Lens distortion model

Definitions

In the following we will describe the mathematical construction used for modelling the non-linear lens distortion as used in 3DE.

We will need a function for applying lens distortion to an image as well as its inverse, a function for removing lens distortion from an image. Since our model function is a non-linear mapping from $\mathbf{R}^2$ to $\mathbf{R}^2$ it will be difficult, to find an analytic expression for the inverse function. In practice, we use an analytic expression for *removing* the lens distortion from images, while the inverse function that *applies* lens distortion is formulated as an iterative, numerical procedure.

We will use the following symbols. Distorted points are denoted by $(x_{\ldots}, y_{\ldots})$, undistorted points are denoted by $(\hat{x}_{\ldots}, \hat{y} \ldots)$. The filmback width and height are called w_{fb} and h_{fb}, respectively. The focal length is f. If we take into account that the lens center can be displaced, we get a camera model as in the figure at the end of this appendix, where $(\Delta x_{\mathrm{fb}}, \Delta y_{\mathrm{fb}})$ is the lens center offset. In case of an anamorphic camera, all parameters are understood as **unsqueezed** quantities!

The camera pyramid can now be described by the parameters

$$(w_{\mathrm{fb}}, h_{\mathrm{fb}}, f, \Delta x_{\mathrm{fb}}, \Delta y_{\mathrm{fb}}). \tag{1}$$

The filmback is represented by points $(x_{\mathrm{fb}}, y_{\mathrm{fb}})$ fulfilling

$$\begin{aligned} x_{\mathrm{fb}} &\in \left[-\frac{w_{\mathrm{fb}}}{2} - \Delta x_{\mathrm{fb}}, +\frac{w_{\mathrm{fb}}}{2} - \Delta x_{\mathrm{fb}}\right] \\ y_{\mathrm{fb}} &\in \left[-\frac{h_{\mathrm{fb}}}{2} - \Delta y_{\mathrm{fb}}, +\frac{h_{\mathrm{fb}}}{2} - \Delta y_{\mathrm{fb}}\right]. \end{aligned} \tag{2}$$

In these coordinates the lens center is $(0,0)$ whereas the center of the filmback is $(-\Delta x_{\mathrm{fb}}, -\Delta y_{\mathrm{fb}})$. The camera pyramid can be replaced by a *dimensionless* camera in which the diagonal radius of the filmback is 1.0. This camera is given by the dimensionless filmback measures

$$w_{\mathrm{dn}} = \frac{w_{\mathrm{fb}}}{R} \qquad h_{\mathrm{dn}} = \frac{h_{\mathrm{fb}}}{R} \tag{3}$$

and the dimensionless lens center offset

$$\Delta x_{\mathrm{dn}} = \frac{\Delta x_{\mathrm{fb}}}{R} \qquad \Delta y_{\mathrm{dn}} = \frac{\Delta y_{\mathrm{fb}}}{R}, \tag{4}$$

where the diagonal radius of the dimensionless filmback is given by

$$R = \sqrt{\left(\frac{w_{\mathrm{fb}}}{2}\right)^2 + \left(\frac{h_{\mathrm{fb}}}{2}\right)^2}. \tag{5}$$

These diagonally normalized, dimensionless coordinates will be denoted by $(x_{\mathrm{dn}}, y_{\mathrm{dn}})$ in the following. The dimensionless filmback is spanned by

$$\begin{aligned} x_{\mathrm{dn}} &\in \left[-\frac{w_{\mathrm{dn}}}{2} - \Delta x_{\mathrm{dn}}, +\frac{w_{\mathrm{dn}}}{2} - \Delta x_{\mathrm{dn}}\right] \\ y_{\mathrm{dn}} &\in \left[-\frac{h_{\mathrm{dn}}}{2} - \Delta y_{\mathrm{dn}}, +\frac{h_{\mathrm{dn}}}{2} - \Delta y_{\mathrm{dn}}\right]. \end{aligned} \tag{6}$$

Again, the lens center is $(0, 0)$. It is also usefull to define what we call field-of-view coordinates. In these coordinates an image ranges from -1 (left) to +1 (right) horizontally and from -1 (bottom) to +1 (top) vertically:

$$\begin{aligned} x_{\mathrm{fov}} &\in [-1, +1] \\ y_{\mathrm{fov}} &\in [-1, +1]. \end{aligned} \tag{7}$$

The center of the image is (0,0), which is not necessarily the lens center (i.e. view direction). We will express the final results in terms of these coordinates. It will be easy for the reader to add some frontend and backend mapping in order to handle pixel coordinates instead of field-of-view coordinates (called fov-coordinates from now on).

The fov-coordinates are quite intuitive, but they are not appropriate for formulating the lens distortion model. Since we use the dimensionless coordinates in the lens distortion model, we have to establish the connection between fov- and dimensionless coordinates:

$$x_{\mathrm{dn}} = \frac{x_{\mathrm{fov}} w_{\mathrm{dn}}}{2} - \Delta x_{\mathrm{dn}} \qquad y_{\mathrm{dn}} = \frac{y_{\mathrm{fov}} h_{\mathrm{dn}}}{2} - \Delta y_{\mathrm{dn}}. \tag{8}$$

The transformation function between these coordinates may be called h:

$$\begin{pmatrix} x_{\mathrm{dn}} \\ y_{\mathrm{dn}} \end{pmatrix} = h(x_{\mathrm{fov}}, y_{\mathrm{fov}}). \tag{9}$$

We will use this function later in order to formulate the distortion function in fov-coordinates.

The distortion model of 3DE V2

Let us review the earlier model used in 3DE V2. The aim was to formaulate a simple non-linear, radially symmetric distortion function. The center of this radial symmetry was fixed to $(0, 0)$, since it was not possible to specify the lens center offset. Radial symmetry of the distortion function makes sense under the assumption that the lens system of the camera to be modelled exhibits radial symmetry.

The diagonally normalized coordinates are appropriate for modelling lens distortion. Since our model is supposed to be radially symmetric, we can introduce polar coordinates

(r, ϕ). The dimensionless coordinate of an distorted image expressed in polar coordinates reads $(x_{\mathrm{dn}}, y_{\mathrm{dn}}) = (r\cos\phi, r\sin\phi)$. The dimensionless coordinate of the undistorted image is now modelled by a function which only depends on the radius r, but not on the angle ϕ.

$$\begin{aligned} g_r &: \mathbf{R} \to \mathbf{R} : \\ g_r(r) &= r(1 + \delta r^2) \end{aligned} \tag{10}$$

which transforms $(x_{\mathrm{dn}}, y_{\mathrm{dn}}) = (r\cos\phi, r\sin\phi)$ into $(\hat{x}_{\mathrm{dn}}, \hat{y}_{\mathrm{dn}}) = (g_r(r)\cos\phi, g_r(r)\sin\phi)$, and which of course preserves the radial symmetry. Note, that the dimensionless coordinates are constructed in a way, that the four corners of the filmback lie on the circle with $r = 1$. The parameter δ was simply called lens distortion in 3DE V2. Although it is extremely simple it worked well for radially symmetric lens systems. However, it failed for anamorphic lenses.

3DE V3: Removing lens distortion

Let us now have a look at the distortion model of 3DE V3. This model is appropriate for describing anamorphic cameras, and as a consequence we need four parameters instead of only one as in 3DE V2. Generally speaking, we consider a function g mapping from $\mathbf{R}^2$ to $\mathbf{R}^2$ of the following form:

$$\begin{gathered} g : \mathbf{R}^2 \to \mathbf{R}^2 : \\ \begin{pmatrix} x_{\mathrm{dn}} \\ y_{\mathrm{dn}} \end{pmatrix} \mapsto \begin{pmatrix} \hat{x}_{\mathrm{dn}} \\ \hat{y}_{\mathrm{dn}} \end{pmatrix} = \begin{pmatrix} x_{\mathrm{dn}}(1 + c_{xx}x_{\mathrm{dn}}^2 + c_{xy}y_{\mathrm{dn}}^2) \\ y_{\mathrm{dn}}(1 + c_{yx}x_{\mathrm{dn}}^2 + c_{yy}y_{\mathrm{dn}}^2) \end{pmatrix}. \end{gathered} \tag{11}$$

This function maps point $(0, 0)$ onto $(0, 0)$, i.e. the lens center is not affected by the distortion. The geometic interpretation of the distortion coefficients $c_{xx}, c_{xy}, c_{yx}, c_{yy}$ is not very intuitive, therefore we use a different representation. We introduce four parameters

$$\begin{aligned} \text{distortion } \delta &\quad \text{default value} : 0.0 \\ \text{anamorphic squeeze } \epsilon &\quad \text{default value} : 1.0 \\ \text{x} - \text{curvature } \eta_x &\quad \text{default value} : 0.0 \\ \text{y} - \text{curvature } \eta_y &\quad \text{default value} : 0.0 \end{aligned} \tag{12}$$

The distortion coefficients relate to these parameters as follows:

$$\begin{aligned} c_{xx} &= \frac{\delta}{\epsilon} \\ c_{xy} &= \frac{\delta + \eta_x}{\epsilon} \end{aligned}$$

$$\begin{aligned} c_{yx} &= \delta + \eta_y \\ c_{yy} &= \delta \end{aligned} \tag{13}$$

These are the parameters which appear in 3DE V3's Camera Adjustment Window and in the Zoom Window in the Distortion Grid Edit mode. For the calculations in this appendix the parameters $c_{\ldots}$ are appropriate. When δ, η_x and η_y are set to their default values, g is the identity mapping and ϵ does not have any influence. The default value of 1.0 for ϵ corresponds to a non-anamorphic camera. When ϵ, η_x and η_y are set to the default values, the parameter δ has the same effect as the parameter δ in 3DE V2, as the reader will easily find out. The function that removes distortion from a fov coordinate point is now simply

$$\begin{pmatrix} \hat{x}_{\text{fov}} \\ \hat{y}_{\text{fov}} \end{pmatrix} = h^{-1} \circ g \circ h(x_{\text{fov}}, y_{\text{fov}}). \tag{14}$$

3DE V3: Applying lens distortion

Essentially, expressions (11) and (13) define the lens distortion model of 3DE V3. As mentioned, this function is used for removing lens distortion from images. In order to apply lens distortion to an image one needs to invert the function g. We do not have an analytic expression for g^{-1}. However, the numeric inversion is quite simple. Let us use e.g. Newton's method.

Assume we are given an undistorted point $(\hat{x}_{\text{dn}}, \hat{y}_{\text{dn}})$. We want to calculate $(x_{\text{dn}}, y_{\text{dn}})$, i.e. we have to solve the nonlinear system of two equations in two variables (dn coordinates):

$$\begin{pmatrix} \hat{x}_{\text{dn}} \\ \hat{y}_{\text{dn}} \end{pmatrix} = g(x_{\text{dn}}, y_{\text{dn}}). \tag{15}$$

The corresponding iterative algorithm reads

$$\begin{pmatrix} x_{\text{dn}} \\ y_{\text{dn}} \end{pmatrix}^{(j+1)} = \begin{pmatrix} x_{\text{dn}} \\ y_{\text{dn}} \end{pmatrix}^{(j)} - \left(\frac{\partial g((x_{\text{dn}}, y_{\text{dn}})^{(j)})}{\partial ((x_{\text{dn}}, y_{\text{dn}})^{(j)})} \right)^{-1} \left(g((x_{\text{dn}}, y_{\text{dn}})^{(j)}) - \begin{pmatrix} \hat{x}_{\text{dn}} \\ \hat{y}_{\text{dn}} \end{pmatrix} \right) \tag{16}$$

where the first derivative matrix is

$$\frac{\partial g(x_{\text{dn}}, y_{\text{dn}})}{\partial (x_{\text{dn}}, y_{\text{dn}})} = \begin{pmatrix} 1 + 3c_{xx}x_{\text{dn}}^2 + c_{xy}y_{\text{dn}}^2 & 2c_{xy}x_{\text{dn}}y_{\text{dn}} \\ 2c_{yx}x_{\text{dn}}y_{\text{dn}} & 1 + 3c_{yy}y_{\text{dn}}^2 + c_{yx}x_{\text{dn}}^2 \end{pmatrix}. \tag{17}$$

In order to run the algorithm we need initial values $(x_{\text{dn}}, y_{\text{dn}})^{(0)}$. We can obtain a good starting point by iterating (15) once.

$$\begin{pmatrix} \hat{x}_{\text{dn}} \\ \hat{y}_{\text{dn}} \end{pmatrix} = \begin{pmatrix} x_{\text{dn}} + c_{xx}x_{\text{dn}}^3 + c_{xy}x_{\text{dn}}y_{\text{dn}}^2 \\ y_{\text{dn}} + c_{yx}y_{\text{dn}}x_{\text{dn}}^2 + c_{yy}y_{\text{dn}}^3 \end{pmatrix}$$

$$= \begin{pmatrix} x_{\text{dn}} + c_{xx}\hat{x}_{\text{dn}}^3 + c_{xy}\hat{x}_{\text{dn}}\hat{y}_{\text{dn}}^2 \\ y_{\text{dn}} + c_{yx}\hat{y}_{\text{dn}}\hat{x}_{\text{dn}}^2 + c_{yy}\hat{y}_{\text{dn}}^3 \end{pmatrix} + O((\hat{x}_{\text{dn}} + \hat{y}_{\text{dn}})^5). \tag{18}$$

Neclecting the higher order terms, the result is

$$\begin{pmatrix} x_{\text{dn}} \\ y_{\text{dn}} \end{pmatrix}^{(0)} = \begin{pmatrix} \hat{x}_{\text{dn}}(1 - c_{xx}\hat{x}_{\text{dn}}^2 - c_{xy}\hat{y}_{\text{dn}}^2) \\ \hat{y}_{\text{dn}}(1 - c_{yy}\hat{x}_{\text{dn}}^2 - c_{yx}\hat{y}_{\text{dn}}^2) \end{pmatrix}. \tag{19}$$

In practice, it turns out that for realistic situations about 5 to 10 iterations are sufficient and that the Newton method converges when the initial values given in (19) are used. The Newton algorithm (16) and the initial values (19) define the inverse function g^{-1}. The function that applies distortion to a fov coordinate point is

$$\begin{pmatrix} x_{\text{fov}} \\ y_{\text{fov}} \end{pmatrix} = h^{-1} \circ g^{-1} \circ h(\hat{x}_{\text{fov}}, \hat{y}_{\text{fov}}). \tag{20}$$

3DE V3: Higher order extensions: the quartic term

In the following, we will investigate how the lens distortion model can be extended to handle more complicated types of non-linear distortion. Originally, the quartic term has been introduced to compensate the distortion of fisheye lenses, but it turned out, that the result for fisheyes is not satisfying. Nevertheless, also common lens distortions can often be handled better, when the quartic term is used. The strategy we follow here is to extend equation (11) by a fourth order term as follows:

$$\begin{aligned} \hat{x}_{\text{dn}} &= x_{\text{dn}}\big(1 + c_{xx}x_{\text{dn}}^2 + c_{xy}y_{\text{dn}}^2 \\ &+ c_{xxx}x_{\text{dn}}^4 + c_{xxy}x_{\text{dn}}^2y_{\text{dn}}^2 + c_{xyy}y_{\text{dn}}^4\big) =: g_x(x_{\text{dn}}, y_{\text{dn}}) \\ \hat{y}_{\text{dn}} &= y_{\text{dn}}\big(1 + c_{yx}x_{\text{dn}}^2 + c_{yy}y_{\text{dn}}^2 \\ &+ c_{yxx}x_{\text{dn}}^4 + c_{yyx}x_{\text{dn}}^2y_{\text{dn}}^2 + c_{yyy}y_{\text{dn}}^4\big) =: g_y(x_{\text{dn}}, y_{\text{dn}}). \end{aligned} \tag{21}$$

Only these terms make sense in the next higher order, so in total we get a model of ten parameters. In practice, however, this model would probably turn out to be too complicated to handle, and at some point it might be easier to describe the distortion function by a two-dimensional spline rather than by a polynome. Nevertheless, by restricting ourselves to radially symmetric fourth order terms we get a reasonable model.

3DE's Camera Adjustment Window contains a parameter called "Quartic Distortion" which we will denote by q in the following. In analogy to equations (13) we specify the relation between q and the new coefficients $c_{xxx}\ldots$:

$$c_{xxx} = \frac{q}{\epsilon}$$

$$\begin{aligned} c_{xxy} &= \frac{2q}{\epsilon} \\ c_{xyy} &= \frac{q}{\epsilon} \\ c_{yxx} &= q \\ c_{yyx} &= 2q \\ c_{yyy} &= q. \end{aligned} \tag{22}$$

We allow dependency on the anamorphic squeeze as we also did for the parameter δ. The radial symmetry becomes obvious when we insert the coefficients in (21),

$$\begin{aligned} \hat{x}_{\mathrm{dn}} &= x_{\mathrm{dn}} \left(1 + c_{xx}x_{\mathrm{dn}}^2 + c_{xy}y_{\mathrm{dn}}^2 + \frac{q}{\epsilon}(x_{\mathrm{dn}}^2 + y_{\mathrm{dn}}^2)^2\right) \\ \hat{y}_{\mathrm{dn}} &= y_{\mathrm{dn}} \left(1 + c_{yx}x_{\mathrm{dn}}^2 + c_{yy}y_{\mathrm{dn}}^2 + q(x_{\mathrm{dn}}^2 + y_{\mathrm{dn}}^2)^2\right), \end{aligned} \tag{23}$$

since all dependency on the parameter q appears as dependency on $(x_{\mathrm{dn}}^2 + y_{\mathrm{dn}}^2) = r_{\mathrm{dn}}^2$. In fact, using only δ and q with $\epsilon = 1$, i.e. no anamorphic squeeze, the distortion mapping can be written as

$$\hat{r}_{\mathrm{dn}} = r_{\mathrm{dn}}(1 + \delta r_{\mathrm{dn}}^2 + q r_{\mathrm{dn}}^4). \tag{24}$$

Using the extended version (21) of the function g we can simply use expression (14) in order to remove lens distortion, as we did before introducing the quartic term.

For applying lens distortion with a quartic term we use Newton's method again. The components of the first derivative matrix are:

$$\begin{aligned} \frac{\partial}{\partial x_{\mathrm{dn}}} g_x(x_{\mathrm{dn}}, y_{\mathrm{dn}}) &= 1 + 3c_{xx}x_{\mathrm{dn}}^2 + c_{xy}y_{\mathrm{dn}}^2 + 5c_{xxx}x_{\mathrm{dn}}^4 + 3c_{xxy}x_{\mathrm{dn}}^2 y_{\mathrm{dn}}^2 + c_{xyy}y_{\mathrm{dn}}^4 \\ \frac{\partial}{\partial y_{\mathrm{dn}}} g_x(x_{\mathrm{dn}}, y_{\mathrm{dn}}) &= 2c_{xy}x_{\mathrm{dn}}y_{\mathrm{dn}} + 2c_{xxy}x_{\mathrm{dn}}^3 y_{\mathrm{dn}} + 4c_{xyy}x_{\mathrm{dn}}y_{\mathrm{dn}}^3 \\ \frac{\partial}{\partial x_{\mathrm{dn}}} g_y(x_{\mathrm{dn}}, y_{\mathrm{dn}}) &= 2c_{yx}y_{\mathrm{dn}}x_{\mathrm{dn}} + 2c_{yyx}y_{\mathrm{dn}}^3 x_{\mathrm{dn}} + 4c_{yxx}y_{\mathrm{dn}}x_{\mathrm{dn}}^3 \\ \frac{\partial}{\partial y_{\mathrm{dn}}} g_y(x_{\mathrm{dn}}, y_{\mathrm{dn}}) &= 1 + 3c_{yy}y_{\mathrm{dn}}^2 + c_{yx}x_{\mathrm{dn}}^2 + 5c_{yyy}y_{\mathrm{dn}}^4 + 3c_{yyx}y_{\mathrm{dn}}^2 x_{\mathrm{dn}}^2 + c_{yxx}x_{\mathrm{dn}}^4, \end{aligned} \tag{25}$$

and the matrix is built from these components according to:

$$\frac{\partial g(x,y)}{\partial(x,y)} = \begin{pmatrix} \frac{\partial}{\partial x} g_x(x,y) & \frac{\partial}{\partial y} g_x(x,y) \\ \frac{\partial}{\partial x} g_y(x,y) & \frac{\partial}{\partial y} g_y(x,y) \end{pmatrix}. \tag{26}$$

We calculate appropriate initial condition as before by iterating (15), using of course the function g modified by the quartic terms. This can be done by hand or using some software for symbolic manimulation like MATHEMATICA or MAPLE. The result is:

$$\begin{aligned} x_{\rm dn}^{(0)} &= \hat{x}_{\rm dn}\big(1 - c_{xx}\hat{x}_{\rm dn}^2 - c_{xy}\hat{y}_{\rm dn}^2 \\ &+ (3c_{xx}^2 - c_{xxx})\hat{x}_{\rm dn}^4 \\ &+ (4c_{xx}c_{xy} + 2c_{xy}c_{yx} - 2c_{xxy})\hat{x}_{\rm dn}^2\hat{y}_{\rm dn}^2 \\ &+ (2c_{xy}c_{yy} + c_{xy}^2 - c_{xyy})\hat{y}_{\rm dn}^4\big) \\ y_{\rm dn}^{(0)} &= \hat{y}_{\rm dn}\big(1 - c_{yx}\hat{x}_{\rm dn}^2 - c_{yy}\hat{y}_{\rm dn}^4 \\ &+ (3c_{yy}^2 - c_{yyy})\hat{y}_{\rm dn}^4 \\ &+ (4c_{yx}c_{yy} + 2c_{xy}c_{yx} - 2c_{yyx})\hat{y}_{\rm dn}^2\hat{x}_{\rm dn}^2 \\ &+ (2c_{xx}c_{yx} + c_{yx}^2 - c_{yxx})\hat{x}_{\rm dn}^4\big)\,. \end{aligned} \tag{27}$$

For MATHEMATICA we give the set of rules in case the reader wants to verify the previous result:

```
Unprotect[Times]
Unprotect[Power]

p ^ m_ := 0 /; IntegerQ[m] && m > 5
q ^ n_ := 0 /; IntegerQ[n] && n > 5
p ^ m_ q ^ n_ := 0 /; IntegerQ[m] && IntegerQ[n] && (m + n) > 5

regelx = x->(p-cxx x^3-cxy x y^2-cxxx x^5-2 cxxy x^3 y^2-cxyy x y^4)
regely = y->(q-cyx x^2 y-cyy y^3-cyxx x^4 y-2 cyyx x^2 y^3-cyyy y^5)

x1 = x/.{regelx,regely}
y1 = y/.{regelx,regely}

x3 = x1/.{regelx,regely}
y3 = y1/.{regelx,regely}

x5 = x3/.{regelx,regely}
y5 = y3/.{regelx,regely}

xf = x5/.{x->0,y->0}
yf = y5/.{x->0,y->0}

collectp[a_] := Collect[a,p]
collectq[a_] := Collect[a,q]

xf = Collect[Expand[xf],p,collectq]
```

```
yf = Collect[Expand[yf],q,collectp]
```

which leads to the following result for `xf` and `yf`:

```
                  2           5    3
xf      = (3 cxx  - cxxx) p  + p

                                                         2
>      (-cxx + (-2 cxxy + 4 cxx cxy + 2 cxy cyx) q ) +

                    2        2                        4
>     p (1 - cxy q  + (cxy  - cxyy + 2 cxy cyy) q )

                      2                    2          4
yf      = (1 - cyx p  + (2 cxx cyx + cyx  - cyxx) p ) q +

                                                       2   3
>     (-cyy + (2 cxy cyx + 4 cyx cyy - 2 cyyx) p ) q  +

              2           5
>     (3 cyy  - cyyy) q
```

where `xf` and `yf` correspond to $x_{\rm dn}^{(0)}$ and $y_{\rm dn}^{(0)}$ and `p` and `q` correspond to $\hat{x}_{\rm dn}$ and $\hat{y}_{\rm dn}$.