Applying and removing lens distortion in post production

Gergely Vass† and Tamás Perlaki

Colorfront Ltd., Budapest

Abstract

Lens distortion is often an issue in post production houses when combining footage taken with different cameras or integrating computer graphics into live action plates. While this error of the imaging process has been studied thoroughly for many years in the field of computer vision, almost none of the existing tools have all the key features users need. Some algorithms work well for any kind of distortion but are hard to calibrate, while others are fully automatic but fail for fisheye or anamorphic lenses. In this paper the different approaches of removing lens distortion are summarized, and a semi-automatic system is introduced that fits the need of post production facilities.

1. Introduction

The image generation of any kind of camera – like film or CCD cameras – may be modelled with the pinhole camera model[7]. However, the images of real cameras suffer from more or less lens distortion, which is a nonlinear and generally radial distortion. The most prevalent form of this effect is the *barrel* and the *pincushion* distortion[11]. The first is due to the fact that many wide angle lenses have higher magnification in the image center than at the periphery. This causes the image edges to shrink around the center and form a shape of a barrel. The pincushion distortion is the inverse effect, when the edges are magnified stronger (see figure (1)).

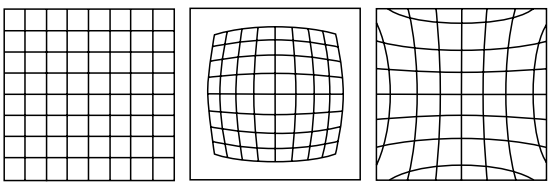

Figure 1: *The original grid and the effect of barrel (center) and pincushion distortion (right).*

† Graduate student at the Budapest University of Technology and Economics

There are different situations using image processing and computer vision algorithms or creating special effects, where the lens distortion is not acceptable. The most crucial one is match moving which is also referred as camera matching. In this process the internal – most importantly focus distance – and external parameters – position and orientation – of the original camera are extracted from the image sequence based on matching points or lines over multiple images[7]. Camera matching is essential when integrating 3D computer graphics elements into real footage, since the virtual camera has to move exactly the same way as the original one.

In the next section the previous work concerning the removal of lens distortion is summarized. In section 3. the typical pipeline of post production studios utilizing this kind of tools is introduced, and their key expectations are collected. As the body of this work a lens distortion removing and applying tool is introduced, that fits these needs. In section 4. the mathematical model of distortion is set up, which is powerful enough and yet simple. The necessary coordinate transformations are described in section 5. introducing a simple, but very important aspect of images with extra border. The issues of application of distortion and our automatic calibration technique are also described.

2. Previous work

The different kind of lens distortions and lens aberrations have been researched for decades. An extensive review of the lens distortion itself is given by Slama[11]. The algorithms correcting lens distortion can be categorized based on two different aspects: the mathematical model of the distortion used, and the way the parameters of the model are estimated.

2.1. Lens distortion models

While it is known that radial distortion is not the only lens aberration altering the geometry of the projected images[3], it is certainly the most – and often the only – significant one[13]. The simplest and most popular way of modelling radial distortion was proposed by Tsai[13] in 1987 (see equation (3)). Many algorithms – just like ours – use this or a slightly modified version[6]. This model works for almost all kind of regular lenses, but fails for wide angle, so called *fish-eye* lenses. For these kind of special lenses different models were proposed[2, 9, 5].

2.2. Calibration techniques

There are different approaches to find the parameters of the lens distortion model that fit the distortion of the actual camera the best. The *automatic calibration* algorithms do not require any user intervention, they find the parameters using an automated – iterative or "one step" – process. Some of these solutions require some knowledge of the 3D scene[11, 3], a reference grid[1] or a set of coplanar points with known coordinates up to a homography[8]. Since these information are not always present, more sophisticated techniques do not need any information of the 3D scene. The "plumb-line" method is one of this kind, since it requires only some straight lines visible on the images, which are curved if distortion occurs[5, 14]. Other calibration techniques do not even need straight lines to be on the image, like the one based on the higher-order correlations in the frequency domain[4]. There are automatic calibration methods that evaluate the lens distortion parameters and the camera parameters simultaneously using iterative[12] search or linear estimation[6]. These process require only the scene to be rigid and the camera to move.

3. Lens distortion in post production

There are different tasks in a post production studio where lens distortion is harmful, like compositing images from different sources or match moving. Match moving or camera matching is the process of calculating the camera parameters – like translation, orientation, focal distance – of the original camera based on only image sequences[7]. This process is essential when integrating 3D computer graphics into live action footage, since the virtual camera has to move exactly the same way the original camera.

3.1. Integrating CG and live action images

The integration of 3D computer graphics and the original images starts by "tracking" the camera, or match moving. If the lens distortion is not removed prior to tracking, the constraints used by the camera matching algorithm – which supposes a pin-hole camera model – will not hold, thus it will not generate a precise enough solution. After removing lens distortion and successful camera matching the computer generated elements may be rendered. Since the 3D rendering algorithms support only pin-hole camera models, the rendered images cannot be combined with the original – and distorted – footage. One may think that the best solution is to composite the CG elements with the undistorted version of the original images used for tracking. However, the undistortion process worsens the quality of the live action images. To overcome this problem lens distortion is *applied* to the CG elements that are later composited with the *original* footage. The advance of this approach is that the rendered images can be generated at any resolution, thus the quality after applying lens distortion remains excellent. Note, that the computer generated images will be rendered always at a higher resolution – or rather larger size – than the original images, since their borders are "pulled" towards the image center (in case of barrel distortion which is the most common type of lens distortion). This issue will be discussed later in detail.

3.2. Requirements for lens distortion correction

Considering the pipe-line of the integration of 3D computer graphics and live action images, and taking into account the general expectations of the post production houses, the requirements for the lens distortion managing solution are the following:

- Work with normal and relatively wide angle lenses. However, extremely wide angle fish-eye lenses are rare in case of camera matching at post production studios.
- Work with asymmetric, or so called anamorphic lenses. These lenses are used for several 35mm film formats and are not rare at all.
- Apply and remove distortion of the same parameters. This means that the inverse mapping of the removal (which is the application of distortion) should be also possible.
- Apply distortion to images of larger size as the original. This means that the application of distortion with specific parameters should be possible for images with extra border around them.
- It is recommended to have some automatic or semi-automatic calibration technique implemented.
- Work fast and reliable with extreme parameters. This means that the tools related to lens distortion will not likely to be used for *only* lens distortion issues, but also for example as an artistic effect.

4. Mathematical model of lens distortion

The distortion of the lenses to be removed is a radial kind of distortion. The simplest way to model this effect is with a shift to the pixel coordinates. The radial shift of coordinates modifies only the distance of every pixel from the image center. Let r denote the distance of the undistorted image coordinates from the center, while $\widehat{r}$ represents the observed distance. With these notations the function that can be used to *remove* lens distortion is:

$$\widehat{r} = f(r) \tag{1}$$

Any distortion $f()$ can be approximated with it's Taylor expansion:

$$\widehat{r} = r + \kappa_1 r^3 + \kappa_2 r^5 + \kappa_3 r^7 + \dots \tag{2}$$

where κ_i are the radial distortion coefficients. The "perfect" approximation would be a polynomial of infinite degree, however, this precision is not needed. Researches and measurements proved, that for average camera lenses the first order is enough and the more elaborated models would only cause numerical instability[13]:

$$\widehat{r} = r + \kappa_1 r^3 = r(1 + \kappa_1 r^2) \tag{3}$$

To model wider angle lenses two coefficients are needed:

$$\widehat{r} = r(1 + \kappa_1 r^2 + \kappa_2 r^4) \tag{4}$$

where κ_1 controls the general behavior of the distortion and κ_2 should be adjusted only if the distortion is so severe that the first order approximation does not give a good enough solution. Note that unfortunately this model is not sufficient for fish-eye lenses. The same equation written in terms of (x,y) components if $(0,0)$ is the image center:

$$\begin{pmatrix} \widehat{p_x} \\ \widehat{p_y} \end{pmatrix} = \begin{pmatrix} p_x(1 + \kappa_1 r + \kappa_2 r^2) \\ p_y(1 + \kappa_1 r + \kappa_2 r^2) \end{pmatrix} \tag{5}$$

If the lens is asymmetric two extra effect has to be compensated:

- The image is shrinken.
- The non-radial, asymmetric distortion might be significant.

To introduce the shrinking effect a *squeeze* term s is added to the formula

$$\begin{pmatrix} \widehat{p_x} \\ \widehat{p_y} \end{pmatrix} = \begin{pmatrix} p_x(1 + \kappa_1 r + \kappa_2 r^2) \\ p_y(1 + \frac{\kappa_1}{s} r + \frac{\kappa_2}{s} r^2) \end{pmatrix} \tag{6}$$

The non-radial distortion is modelled by two distortion functions corresponding to the *horizontal* and *vertical* directions. Just like the radial distortion these functions can be approximated by their Taylor expansion. However, it is not necessary to approximate them with two or more coefficients:

$$\begin{aligned} \widehat{p_x} &= f_x(p_x) = p_x(1 + \lambda_x p_x^2) \\ \widehat{p_y} &= f_y(p_y) = p_y(1 + \lambda_y p_y^2) \end{aligned} \tag{7}$$

The combination of these distortions gives the model used in our system, which is sufficient for almost all lenses film makers use for shots involving camera matching. This kind of mathematical model is used in the match moving product called 3D Equalizer[10], however the derivation of the equations introduced in this paper above is much simpler and elegant. The final result for removing distortion:

$$\begin{pmatrix} \widehat{p_x} \\ \widehat{p_y} \end{pmatrix} = \tag{8}$$

$$= \begin{pmatrix} p_x(1 + \kappa_1 p_x^2 + \kappa_1(1 + \lambda_x) p_y^2 + \kappa_2 (p_x^2 + p_y^2)^2) \\ p_y(1 + \frac{\kappa_1}{s} p_x^2 + \frac{\kappa_1}{s}(1 + \lambda_y) p_y^2 + \frac{\kappa_2}{s}(p_x^2 + p_y^2)^2) \end{pmatrix}$$

where

- κ_1 controls the primary distortion (default 0).
- κ_2 adjusts the secondary distortion especially at the borders (default 0).
- s controls the squeeze factor (default 1 means no squeeze).
- λ_x and λ_y control the asymmetric distortion, also called x and y curvature (default 0).

5. Normalization

The mathematical relations of the previous section suppose the origin to be the center of the distortion. However, it is not enough to translate the pixel coordinates such that this requirement is fulfilled. All pixels should be translated to a *dimensionless* frame, where the image resolution is not important. The model would be useless if images with the same distortion but different resolution would have different distortion parameters. In the dimensionless frame the diagonal radius of the image is always 1, and the lens center is $(0,0)$ (see figure (2)).

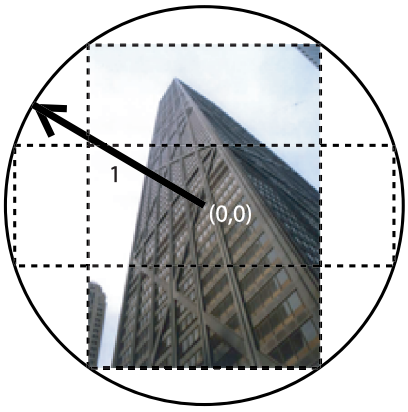

Figure 2: *On the figure the dimensionless version of two images of different aspect ratio are shown.*

The formula to transform the pixel coordinates to dimensionless coordinates is the following:

$$\begin{pmatrix} p_x \\ p_y \end{pmatrix} = \begin{pmatrix} (p'_x - c'_x)/\sqrt{(\frac{w_x}{2})^2 + (\frac{w_y}{2})^2} \\ (p'_y - c'_y)/\sqrt{(\frac{w_x}{2})^2 + (\frac{w_y}{2})^2} \end{pmatrix} \quad (9)$$

where (p_x, p_y) are the dimensionless and (p'_x, p'_y) the pixel coordinates, (c'_x, c'_y) is the lens center in pixel coordinates and w_x, w_y are the image width and height in pixels. The relation of the inverse transformation – which is not shown here – is easy to produce and is also a simple formula. Note that (c'_x, c'_y) is not necessarily the image center[15] $(\frac{w_x}{2}, \frac{w_y}{2})$.

As mentioned in section 3.2. it is sometimes necessary to apply distortion to a images with extra border. This means that instead of just applying distortion to the image, we generate a version with larger field of view, apply distortion a special way to this large image, and then crop the inner part to the original size. Using this approach there will be obviously no gaps at the borders. An example is shown in figure (3):
Let us suppose digital background – with a bottle of oil – has to be inserted into video images. The lens distortion parameters are acquired from the original, video-sized source. The computer generated picture with barrel distortion applied to it – if rendered at regular size – cannot be put in the background because of the visible black area around (top row). If larger images are rendered and the application of distortion is handled correctly – *not* the regular way – the cropped result will fit perfectly into the background (bottom row).

Let r denote the ratio of the normal image size (w_x, w_y) and the size of the large image with extra border (l_x, l_y):

$$r = \frac{l_x}{w_x} = \frac{l_y}{w_y} \quad (10)$$

To be able to distort images with extra border correctly, the transformation from pixel coordinates to dimensionless coordinates should be slightly altered: the inner part – of original size – has to have coordinates such that it's diagonal radius is 1. To achieve this the only change we have to make is to apply transformation (9) to *large* images substituting the dimensions of the *original*, *small* images. In our implementation the user defines only the ratio r, for example $r = 1.5$ if the image size is 150% of original. If $r = 1$ the image is distorted the regular way, if $r > 1$ the image dimensions substituted into (9) for the transformation will be $(w_x, w_y) = (\frac{l_x}{r}, \frac{l_y}{r})$. The result of this transformation is exactly what we want as shown in figure (4)

6. Applying distortion

The removal of distortion is realized by the transformation introduced in section 4. To apply distortion we need the inverse transformation:

$$\begin{pmatrix} p_x \\ p_y \end{pmatrix} = f^{-1} \begin{pmatrix} \widehat{p_x} \\ \widehat{p_y} \end{pmatrix} \quad (11)$$

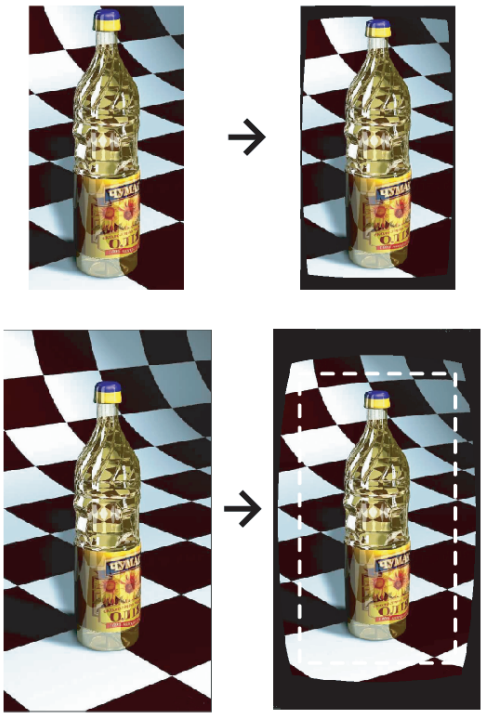

Figure 3: *Computer generated elements should be rendered at larger size to avoid black gaps. If the large image is distorted using the proposed way, the cropped portion – and not the whole image – will match the desired distortion while avoiding gaps.*

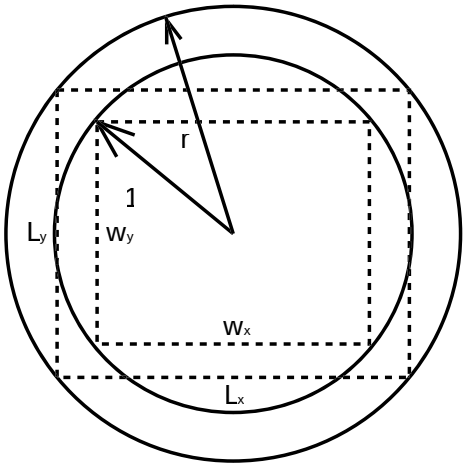

Figure 4: *The diagonal radius of the whole image will be r instead of* 1. *The inner part will be distorted correctly*

However, there is no closed form solution for the inverse of (8), thus numerical calculations have to be used. We used the Newton method, which is based on the following iteration[10]:

$$\begin{pmatrix} p_x \\ p_y \end{pmatrix}^{(j+1)} = \tag{12}$$

$$= \begin{pmatrix} p_x \\ p_y \end{pmatrix}^{(j)} - \left(\frac{\partial g((p_x,p_y)^{(j)})}{\partial ((p_x,p_y)^{(j)})} \right)^{-1} \left(f \begin{pmatrix} p_x \\ p_x \end{pmatrix}^{(j)} - \begin{pmatrix} \widehat{p_x} \\ \widehat{p_y} \end{pmatrix} \right)$$

where

$$\left(\frac{\partial f((p_x,p_y)^{(j)})}{\partial ((p_x,p_y)^{(j)})} \right) = \begin{pmatrix} \frac{\partial}{\partial p_x} f_x(p_x,p_y) & \frac{\partial}{\partial p_y} f_x(p_x,p_y) \\ \frac{\partial}{\partial p_x} f_y(p_x,p_y) & \frac{\partial}{\partial p_y} f_y(p_x,p_y) \end{pmatrix}$$

is the first derivative (Jacobian) matrix. The values of the entries of this matrix can be evaluated using some software capable of symbolic manipulation like MATHEMATICA or MAPLE, or refer to the Science-D-Visions paper[10]. The convergence of such iterative methods depend on basically two things: the initial estimation $(p_x,p_y)^{(0)}$ and the number of iteration steps.

If the lens distortion attributes are set to values corresponding to real lenses, this algorithm converges in 5-10 steps with the initial estimation introduced for 3DE[10]. However, we suppose that some users will use the tool to create artistic effects, thus the convergence should be guarantied at more extreme parameters as well. The most problematic situations are when the mapping $f()$ is even not one-to-one (an example is shown on figure (5)).

We have found that using the initial estimation of $(0,0)$ – which is the lens center – the iteration always converges. We make use of the fact, that $f()$ is monotonic in the central region, where one of the solution must be found (on figure (5) this region is the inner part of the dashed circle). Using $(0,0)$ as the initial estimate the original image can be reconstructed from the bottom (distorted) image of figure (5) using Newton iteration, see figure (6). Note that in the outer black area there is no solution (eg. the original pixels are not visible on the source image), thus the Newton iteration stops at an arbitrary position (if this position is on the image the pixel has some color, otherwise it is black).

7. Automatic calibration

There are numerous algorithms that are capable of calibrating the lens distortion parameters without any reference object (see section (2.2)). We have chosen a process based on straightening curved lines that are supposed to be straight. The advances of this approach are:

- It works with still images as well.
- The implementation is rather simple.
- If calibration grid is available the accuracy of the method is superior.

Figure 5: *Example: with parameters $\kappa_1 = -0.2$ and $\kappa_2 = -0.5$ some pixels are moved to multiple positions. Watch Nóra's face being duplicated on the right side.*

Figure 6: *Using Newton iteration with initial estimation of $(0,0)$ the application of distortion reconstructs the top image from the bottom of figure (5) successfully.*

The calibration tries to evaluate the two main parameters of the distortion, κ_1 and κ_2, based on user defined lines. The lens center would be the third main parameter, however, our tests have shown that the user defined lines are not precise enough to handle the numerical instability introduced by the new variable. For anamorphic lenses manual calibration is needed, but in fact the productions using such equipment make always calibration images anyway. The algorithm for calibration is based on an extremely simple search-loop finding one parameter (k) at a time. (k) is either κ_1 or κ_2:

```
while(step>minstep)
{
 calculate_error_at(k-step);
 calculate_error_at(k+step);
 calculate_error_at(k);
 if(k is smallest)
 {
  step:=step/4;
 }
 else
 {
  if((k-step) was better)
    k:=k-step;
  else
    k:=k-step;
 }   }
```

The loop starts with testing whether the `(k-step)`or `(k+step)`values are better. If non of they are, we know that the solution is “near”, thus the step size is set to finer. As stated before we suppose that there are no local minimums over the search region that would trap the search loop. The error – calculated in `calculate_error_at()`– is defined as follows:

$$\sum_{i=1}^{n} \sum_{k=2}^{m_i-1} d(p_{ik}, l_i) \tag{13}$$

where n is the number of lines, m_i is the number of points defining the i^{th} line, p_{ik} is the k^{th} point of the i^{th} line and $d(p_{ik}, l_i)$ is the distance between point p_{ik} and the line connecting p_{i1} and p_{im_i}.

The search loop is run several times – for the first run to find κ_1, the second to evaluate κ_2, the third refines κ_1 ect. – until the solution is stable. Our tests have proven that the majority of the lens distortions can be handled with the simple, one variable model (when $\kappa_2 = 0$), since the second search loop of κ_2 could not improve the solution. The automatic calibration process is illustrated on figure (7), where the distortion is successfully removed.

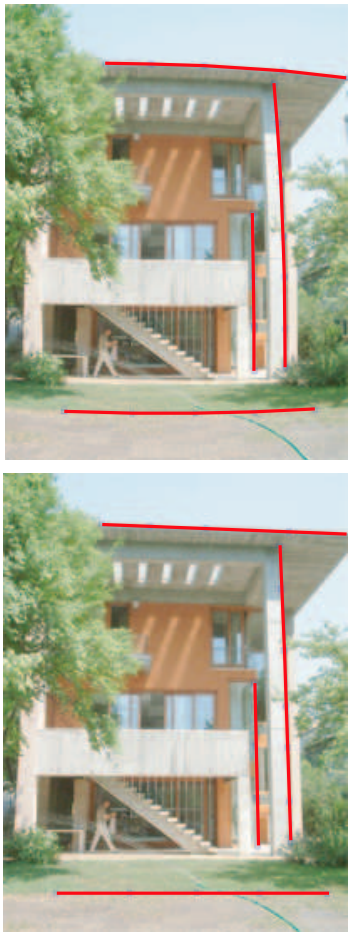

Figure 7: κ_1 *and* κ_2 *are calibrated based on the user defined curved lines. The lines on the corrected result remain straight.*

8. Applying distortion fast

The implementation of the lens distortion removing and applying methods based on equations (8), (9) and (12) are very straightforward. For every pixel – using the mathematical model – we calculate it’s position on the source image at sub-pixel accuracy. Based on this position the pixel’s color may be sampled from the input image. In our implementation bicubic interpolation was used. However, to recalculate these vectors for every frame of a long sequence is waste of time. It is strongly recommended to use some buffer to store these vectors, thus the distortion of the images is no more then re-sampling.

If the lens distortion tool is used for managing lens distortion, only the processing speed of long image sequences matters. However, if the tool is used to make artistic effects – which is likely to happen in real situations – the feedback-time of the interactive changes in the parameters should be also fast. This is not a problem if distortion is removed, since in that case only a simple, closed-form formula has to be evaluated for the pixels. Unfortunately the Newton it-

eration is much slower: complex calculations have to be iterated many times for each pixel, which produces long (1-2 sec) rendering time. To have fast feedback for this case as well, we implemented a different method for applying distortion fast. The relation (8) to remove distortion has the form $(\widehat{p}_x, \widehat{p}_y) = f(p_x, p_y)$, where (p_x, p_y) are undistorted and $(\widehat{p}_x, \widehat{p}_y)$ are the distorted pixel coordinates. If we want to produce the distortion free image from captured images, we should calculate *for each output pixel* a sample position on the source picture to build up the whole image (see figure (8)) using the formula. Our method does not try to evaluate the inverse mapping of equation (8), but use it to *displace* the pixels of the source images and produce the inverse effect this way. If we calculate for each *undistorted input pixel* it's distorted position, it can be displaced correctly to produce a distorted output (see figure (9)).

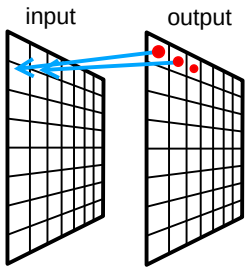

Figure 8: *Using the standard approach, for* ***each output*** *pixel the RGB values are sampled from the source.*

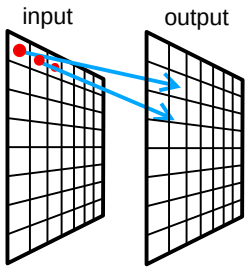

Figure 9: *An alternative way to generate the inverse mapping is to displace all the* ***source image pixels****.*

Of course this approach has several major problems (which may be the cause why it is not used in other systems):

- Some input pixels will be shifted to the same output pixel.
- There will be "empty" pixels on the result image, where no input pixels are placed.
- The process is based on shifting pixels which may cause aliasing.

Since this way of applying distortion is only for previewing in our system, quality is of minor importance. This is why the first and the last problems are ignored, they produce only hardly visible deficiencies. There are basically two cases of problem number two. If barrel distortion is applied the image is shrinken, thus the border pixels will remain empty. In this case the filling of the whole output image with black pixels prior to the process solves the problem. If pincushion distortion is applied, the image is scaled up thus empty gaps will occur as seen on figure (10). This very annoying error is solved in our solution in a very simple way: black pixels are filled with the color of their closest neighbor. Using the proposed method to apply distortion the rendering time of video footage decreases to 2/3 frames per second on a 1.5 GHz PC.

Figure 10: *In case of pincushion distortion there will be pixels on the final image that remain empty (left image). Our algorithm fills these pixels with the nearest neighbor (right image).*

Figure 11: *The left image is distorted using our fast method, the right is produced with Newton iteration. The quality of the left image is somewhat worse, but the processing time is reduced by 50%.*

9. Conclusions and future work

While there are many algorithms developed to remove and apply distortion none of them fulfills all the requirements of post production studios. In this paper we introduced our semi-automatic tool that is capable to handle almost all of the possible situations. We have also proposed a simple way to apply distortion fast. The very common problem of distorting computer generated images to match live action footage – without black areas at the border – has been discussed in detail as well. Our tests have proven that the algorithms work well in production. There are two aspects of future development: supporting more kinds of lens distortion and to have other – possibly automatic – calibration techniques as well. For these improvements the results in the literature discussed in section (2.1) and (2.2) may be used.

Acknowledgements

We are grateful to Prof. Szirmay-Kalos for bringing this conference together and to Márk Jászberényi (CEO of Colorfront) for supporting us.

References

1. D. G. Bailey. A New Approach to Lens Distortion Correction. *Proceedings of Image and Vision Computing, New Zealand*, 2003.
2. A. Basu and S. Licardie. Alternative models for fish-eye lenses. *Pattern Recognition Letters*, **16**: 433–441, 1995.
3. S. S. Beauchemin and R. Bajcsy. Modelling and Removing Radial and Tangential Distortions in Spherical Lenses *Theoretical Foundations of Computer Vision 2000*, pp. 1–21.
4. H. Farid and A. C. Popescu. Blind Removal of Lens Distortion *Journal of the Optical Society of America*, **18**(9) pp.: 2072-2078, 2001.
5. F. Devernay and O. D. Faugeras. Straight lines have to be straight, *Machine Vision and Applications*, **13**(1): 14–24, 2001.
6. A. W. Fitzgibbon. Simultaneous linear estimation of multiple view geometry and lens distortion. *CVPR*, 2001.
7. R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision* Cambridge, 2000.
8. T. Pajdla, T. Werner and V. Hlaváč. Correcting Radial Lens Distortion without Knowledge of 3-D Structure *Technical report TR97-138*, FEL ČVUT, Karlovo náměstí 13, Prague, Czech Republic, 1997.
9. J. Perš and S. Kovačič. Nonparametric, Model-Based Radial Lens Distortion Correction Using Tilted Camera Assumption *Proceedings of the Computer Vision Winter Workshop 2002, Bad Aussee, Austria*, pp 286–295, 2002.
10. U. Sassenberg. "Lens distortion model of 3DE V3", *On-line technical documentation*, http:// www. 3de-qualizer. com/ sdv_tech_art/ paper/ distortion.html (2001).
11. C. Slama. *Manual of Photogrammetry.* American Society of Photogrammetry, Falls Church, VA, USA, 4th edition, 1980.
12. G. P. Stein. Lens distortion calibration using point correspondences. *Proceedings of CVPR*, 1997.
13. R. Y. Tsai. A versatile camera calibration technique for high-accuracy 3D machine vision metrology usin off-the-shelf TV cameras and lenses. *IEEE Journal of Robotics and Automation*, **3**(4): 323–344, 1987.
14. R. J. Valkenburg and P. L. Evans. Lens Distortion Calibration by Straightening Lines. *Proceedings of Image and Vision Computing, New Zeland*, 2002.
15. R. G. Willson and S. A. Shafer. What is the Center of the Image *Proc. IEEE Conf. on Computer Vision and Pattern Recognition*, 1994, pp. 2946–2955.