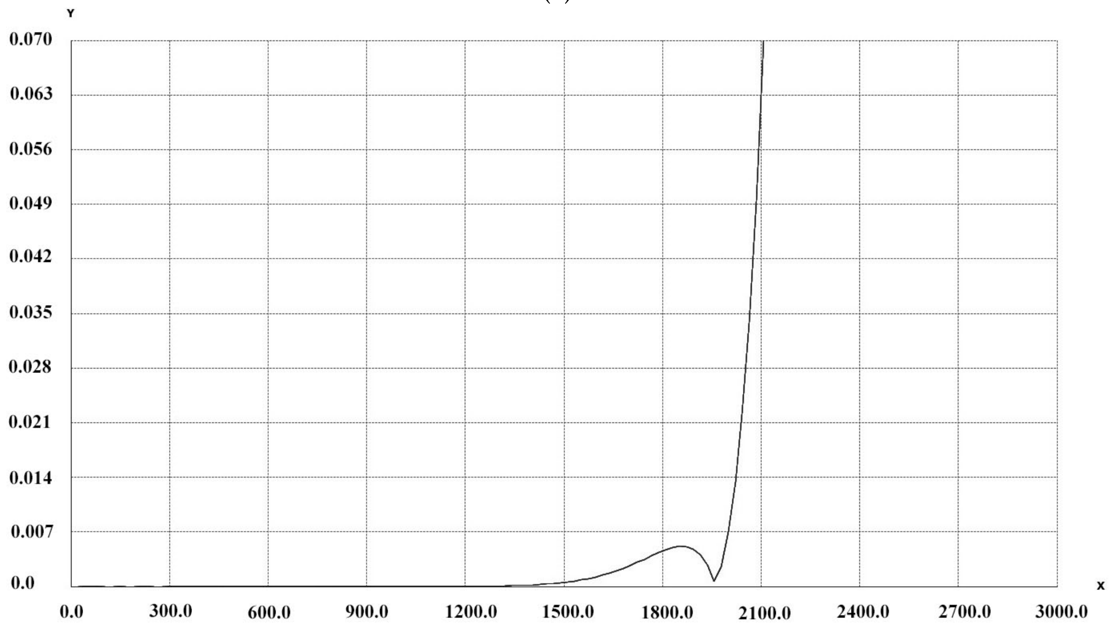
In Figure 4, below, we present the residual computed from 0 to max X frame using coeficients $k_1 ... k_9$ for inverse distortion.

The results are very good, less than 0.07 pixel on the frame border along 0X axis and the performance is quite the same as for compensating the original distortion.

We can see that in almost all the images the residuals are close to the ones presented in Figure 5. Nevertheless, we can observe in Figure 4 higher residual in the corners, where the distance to the CoD is the greatest.



(**a**)



(**b**)

Figure 4. Inverse residual distortion with $k_1..k_9$ Coefficient Computed for the entire frame. (**a**) Inverse residual distortion with $k_1...k_4$ Coefficient Computed from 0 to max X frame; (**b**) Inverse residual distortion with $k_1..k_9$ Coefficient Computed from 0 to max X frame.

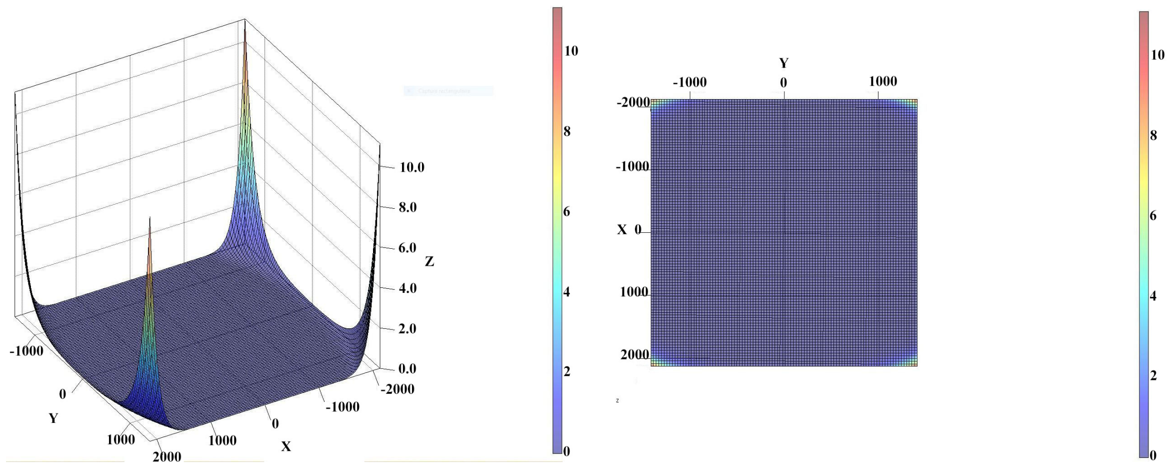


Figure 5. Inverse residual distortion with $k_1 ... k_9$ coef. Computed on the entire frame. (**a**) Axonometric view; (**b**) Top view.

Here follows a brief analysis of the residuals:

These two experiments show that the results are totally acceptable even if the residuals are higher in zone furthest from the CoD, *i.e.*, in the diagonal of the frame. As shown in Table 3, only 2.7 % of the frames have residuals > 1 pixel.

Table 3. Residuals on the full frame format.

Pixel by Residual	Nb Concerned Pixel
Pixels	10,000.0
Pixels with residual < 0.2	9344.0 (93.44 %)
Pixels with residual < 1.0	9732.0 (97.32 %)
Pixels with residual > 1.0	268.0 (2.68 %)

4.3. Inverse Distortion Computation on an Image Done with a Metric Camera

This short experiment used an image taken with a Wild P32 metric camera in order to work on an image without distortion. The Wild P32 terrestrial camera is a photogrammetric camera designed for close-range photogrammetry, topography, architectural and other special photography and survey applications.

This camera was used as film based, the film is pressed onto a glass plate fixed to the camera body on which 5 fiducial marks are incised. The glass plate prevents any film deformation.

The film format is 65 mm $\times$ 80 mm and the focal length, fixed, is 64 mm. Designed for architectural survey the camera has a high eccentricity and the 5 fiducial marks were used in this paper to compute the CoD. In Figure 6a four fiducial marks are visible (the fifth is overexposed in the sky). The fiducial marks are organized as follows: one at the principal point (PP), three at 37.5 mm from the PP (left,right,top) and one at 17.5 mm (bottom).

This image was taken in 2000 in the remains of the Romanesque Aleyrac Priory, in northern Provence (France) [42]. Its semi-ruinous state gives a clear insight into the constructional details of its fine ashlar masonry as witnessed by this image taken using a Wild P32 during a photogrammetric survey.

As this image did not have any distortion, we used a polynomial distortion coming from another calibration and adapted it to the P32 file format (see Table 4). The initial values of the coefficients have been conserved and the distortion polynom expressed in millimeters is the compensation due at any point of the file format. The important eccentricity of the CoD is used in the image rectification: the COD is positioned on the central fiducial marks visible on the images in Figure 6a,b.

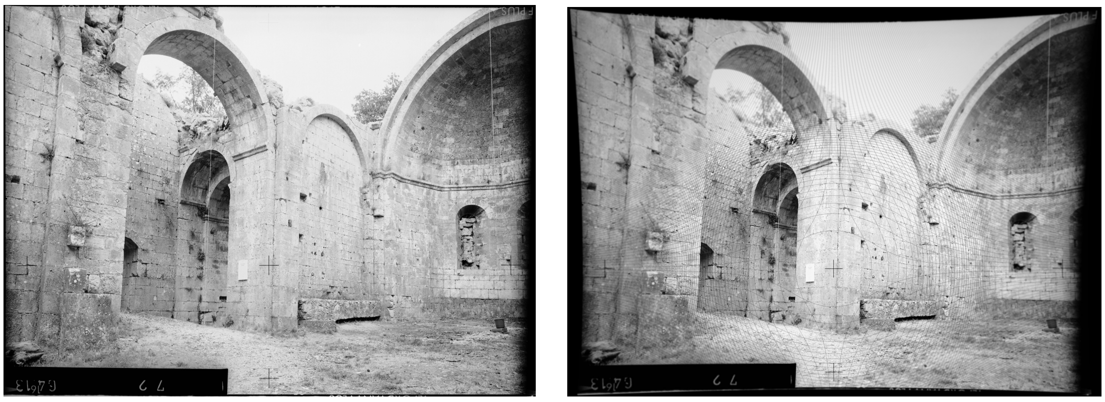


Figure 6. Distortion-less image taken using a small-format Wild P32 metric camera and application of an artificial distortion. (**a**) On the left, original image taken with P32 Wild metric camera; (**b**) On the right, pincushion distortion applied on this original image whithout interpolation. As images have not the same pixel size some vacant pixel are visible as black lines (see Hughues [32]). These lines surround the distortion center, here located on a fiducial mark, strongly shifted from the image center.

After scanning the image (the film was scanned by Kodak and the result file is a 4860 × 3575 pixel image), we first measure the five fiducial marks in pixels on the scanned image and then compute an affine transformation to pass from the scanned image in pixels to the camera reference system in millimeters where the central cross is located at (0.0, 0.0). This is done according to a camera calibration provided by the vendor, which gives the coordinates of each fiducial mark in millimeters in the camera reference system. Table 5 shows the coordinates of the fiducial marks and highlights the high eccentricity of the camera built for architectural survey. This operation is called internal orientation in photogrammetry and it is essential when using images coming from film-based camera that were scanned. The results of these measurements are shown in Table 5.

In Figure 6a we can see the original image taken in Aleyrac while in the Figure 6b we can see the result of the radial distortion inversion. Figure 7 shows the original image in grey and the image computed after a double inversion of the radial distortion model in green.

We can observe no visible difference in the image. This is correlated with the previous results in the second experiment, see Figure 4.

Table 4. Radial distortion comensation and then application of the inverse used with the image taken with the P32 camera.

Coef.	Original	Inverse
k_1	0.09532	-0.09532
k_2	-9.656×10^{-8}	0.02725780376
k_3	7.245×10^{-11}	-0.010392892306459602
k_4	0.0	0.004540497555744342
k_5	0.0	-0.0021482705738196948
k_6	0.0	0.0010711249019932042
k_7	0.0	$-5.542464764540273 \times 10^{-4}$
k_8	0.0	$2.948490225469636 \times 10^{-4}$
k_9	0.0	$-1.6024842649677896 \times 10^{-4}$

Table 5. Photograph taken with the Wild P32 camera: data and some results of the Internal Orientation.

Param.	X Value	Y Value
mm 2 Pixel	58.885	58.885
Frame size in pixel	4860	3575
Frame size in mm	82.54	60.71
CoD, ppx ppy (pixel)	2423.212	2377.528
Fiducial mark-up- (mm)	0.0	37.5
Fiducial mark-right- (mm)	37.5	0.0
Fiducial mark-down- (mm)	0.0	−17.5
Fiducial mark-left- (mm)	−37.5	37.5
Fiducial mark-center- (mm)	0.0	0.0

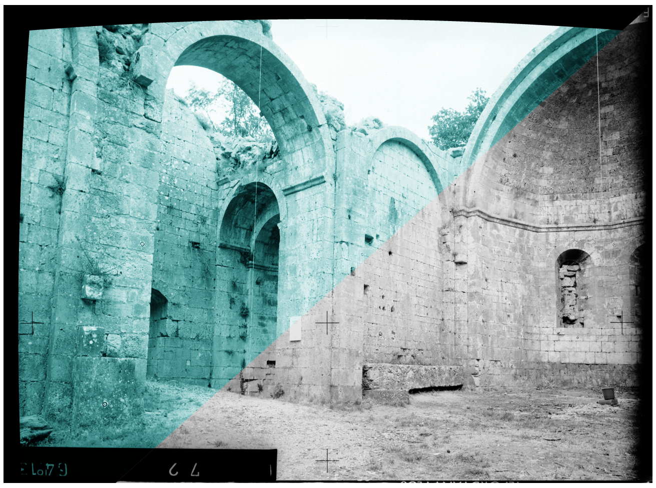


Figure 7. Distortion compensation applied on the pincushion image obtained in Figure 6b. In green the image corrected by inverse distortion; in black and white the original image.

5. Conclusions and Discussion

The experiments presented in this article show the relevance of the proposed methodology and the reliability of the result. However, a significant difference exists depending on whether the set of coefficients $k_1 ... k_4$ or $k_1 ... k_9$. For large distortion the number of parameters should be significant. See Figures 4 and 5 for the influence of the number of coefficients. We can note that since the formulation by Brown, the number of coefficients used to characterize the distortion has increased. In 2015 the Agisoft company added k_4 in their radial distortion model while at the same time many software still use only k_1, k_2 and in 2016 they add p_3 and p_4 to the tangential distortion model.

Even when $k_1 ... k_4$ are sufficient for compensating the radial distortion, it is however necessary to increase the degree of the polynomial to correctly compute the inverse.

5.1. A Bridge between PhotoModeler and Agisoft for Radial Distortion

One of the applications for using such a formula to compute the inverse distortion coefficient in function of $k_1 ... k_4$ is to convert distortion models between two software programs that use the inverse distortion model, as for example PhotoModeler and PhotoScan from Agisoft. Indeed PhotoModeler uses the Brown distortion model to compensate for observations made on images and so to obtain a theoretical observation without distortion effect. In contrary, PhotoScan from Agisoft uses a similar model but it adds the distortion to a point projected onto the image. To convert a distortion model from PhotoModeler to PhotoScan, or vice versa, we need to compute the inverse distortion model.

We need to take in consideration the unit used to express the 2D point coordinate: in PhotoModeler the points are measured in millimeters and their range is limited to the camera frame; whereas in PhotoScan, the points are normalized by the focal length.

To convert a distortion model from PhotoModeler to PhotoScan the following steps are necessary:

1. Given $k_1 ... k_3$ as the coefficients of the polynom modeling the radial distortion in PhotoModeler. Note that PhotoModeler uses only $k_1 ... k_3$ coefficient.
2. $k_1 = k_1 * focalmm^2$
   $k_2 = k_2 * focalmm^4$
   $k_3 = k_3 * focalmm^6$
3. Compute $k'_1 ... k'_4$ (PhotoScan uses k_4) according to Appendix C.

And to obtain the $k_1 ... k_3$ for PhotoModeler starting from $k_1 .. k_4$ given by PhotoScan we need:

1. Given $k_1 ... k_4$ as the coefficient of the polynom modeling the radial distortion in PhotoScan.
2. $k_1 = k_1 / focalmm^2$
   $k_2 = k_2 / focalmm^4$
   $k_3 = k_3 / focalmm^6$
   $k_4 = k_4 / focalmm^8$
3. compute $k'_1 ... k'_3$ (PhotoModeler uses k_3) according to Appendix C.

The proposed approach in this article allows to compute the new coefficients in function of $k_1 ... k_4$

5.2. Possible Limitations

Our results on inverse residual distortion suggests a decrease with the order of approximation. In the future it could be interesting to determine some analytical bounds on the maximal residual distortion. This bound would depend on the distortion coefficients k_1, k_2, k_3, k_4, $maxX$ and the order N. The crucial question would be to know whether this bound converges when N goes to infinity. This is absolutely not guaranteed since the formula of proposition 1 has been obtained by purely formal manipulations and the power series Q could be divergent (which implies the divergence of the residual). In this case, one could however expect good behavior, similarly to formal solutions of differential equations, whose approximations can be controled for $r < R$ until a bound N depending on R [43] (It is also explained more briefly in [44], Section 3 page 103). At a given N one could also expect that this bound will decrease if the distortion given by P is decreased.

We thought such technical questions may be of great interest, both from a mathematical perspective as well as an applied one. Obtaining theoretical results on the inverse residual distortion might influence the software community in adding more coefficients in the polynomial models.

Acknowledgments: The authors wish to thank two postdoctoral researchers in the team; Motasem Nawaf for his implication in the iterative inverse distortion method and Jean-Philip Royer for his implementation of the inverse distortion in Python in PhotoScan Software in order to import and export camera distortion toward other photogrammetric software.

Author Contributions: Pierre Drap designed the research, implemented the inverse distortion method and analyzed the results. Julien Lefèvre proved the mathematical part. Both authors wrote the manuscript.

Conflicts of Interest: The authors declare no conflict of interest.

Appendix A

In Equation (8) we replace P and Q by their power series in order to identify coefficients

$$1 = \sum_{m=0}^{+\infty} a_m r^{2m} \sum_{n=0}^{+\infty} b_n \left(rP(r)\right)^{2n}$$

For k fixed $P(r)^k$ can be rewritten as a product:

$$P(r)^k = \sum_{n_1=0}^{4} a_{n_1} r^{2n_1} \dots \sum_{n_k=0}^{4} a_{n_k} r^{2n_k}$$

which gives a more compact expression

$$P(r)^k = \sum_{m=0}^{4k} r^{2m} \sum_{\substack{n_1+\dots+n_k=m \\ 0\leq n_i\leq 4}} a_{n_1}\dots a_{n_k} = \sum_{m=0}^{4k} r^{2m} p(m,k)$$

Note that $p(m,k) = 0$ as soon as $m > 4k$. Then we obtain:

$$Q\big(rP(r)\big) = \sum_{n=0}^{+\infty} b_n r^{2n} \sum_{m=0}^{8n} r^{2m} p(m,2k)$$

$$Q\big(rP(r)\big) = \sum_{k=0}^{+\infty} r^{2k} \underbrace{\sum_{\substack{m+n=k \\ 0\leq n \\ 0\leq m\leq 8n}} b_n p(m,2n)}_{t(k)}$$

Let us call $t(k)$ the coefficient in the previous sum. Actually $t(k)$ will turn out to be $q(k)$, defined in Proposition 1. Last we can express the initial equality $P(r)Q\big(r(P(r)\big) = 1$:

$$\sum_{k=0}^{+\infty} a_k r^{2k} \sum_{j=0}^{+\infty} r^{2j} t(j) = 1$$

$$\sum_{l=0}^{+\infty} r^{2j} \sum_{k+j=l} a_k t(j) = 1$$

To obtain the equality:

$$\sum_{k=0}^{l} a_k t(l-k) = 0 \text{ for } l \geq 1$$

We decompose this sum:

$$a_0 t(l) + \sum_{k=1}^{l} a_k t(l-k) = 0$$

and since $a_0 = 1$ we conclude that $t(l)$ satisfy the same recurrence relationship as q in Proposition 1. The two quantities are therefore equal since they have the same initial value. We have also an alternative expression for q given by the definition of t depending on b_n and $p(m,2n)$ that gives:

$$b_k p(0,2k) + \sum_{\substack{m+n=k \\ 0\leq n \\ 1\leq m\leq 8n}} b_n p(m,2n) = t(k)$$

By remarking that $p(0,2k) = 1$ we obtain Proposition 1.

Appendix B

There are two ways of implementing the result of Proposition 1. One can be interested in having a symbolic representation of coefficients b_n with respect to a_1, a_2, a_3, a_4. But one can also simply obtain numeric values for b_n given numeric values for a_n.

The main ingredient in both cases is to compute efficiently the coefficients $p(j,k)$. One can start by the trivial result that if $n_1 + \dots + n_k = j$ then $n_1 + \dots + n_{k-1} = j - n_k$. But n_k can take only 5 values between 0 and 4 (there are no other coefficient a_n in our case). Therefore one can easily derive the recursive identity:

$$p(j,k) = p(j,k-1) + a_1 p(j-1,k-1) + a_2 p(j-2,k-1) + a_3 p(j-3,k-1) + a_4 p(j-4,k-1)$$

To compute b_n we need coefficients $p(j,2k)$ with the constraints $j+k=n$, $0 \leq k$ and $1 \leq j \leq 8k$. A table $(n-1) \times 2(n-1)$ is defined and coefficients are progressively filled by varying the coefficient k from 1 to $2(n-1)$ thanks to dynamic programming. This step is summarized in Algorithm B1.

Algorithm B1 Computation of $p(j,k)$

```
Require: coefficients $a_1, a_2, a_3, a_4$, integer $N$
  Define an array $p$ of size $N \times 2N-1$
  for $k = 0 : 2(N-1)$ do
      $p(0,k) = 1$
  end for
  for $k = 1 : 2(N-1)$ do
      for $j = 1 : 2(N-1)$ do
          $p(j,k) = p(j,k-1) + a_1 p(j-1,k-1) + a_2 p(j-2,k-1) + a_3 p(j-3,k-1) + a_4 p(j-4,k-1)$
                                                  ▷ With $p(j,k) = 0$ as soon as $j < 0$ or $k \leq 0$
      end for
  end for
  return $p$
```

The most tricky aspect is to manipulate formal terms with a_1, a_2, a_3, a_4. For that it is good to remark that coefficients b_n are made of terms $a_1^{n_1} a_2^{n_2} a_3^{n_3} a_4^{n_4}$ such as $n_1 + 2n_2 + 3n_3 + 4n_4 = n$. We can therefore have an a priori bound on each exponents, n, $n/2$, $n/3$, $n/4$ respectively. Given that, each multinomial term can be represented as a coefficient in a 4D array of bounded size. It is also very convenient to use a sparse representation for it due to many vanishing terms. Addition of terms are simply additions of 4D arrays of size bounded by n^4. Multiplication requires shifting operations on dimensions of the array, basically multiplying by $a_1^{n_1}$ corresponds to a translation by n_1 of the first dimension.

Algorithm B2 Computation of coefficients b_n

```
Require: coefficients $a_1, a_2, a_3, a_4$, integer $N$
  Define an array $q = [0,0,0,0]$
  Define an array $b$ of size $N$
  $b(0) = 1$
  Compute $p$ with Algorithm B1
  for $n = 1 : N$ do
      $tmp = 0$
      for $k = 1 : 4$ do
          $tmp = tmp - a_k q(k-1)$
      end for
      $b(n) = b(n) + tmp$
      for $k = n/9 : (n-1)$ do
          $b(n) = b(n) - b(k)p(n-k, 2k)$
      end for
      $q(1:3) = q(0:2)$ and $q(0) = tmp$
  end for
  return $b$
```

We summarize the computation (Algorithm B2).

Appendix C

Here are the formula for the nine first coefficients b_n. Note that there are no more coefficients $-k_n$ for b_n, when $n \geq 5$, since $k_n = 0$.

$$b_1 = -k_1$$

$$b_2 = 3k_1^2 - k_2$$

$$b_3 = -12k_1^3 + 8k_1k_2 - k_3$$

$$b_4 = 55k_1^4 - 55k_1^2k_2 + 5k_2^2 + 10k_1k_3 - k_4$$

$$b_5 = -273k_1^5 + 364k_1^3k_2 - 78k_1k_2^2 - 78k_1^2k_3 + 12k_2k_3 + 12k_1k_4$$

$$b_6 = 1428k_1^6 - 2380k_1^4k_2 + 840k_1^2k_2^2 - 35k_2^3 + 560k_1^3k_3 - 210k_1k_2k_3 + 7k_3^2 - 105k_1^2k_4 + 14k_2k_4$$

$$\begin{aligned}b_7 = {}&-7752k_1^7 + 15504k_1^5k_2 - 7752k_1^3k_2^2 + 816k_1k_2^3 - 3876k_1^4k_3 + 2448k_1^2k_2k_3 - 136k_2^2k_3 - 136k_1k_3^2\\ &+ 816k_1^3k_4 - 272k_1k_2k_4 + 16k_3k_4\end{aligned}$$

$$\begin{aligned}b_8 = {}&43263k_1^8 - 100947k_1^6k_2 + 65835k_1^4k_2^2 - 11970k_1^2k_2^3 + 285k_2^4 + 26334k_1^5k_3 - 23940k_1^3k_2k_3\\ &+ 3420k_1k_2^2k_3 + 1710k_1^2k_3^2 - 171k_2k_3^2 - 5985k_1^4k_4 + 3420k_1^2k_2k_4 - 171k_2^2k_4 - 342k_1k_3k_4 + 9k_4^2\end{aligned}$$

$$\begin{aligned}b_9 = {}&-246675k_1^9 + 657800k_1^7k_2 - 531300k_1^5k_2^2 + 141680k_1^3k_2^3 - 8855k_1k_2^4 - 177100k_1^6k_3\\ &+ 212520k_1^4k_2k_3 - 53130k_1^2k_2^2k_3 + 1540k_2^3k_3 - 17710k_1^3k_3^2 + 4620k_1k_2k_3^2 - 70k_3^3 + 42504k_1^5k_4\\ &- 35420k_1^3k_2k_4 + 4620k_1k_2^2k_4 + 4620k_1^2k_3k_4 - 420k_2k_3k_4 - 210k_1k_4^2\end{aligned}$$

References

1. Arfaoui, A. Geometric image rectification: A review of most commonly used calibration patterns. *Int. J. Signal Image Process. Issues* **2015**, *1*, 1–8.
2. Rahul, S.; Nayar, S.K. Nonmetric calibration of Wide-Angle Lenses and Polycameras. *IEEE Trans. Pattern Anal. Mach. Intell.* **2000**, *8*, 1172–1178.
3. Michael, C.; Wolfgang, K.; Jan, S.; Norbert, H.; Silvia, N.; Wallgrun, J. Data Capture. In *Handbook of Geographic Information*; Wolfgang, K., Danko, D.M., Eds.; Publishing House: Berlin, Germany, 2012; pp. 212–297.
4. Ben, T.; Murray, D.W. The impact of radial distortion on the self-calibration of rotating cameras. *Comput. Vis. Image Underst.* **2004**, *96*, 17–34.
5. Conrady, A.E. Lens-systems, Decentered. *Mon. Not. R. Astron. Soc.* **1919**, *79*, 384–390.
6. Brown, D.C. Close-range camera calibration. *Photom. Eng.* **1971**, *37*, 855–866.
7. Brown, D.C. Decentering Distortion of Lenses. *Photom. Eng.* **1966**, *32*, 444–462.
8. Santana-Cedrés, D.; Gómez, L.; Alemán-Flores, M.; Salgado, A.; Esclarín, J.; Mazorra, L.; Álvarez, L. An Iterative Optimization Algorithm for Lens Distortion Correction Using Two-Parameter Models. *IPOL J. Image Process. Line* **2015**, *1*, doi:10.5201/ipol.

9. Tordoff, B.; Murray, D.W. Violating rotating camera geometry: The effect of radial distortion on self-calibration. In Proceedings of the 15th International Conference on Pattern Recognition, Barcelona, Spain, 3 September 2000; pp. 423–427.
10. Dougherty, G. *Digital Image Processing for Medical Applications*; Editor, F., Meditor, A., Eds.; Cambridge University Press: New York, NY, USA, 2009.
11. De Villiers, J.P.; Wilhelm, L.F.; Geldenhuys, R. Centi-pixel accurate real-time inverse distortion correction. In Proceedings of the Optomechatronic Technologies, 7266 726611-1, San Diego, CA, USA, 17 November 2008.
12. Mondrian, P. Public Domain, 2015. Available online: http://www.aventdudomainepublic.org/mondrian (accessed on 25 May 2016).
13. Papadaki, A.I.; Georgopoulos, A. Development, comparison, and evaluation of software for radial distortion elimination. In Proceedings of the SPIE 9528, Videometrics, Range Imaging, and Applications XIII, Munich, Germany, 21 June 2015; doi:10.1117/12.2184569.
14. Kasser, M. Photogrammetrie et vision par ordinateur. *XYZ* **2008**, *12*, 49–54.
15. Eos Systems Inc. PhotoModeler Software. 2016. Available online: http://www.photomodeler.com/index.html (accessed on 29 May 2016).
16. Agisoft. PhotoScan Software. 2016. Available online: http://www.agisoft.com/ (accessed on 29 May 2016).
17. OpenCV. Toolbox. 2016. Available online: http://opencv.org// (accessed on 29 May 2016).
18. Fraser, C.S. Photogrammetric Camera Component Calibration: A Review of Analytical Techniques. In *Calibration and Orientation of Cameras in Computer Vision*; Gruen A., Huang, T., Eds.; Springer Verlag: Berlin, Germany, 2001; pp. 95–119.
19. Zhang, Z. Camera Calibration. In *Emerging Topics in Computer Vision*; Medioni, G., Kang, S.B., Eds.; Prentice Hall Technical References: Upper Saddle River, NJ, USA, 2004; pp. 1–37.
20. Shortis, M. Calibration Techniques for Accurate Measurements by Underwater Camera Systems. *Sensors* **2015**, *15*, 30810–30826.
21. Hartley, R.I. Camera Calibration Using Line Correspondences. In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition, Seattle, WA, USA, 21–23 June 1994; pp. 361–366.
22. Hartley, R.I. Projective reconstruction from line correspondences. In Proceedings of the IEEE Computer Society Conference on Computer Vision and Pattern Recognition (CVPR '94), Seattle, WA, USA, 21–23 June 1994; pp. 903–907.
23. Devernay, F.; Faugeras, O.D. Straight lines have to be straight. *Mach. Vis. Appl.* **2001**, *13*, 14–24.
24. Nomura, Y.; Sagara, M.; Naruse, H.; Ide, A. Self-calibration of a general radially symmetric distortion model. *IEEE Trans. Pattern Anal. Mach. Intell.* **2002**, *14*, 1095–1099.
25. Claus, D.; Fitzgibbon, A.W. A Plumbline Constraint for the Rational Function Lens Distortion Model. In Proceedings of the British Machine Vision Conference, Oxford, UK, 15 September 2005; pp. 99–108.
26. Tardif, J.-P.; Sturm, P.; Roy, S. Self-calibration of a general radially symmetric distortion model. In Proceedings of the 9th European Conference on Computer Vision, Graz, Austria, 7–13 May 2006.
27. Rosten, E.; Loveland, R. Camera distortion self-calibration using the plumb-line constraint and minimal Hough entropy. *Mach. Vis. Appl.* **2009**, *22*, 77–85.
28. Slama, C.C. *Manual of Photogrammetry 4 Edition*; Slama, C.C., Theurer, C., Hendrikson, S.W., Eds.; American Society of Photogrammetry: Falls Church, VA, USA, 1980; pp. 32–58.
29. Atkinson, K. *Close Range Photogrammetry and Machine Vision*; Atkinson, F.K., Fryer, J.G., Eds.; Whittles Publishing: Dunbeath, UK, 2001.
30. Heikkilä, J.; Silven, O. A Four-Step Camera Calibration Procedure with Implicit Image Correction. *CVPR97* **1997**, *22*, 1106–1112.
31. Basu, A.; Licardie, S. Alternative models for fish-eye lenses. *Pattern Recognit. Lett.* **1995**, *16*, 433–441.
32. Hughes, C.; Glavin, M.; Jones, E.; Denny, P. Review of geometric distortion compensation in fish-eye cameras. In Proceedings of the Signals and Systems Conference, Galway, Irish, 18–19 June 2008; pp. 162–167.
33. Hughes, C.; Glavin, M.; Jones, E.; Denny, P. Accuracy of fish-eye lens models. *Appl. Opt.* **2010**, *49*, 3338–3347.
34. De Villiers, J.P.; Nicolls, F. Application of neural networks to inverse lens distortion modelling. In Proceedings of the 21st Annual Symposium of the Pattern Recognition Society of South Africa (PRASA), Stellenbosch, South Africa, 22–23 November 2010; pp. 63–68.

35. Mallon, J.; Whelan, P.F. Precise Radial Un-distortion of Images. In Proceedings of the 17th International Conference on Pattern Recognition (ICPR 04), Cambridge, UK, 23–26 August 2004.
36. Heikkilä, J. Geometric Camera Calibration Using Circular Control Points. *IEEE Trans. Pattern Anal. Mach. Intell.* **2000**, *22*, 1066–1077.
37. Wei, G.-Q.; Ma, S. Implicit and explicit camera calibration: Theory and experiments. *IEEE Trans. Pattern Anal. Mach. Intell.* **2002**, *16*, 469–480.
38. Jin, H. Method and Apparatus for Removing General Lens Distortion From Images. US Patent 8,265,422, 23 September 2014.
39. Han, D. Real-Time Digital Image Warping for Display Distortion Correction. In Proceedings of the Second International Conference on Image Analysis and Recognition ICIAR'05, Toronto, ON, Canada, 2005; pp. 1258–1265.
40. Abeles, P. Inverse Radial Distortion Formula. Less Than Optimal. Available online: http://peterabeles.com/blog/?p=73 (accessed on 29 May 2016).
41. Alvarez, L.; Gómez, L.; Sendra, J.R. An Algebraic Approach to Lens Distortion by Line Rectification. *J. Math. Imaging Vis.* **2009**, *35*, doi:10.1007/s10851-009-0153-2.
42. Drap, P.; Grussenmeter, P.; Hartmann-Virnich, A. Photogrammetric stone-bystone survey and archeological knowledge: An application on the Romanesque Priory Church Notre-Dame d' Aleyrac (Provence, France). In Proceedings of the VAST2000 Euroconference, Prato, Italy, 24–25 November 2000; pp. 139–145.
43. Ramis, J.P. Les séries k-sommables et leurs applications, Complex Analysis, Microlocal Calculus and Relativistic Quantum Theory. In Proceedings of the Colloquium Held at Les Houches, Centre de Physique, Berlin, Germany, 23 September 1979; pp. 178–199.
44. Ramis, J.P. Séries divergentes et procédés de resommation. Available online: http://www.math.polytechnique.fr/xups/xups91.pdf (accessed on 29 May 2016).