Correcting Lens Distortions in Digital Photographs

Wolfgang Hugemann

Abstract

The wide-angle lenses (or rather zoom lenses when set to short focal length) typically produce a pronounced barrel distortion. This lens distortion affects damage mappings (i. e. the superposition of damage photographs) as well as perspective rectifications. Lens distortion can however be mostly corrected by applying suitable algorithmic transformations to the digital photograph.

The paper presents the algorithms used for this correction, together with programs that perform either the entire task of correction or that allow one to determine the lens correction parameters. The paper concludes with some (rectified) example images and an estimation of the gains in accuracy achieved by applying lens correction algorithms.

Introduction

Ideally, a photograph should be a perfect perspective mapping of the photographed scene. This holds especially when the photograph is further processed into a to-scale representation of the pictured object, as often is the case in accident reconstruction, e. g. for shots of damaged vehicles or of the road surface – the latter often taken from an elevated position and then rectified.

To keep lens distortions reasonably small, the general advice is to use a small-angle lens, if possible a telephoto lens. In practice, this is however often impossible, as the space around the photographed object is limited and the required distance to the object cannot be achieved. This holds especially for shots of the road surface, which are mostly taken by use of wide-angle lenses in order to cover the desired space. Furthermore, there are a lot of 'external' photographs, over which the reconstructionist has no influence on the camera settings. These are pictures taken by the people involved in the accident or by the police, who – at least in Germany – often have to make do with inadequate equipment.

By the use of suitable software, lens distortions can be corrected in retrospect, eliminating much of the error produced by unsatisfactory shots. In the following, we will present mathematical approaches to describe lens distortion, present some programs that will do most of the job for you, and demonstrate the accuracy achieved.

Modelling lens distortion

When transforming picture coordinates into real-world coordinates, we mostly use cartesian systems for both. This coordinate system seems to fit the problem most naturally, as photographs are rectangular and some (not very) special set-up situations (so-called nadir or coplanar photographs) are simply to-scale mappings of the photographed plane.

When describing lens distortions, we should however use a polar coordinate system, with the lens's main axis as its origin: obviously, the lens is an axially symmetric object, so we should expect all distortions to be rotation-symmetric. We assume the lens's main axis (the *principal axis*) to meet the image plane at an exact right angle, at the *principal point*.

In order to describe lens distortions, it is sufficient to investigate coplanar photographs, as any additional distortion created by the camera set-up will just be to the perspective and can be attended to in a later step. In coplanar perspective mapping, the real-world coordinates are just a fixed multiple of the image coordinates. It is common to denote real-world coordinates by capital letters X, Y and image coordinates by small letters x, y. So for a coplanar perspective mapping we arrive at

$$X - X_0 = c_1\,(\hat{x} - x_0) \tag{1}$$

$$Y - Y_0 = c_2\,(\hat{y} - y_0) \tag{2}$$

These equations allow one to define the origins of both coordinate systems freely. Common choices for the coordinate origin in digital photographs are the upper left corner and the principal point, which should ideally coincide with the middle point of the image. Furthermore, the equations consider different scale factors for the x- and y-directions, c_1 and c_2. For digital photographs, these scale factors are identical, but video cameras might (virtually) use non-square pixels, as is the case in DV cameras.

In the equations above we used $\hat{x}, \hat{y}$ rather than x, y to denote the image coordinates of the ideal perspective mapping: this is the common way to denote estimates, and estimates they are, having to be derived from the physical image coordinates x, y.

In order to describe lens distortion, it suffices to establish the relationship between the physical coordinates of a pixel x, y and the coordinates $\hat{x}, \hat{y}$ of the ideal perspective mapping which should be used in the above equations. In the following, we assume $\hat{x}, \hat{y}$ to refer to the principal point and x, y to refer to the physical centre of the image, the offset of the principal point being denoted by $\hat{x}_p, \hat{y}_p$. We can split lens distortion into a radial and a tangential (torsional) part

$$\hat{r}^2 = \hat{x}^2 + \hat{y}^2 \tag{3}$$

$$r^2 = (x - \hat{x}_p)^2 + (y - \hat{y}_p)^2 \tag{4}$$

$$\hat{r} = f(r)\,r \tag{5}$$

$$\hat{r}\alpha = g(r, \alpha)\,r\alpha \tag{6}$$

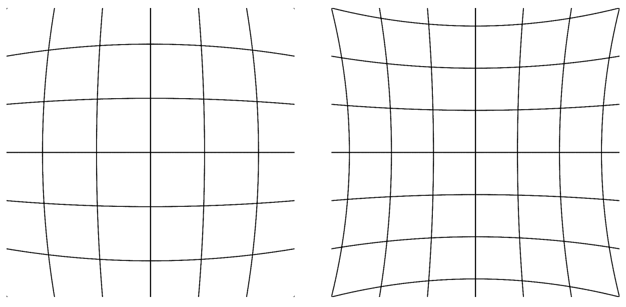


Fig. 1: Common lens distortions [2]

This offset of the principal point $\hat{x}_p, \hat{y}_c$ is camera-specific, as it results from manufacturing tolerances in regard to the mutual mounting of the camera sensor and the lens. The distortion functions $f(r), g(r)$ are however specific for a certain make and model of camera, if the (zoom) lens is non-interchangeable, as is the case for consumer and mobile phone cameras. For interchangeable lenses, the distortion is lens-specific, possibly needing some correction in regard to the exact sensor size of the camera it is mounted on.

It is common to model the distortion by dimensionless functions $f(r), g(r, \alpha)$, like in the above equations. Moreover, the radius r is often normalised to the dimensions of the sensor, such that $f(r)$ and r are both of magnitude one.

Experience shows that the effects of radial lens distortion $f(r)$ exceed those of torsional distortion $g(r, \alpha)$ at least by an order of magnitude. Consequently, most software only models radial distortion, i. e. tries to determine the functional relationship $f(r)$ for a certain make and model of camera, for example, an SLR camera combined with a specific lens. For zoom lenses, the functional relationship $f(r)$ is specific for a certain focal length, i. e. the functional relationship has to be modelled for different settings of the focal length. Basically, we have to distinguish between, figure 1 [2]:

barrel distortion $f(r) < 1$
: The apparent effect is that of an image which has been mapped around a sphere or barrel.

pincushion distortion $f(r) > 1$
: The visible effect is that lines that do not go through the centre of the image are bowed inwards, towards the centre of the image, like a pincushion.

Most consumer cameras show pronounced barrel distortion when set to short focal length.

Modelling radial distortion

The most common functional approach for $f(r)$ is a polynomial

$$f(r) = 1 + a_1 r + a_2 r^2 + \ldots + a_n r^n \tag{7}$$

Optical theory shows that this polynomial should only feature even powers

$$f(r) = 1 + a_2 r^2 + a_4 r^4 + \ldots + a_{2n} r^{2n} \tag{8}$$

In practice, the number of parameters is often limited to about three, i. e. either

$$f(r) = 1 + a_1 r + a_2 r^2 + a_3 r^3 \tag{9}$$

or

$$f(r) = 1 + a_1 r^2 + a_2 r^4 + a_3 r^6 \tag{10}$$

with some of the coefficients possibly being zero.

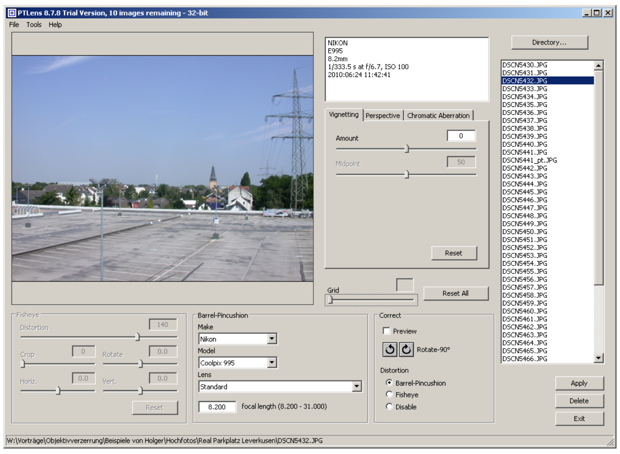
Tangential distortion

The most common model incorporating tangential distortion is the Brown-Conrady model [6, 5], which adds a tangential component to the radial distortion

$$\begin{pmatrix} \hat{x} \\ \hat{y} \end{pmatrix} = (1 + a_1 r^2 + a_2 r^4 + a_3 r^6) \begin{pmatrix} x \\ y \end{pmatrix} + \begin{pmatrix} 2a_4 xy + a_5(r^2 + 2x^2) \\ a_4(r^2 + 2y^2) + 2a_5 xy \end{pmatrix} \tag{11}$$

Determining the lens parameters

When determining the lens parameters, all programs rely on the same paradigm: the ideal perspective mapping should map real world straight lines to straight lines in the image. So if a set of



PTLens 8.7.8 Trial Version, 10 images remaining - 32-bit

```
NIKON
E995
8.2mm
1/333.5 s at f/6.7, ISO 100
2010:06:24 11:42:41
```

Amount 0
Midpoint 50

Fisheye: Distortion 140; Crop 0; Rotate 0.0; Horiz. 0.0; Vert. 0.0

Barrel-Pincushion: Make Nikon; Model Coolpix 995; Lens Standard; 8.200 focal length (8.200 - 31.000)

Correct: Preview; Rotate-90°

Distortion: [✓ Barrel-Pincushion] Fisheye Disable

```
DSCN5430.JPG
DSCN5431.JPG
DSCN5432.JPG
DSCN5433.JPG
DSCN5434.JPG
DSCN5435.JPG
DSCN5436.JPG
DSCN5437.JPG
DSCN5438.JPG
DSCN5439.JPG
DSCN5440.JPG
DSCN5441.JPG
DSCN5441_pt.JPG
DSCN5442.JPG
DSCN5443.JPG
DSCN5444.JPG
DSCN5445.JPG
DSCN5446.JPG
DSCN5447.JPG
DSCN5448.JPG
DSCN5449.JPG
DSCN5450.JPG
DSCN5451.JPG
DSCN5452.JPG
DSCN5453.JPG
DSCN5454.JPG
DSCN5455.JPG
DSCN5456.JPG
DSCN5457.JPG
DSCN5458.JPG
DSCN5459.JPG
DSCN5460.JPG
DSCN5461.JPG
DSCN5462.JPG
DSCN5463.JPG
DSCN5464.JPG
DSCN5465.JPG
DSCN5466.JPG
```

```
W:\Vorträge\Objektivverzerrung\Beispiele von Holger\Hochfotos\Real Parkplatz Leverkusen\DSCN5432.JPG
```

Fig. 2: PTlens's program window (stand-alone version)

points $P_0, P_1, \ldots, P_n$ is known to lie on a straight line, their images $p_0, p_1, \ldots, p_n$ should also fall onto a straight line, i. e. satisfy the equation

$$\vec{p_i} = \vec{p_0} + \varrho_i \vec{e} \tag{12}$$

Any deviation from this rule has to be attributed to lens distortion

$$\vec{p_i} = \vec{f}(\vec{p_i}) \tag{13}$$

We need two points to determine the two parameters defining a straight line. Each additional point will then provide one more equation to determine the parameters used in $\vec{f}(\vec{r})$. So if our functional approach is

$$\hat{r} = \left(1 + a_1 r + a_2 r^2\right) r \tag{14}$$

we would have to provide at least four points on one straight line in order to determine the two sought parameters a_1 and a_2. In practice, calibration programs mostly use a rectangular grid of straight lines, often a chequerboard, to generate a set of equations and then calculate the mapping parameters by a nonlinear least-squares fit. Some programs generate the set of control points on their own, often using pre-defined templates; other programs require the user to select the control points from the image.

Ready-made solutions

Fortunately, there are quite a few programs that will correct lens distortion in arbitrary photographs automatically. These programs rely on the EXIF information embedded in the digital photograph, providing the make, model and focal length. They are shipped with an extensive camera/lens database, providing the correction parameters for a variety of camera/lens combinations.

The most-used program is probably *PTlens*, which perfectly meets the needs of the reconstructionist: a graphical Windows interface, a vast camera/lens database, a good accuracy and a reasonable price. The program can be used as a stand-alone application as well as a plug-in for Adobe Photoshop, figure 2.

PTlens will read the EXIF information from the digital photograph, look up the camera in its database and apply its lens correction, all of it with minimal user interaction. If your camera cannot be found in its database, you may provide calibration photographs (shot at various focal lengths) to Tom Niemann, the author of *PTlens*, who will update the camera database within a few days. In fact, *PTlens's* vast database has been set-up this way, i. e. members of its wide-spread user community providing calibration photographs to its author. Problems may however arise from 'external' photographs shot with a camera model unknown to *PTlens*. In this case, you will either have to get access to a camera of that make and model (in orer to shot the required calibration photographs) or use the parameters of a 'comparable' camera model.

PTlens obviously relies on the correction algorithms provided by *Panorama Tools*, an early panorama stitching tool developed by Harry Deutsch. (Which explains its somewhat crude name.) *Panorama Tools* uses a third-order polynomial approach to describe radial distortion, neglecting the camera-specific offset between the centre of the image and the principal point. This allows to establish lens correction parameters based on makes and models of cameras, not individual cameras.

We have to keep in mind that the offset of the lens's axis to the centre of the image does not directly generate distortion. Neglecting this offset this will only mean that the lens correction applied to a certain image point is based on the 'wrong' radius. So it only affects the accuracy of the correction algorithm and may therefore be considered as a correction term of lower order.

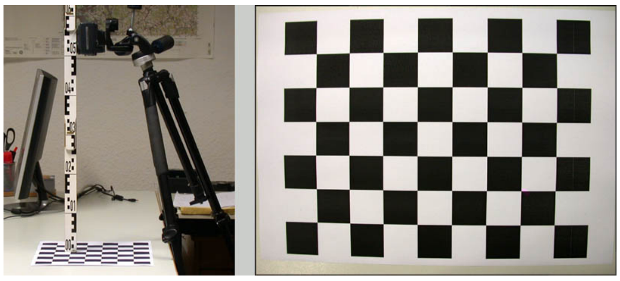

a) camera setup b) test photograph

Fig. 3: Experimental setup for *PC-Rect's* lens calibration

Furthermore, it has to be pointed out that the estimation of the image centre is an ill-posed problem [7], i. e. it is difficult to achieve stable estimates for it by evaluation of the image content. In practice, the extimates for the offsets x_c, y_c will depend on the lens distortion model we use [8]. In our calibrations of three Nikon Coolpix 995 cameras, we observed the estimates for the principal point fall into a circle of about 30 pixels radius around the image centre, corresponding to a manufacting tolerance of ± 0.1 mm, which seems quite large. We are not aware of manufacturer data on the production tolerances in this regard.

PTlens uses half of the smaller dimension of the image (i. e. its height h in landscape format) for the normalisation of the radius. It then chooses the coefficients such that the height on the vertical middle line of the image remains unaltered (non-scaling restraint)

$$\varrho = 2r/h \tag{15}$$

$$f(\varrho) = 1 + c_1\varrho + c_2\varrho^2 + c_3\varrho^3 \tag{16}$$

$$0 = c_1 + c_2 + c_3 \tag{17}$$

The last line guarantees that $f(\varrho = 1) = 1$. *Panorama Tools* and its graphical user interfaces are still under continuous development in projects like *PTGui* (commercial) and *Hugin* (open source).

PTlens's current database, being the 'marrow' of the program, is encrypted and can only be read by *PTlens* itself and very few other applications whose authors licensed it. According to recent information given to me by the author of *PTlens*, he intends to maintaining its lens database in the foreseeable future.

Special solutions for photogrammetry

In photogrammetry, the parameters describing the lens distortion are treated as a subset of the inner orientation of the camera (in contrast to its outer orientation, as described by the linear coordinates of its position and the angular coordinates of its viewing direction). The parameters of the inner orientation are either known from the start or have to be determined during the calibration process for the photograph. An obvious example of the former parameter type is the focal length, which can be read from the EXIF data.

Photogrammetry in accident reconstruction mostly means the rectification of a single photograph of a flat surface such as the roadway, whereby multiple photographs may be mounted to a mosaic image. In this approach, the calibration of a perfectly perspective mapping needs four match points. So if the inner orientation of the camera is unknown, more than four match points can be used for the calibration, allowing one to determine the distortion parameters of the lens as one proceeds. This approach is followed by some modern 3D photogrammtry programs, when the user provides more match points than are actually needed.

A related technique is well-known to reconstructions from former times: the Rollei Réseau cameras had a glass plate with a crosshair grid mounted in front of the film. When calibrating the image, the coordinates of these crosshair markings on the photopositive had to determined in advance. Thereby one eliminated effects such as film undulation in the camera and film shrinkage during the watery development process. These effects, which had to be considered as part of the inner orientation of the camera, are absent in digital photography.

In recent years, the first mentioned approach, i. e. calibration the camera in advance, independently from the scene taken, has become the more popular: it simplifies matters, as the calibration process is performed only once and for all, and from then on, four match points suffice to calibrate the rectification of a single photograph. Furthermore, the lens parameters can be determined more accurately by special calibration photographs.

The trade-off of this approach however is that the lens distortion of an unknown camera cannot easily be compensated for by taking additional measurements at the photographed scene, to be used as additional match points.

Most relevant to reconstructionists, *PC-Rect* allows one to correct the lens distortion for a specific camera at a certain focal length, using the Brown-Conrady model eq. (11) with a fourth-order approach for the radial distortion. The remaining four lens parameters a_1 ... a_4 are automatically determined from a photograph of a test pattern, consisting of a 9×7 chequerboard, figure 3. The parameters are only valid for a specific focal length of a certain make and model of camera. At the moment, it is however impossible to store a general model for a zoom camera, covering all focal lengths, in *PC-Rect*.

Lens correction in panoramas

Sophisticated panorama stitching programs will take care of lens distortion, as dedicated panorama cameras use very wide-angle lenses or even fisheye lenses, with severe lens distortions. In the automatic stitching mode (i. e. the default mode), the control point sets, defining the matching between the single photographs, are generated automatically. These sets comprise dozens of control points, i. e. more than enough to determine the lens distortion parameters during the optimisation.

Consequently, these parameters are part of the optimisation process and are estimated along with other parameters, like the horizontal field of view and the overlap of the single shots. In some programs (*Hugin*) the lens correction parameters can be read from the program's output.

Script-based lens correction

The trick of reading the camera's make and model as well as the focal length from the EXIF data and applying the adequate lens correction can even be performed by *ImageMagick*, which has *PTlens's* correction model built-in. With this approach, one will not easily cover a vast amount of camera makes and models, but the lens correction for ones own camera(s) can be automated by a VisualBasic script along the lines of that presented in Appendix A. This does however require knowledge of the lens correction coefficients and their dependency on focal length, as implemented in the sample script.

Getting lens correction coefficients

Each time you create a panorama with *Hugin* or *PTGui*, the radial distortion parameters are estimated and used to correct the images before they are stitched together into a panorama. Thus the easiest approach would be to create a panorama with, say, *Hugin* and read the parameters from the program's output. However, this estimation of the lens parameters would not be stable enough for a general use in image correction. In order to get a stable estimate, ome rather has to invest some work by hand.

Ready-made lens parameter sets

Until February 2006 *PTlens's* database was coded in XML format, i. e. an easily editable text format. This 2006 version of *PTlens's* XML database is still (legally) available at *Hugin's SourceForge* website [3] and provides data for a lot of older camera models.

When *PTlens's* database became encrypted, the authors of *Hugin* tried to establish a free XML coded lens database as an alternative. This database is called *LensFun* and can be downloaded at Berlios [4]. It comes with a complete programming interface, but all we basically need is the information for our camera in the XML file.

```
<lens>
  <maker>Nikon</maker>
  <model>Standard</model>
  <mount>nikon995</mount>
  <cropfactor>4.843</cropfactor>
  <calibration>
    <distortion model="ptlens" focal="8.2" a="0" b="-0.019966" c="0" />
    <distortion model="ptlens" focal="10.1" a="0" b="-0.010931" c="0" />
    <distortion model="ptlens" focal="13.6" a="0" b="-0.002049" c="0" />
    <distortion model="ptlens" focal="18.4" a="0" b="0.003845" c="0" />
    <distortion model="ptlens" focal="23.4" a="0" b="0.006884" c="0" />
    <distortion model="ptlens" focal="28.3" a="0" b="0.008666" c="0" />
    <distortion model="ptlens" focal="31" a="0" b="0.009298" c="0" />
  </calibration>
</lens>
```

Fig. 4: XML code for the Nikon Coolpix 995 in the LensFun database

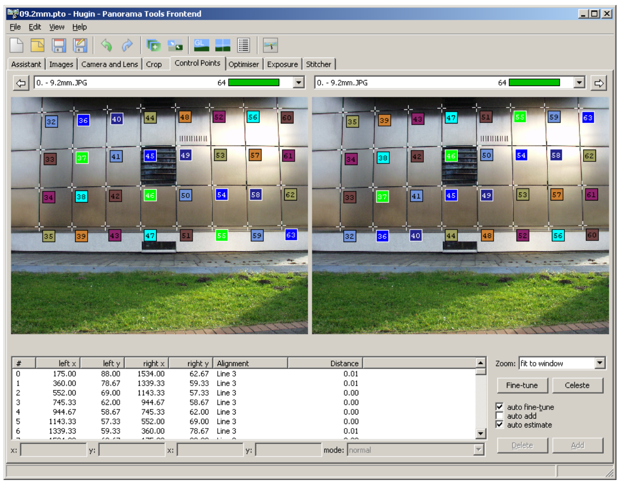
As an example, we will pick the lens correction parameters for the once popular Nikon Coolpix 995 in the following, figure 4. The information is found in the file `compact-nikon.xml`, which resides in the directory `\data\db`. The file can be examined by the use of a text editor or an XML viewer such as *XML Marker*.

As can be taken from the technical data sheet, the zoom range of the Nikon Coolpix 995 is 8.2 – 31.0 mm, corresponding to 38 – 152 mm in 35 mm format. This gives a crop factor of 152 / 31 = 4.90, which roughly corresponds to the 4.843 given the XML file. The coefficients of the distortion equation are supplied for six focal lengths, namely 8.2 mm, 10.1 mm, 13.6 mm, 18.4 mm, 23.4 mm, 28.3 mm and 31.0 mm. The coefficients c_1 (`a`) and c_3 (`c`) are set to zero, i. e. the distortion is described only by the second-order term c_2, as predicted by theory, eq. (8).

For the six focal lengths provided, the correction coefficient c_2 can be read from the XML file. For other focal lengths a suitable correction parameter can be determined by linear interpolation between the two neighbouring focal lengths. As an alternative, the dependency of c_2 on the focal length f can be approximated by the polynomial

$$\begin{aligned} c_2 = 0.000\,005\,142 f^3 - 0.000\,380\,839 f^2 \\ + 0.009\,606\,325 f - 0.075\,316\,854 \end{aligned} \tag{18}$$

So the focal length read from the EXIF information is used to calculate the lens correction parameter c_2 by eq. (18) in the first step, and then, in a second step, the lens correction is performed according to eq. (16).


09.2mm.pto - Hugin - Panorama Tools Frontend
File Edit View Help
Assistant | Images | Camera and Lens | Crop | Control Points | Optimiser | Exposure | Stitcher
0. - 9.2mm.JPG
0. - 9.2mm.JPG
Fine-tune
Celeste
Delete
Add

#	left x	left y	right x	right y	Alignment	Distance
0	175.00	88.00	1534.00	62.67	Line 3	0.01
1	360.00	78.67	1339.33	59.33	Line 3	0.01
2	552.00	69.00	1143.33	57.33	Line 3	0.00
3	745.33	62.00	944.67	58.67	Line 3	0.00
4	944.67	58.67	745.33	62.00	Line 3	0.00
5	1143.33	57.33	552.00	69.00	Line 3	0.00
6	1339.33	59.33	360.00	78.67	Line 3	0.01

Zoom: fit to window
[✓] auto fine-tune
[ ] auto add
[✓] auto estimate
mode: normal

Fig. 5: Selecting control points for the calculation of lens parameters. Modern architecture provides excellent targets with lots of straight lines.

Calibrating from scratch

As pointed out in the section 'Determining lens parameters', the basic idea is to take a photograph of an object that has a lot of guaranteed straight lines, tell these to a suitable program and let it determine the appropriate lens correction parameters. In the following, we will demonstrate this technique by the use of *Hugin* and the lens model of *Panorama Tools*, but one could use *PC-Rect* and its Brown-Conrady lens model just as well.

The basic approach when using *Hugin* is described in the 'Simple Lens Calibration Tutorial' on *Hugin's* website. However, the image presented there is not well-suited, and using only two straight lines for the lens calibration will not yield stable estimates for the parameters.

Instead, you should take a photograph of a modern building, as proposed on the *PTlens's* website. (Follow all the instructions given there.) Then establish a grid on this photograph by determining the pixel coordinates of a lot of grid points. You can use any image viewer to do this, namely one that can store such data. We use polylines in *WinMorph* to do so, figure 5. Then open the calibration image with *Hugin*, set the panoramic mapping to rectilinear (on the last tab page) and store the project. Open the *Hugin* file (which is a plain text file with the extension PTO) with a text editor and supply a point list. A single line looks like this:

```
c n0 N0 x175.0 y87.8 X1533.3 Y62.6 t3
```

where `x`, `y` are the point coordinates in the source image and `X`, `Y` are the point coordinates in the target image – which actually are two versions of the same image in this special case. (Usually these would be two different images lying next to each other in a panorama.) The intro `c n0 N0` is standard code and the trailer `t3` (respectively `t4`, `t5`, `t6` ...) is the numbering of the associated straight line, starting with the index 3.

Of course, `x, y` and `X, Y` have to lie on the same straight line. They must however not be identical, as the optimiser would refuse to work under such conditions. The easiest approach is to use the reverse ordering for the target coordinates `X, Y`. You can use any program to establish a set of lines based on the intersection points derived from the photograph. (We used Excel to perform this task.) When ready, copy the point list to the corresponding section of the PTO file, save it and re-open it with *Hugin*.

Then switch to *Hugin's* optimiser, choose 'Optimize the custom parameters below', pick a, b, c and then press 'Optimize now!'. The result should give parameters which are close to 0.01. If not, check the point settings on the tab page 'Control points' (which is limited by the fact that each point is used twice, such that you will only see the second half of the control point set). If you have calculated large values for the parameters, the control points are probably out of order or not correctly associated with their corresponding lines.

If you have to re-run the optimiser, turn on the check box 'Edit script before optimising' on the right button of the according tab page: Set the start vector a, b, c back to `a0.0 b0.0 c0.0` before re-starting the optimiser. Otherwise the (non-linear) iteration will start at an off-point and would probably not yield the correct result.

For a camera equipped with a fixed lens, one does this calibration once and for all. For a camera with a zoom lens, one has to cover the

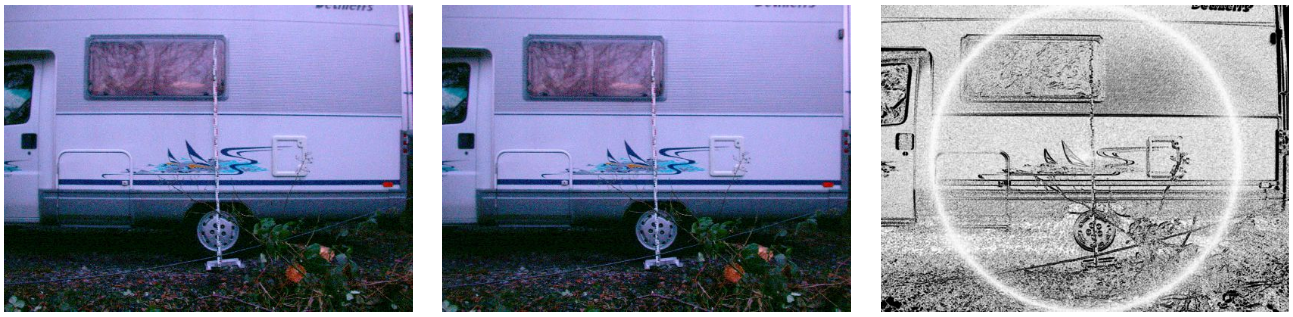


Fig. 6: Wide-angle photograph of a motor caravan, shot with a Nikon Coolpix 995 at $8.2\,\mathrm{mm}$ focal length

entire range of focal lengths by calibrating at about five different focal lengths. A ready-made example, both with a calibration image and the corresponding *Hugin* project can be downloaded at [3].

Examples

To-scale vehicle photographs

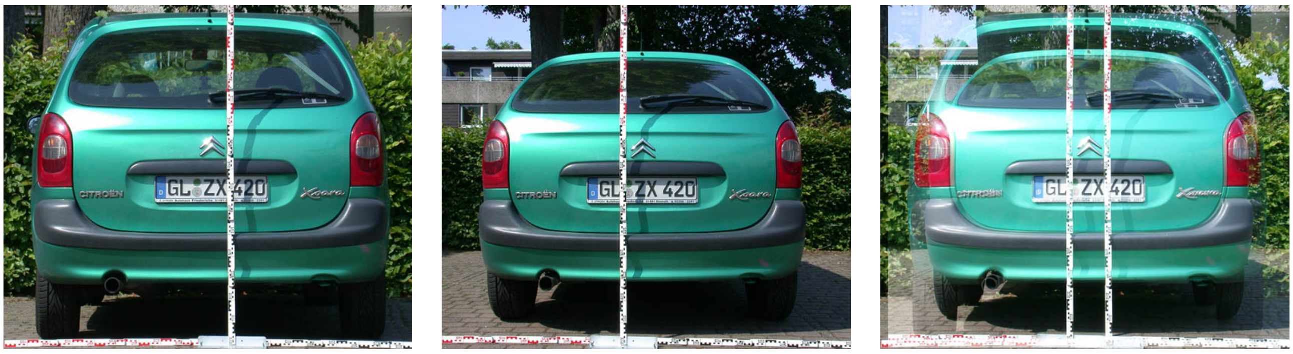
The original photograph in figure 6a had to be taken from a rather short distance during dusk, as space was limited due to a steep declivity at the back of the photographer. (The poor lighting conditions explain the blue tint which stems from severe lightening in the post-processsing.) The original photograph shows pronounced barrel distortion, visible especially in the horizontal stripe near the top of the image and for the back corner of the build-up. The Nikon Coolpix 995 used for this shot is found in *PTlens's* database, so the distortion could readily be corrected, figure 6b.

Figure 6c shows the difference between greyscale versions the two photographs, calculated by subtraction of the two, followed by negation and extreme clipping and Gamma correction. Again, the effects of the correction are best illustrated by the horizontal stripe at the top. The white circle (indicating zero difference) results from the non-scaling restriction eq. (17): the points on a circle with a diameter equal to the smaller dimension of the image remain unaltered.

However, lens correction of wide-angle photographs is no cure-all, figure 7. Although the horizontal folding rule becomes straight in the corrected wide-angle photograph (whereas being markedly bended in the original photograph), the overall picture is still not to-scale. This becomes evident when being overlayed to a telelens shot, figure 7c: although the scalings of the vertical rulers perfectly match, the horizontal stretch of the wide-angle shot is still too narrow, which is best illustrated by its rear lights. Furthermore, the upper part of the vehicle is compressed in the wide-angle shot.

These effects mostly originate from the fact that the back of the vehicle is of course no perfect plane: the bumper (in front of which the vertical ruler is placed) sticks out of the back of the vehicle by about $10\,\mathrm{cm}$ and the rear window (and its surroundings) are even more off-plane. Although the rulers establish a perfect scaling in the plane they span, other points, which fall out of this plane are off-scale (so-called parallax error).

This is to say that lens correction will help to get wide-angle shots a little bit more to-scale, but cannot eliminate their shortcomings. Wide-angle shots – as they are usually taken by loss adjusters and policemen – remain problematic when used in damage mappings. At least, lens correction can avoid the additional distortions introduced by wide-angle shots.



a) $f = 32\,\mathrm{mm}$ b) $f = 8.2\,\mathrm{mm}$, corrected c) comparison

Fig. 7: Comparison: wide-angle photograph and tele-shot of a passenger car (Nikon Coolpix 995)

Photogrammetry

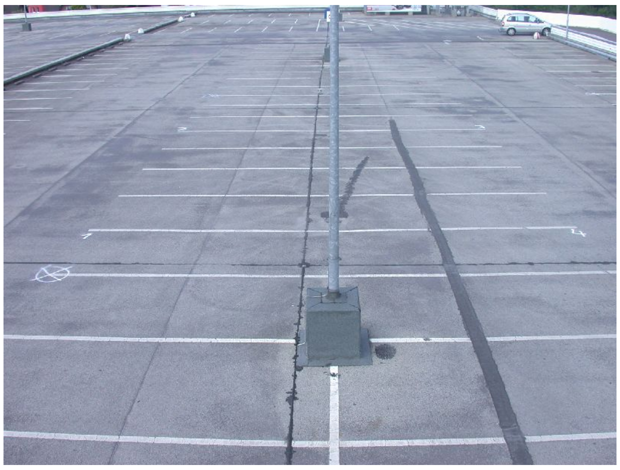
When using wide-angle photographs in photogrammetry, the effects of lens distortion can severely be enlarged by those of the perspective mapping, figure 8. The photographs on the left where taken by use of a window cleaning rod from about 5 m height with a camera pitch that bans the horizon from the image, i. e. the entire image area can be used for photogrammtric purposes. To the contrast, the photograph on upper the right side of the figure is taken from eye height with the horizon at the middle of the image, representing the standard set-up of 'external' photographs.

After the rectification the four match points fall exactly onto the corresponding points of the aerial photographs, regardless whether lens correction is applied or not, as this is an intrinsic feature of the perspective mapping. The mapping of points which fall out of the quadrilateral circumscribed by the control points can however become severely wrong. Obviously, the shortcomings of the ideal perspective mapping and the benefits of the lens correction become more pronounced if the photograph is taken from eye-height – as generally is the case when shot by the (German) police or participants of the accident. In the photograph taken from eye height it becomes also obvious that the surface of the parking lot is not exactly level, causing significant warping of the long middle line running between the opposite parking spaces.

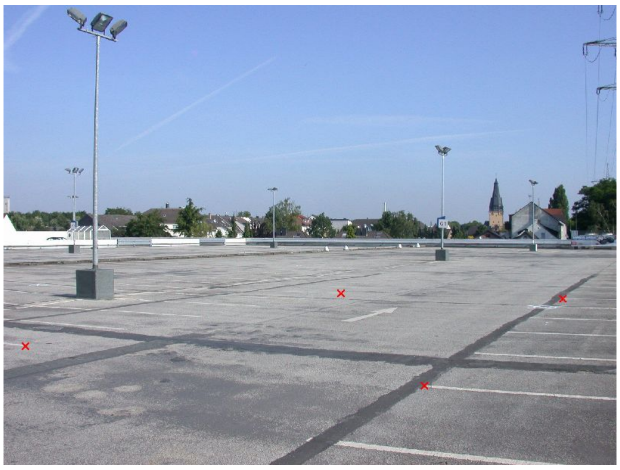
The images in figure 8 were calculated by the use of *ImageMagick*, performing the lens correction as well as the rectification. By this way of proceding, we do not have to pick the calibration points twice, in the original and in the lens-corrected image, as would have to be done in a two-step procedure. Instead, the coordinates of the calibration points in the lens-corrected image were calculated from those chosen in the original image, using the same transformation formula as for the lens correction. Furthermore, *ImageMagick* can handle sub-pixel coordinates (i. e. real numbers), mitigating rounding errors.

The chosen (sub-) pixels were then mapped onto the same pixels in the rectified images, such that the calibration points exactly match in the rectified images. *ImageMagick* can restrict the size of the rectified image to that of the original (i. e. restrict the amount of information per pixel roughly to that of the original), so the two rectified versions can readily be placed onto the corresponding aerial photograph, allowing a definitive comparison.

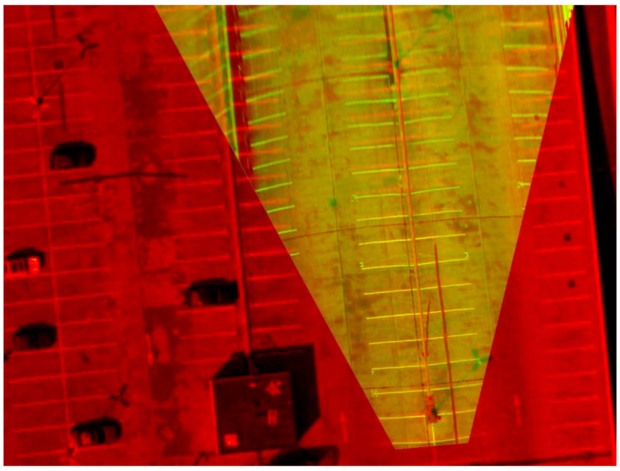
The other option would have been to use a commercial image rectification program with built-in lens correction, such as *PC-Rect*. This would have meant that we have had to pick each match point twice, in the original photograph and in the lens-corrected image, intruducing ano-



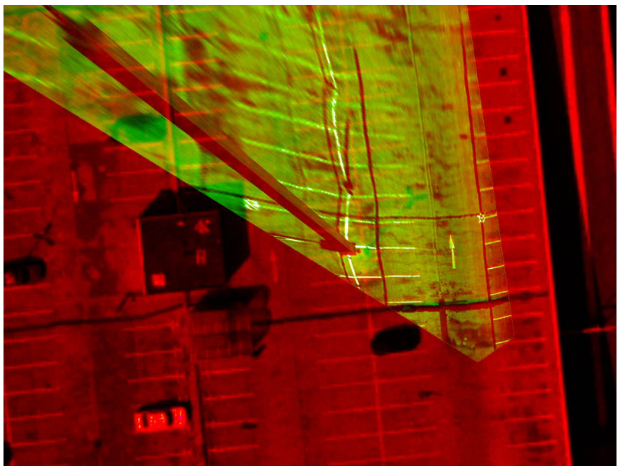
a) original – elevated position ($5\,\mathrm{m}$) – match points are labelled 1 ... 4



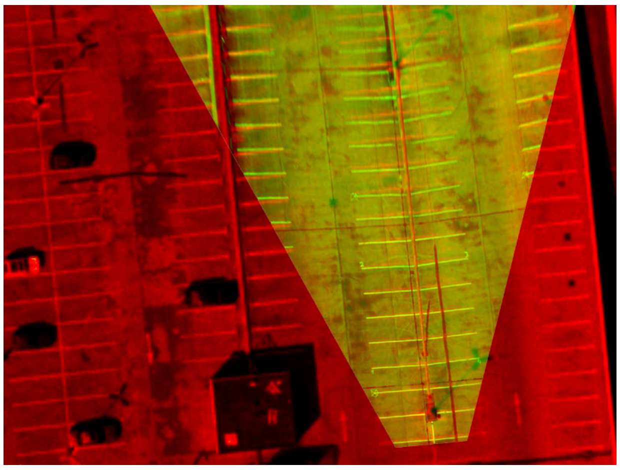
d) original – eye height – match points are marked by red crosses



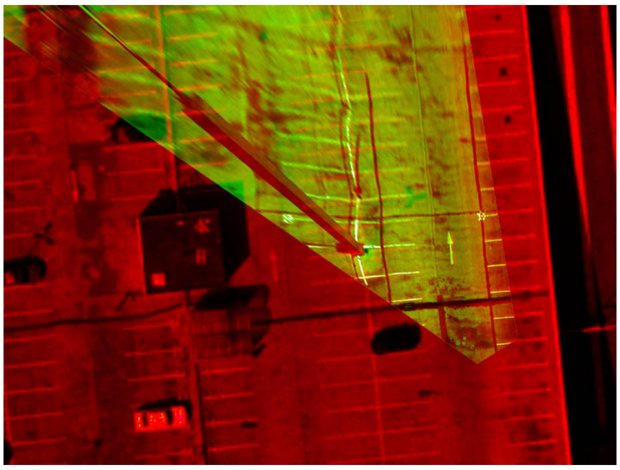
b) rectified version compared to aerial photograph



e) rectified version compared to aerial photograph



c) same comparison for lens corrected rectified version



f) same comparison for lens corrected rectified version

Fig. 8: Rectifications of wide-angle photographs taken with a Nikon Coolpix 995 in comparison to aerial photographs (ground resolution $20 \times 20\,\mathrm{cm}$). The aerial photograph is tinted red, the rectified versions are tinted green.

ther source of error by the user's choice. This would have also meant different overall dimensions and orientations for the rectified images and would thereby have introduced some degree of freedom how to exactly place the rectified versions onto the aerial photograph.

Conclusion

Low- and mid-priced wide-angle lenses – especially zoom lenses when set to short focal length – show pronounced lens distortion. This mostly comes as barrel distortion, i. e. the off-centre distances (radii) of points near the image borders are less than than they should be in an ideal perspective mapping. In contrast to that, the effects of tangential distorsion are mostly negligable.

Due to manufacturing tolerances, the lens's axis does not meet the camera sensor at the exact centre of its sensitive area – as it ideally should. This does not directly create distortion, but an offset of the distortion centre, affecting the correction algorithm. The effects created by this are specific to an individual camera and cannot be eliminated by a database referring to makes and models.

Lens distortion is present in most photographs used for accident reconstruction purposes, as the average user – including policemen and loss adjusters – usually sets the camera to the shortest focal length, in order to 'catch as much information as possible'. For the same reason, even photographs taken specifically for photogrammetric purposes often use short focal lengths. The perspective mapping can amplify the effects of lens distortion quite drastically.

Radial lens distortion can nowadays easily be compensated by the use of lens correction algorithms and camera databases. For reconstruction purposes, *PTlens* is the tool of choice, offering good correction results and a vast camera/lens database at a reasonable price. Especially photographs used for damage mappings and photogrammetry should be lens-corrected prior to their use.

The result of the lens correction can be somewhat enhanced when calibrating an individual camera/lens combination, especially in regard to the offset of the lens's axis to the centre of the image sensor's sensitive area. This may be relevant to high-precision photogrammetry, but hardly pays for accident reconstruction purposes.

References

[1] http://epaperpress.com/ptlens

[2] www.wikipedia.de

[3] http://sourceforge.net/projects/hugin/files/PTLens%20Database

[4] http://lensfun.berlios.de

[5] Brown, D. C.
Decentering Distortion and the Definitive Calibration of Metric Cameras
29þAnnual Meeting of the Americamn Society of Photogrammtric Engineering (1965)

[6] Conrady, A.
Decentered Lens Systems
Monthly Notices of the Royal Astronomical Society 79 (1919), pp. 384 – 390.

[7] Ruiz, A.; López-de-Teruel, P. E.; García-Mateos, G.
A Note on Principal Point Estimability
16th International Conference on Pattern Recognition (ICPR 2002)

[8] Läbe, T.; Förstner, W.
Geometric Stability of Low-cost Digital Cameras
http://www.isprs.org/proceedings/XXXV/congress/comm1/papers/95.pdf

Contact

Wolfgang Hugemann
Ingenieurbüro Morawski + Hugemann
von-Diergardt-Str. 19
51375 Leverkusen
Germany
hugemann@unfallrekonstruktion.de

English version suplied by the author(s).

Proofread by Richard Lambourn

Appendix A

```
'********************************************
' This script corrects the barrel distortion of a Nikon Coolpix 995
' The correction factors depend on the focal length, which is read
' from the EXIF header in the first step. The corrected version
' of the image is stored with a JPEG quality of 80% in a file carrying the
' trailer "_ptr"
'
' Version 1.0 vom 30.12.2009
' © by Wolfgang Hugemann, IB Morawski + Hugemann
'********************************************
'
const strConv = "Convert" ' filename of ImageMagick's convert tool
const strAdd = "_ptr"     ' trailer for the modified file
Dim wsh, fs
Set wsh = CreateObject("Wscript.Shell")
Set fs = CreateObject("Scripting.FileSystemObject")
'
' Name of input and ouput file
strFileIn = WScript.Arguments(0)
Pos = InStrRev(strFileIn,".")
strFileOut = Left(strFileIn,Pos - 1) & strAdd & Mid(strFileIn, Pos)
'
' Determination of the focal length
' This is given as a fraction of two LONG values, e.g. 8.2 mm = 82/10
' In order to calculate the real value, we make use of the EVAL function
command = "cmd /k identify -format ""%[EXIF:FocalLength]"" " & strFileIn
Set objExec = wsh.Exec(command)
strf = objExec.StdOut.Readline
f = eval(strf)
'
' For the Nikon Coolpix 995, we need only the coefficient 'b' (c_2)
' of the square term. Its dependence on the focal length has been modelled
' by a third-order polynomial
b =  0.000005142 * f * f * f -0.000380839 * f * f + _
     0.009606325 * f  -0.075316854
d = 1 - b
b=replace(b,",",".")
d=replace(d,",",".")
'
' Setting the adequate of the convert command
Command = strConv & " """ & strFileIn _
& """ -quality 80%% -virtual-pixel black -filter point -distort Barrel ""0.0 " _
& b & " 0.0 " & d & """ """ & strFileOut & """"
MsgBox command
wsh.run command, 7, true
```