1 Fisheye Transformation

The objective of this task is to implement a fish-eye transformation in order to reconstruct a given distorted image. An illustration of such transformation is provided in the lectures of [12] and depicted in Figure 1. To address this task, we first explored the theoretical foundations described in section 2 and then implemented the code as described in section 3.

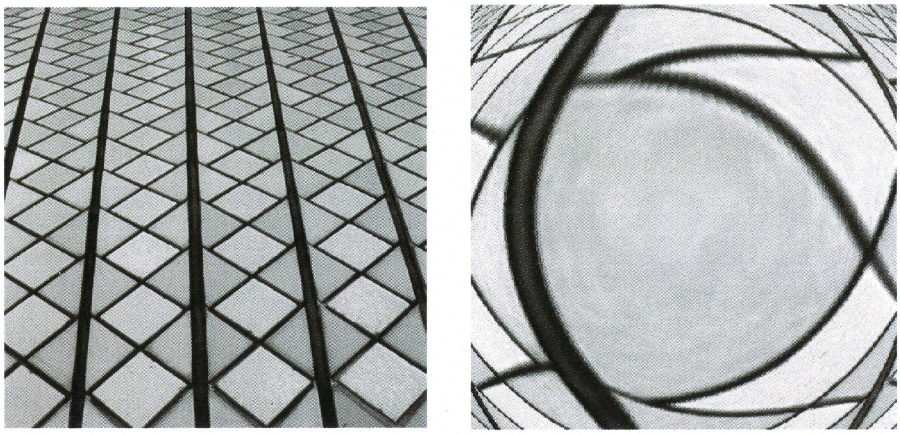


Figure 1: The lecture notes referenced in [12] present this figure, which demonstrates the fish-eye distortion task. On the left side, there is an image depicting tiles, captured from a top-down perspective at a slight angle. On the right side, the image illustrates the same left image after a fish-eye transformation.

A comprehensive list of lecture material [12] and additional literature can be found in the bibliography. Among these sources, the article titled 'Review of Geometric Distortion Compensation in Fish-Eye Cameras' was particularly useful [9]. The acquired knowledge was implemented using a Python script, specifically in a Jupyter Notebook. This choice was made because of Python's intuitive and abstract nature, combined with the versatile data processing and visualization capabilities offered by the Python Notebook.

2 Theoretical Background

The fish-eye transformation is a mathematical or algorithmic method that recreates or corrects an image distortion caused by a fish-eye lens. This chapter explains the background knowledge necessary to understand the experiments and results, starting with the basic camera characteristics described in section 2.1. An overview about camera calibration techniques can be found in 2.2. Techniques and algorithms for radial transformation of an image are described in sections 2.3, 2.4, 2.5, 2.6 and 2.7.

2.1 Pinhole Camera

The pinhole camera is a basic model that describes the process of making a 2D image of the 3D world. An illustration of this model is shown in Figure 2. The light beams go through a small hole and land on the image plane. The hole is called the focal point. The distance between the image plane and the focal point is called focal length. The center point of the image is called the principal point [13].

As no lenses are used, this represents a linear projective projection. A camera typically has lenses that bend the light beams on the way from the outside world to the camera

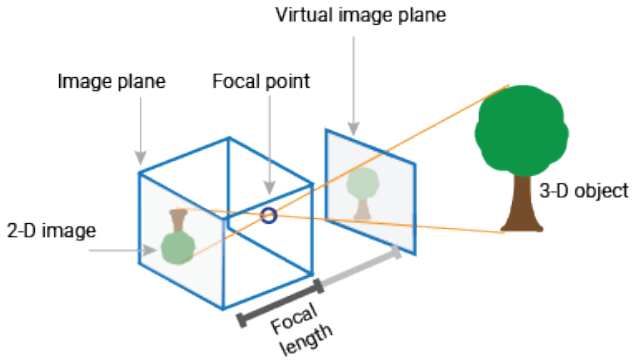


Figure 2: Pinhole camera model. The 3D tree is projected without distortion but inverted onto a 2D planar image plane. Source: [13]

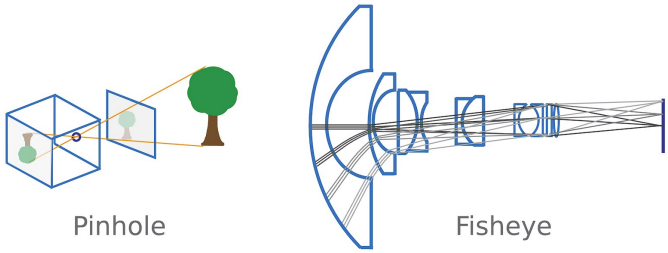


Figure 3: This figure illustrates the difference between the pinhole camera and a fisheye lens. While in a pinhole camera the light is not redirected, the fisheye lens captures beams from a wider angle and redirects them through several lenses on the image plane. Source: [13]

image sensor. This is depicted in Figure 3. In the pinhole camera, every light beam forms a straight line. This limits the maximum viewing angle. To increase the angle, the focal length needs to be reduced. But in real cameras, there are physical limitations and the focal length cannot be shortened infinitely. Therefore, lenses capture light outside of the pinhole camera view angle.

2.2 Camera calibration

According to Tsai[2], camera calibration in 3D machine vision involves determining intrinsic parameters (geometric and optical characteristics of the camera lenses) and extrinsic parameters (camera position and orientation) relative to a world coordinate system. A calibrated camera is essential for the following computer vision tasks:

- Extracting 3D information from the 2D image coordinates mainly to determine the location of a specific feature or to estimate the pose of a robot with a mounted camera inside a world coordinate system.
- Extracting 2D image coordinates from the 3D world. This is commonly used in model-driven inspection to confirm or verify a hypothesis about the state of an object by comparing its actual image coordinates against predicted image coordinates calculated from a 3D model.

Tsai[2] mentions five important characteristics that camera calibration techniques have to fulfill:

1. There must be no human intervention for the process to work.
2. It must be accurate enough to fulfill the requirements of the application in which the camera is used.
3. It has to be efficient, which means it should not contain a nonlinear high-dimensional optimization problem.
4. It should work on any common camera.
5. It should only require the camera and no additional pre- or post-processing equipment.

There are a large number of calibration techniques. Tsai[2] categorizes them into four categories, which are listed here with their respective advantages and disadvantages (refer to [2] for examples of each category) in Table 1.

Cat.	**Technique**	**Pro's**	**Con's**
1	Using a full-scale non-linear optimization.	Versatile because it can easily be adapted to any complex model.	Prior information required and high computational cost.
2	Calculating a perspective transformation by solving a linear equation.	Optimization problem is linear.	Lens distortion is disregarded. More number of unknowns than degrees of freedom. Limited accuracy in noisy conditions.
3	Using two planes.	Only linear equations needed.	Much higher number of unknowns than degrees of freedom. Transformation derived empirically and not analytically.
4	Using geometry.	Only a linear search required.	Lens distortion is disregarded. Focal length is assumed to be known. Image scale must be known.

Table 1: Table which describes categories of calibration techniques with their corresponding pros and cons. Source: [2]

To calibrate an image using a category 1 calibration technique, the camera model is described by Tsai in Figure 4.

The first step is a rigid transformation, changing the camera position inside the world coordinate system. Parameters are the translation (T) and rotation (R) matrices.

$$R \equiv \begin{bmatrix} r_1 & r_2 & r_3 \\ r_4 & r_5 & r_6 \\ r_7 & r_8 & r_9 \end{bmatrix} \quad T \equiv \begin{bmatrix} T_x \\ T_y \\ T_z \end{bmatrix} \tag{1}$$

The second step involves the projective transformation of the 3D camera coordinate system to a 2D image coordinate system. The parameter is the focal length of the camera.

$$X_u = f \cdot \frac{x}{z} \qquad Y_u = f \cdot \frac{y}{z} \tag{2}$$

In the third step, the radial lens distortion is calculated. Tsai uses a polynomial radial distortion with parameters κ_i. In this report, other distortion models were also used. Aside from radial, there is tangential distortion also. However, this is neglected both in the paper and in this report.

$$D_x = X_d(\kappa_1 r^2 + \kappa_2 r^4 + ...) \tag{3}$$
$$D_y = Y_d(\kappa_1 r^2 + \kappa_2 r^4 + ...) \tag{4}$$
$$r = \sqrt{X_d^2 + Y_d^2} \tag{5}$$

Lastly, in the fourth step, the real-world coordinates are transformed into image coordinates.

$$X_f = s_x d_x'^{-1} X_d + C_x \tag{6}$$

$$Y_f = d_y^{-1} Y_d + C_y \tag{7}$$

where:

(X_f, Y_f) = row and column numbers of the image pixel in computer frame memory

(C_x, C_y) = row and column numbers of the center of computer frame memory

$d_x' = d_x \frac{N_{cx}}{N_{fx}}$

d_x = center-to-center distance between adjacent sensor elements in X (scan line) direction

d_y = center-to-center distance between adjacent CCD sensors in the Y direction

N_{cx} = number of sensor elements in the X direction

N_{fx} = number of pixels in a line as sampled by the computer

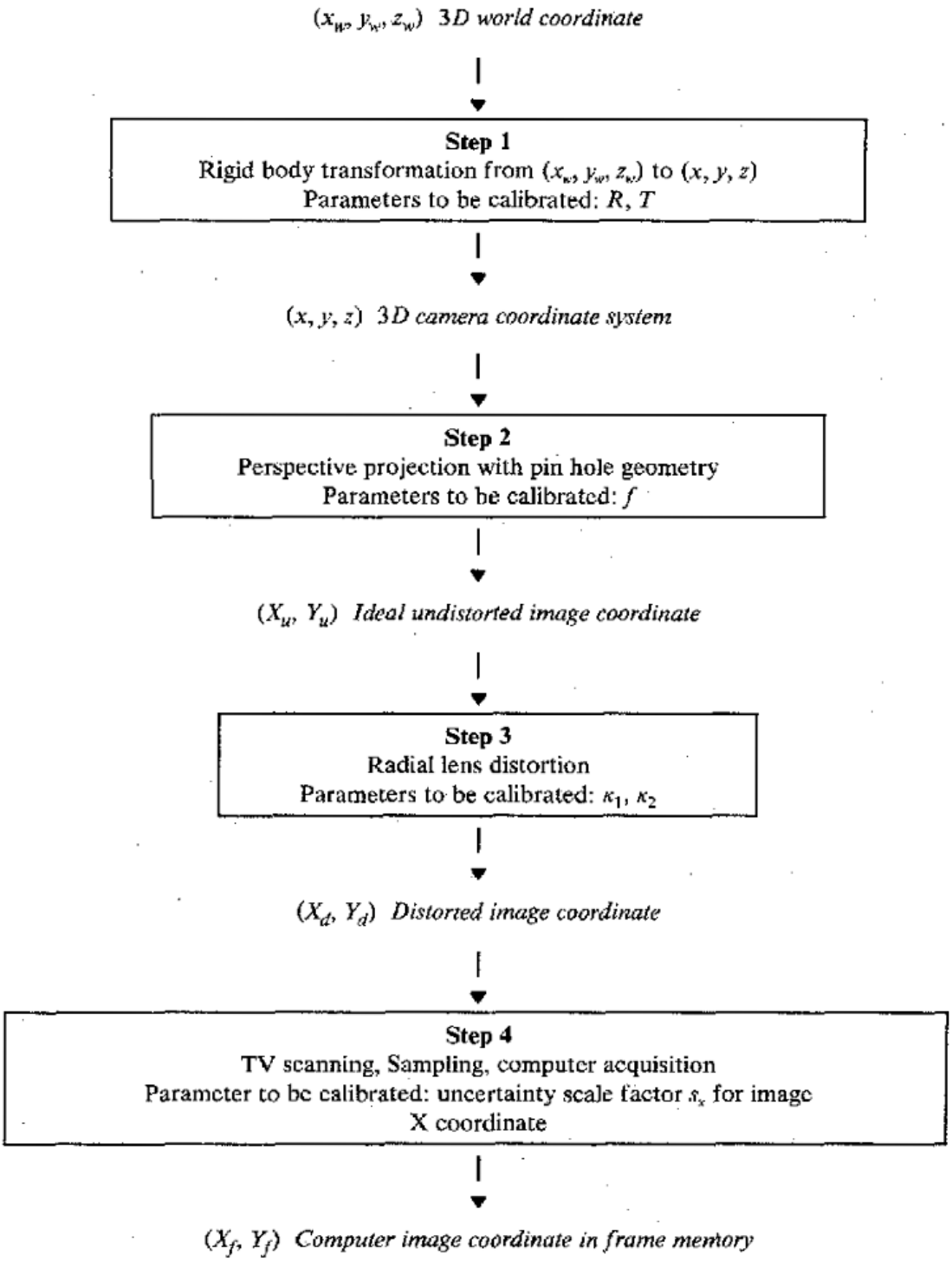


Figure 4: This figure describes four steps to transform the 3D world coordinate in which a camera is embedded into 2D image coordinates. Source: [2]

2.3 Radial Distortion

Radial distortion describes the key concept necessary to understand the effect of a fisheye lens on an image. Fisheye cameras can capture light from a field of view up to 180 degrees. As more information is compressed onto the image plane, the 2D image is non-linearly distorted. This is called radial distortion [9].

There are two main types of radial distortion. These are illustrated by [13] in Figure 5. In the center, a grid is visible without any distortion. This acts as a reference to see the differences between the distortion types. All parallel horizontal and vertical lines are straight. On the left side of the figure, the pincushion distortion is visible. All lines are bent beside the center horizontal and vertical lines that cross the principal point. All other lines bend away from the corresponding parallel central line as the distance to the principal point increases. On the right side, the barrel distortion is visible. Here the inverse effect can be seen. As the distance from the principal point increases, all lines bend toward the corresponding central lines.

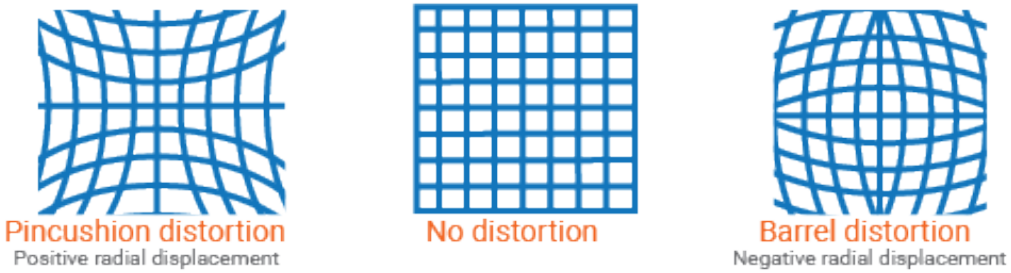


Figure 5: Radial distortion types. Left: Pincushion distortion. Center: No distortion. Right: Barrel distortion. Source: [13]

Fisheye models used for radial distortion: The following models described in [9] were used to generate and rectify a fisheye transformation:

- Polynomial model described by Mallon and Whelan [7]
- Perspective model described by Ishii, Sudo, and Hashimoto [6]
- The fish-eye transform by Basu and Licardie [3]
- The field-of-view model by Devernay and Faugeras [5]

2.4 Polynomial model described by Mallon and Whelan [7]

According to [9], polynomials are commonly used to model radial distortion of wide angle or low focal lenses because their implementations are computational efficient. But they often lack an analytical method for inversion. Also, Mallon and Whelan mention that there is no benefit using a polynomial model higher than fifth order.

Following, the formula for the odd polynomial distortion is depicted:

$$p_d = p_u + f_u, \tag{1}$$

where

$$f_x(x_u, y_u) = k_1 x_u r_u^2 + k_2 x_u r_u^4,$$
$$f_y(x_u, y_u) = k_1 y_u r_u^2 + k_2 y_u r_u^4.$$

with

$$p_u = (x_u, y_u)^T$$
$$r_u = \sqrt{x_u^2 + y_u^2}$$

Now the formula for the odd polynomial correction is depicted:

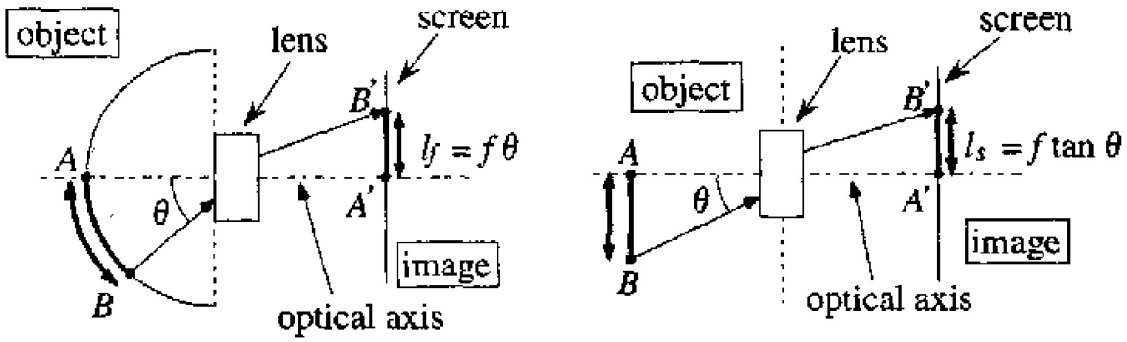
$$r_u = r_d - r_d \left(\frac{\kappa_1 r_d^2 + \kappa_2 r_d^4 + \kappa_1^2 r_d^4 + \kappa_2^2 r_d^8 + 2\kappa_1\kappa_2 r_d^6}{1 + 4\kappa_1 r_d^2 + 6\kappa_2 r_d^4} \right) \tag{8}$$

with

$$r_d = \sqrt{x_d^2 + y_d^2} \tag{9}$$

2.5 Perspective Model described by Ishii, Sudo, and Hashimoto [6]

Common scientific fisheye lenses use equidistant projections like depicted in Figure 6a. This model projects an arbitrary image point B to the image plane using a circle with the central point at the focal point and radius defined by the focal length. This distance l_f of B to the optical axis A is the circle distance on the aforementioned circle and therefore given by $l_f = f \cdot \theta$. Common wide-angle lenses instead use a gnomonic projection as depicted in Figure 6b. Here the distance of a point B to the optical axis is defined with a line orthogonal to the optical axis. This distance is calculated using the trigonometric formula $l_s = f \cdot \tan\theta$.



(a) (b)

Figure 6: Illustration of the equidistant projection principle (left) in direct comparison to the central projection method (right). [6]

The core idea of the transformation described by Ishii, Sudo, and Hashimoto [6] is that they assume that the distances l_f and l_s are proportional to the angle θ:

$$l_f \propto l_s \tag{10}$$
$$f \cdot \theta \propto f \cdot \tan(\theta) \tag{11}$$

Now the angle can be estimated from the equidistant projection

$$l_f = f \cdot \theta \tag{12}$$
$$\Leftrightarrow \quad \frac{l_f}{f} = \theta \tag{13}$$

and inserted into the formula for the gnomonic projection

$$l_s = f \cdot \tan\left(\frac{l_f}{f}\right) \tag{14}$$

After the signs l_f and l_s are relabeled to r_d and r_u the final formula for the corrected point is

$$r_u = f \tan\left(\frac{r_d}{f}\right) \tag{15}$$

and for the distorted point

$$r_d = f \arctan\left(\frac{r_u}{f}\right) \tag{16}$$

2.6 Fisheye Transformation described by Basu and Licardie [3]

The fisheye transform described by Basu and Licardie [3] is a variable-resolution mapping method inspired by the retino-cortical mapping in human and primate vision. It creates images with a high-resolution focus (fovea) and non-linearly decreasing resolution toward the periphery. The distortion is represented by

$$r_d = s \ln(1 + \lambda r_u) \tag{17}$$

and the inverse by

$$r_u = \frac{e^{\frac{r_d}{s}} - 1}{\lambda} \tag{18}$$

2.7 Fisheye Transform by Devernay and Faugeras [5]

The core idea behind this concept is that the distance of an image point on the 2D planar plane to the principal point is approximately proportional to the angle of the projected 3D point and the central optical axis. This principle is depicted in Figure 7.

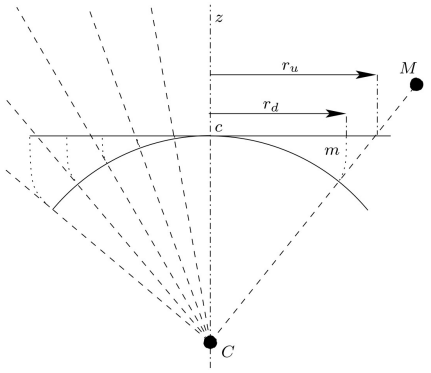


Figure 7: Illustration of the fisheye distortion model described by Devernay and Faugeras. r_u and r_d are both proportional to the angle of those points to the optical axis. Source: [5]

The corresponding distortion formula is

$$r_d = \frac{1}{\omega} \arctan\left(2r_u \tan\frac{\omega}{2}\right) \tag{19}$$

and the inverse

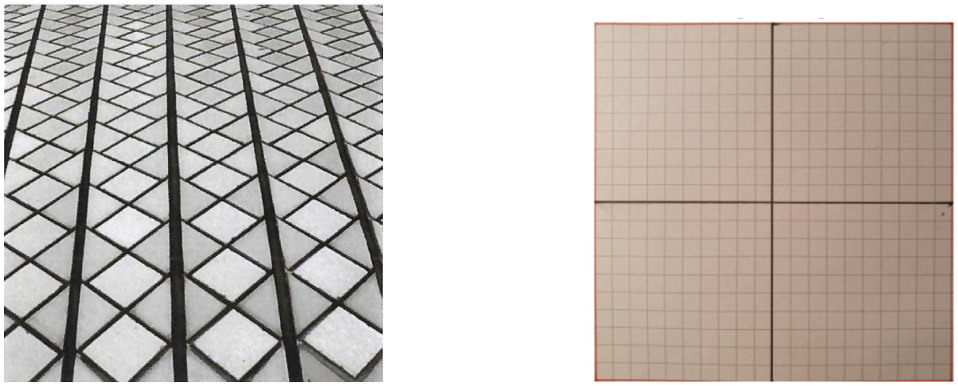
$$r_u = \frac{\tan(r_d\omega)}{2\tan(\frac{\omega}{2})} \tag{20}$$

where the parameter ω represents the field of view of the "ideal" fisheye lens. Devernay and Faugeras note that this model might not be sufficient to correct complex distortion. Therefore, the image can be preprocessed with a polynomial correction model.

3 Implementation

The task was implemented with the Python programming language in a Jupyter Notebook. Additionally, the libraries cv2, pyplot, and numpy were used for processing and visualizing the images.

The implementation was divided into two phases. In the first phase, we performed radial distortion correction and reconstruction on a sample image. Similar to the provided image, which shows tiles from a top-down perspective at a slight angle 8a, the sample image contained vertical and horizontal lines in a top-down arrangement. For simplicity, we omitted the diagonal lines and the slight angle as shown in the figure 8b. This approach allowed a clearer observation of the transformation effects and helped to estimate the most appropriate models and parameters. In the second phase, we applied the knowledge and data obtained in the first phase to reconstruct the given fisheye distorted image shown in Figure 1 and therefore complete the task.



(a) (b)

Figure 8: Illustration of the two images used for the two implementation phases. Left: the given image with tiles, taken from a top-down perspective at a slight angle. Right: A sample image with vertical and horizontal lines taken from a top-down perspective, but without diagonal lines and the slight angle.

The processes of distorting and rectifying an image follow the same pattern, which can be summarized in the following six steps. Figure 9 provides an intuitive and visual illustration of these steps for a fictive image and transformation.

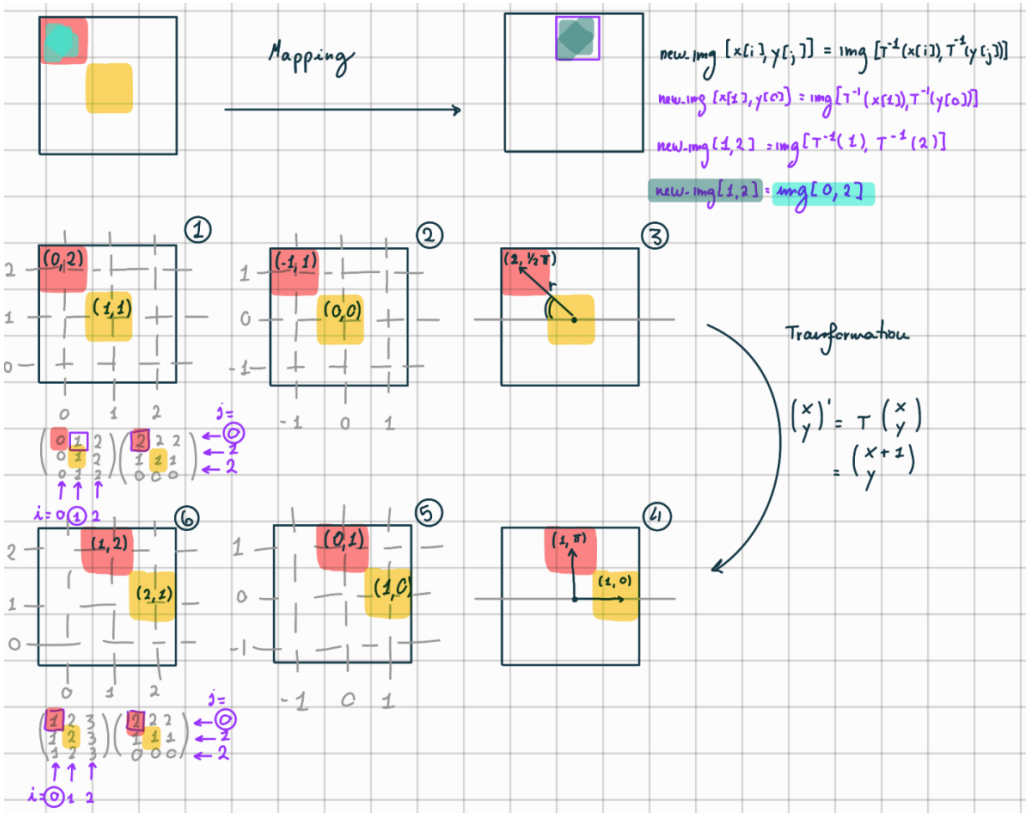


Figure 9: An intuitive and visual illustration of the six implementation steps for a fictional image transformation. The six steps are numbered and include (1) creating a coordinate grid, (2) normalising the coordinates, (3) converting to polar coordinates, (4) applying or correcting the distortion, (5-6) converting back to Cartesian coordinates and converting back to pixel coordinates. The backward mapping is illustrated in the top half of the figure with an example calculation to assign the correct color value of the corresponding pixel in the original image to the pixel at coordinates (1,2) in the transformed image.

3.1 Creating a coordinate grid

The first step is to create a grid that covers the position of each pixel in the image. These are stored in 2D arrays whose elements correspond to the respective x and y coordinates in the image grid. For this and the following steps, it is assumed that all distortion centroids coincide with the image centroids.

3.2 Normalization of the coordinates

To make the calculations independent of the actual image size, the coordinates are normalized with

$$x_{\text{norm}} = \frac{2x}{\text{width}} - 1 \tag{21}$$

and

$$y_{\text{norm}} = \frac{2y}{\text{height}} - 1 \tag{22}$$

This scales the image in a range between -1 and 1 and its center to the position (0,0).

3.3 Conversion to polar coordinates

To facilitate the application of the transformation models, the previously calculated normalized Cartesian coordinates are converted to polar coordinates according to Ngo and Asari [8] and M.Mohankumar, T.Thamaraimanalan, and N.Sanjeev [1]. These coordinates depend exclusively on the distance from the image center and are calculated with the parameters

$$r = \sqrt{x_{\text{norm}}^2 + y_{\text{norm}}^2} \tag{23}$$

and

$$\phi = \arctan(y_{\text{norm}}, x_{\text{norm}}) \tag{24}$$

Here r is the radius from the pixel to the image center, and ϕ is the angle relative to the horizontal axis [8] [1].

3.4 Application or correction of the distortion

To compute the new coordinates of the image, it is essential to first determine the appropriate parameters and then apply a suitable transformation method. Table 2 presents the most optimal parameter values for each of the four previously described methods. These values were identified through a trial-and-error process, where the default parameters provided by Hughes et al. were incrementally adjusted to assess their impact on the transformation and calibrate them until satisfactory results were obtained.

Polynomial Distortion model	k1 = 0.2 and k2 = 0.2
Fisheye transformation by Basu and Licardie	l = 400 and s = 0.22
Fisheye transform by Devernay and Faugeras	omega = 2.5
Perspective model by Ishii, Sudo, and Hashimoto	f = 0.105

Table 2: Model parameters: This table shows, for each radial transformation model described in section 2.3, the required parameters and their most appropriate values.

3.5 Conversion back to Cartesian and Pixel coordinates

To project the coordinates back into the original latitude and altitude ranges, the transformed polar coordinates are converted back into Cartesian coordinates by

$$x_{\text{new}} = r \cdot \cos(\phi) \tag{25}$$

and

$$y_{\text{new}} = r \cdot \sin(\phi) \tag{26}$$

and these are then converted back to the original pixel area by

$$x_{\text{pixel}} = \frac{x_{\text{new}} + 1}{2} \cdot \text{width} - 1 \tag{27}$$

and

$$y_{\text{pixel}} = \frac{y_{\text{new}} + 1}{2} \cdot \text{height} - 1 \tag{28}$$

3.6 Backward mapping

Since the models used for coordinate transformation are not inherently invertible and implicitly produce back-mapped coordinates [9], we chose back-mapping over forward-mapping as our approach to image warping. In addition, this approach is widely used for its advantage of avoiding floating-point gaps by ensuring that each target pixel is assigned a value from the source image [9, 4]. In backward mapping, for each pixel (X_t, Y_t) in the target image, its corresponding position (X_s, Y_s) in the source image is determined using the inverse transformation [4]:

$$(X_s, Y_s) = T^{-1} \cdot (X_t, Y_t) \tag{29}$$

where T is the transformation function. Conversely, forward mapping determines the transformed position for each pixel (X_s, Y_s) in the source image using [4]:

$$(X_t, Y_t) = T \cdot (X_s, Y_s). \tag{30}$$

In our implementation, we iterate over all pixels in the target image. For each target pixel (i, j), the corresponding pixel value is fetched from the input image at $(X_{\text{new}}, Y_{\text{new}})$ where

$$X_{\text{new}} = X_t[i, j] \tag{31}$$

and

$$Y_{\text{new}} = Y_t[i, j]. \tag{32}$$

The final result is a new image with the fetched values for all coordinates (i, j) in the target image. However, due to the mismatch between the discrete pixel coordinates and the continuous inverse computed coordinates shown in Figure 10, the final result can suffer from aliasing artefacts and loss of detail. These issues can be mitigated using interpolation techniques such as nearest neighbour, bilinear, bicubic or spline interpolation [10, 11]. Since these discrepancies were minimal in our results, we chose to address this mismatch by rounding pixel positions to their nearest discrete values [10].

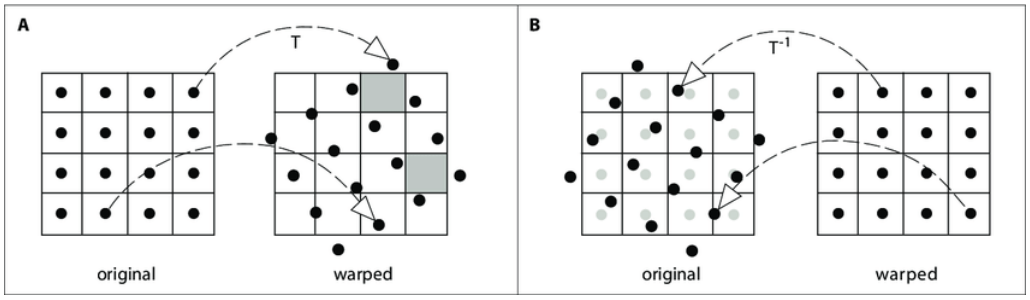


Figure 10: Comparison of the the mapping methods: Left: Foreward mapping. Right: Backwardard mapping

4 Results

The figures below demonstrate that all four transformation models described above can be effectively applied to our sample image, resulting in both fisheye distortion and accurate correction to the original image.

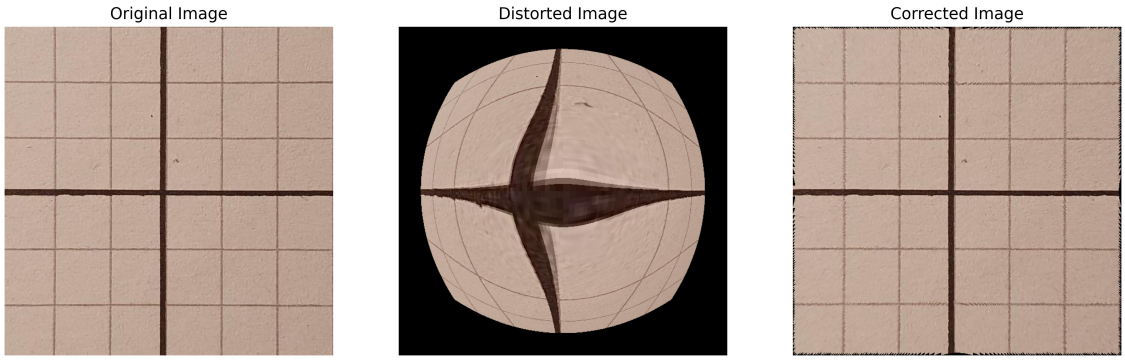


Figure 11: Fisheye model from Basu and Licardie [3]: Left: sample image resembling the given original fisheye-distorted image. Center: Distorted image according to Basu and Licardie [3]. Right: Reconstructed image according to Basu and Licardie [3]

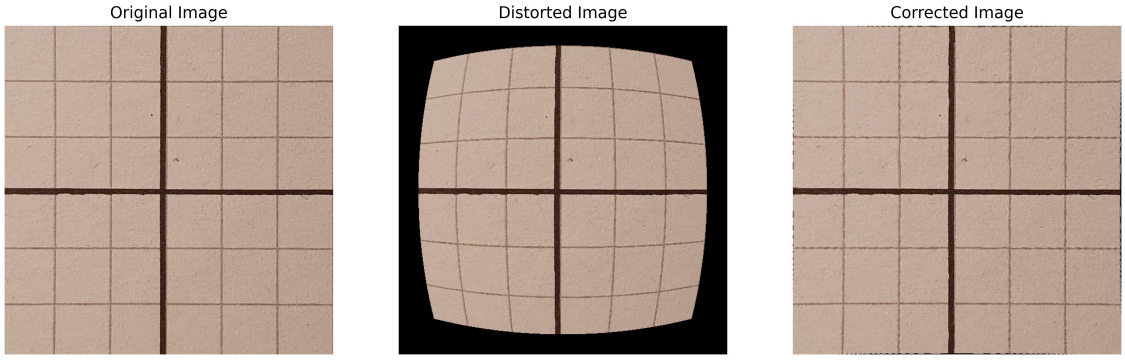


Figure 12: Odd polynomial model from Mallon and Whelan [7]: Left: sample image resembling the given original fisheye-distorted image. Center: Distorted image according to Mallon and Whelan [7]. Right: Reconstructed image according to Mallon and Whelan [7]

Among these models, the one proposed by Basu and Licardie, shown in Figure 11, proves to be the most suitable for our task. It produces a more pronounced distortion effect that closely matches the given fisheye-distorted image in Figure 1.

This result is further supported by the direct comparison of image reconstructions in Figure 15. The model from Basu and Licardie correctly restores the distorted image to its original state, while the other models leave residual distortions, especially in the central region. However, a notable limitation of this model as well as the ones from Ishii, Sudo, and Hashimoto [6] and Devernay and Faugeras [5], compared to that of Mallon and Whelan [7], is the "S-shape" reconstruction of the image, which captures only a partial view of the photographed object.

5 Conclusion

This project successfully implemented a fisheye reconstruction of the given distorted image shown in Figure 1, combining theoretical exploration with practical implementation in Python.

We started by applying radial transformation models to distort and reconstruct a sample image 8b with characteristics similar to the given original image 8a. Using the insights and data gained from this first phase, we then proceeded to solve the main task of recon-

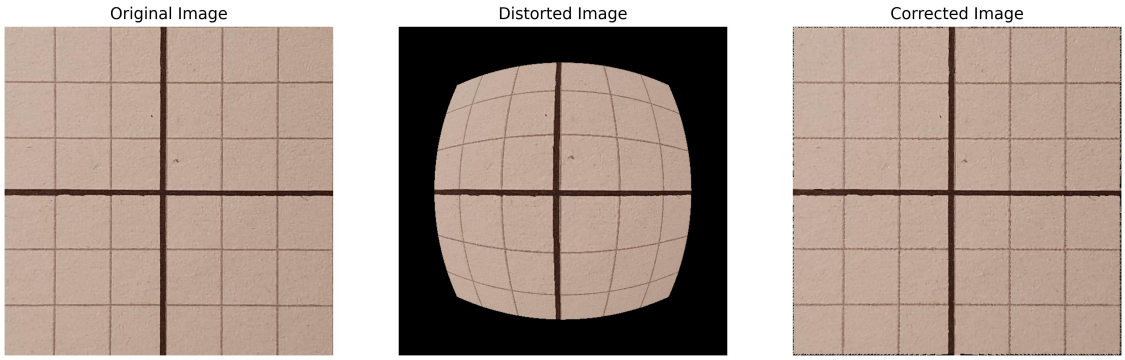


Figure 13: Fisheye model from Devernay and Faugeras [5]: Left: sample image resembling the given original fisheye-distorted image. Center: Distorted image according to Devernay and Faugeras [5]. Right: Reconstructed image according to Devernay and Faugeras [5]

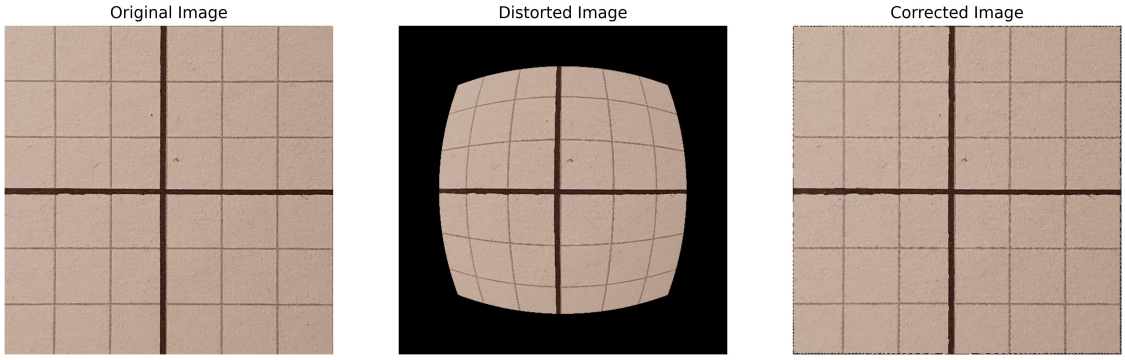


Figure 14: Perspective fisheye model from Ishii, Sudo, and Hashimoto [6]: Left: sample image resembling the given original fisheye-distorted image. Center: Distorted image according to Ishii, Sudo, and Hashimoto [6]. Right: Reconstructed image according to Ishii, Sudo, and Hashimoto [6]

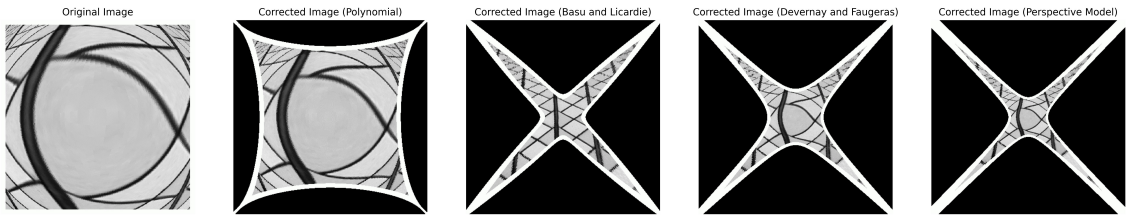


Figure 15: Corrected images comparison: The original fisheye-distorted image (Figure 1) is reconstructed by applying the models from Mallon and Whelan [7], Basu and Licardie [3], Devernay and Faugeras [5], and Ishii, Sudo, and Hashimoto [6] (from the second on the left to the fifth on the right)

structing the given fisheye-distorted image. This process involved six steps: (1) creating a coordinate grid, (2) normalizing the coordinates, (3) converting to polar coordinates, (4) applying a fish-eye reconstruction or distortion, (5) converting back to Cartesian and pixel coordinates, and (6) mapping each transformed coordinate to the original pixel grid. We tested four transformation models, with the approach of Basu and Licardie proving to be the most effective for our task 15.

For future research, two aspects of our project could be improved. First, the use of more precise and structured methods for parameter calibration, beyond the simple trial-and-error approach. This could improve the accuracy of the results and scalability of the project . Second, incorporating interpolation techniques such as nearest neighbor, bilinear, bicubic or spline interpolation, rather than simply rounding pixel positions to their nearest discrete values. This could help to eliminate potential aliasing artifacts and information loss during the backward mapping process.