An Exact Formula for Calculating Inverse Radial Lens Distortions
Pierre Drap, Julien Lefèvre

To cite this version:
Pierre Drap, Julien Lefèvre. An Exact Formula for Calculating Inverse Radial Lens Distortions. Sensors, MDPI, 2016, 16 (6), pp.807. 10.3390/s16060807. hal-01458821

HAL Id: hal-01458821
https://hal-amu.archives-ouvertes.fr/hal-01458821
Submitted on 7 Feb 2017

HAL is a multi-disciplinary open access archive for the deposit and dissemination of scientific research documents, whether they are published or not. The documents may come from teaching and research institutions in France or abroad, or from public or private research centers. L’archive ouverte pluridisciplinaire HAL, est destinée au dépôt et à la diffusion de documents scientifiques de niveau recherche, publiés ou non, émanant des établissements d’enseignement et de recherche français ou étrangers, des laboratoires publics ou privés.
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.

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 planar 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 fisheye 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.](image)

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
\[
    r = ||P_c||
\]
\[
dr = 1 + k_2r^2 + k_4r^4 + ...
\]
\[
P_c = P_n / dr
\]
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 modeled 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:

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

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

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

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 $T_2$:

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

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

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

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

$$
Q(r') := \sum_{n=0}^{+\infty} b_n r'^{2n}
$$
with \(a_0 = 1, a_1 = k_1, \ldots, a_n = k_n\) (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 \(r' = ar\) in order to have the same domain of definition for \(P\) and \(Q\). So \(Q\) reads:

\[
Q(r') = \sum_{n=0}^{\infty} b'_n r^{2n}
\]

with \(b'_n = b_n a^{2n}\). 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:

\[
r = rP(r)Q(rP(r))
\]

and at the end:

\[
1 = P(r)Q(rP(r))
\]

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 \(a_1, \ldots, a_4\) are given. It is summarized in the following result:

**Proposition 1.** Given the sequence \(a_1, \ldots, a_4\) it is possible to obtain the recursive relation:

\[
b_0 = 1 \text{ and for } n \geq 0 \quad b_n = -\sum_{k=1}^{4} a_k q(n-k) - \sum_{1 \leq k \leq 8} b_k p(j, 2k) \]

where we use the following intermediate coefficients:

\[
p(j, k) = \sum_{0 \leq n_i \leq 4} a_{n_1} \ldots a_{n_k}
\]

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

We will derive this expression in Appendix A and show how the coefficients \(b_1, \ldots, b_n\) 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 \(a_0, \ldots, a_4\) and \(q(n-1), \ldots, q(n-4)\) that only depends on the sequence \(a_n\). Similarly the second summation involves \(b_0, \ldots, b_{n-1}\) 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:

$$b_1 = -a_1$$  \hspace{1cm} (10)

$$b_2 = 3a_1^2 - a_2$$  \hspace{1cm} (11)

$$b_3 = 8a_1a_2 - 12a_1^3 - a_3$$  \hspace{1cm} (12)

$$b_4 = 55a_1^4 + 10a_1a_3 - 55a_1^2a_2 + 5a_2^2 - a_4$$  \hspace{1cm} (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>
<thead>
<tr>
<th>Radial Distortion Coefficients</th>
<th>Original Value</th>
<th>Computed Inverse Value</th>
</tr>
</thead>
<tbody>
<tr>
<td>$k_1$</td>
<td>$1.532 \times 10^{-4}$</td>
<td>$1.532 \times 10^{-4}$</td>
</tr>
<tr>
<td>$k_2$</td>
<td>$-9.656 \times 10^{-8}$</td>
<td>$1.6697072 \times 10^{-7}$</td>
</tr>
<tr>
<td>$k_3$</td>
<td>$7.245 \times 10^{-11}$</td>
<td>$-2.33941625216 \times 10^{-10}$</td>
</tr>
<tr>
<td>$k_4$</td>
<td>0.0</td>
<td>$3.1255518770316804 \times 10^{-13}$</td>
</tr>
<tr>
<td>$k_5$</td>
<td>0.0</td>
<td>$-4.774156462972984 \times 10^{-16}$</td>
</tr>
<tr>
<td>$k_6$</td>
<td>0.0</td>
<td>$7.680785197322419 \times 10^{-19}$</td>
</tr>
<tr>
<td>$k_7$</td>
<td>0.0</td>
<td>$-1.1582853960835112 \times 10^{-21}$</td>
</tr>
<tr>
<td>$k_8$</td>
<td>0.0</td>
<td>$2.1694555835054252 \times 10^{-24}$</td>
</tr>
<tr>
<td>$k_9$</td>
<td>0.0</td>
<td>$-3.779164309884112 \times 10^{-27}$</td>
</tr>
</tbody>
</table>

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 original distortion and after \( n \) inversions (\( n = 2 \) and \( n = 10,000 \)).

<table>
<thead>
<tr>
<th>Coefficient</th>
<th>Original</th>
<th>Inverse 1</th>
<th>Delta Loop 1</th>
<th>Delta Loop 10,000</th>
</tr>
</thead>
<tbody>
<tr>
<td>( k_1 )</td>
<td>( 1.532 \times 10^{-4} )</td>
<td>(-1.532 \times 10^{-4} )</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>( k_2 )</td>
<td>(-9.656 \times 10^{-8} )</td>
<td>(1.6697072 \times 10^{-7} )</td>
<td>0.0</td>
<td>0.0</td>
</tr>
<tr>
<td>( k_3 )</td>
<td>( 7.245 \times 10^{-11} )</td>
<td>(-2.33941625216 \times 10^{-10} )</td>
<td>(1.292469707114 \times 10^{-26} )</td>
<td>(1.292469707 \times 10^{-26} )</td>
</tr>
<tr>
<td>( k_4 )</td>
<td>0.0</td>
<td>(3.1255518770316804 \times 10^{-13} )</td>
<td>(1.009741958682 \times 10^{-28} )</td>
<td>(1.009842932 \times 10^{-24} )</td>
</tr>
</tbody>
</table>

This first experiment 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 \( \times \) 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 \( \text{dist}(p, p_2) \).

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 coefficients \( k_1 \)...\( k_9 \) 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.
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.
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.

<table>
<thead>
<tr>
<th>Pixel by Residual</th>
<th>Nb Concerned Pixel</th>
</tr>
</thead>
<tbody>
<tr>
<td>Pixels</td>
<td>10,000.0</td>
</tr>
<tr>
<td>Pixels with residual &lt; 0.2</td>
<td>9344.0 (93.44 %)</td>
</tr>
<tr>
<td>Pixels with residual &lt; 1.0</td>
<td>9732.0 (97.32 %)</td>
</tr>
<tr>
<td>Pixels with residual &gt; 1.0</td>
<td>268.0 (2.68 %)</td>
</tr>
</tbody>
</table>

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 × 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](image1.png)

**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 without interpolation. As images have not the same pixel size some vacant pixel are visible as black lines (see Hughes [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>
<thead>
<tr>
<th>Coef.</th>
<th>Original</th>
<th>Inverse</th>
</tr>
</thead>
<tbody>
<tr>
<td>$k_1$</td>
<td>0.09532</td>
<td>−0.09532</td>
</tr>
<tr>
<td>$k_2$</td>
<td>$-9.656 \times 10^{-8}$</td>
<td>0.02725780376</td>
</tr>
<tr>
<td>$k_3$</td>
<td>$7.245 \times 10^{-11}$</td>
<td>−0.010392892306459602</td>
</tr>
<tr>
<td>$k_4$</td>
<td>0.0</td>
<td>0.00454049755574342</td>
</tr>
<tr>
<td>$k_5$</td>
<td>0.0</td>
<td>−0.002148705738196948</td>
</tr>
<tr>
<td>$k_6$</td>
<td>0.0</td>
<td>0.0010711249019932042</td>
</tr>
<tr>
<td>$k_7$</td>
<td>0.0</td>
<td>−5.542464764540273 $\times 10^{-4}$</td>
</tr>
<tr>
<td>$k_8$</td>
<td>0.0</td>
<td>2.948490225469636 $\times 10^{-4}$</td>
</tr>
<tr>
<td>$k_9$</td>
<td>0.0</td>
<td>−1.6024842649677896 $\times 10^{-4}$</td>
</tr>
</tbody>
</table>
Table 5. Photograph taken with the Wild P32 camera: data and some results of the Internal Orientation.

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

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 into 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. 
   \[ k_1 = k_1 \times \text{focalmm}^2 \]
   \[ k_2 = k_2 \times \text{focalmm}^4 \]
   \[ k_3 = k_3 \times \text{focalmm}^6 \]

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. 
   \[ k_1 = \frac{k_1}{\text{focalmm}^2} \]
   \[ k_2 = \frac{k_2}{\text{focalmm}^4} \]
   \[ k_3 = \frac{k_3}{\text{focalmm}^6} \]
   \[ k_4 = \frac{k_4}{\text{focalmm}^8} \]

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 suggest 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 $k_1,k_2,k_3,k_4$, $\text{maxX}$ 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 controlled for $r < R$ until a bound 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

\[
1 = \sum_{m=0}^{+\infty} a_m r^{2m} + \sum_{n=0}^{+\infty} b_n (rP(r))^{2n}
\]

For $k$ fixed $P(r)^k$ can be rewritten as a product:
\[ P(r)^k = \sum_{n_1=0}^{4} a_{n_1} r^{2n_1} \cdots \sum_{n_k=0}^{4} a_{n_k} r^{2n_k} \]

which gives a more compact expression

\[ P(r)^k = \sum_{m=0}^{4k} r^{2m} \sum_{0 \leq n_1 \leq 4} a_{n_1} \cdots a_{n_k} = \sum_{m=0}^{4k} r^{2m} p(m, k) \]

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

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

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 \( P(r)Q(rP(r)) = 1 \):

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

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

To obtain the equality:

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

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:

\[ b_n p(0, 2k) + \sum_{m+n=k}^{1 \leq m \leq 8n} b_n p(m, 2n) = t(k) \]

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 \( n_1 + \ldots + n_k = j \) then \( n_1 + \ldots + n_{k-1} = j - n_k \). 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
    $p(j, k) = p(j, k - 1) + a_1 p(j - 1, k - 1) + a_2 p(j - 2, k - 1) + a_3 p(j - 3, k - 1) + a_4 p(j - 4, k - 1)$
  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 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
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 $k_n = 0.$

$$b_1 = -k_1$$

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

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

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

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

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

$$b_7 = -7752k_1^7 + 15504k_1^5k_2 - 7752k_1^3k_2^2 + 816k_1^2k_2^3 - 3876k_1^4k_3 + 2448k_1^2k_2k_3 - 136k_2^3k_3 - 136k_1k_3^2$$

$$+ 816k_1^3k_4 - 272k_1k_2k_4 + 16k_3k_4$$

$$b_8 = 43263k_1^8 - 100947k_1^6k_2 + 65835k_1^4k_2^2 - 11970k_1^2k_2^3 + 285k_2^4 + 26334k_1^3k_3 - 23940k_1^2k_2k_3$$

$$+ 3420k_1k_2^2k_3 + 1710k_1^3k_4 - 171k_1k_2k_3 - 5985k_1^2k_4 + 3420k_1k_2k_3 - 171k_2^2k_4 - 342k_1k_3k_4 + 9k_4$$

$$b_9 = -246675k_1^9 + 657800k_1^7k_2 - 531300k_1^5k_2^2 - 141680k_1^3k_2^3 - 8855k_1^2k_2^4 + 177100k_1^2k_3$$

$$+ 212520k_1k_2k_3 - 531300k_1^3k_3 + 1540k_2^3k_3 - 17710k_1^2k_3^2 + 4620k_1k_2k_3^2 - 70k_3^3 + 42504k_1^2k_4$$

$$- 35420k_1k_2k_4 + 4620k_1^2k_4 + 4620k_1^2k_3k_4 - 420k_2k_3k_4 - 210k_1k_4$$

References


