CORRECTION OF RADIAL DISTORTION IN DIGITAL IMAGES

J.Jedlička, M.Potůčková

Charles University in Prague
Faculty of Science

Abstract

This article describes a method of the correction of radial distortion in digital images using Matlab computing environment. The article focuses on two problems. First one is an implementation of an algorithm for the radial distortion correction and the second one is building a simple and easy to use application with Matlab GUI for the radial distortion correction. Matlab was chosen for solving this task because it supports fast and easy calculations with matrixes (in this case raster files) and it also contains a set of build-in functions for 1D and 2D interpolation. Thanks to these the development of the new application was really fast.

1. Introduction

Lens distortions are phenomena that prohibit applying the simple pinhole camera principle in most of photogrammetric applications. Distortions belong to optic deficiencies called aberrations that cause a degradation of the final image. In contrary to other aberrations, distortions do not affect quality of the image but have a significant impact on the image geometry. There can be found two types of distortions, radial distortion and tangential distortion. However, only radial distortion has a significant influence on the image geometry. Tangential distortion is usually insignificant and is not included into computing of distortion correction. Radial distortion is a deficiency in straight lines transmission. The effect of radial distortion is that straight lines are bended as general curves and points are moved in the radial direction from their correct position. Together with a spatial transformation, the correction of radial distortion is the key step in the image rectification or orthorectification. Especially when working with non-metric digital cameras, the radial distortion reaches significant values and a correction of this distortion should be the first step in image processing.

2. Radial distortion

As mentioned before, radial distortion is a deficiency in a straight line transmission. There are two major types of radial distortion. The first one is a negative displacement also called barrel distortion. Barrel distortion occurs when points are moved from their correct position towards the centre of the image. The second type of radial distortion is a positive displacement, which occurs when points are displaced further away from the optical axis. This type is also called pincushion distortion. Barrel distortion is common for wide angle lenses and pincushion distortion for narrow angle lenses [1].

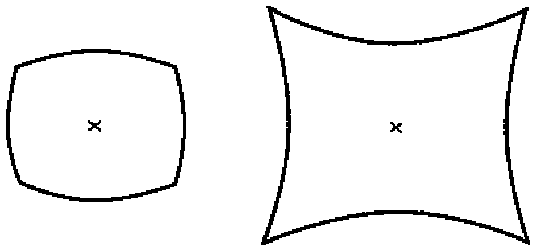

Figure 1: Effect of radial distortion on image geometry. Doted lines represent original image without any distortion. Solid lines show the effect of barrel (left) and pincushion (right) distortion.

Radial distortion is usually not perfectly rotationally symmetrical but for a computation of distortion it is assumed to be symmetrical. If the image should be used for the measurement of distances, radial distortion of lenses should be rotationally symmetrical (or at least nearly rotationally

symmetrical). If it is not, the correction for symmetrical distortion can cause errors in the position of some points.

Due to the presumption of its rotational symmetry, radial distortion over an image can be presented as a general curve describing a dependency between a radial distance from the image center and radial distortion. Because an underlying distortion function is usually not known and can not be obtained by analytic means, the polynomial approximation of radial distortion function is used [6]. When using photogrammetric cameras, the polynomial approximation of radial distortion is granted from a camera producer. In these days using of non-metric cameras is more and more popular because of their low cost and easy manipulating. In case of non-metric camera, a calibration must be carried out before its use in photogrammetric applications [2]. The camera calibration report contains necessary information about camera's internal parameters including radial distortion. Camera calibration is based on the measurement of control points. Differences between measured and calculated control point coordinates is used for a construction of the polynomial approximation of radial distortion. Specialized software packages as Photomodeler or Camera Calibration toolbox for Matlab are designed for the calibration of non-metric cameras. When polynomial approximation of radial distortion is known, it can be used for correction of radial distortion in an image.

3. ALGORTIHM DEVELOPMENT

The aim of this work was to develop a function that could correct radial distortion over an image. We were focused on correction of radial distortion only, not on estimating of a polynomial approximation of radial distortion (for this purpose Camera Calibration toolbox should be used in Matlab computing environment [3]). The basic idea was quite simple. If we know radial distortion values over the image, we are able to estimate a radial shift for each pixel and move this pixel to the desired position. Basic trigonometric rules were used to compute the radial distance of a pixel from the image center. After estimating the radial distance, its value was an input for computing a magnitude of the radial shift in this point. By subtracting the magnitude of the radial shift from the radial distance, the correct radial distance of a pixel from the image center is estimated. This radial distance is then converted to x, y [row, column] coordinates which represent the desired position of a pixel in the image (in image coordinates). If this procedure is applied to every pixel in the image, the result is a set of non-uniformly spaced points. These points are associated with the appropriate pixel values, so the corrected image can be interpolated.

The first step in developing the function was a construction of a universal interface that allows for inserting polynomial approximation of radial distortion. Two possibilities of inserting these parameters were implemented. The first one is passing the polynomial approximation as an executable string (must contain a variable R which represents the radial distance), for example "0.005*R.^2-35" (keeping Matlab rules for operators). This approach is really universal because every polynomial approximation can be written as a function. The second option expects that radial distortion values are known in some points of the image. The polynomial approximation is not computed but it is possible to pass two vectors, first is the radial distance and the second is appropriate radial distortion. Another necessary parameter is the type of interpolation between inserted pairs of values which is used to estimate the polynomial approximation (in fact the order of inputs is: type of interpolation, radial distance, radial distortion). Three vectors that represent row and column coordinates of corrected pixels and appropriate pixel values are the output of this function. The output is in a vector form rather than matrix because it is farther used with the function "griddata" which requires vector inputs.

After running this function on a testing dataset, some problems appeared. First problem was that when a large image file was used as an input for our function, output vectors were too long. This caused a problem when performing the griddata function (Matlab run out of memory). This problem was partially fixed by adding two new input parameters to the function interface. These parameters allow for specifying the sampling interval along rows and columns. This spacing can significantly reduce the amount of output data but it also reduces the quality of a corrected picture. Other problem with the griddata function was that sometimes this function broke down because it can not initialize the first Qhull for the Delaunay triangulation. This can be fixed by adding some extra parameters to the griddata function. For more information about the griddata and Qhull [7].

Another problem that cost some extra work was the computing speed. Initially the core computing code was written with a use of for loops. This seems more natural when trying to write an

algorithm in some programming languages. However, Matlab is built especially for working with matrixes and matrix computations are much faster than for or while loops in the Matlab environment. Due to this fact, we transformed for loop operations to matrixes operations. In Matlab this process is known as loops vectorizing [4]. This transform significantly speeds up computing process. The newly implemented function with matrixes operations instead of loops runs approximately 140 times faster than the old one.

Last thing we had to solve was related to the final corrected output image. First idea was that the pixel size of the output corrected image must be the same as the pixel size of an input image. This seems reasonable when we want to used the corrected image for measurement. Problems occur when extreme values of radial distortion are passed as an input. Passing huge values of radial distortion is useful for education purpose because it well illustrates the effect of radial distortion on the image geometry. But when the values are too big, a matrix designed for an output image exceeds the maximal size for a variable allowed in Matlab. This problem was solved by adding a switcher into GUI. Using this switcher (two radio buttons in group) enables a user to choose between saving the pixel size, or saving the number of pixels in the final image. The first option is useful when the final picture is used for the distance measurement (the pixel size is the same in the input and output images). The second option is necessary when radial distortion is of huge values (the number of pixels in the input and output images is same but pixel size is changed).

4. Building GUI

The final step in this work was creating the graphic user interface which allows for an easy to use environment for performing the radial distortion correction without knowledge of Matlab programming. For GUI development Matlab GUI builder was used. One main and bunch of independent GUIs were developed to afford necessary functionality. Data important for the farther use were exchanged among these GUIs in the predefined manner. All common data shared by GUIs where saved in UserData property of main GUI as a structured variable [5]. Each GUI can access these shared data easily and each GUI must update UserData property of main GUI every time when shared data was changed. This guarantees data consistence and using of actual data in every operation. Central storage of shared variables also improves orientation in application code.

5. Conclusion

The presented function for the correction of images from the radial lens distortion has several applications. In the Matlab environment, it can be used a pre-processing step for image registration. In the same manner the images can be corrected before georeferencing in CAD or GIS software packages that do not provide any function for correcting lens distortion that is a standard part of professional photogrammetric systems. Another useful application we see in education for a demonstration of geometric distortions caused by aberrations of objective lens of the non-metric cameras.

Acknowledgement

The presented work was supported by the grant MSM0021620831 of the Czech Ministry of the Education, Youth and Sports.

References

[1] E. M. Mikhail et al., *Introduction to modern photogrammetry*. Willey, New York, 2001. ISBN 0-471-30924-9

[2] K. B. Atkinson, *Close range photogrammetry and machine vision*. Whittles Publishing, 1996.

[3] Camera Calibration Toolbox for Matlab [http://www.vision.caltech.edu/bouguetj/calib_doc/htmls/parameters.html]

[4] R.Gonzales, R.E. Woods, S.L. Eddins, *Digital image processing using MATLAB*. Prentise Hall, Upper Saddle River 2004. ISBN 0-13-008519-7.

[5] P. Marchand, T.O. Holland, *Graphica and GUIs with MATLAB, third edition.* Chapman & Hall/CRC, New York 2003. ISBN 1-58488-320-0.

[6] J. Perš, S. Kovačič. Nonparametric Model-Based Radial Lens Distortion Correction Using Tilted Camera Assumption [http://vision.fe.uni-lj.si/docs/janezp/pers-wwk2002.pdf]

[7] Qhull [http://www.qhull.org/]

Appendix

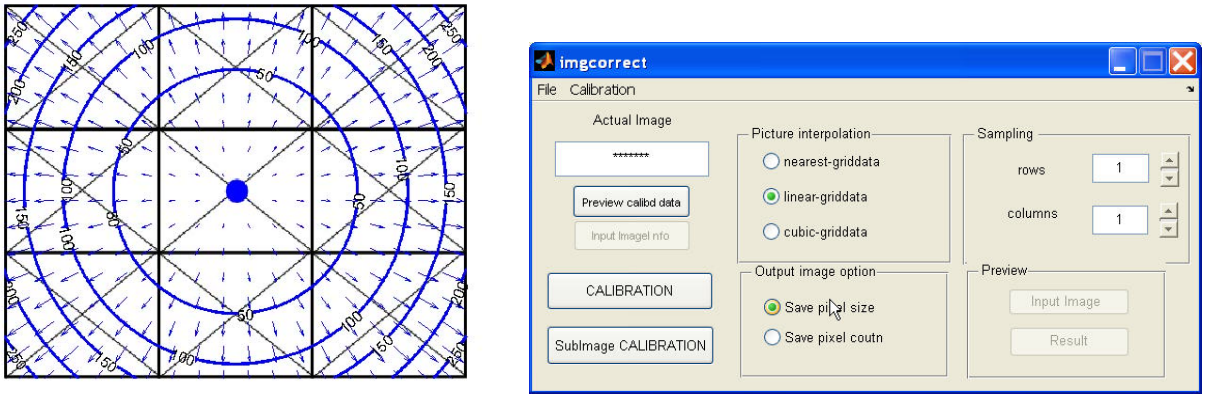

Figure 2. a) shows contours of the same radial distortion value and also the magnitude and the direction of radial distortion over the image (arrows), b) shows main application GUI

Part of radialshift function (using matrixes operation)

```
%A is input image
A=double(A);
S=size(A);
sinuhel=zeros(S(1),S(2));
cosuhel=zeros(S(1),S(2));
%coordiante of image center
stred(1)=S(1)/2;
stred(2)=S(2)/2;
disp('COMPUTING RADIAL SHIFT');
try
%creating vectors for mashgrid
r=0.5:sr:S(1);
c=0.5:ss:S(2);
[C,ROW]=meshgrid(c,r);
R=sqrt((C-stred(2)).^2+(ROW-stred(1)).^2);
sinuhel=(ROW-stred(1))./R;
cosuhel=(C-stred(2))./R;
switch intmetod
            case 'spline'
                  posun=interp1(varargin{1},varargin{2},R,'spline','extrap');
            case 'nearest'
                  posun=interp1(varargin{1},varargin{2},R,'nearest','extrap');
            case 'linear'
                  posun=interp1(varargin{1},varargin{2},R,'linear','extrap');
            case 'cubic'
                  posun=interp1(varargin{1},varargin{2},R,'cubic','extrap');
            case 'custom'
                   posun=eval(varargin{1});
            otherwise
                disp('error');
        end
        y=ROW-sinuhel.*posun;
        x=C-cosuhel.*posun;
        z=A;
catch
    error('error during computing radial shift');
end
```

Part of radialshift function before loops vectorizing (using for loops)

```
A=double(A);
S=size(A);
stred(1)=S(1)/2;
stred(2)=S(2)/2;
disp('COMPUTING RADIAL SHIFT');
try
for j=1:sr:S(1)
    for k=1:ss:S(2)
        rp=j-0.5;
        sp=k-0.5;
         R=sqrt((rp-stred(1)).^2+(sp-stred(2)).^2);
         sinuhel=(rp-stred(1))/R;
        cosuhel=(sp-stred(2))/R ;
        switch intmetod
            case 'spline'
                  posun=interp1(rx,ry,R,'spline','extrap');
            case 'nearest'
                     posun=interp1(rx,ry,R,'nearest','extrap');
            case 'linear'
                  posun=interp1(rx,ry,R,'linear','extrap');
            case 'cubic'
                   posun=interp1(rx,ry,R,'cubic','extrap')
               case 'custom'
                    posun=eval(varargin{1});
            otherwise
                disp('error');
        end
```

Jan Jedlička
Charles University in Prague
Faculty of Science
Department of Applied Geoinformatics and Science
e-mail: jan_jedlicka@centrum.cz

Marketa Potuckova
Charles University in Prague
Faculty of Science
Department of Applied Geoinformatics and Science
e-mail: mpot@natur.cuni.cz
www.natur.cuni.cz/gis