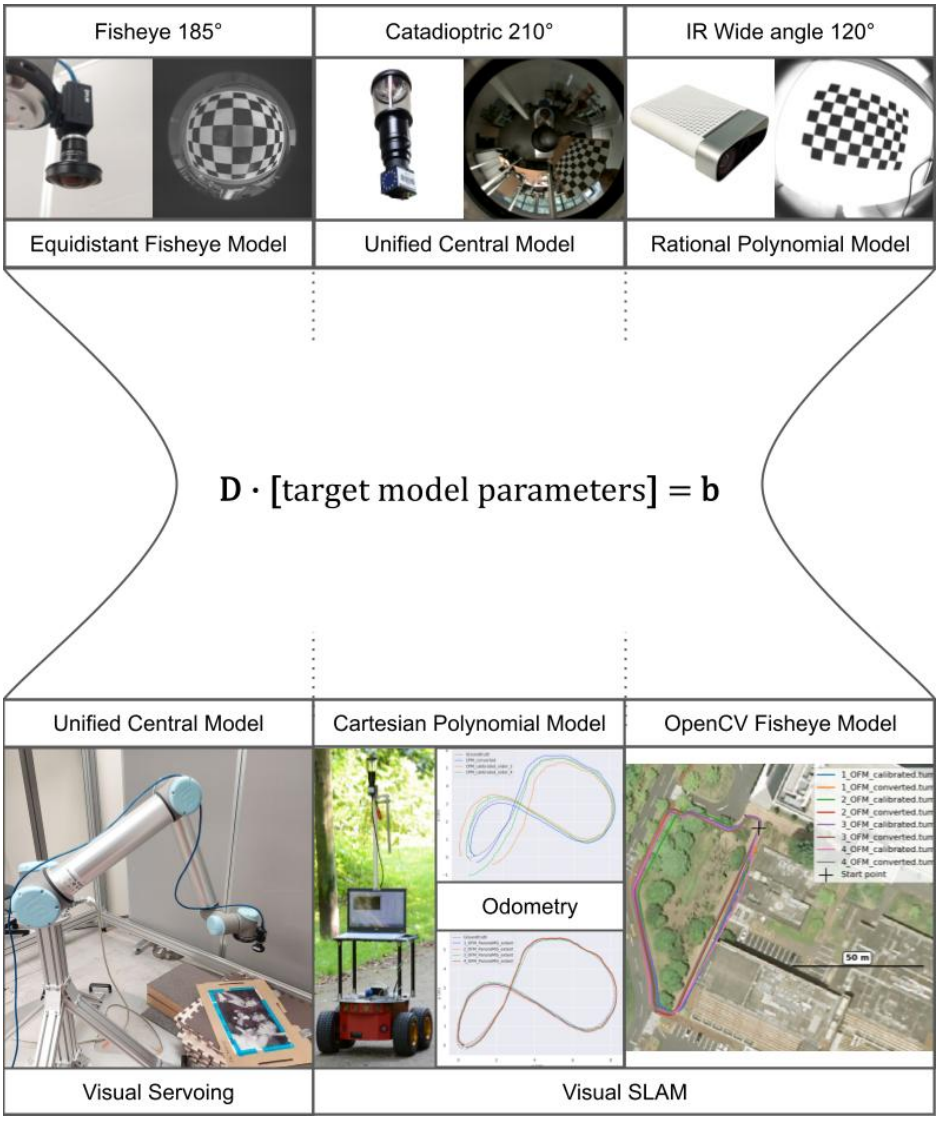
Linear conversions of nonlinear camera models for robotic vision applications

Eva Goichon, Guillaume Caron, Pascal Vasseur, Fumio Kanehiro

Graphical Abstract

Linear conversions of nonlinear camera models for robotic vision applications

Eva Goichon, Guillaume Caron, Pascal Vasseur, Fumio Kanehiro



Highlights

Linear conversions of nonlinear camera models for robotic vision applications

Eva Goichon, Guillaume Caron, Pascal Vasseur, Fumio Kanehiro

- Converting the camera model allows a camera to be used with a new camera model without calibration.
- Validation by undistortion accuracy and comparison with calibration.
- Standard error metrics (SSIM, APE, RPE) confirm the reliability of the converted parameters.
- Application to various unconventional cameras for visual odometry, visual servoing and SLAM.
- Open-source software: `libPeR` https://github.com/PerceptionRobotique/libPeR_base.
- Open-source dataset: WAIr-JaM https://extra.u-picardie.fr/nextcloud/index.php/s/Ltj322skRmkTmQC.

Linear conversions of nonlinear camera models for robotic vision applications

Eva Goichon[a,b,*], Guillaume Caron[a,b], Pascal Vasseur[a], Fumio Kanehiro[b]

[a] *MIS laboratory (Modeling, Information & Systems), Université de Picardie Jules Verne (UPJV), 33 Rue Saint-Leu, Amiens, 80039, France*
pascal.vasseur/guillaume.caron @u-picardie.fr
[b] *CNRS-AIST JRL (Joint Robotics Laboratory), IRL, National Institute of Advanced Industrial Science and Technology (AIST), 1-1-1 Umezono, Tsukuba, 305-8560, Japan*
f-kanehiro@aist.go.jp

Abstract

Camera models play a crucial role in robot vision applications. Yet their diversity poses a challenge when working with data captured with cameras calibrated using different models. In this paper, we address this issue by introducing a mathematical framework that enables conversion between various camera projection models. This approach allows algorithms designed for a specific model to process data from cameras calibrated with other models, eliminating the need for recalibration and enabling the reuse of pre-existing datasets that do not provide access to calibration images.

We present the general conversion method for state-of-the-art camera models that we derive for three new camera model conversions, covering various camera types, including fisheye and catadioptric systems. Quantitative evaluation is conducted with respect to well-known calibration methods. We compare our method on image undistortion, as well as in practical applications such as SLAM, visual servoing, and visual odometry. The results demonstrate that our conversion approach achieves performances comparable to calibration without the need for explicit calibration.

This work contributes to a more flexible and adaptive use of cameras in robot applications. The proposed camera model conversion framework is implemented in the open-source `libPeR` library, available at: https://github.com/PerceptionRobotique/libPeR_base.

*Corresponding author: Eva Goichon, email: eva.goichon@outlook.fr

Keywords: Camera model conversion, Simultaneous Localization and Mapping (SLAM), Visual Odometry, Visual Servoing, Camera calibration

---

Funding

This work was supported by CR Hauts-de-France and AIST.

Author contributions

Eva Goichon: Conceptualization, Data curation, Formal analysis, Investigation, Software, Validation, Visualization, Writing - original draft. **Guillaume Caron**: Conceptualization, Data curation, Formal analysis, Funding acquisition, Investigation, Methodology, Resources, Software, Validation, Supervision, Visualization, Writing - original draft, Writing - review and editing. **Pascal Vasseur**: Conceptualization, Funding acquisition, Resources, Supervision, Writing - review and editing. **Fumio Kanehiro**: Conceptualization, Funding acquisition, Resources, Supervision, Writing - review and editing.

Acronyms

AKDK Azure Kinect DK

APE Absolute Pose Error

CAHV Center Axis Horizontal Vertical vectors

CAHVOR Center Axis Horizontal Vertical Optical Radial vectors

CP2OF Cartesian Polynomial to OpenCV's Fisheye model conversion

CPM Cartesian Polynomial Model

DVS Direct Visual Servoing

EF2UC Equidistant Fisheye to Unified Central model conversion

EFM Equidistant Fisheye Model

FoV Field of View

GT Ground Truth

IR Infrared

OCamCalib Omnidirectional Camera Calibration toolbox

OFM OpenCV's Fisheye Model

PM Polynomial radial distortion Model

RANSAC Random Sample Consensus

RGB Red Blue Green

RP2OF Rational Polynomial radial distortion to OpenCV's Fisheye model conversion

RP2P Rational Polynomial radial distortion to Polynomial radial distortion model conversion

RPE Relative Pose Error

RPM Rational Polynomial radial distortion Model

SLAM Simultaneous Localization And Mapping

SSIM Structural Similarity Index Measure

SVO Semi-direct Visual Odometry

UC2CP Unified Central to Cartesian Polynomial model conversion

UCM Unified Central Model

1. INTRODUCTION

When performing geometric calculations from images, it is crucial to adopt projection models that accurately describe the image formation process captured by the camera [1]. In recent years, unconventional cameras have gained growing popularity due to their enhanced capabilities, particularly their extended Field of View (FoV). Among these, wide-angle, fisheye, and omnidirectional cameras offer a FoV far superior to that of pinhole cameras [2].

These cameras are increasingly used in various computer vision applications, including localization tasks (visual odometry [3], SLAM [4]), as well as vision-based control (visual servoing [5]). However, each algorithm tailored for one of the latter tasks relies on a specific camera model, which may differ from the one used for camera calibration, for instance done accurately at the factory. Such incompatibility poses a major challenge for the seamless integration of these cameras into robotic systems without requiring strong expertise in various and unconventional camera recalibration methods.

For unconventional cameras, such as fisheye panoramic cameras [6] or catadioptric systems combining image sensors with curved mirrors [7], several different models have been proposed [8] to overcome the limits of perspective and Polynomial radial distortion Model (PM) or Rational Polynomial radial distortion Model (RPM) distortion models. These models aim to address the singularities of the classical pinhole model due to their very wide FoV, often exceeding 180 degrees, making the pinhole model inadequate. Among the most common models for fisheye cameras are the seminal equidistant and equisolid models [6], as well as more general formulations like the polynomial angular model of Kannala and Brandt [9].

One also finds the Center Axis Horizontal Vertical Optical Radial vectors (CAHVOR) model [10], an extension of the pinhole model, primarily used in the space industry for wide-angle lenses. It accounts for both geometric and optical distortions but is generally not suited for hyper fisheye cameras, which have a FoV exceeding 180°, or omnidirectional cameras. While widely adopted in the space sector, particularly in planetary surface mapping applications, this model is however not employed in robotics contrary to OpenCV's Fisheye Model (OFM) that ships with the famous open source OpenCV library [11]. It is also an extension of the pinhole model but inspired of the angular polynomial model of Kannala and Brandt [9] to model the radial distortions missed by the former. The OFM, is commonly employed to model fisheye images in robot vision software integrating OpenCV but due to the pinhole projection involved, it fails to model accurately cameras of FoV close or beyond 180°, such as catadioptric systems. These systems require specific calibration techniques adapted to the geometry of the reflecting mirror. For example, calibration can be performed using models dedicated to the mirror shape, such as the ad hoc parabolic model [12], or by employing the Unified Central Model (UCM) for camera-mirror systems that feature a single viewpoint [13, 14].

When the mirror shape or the camera-mirror alignment is imperfect, the UCM extends by incorporating the same classical polynomial radial distortion models used for pinhole cameras [2]. However, this approach relies on a two-step modeling process leading to a large number of parameters with redundancies. To overcome this limitation and inspired by the polynomial angular model designed for fisheye cameras, the Cartesian Polynomial Model (CPM) was introduced to simultaneously account for mirror geometry and distortions [15]. This model can also describe certain fisheye cameras [16].

Thus, the landscape of projection models is vast and diverse.

In practice, it is often necessary to recalibrate a camera to adopt the model required by a given robot vision algorithm. Recalibration is not only time-consuming but can also be less accurate than the factory calibration performed by the manufacturer with high-grade calibration objects and procedures. Moreover, in certain situations, such as when calibration images are unavailable in a dataset, classical calibration becomes impossible.

To overcome these constraints, we propose a linear computation method that converts models, which can be non-linear. The parameters of input models are derived from calibration or manufacturer specifications. Our conversion computes the parameters of the output model required by the target algorithm, without requiring calibration images. Even though the idea of converting between camera models was introduced for converting the parameters of the CAHVOR model to those of the perspective projection with polynomial distortions in [10], it was only for equivalent models allowing to directly express the output parameters only from those of the input model. In this work, we instead leverage [17] that paved the way to develop conversions between models that are not equivalent (RPM to PM, Equidistant Fisheye Model (EFM) to UCM and UCM to CPM), thus with the impossibility to express the intrinsic parameters of the output camera model only from those of the input model. [18] is another approach to convert between camera models concentrating on fisheye cameras. The conversion is solved by non-linear optimization using inverse projection functions of an input model applied to a set of generated points and forward projection functions of the output model among UCM, CPM, PM, Enhanced UCM [19], Double Sphere [20], polynomial angular model [9]. The initial guesses are a mix of manually fixed intrinsic parameters and computed ones using similar ideas than [17].

Compared to prior works [17] and [18], this new article provides the following contributions:

- a generalization of the camera model conversion methods explored in [17] that can handle input and output models that are not equivalent;
- two entirely new camera model conversions to the OpenCV Fisheye Model from the Rational Polynomial distortions model and from the Cartesian Polynomial Model;
- a thorough evaluation of four conversion processes themselves with both the approximation error and undistortion quality criteria;
- new applications to visual servoing, visual odometry and more applications to SLAM.

The validation of our theoretical developments on real camera data for several visual tasks in robotics demonstrate their effectiveness in practical scenarios.

The rest of the paper is organized as follows. Section 2 presents the related work on camera models and model conversion in the state-of-the-art. Section 3 details the general framework and the various camera model conversions considered in this work. Section 4 reports the evaluation protocol and the results obtained for four model conversions. Then, Section 5 explores the application of these conversions in different visual algorithms, such as visual servoing, visual odometry, and fisheye SLAM, for robot arm and mobile robot, before discussing the limitations (Sec. 6) and conclusion (Sec. 7).

2. Related Work

This section reviews the camera models (Sec. 2.1) used in existing camera model conversions methods that are recalled in Section 2.2. Section 2.1 also recalls the OFM that is used for the first time in a linear conversion in this article (Sec. 3).

2.1. Camera Models

2.1.1. Pinhole model

The pinhole model [8] is an ideal camera model based on perspective projection, where a 3D point $\mathbf{X} = (X, Y, Z)^\top \in \mathbb{R}^3$ in the camera frame is projected onto the normalized image plane using:

$$x = \frac{X}{Z} \quad \text{and} \quad y = \frac{Y}{Z}, \tag{1}$$

where $\mathbf{x} = (x, y)^\top \in \mathbb{R}^2$ are the normalized image coordinates of the projected point. Then, the pixel coordinates $(u, v)^\top \in \mathbb{R}^2$ are obtained after a scale change involving the camera focal length $f \in \mathbb{R}_+$ and the actual pixel size on the sensor (width $K_u \in \mathbb{R}_+$ and height $K_v \in \mathbb{R}_+$) and a coordinate system shift using the principal point coordinates in pixels $(u_0, v_0)^\top \in \mathbb{R}^2$:

$$u = \frac{f}{K_u} x + u_0 \quad \text{and} \quad v = \frac{f}{K_v} y + v_0. \tag{2}$$

The pinhole model assumes a perfect, distortion-free projection and serves as the foundation for many camera models. However, real cameras exhibit optical distortions that the pinhole model does not account for. To address this, models introduce distortion parameters to better represent real lens characteristics by applying a correction between the projected coordinates and the source 3D point line-of-sight.

2.1.2. CAHVOR model

Among the camera models employed in model conversions, the CAHVOR model [10] used in the machine vision community extends the distortion-free Center Axis Horizontal Vertical vectors (CAHV) framework [21], leveraging the same perspective relationships than (1) but expressing 3D coordinates in the world frame from 2D coordinates in the digital image plane, by incorporating radial and tangential distortion terms. This model defines the camera geometry using six vectors: $\mathbf{C}, \mathbf{A}, \mathbf{H}, \mathbf{V}, \mathbf{O}, \mathbf{R}$, each one belonging to $\mathbb{R}^3$, where $\mathbf{C}$, is the camera center, $\mathbf{A}$ is the optical axis direction, $\mathbf{H}$ and $\mathbf{V}$ are the image plane basis vectors, $\mathbf{O}$ is the offset vector, and $\mathbf{R}$ is the radial distortion vector. Unlike the pinhole model, CAHVOR accounts for lens distortions by introducing the radial distortion coefficients $\mathbf{R} = (r_0, r_1, r_2)^\top$ of a second-order polynomial, where r_0 represents the primary radial distortion term, while r_1 and r_2 introduce higher-order terms. Rewritten in the 3D camera coordinate system, CAHVOR's distortion model expresses 3D point $\mathbf{X}$ from the 3D point $\mathbf{X}_d = (X_d, Y_d, Z_d)^\top \in \mathbb{R}^3$ computed from point coordinates $\mathbf{u}_d = (u_d, v_d)^\top$ in the digital image plane suffering distortions, using the inverse of (2) and (1) (Z_d known), as:

$$\mathbf{X} = \mathbf{X}_d + \left(r_0 + r_1 \left(\frac{X_d^2 + Y_d^2}{Z_d^2} \right)^2 + r_2 \left(\frac{X_d^2 + Y_d^2}{Z_d^2} \right)^4 \right) [X_d, Y_d, 0]^\top. \tag{3}$$

2.1.3. Photogrammetric model

On the other hand, the photogrammetric model [22], widely used in aerial and terrestrial imaging, describes the imaging process with (1) but with a polynomial radial distortion term with coefficients $k_{G_0}, k_{G_1}, k_{G_2}$, each one belonging to $\mathbb{R}$. Unlike CAHVOR, it directly expresses distortion corrections to digital image plane coordinates $\mathbf{u}_d$, using $\mathbf{u}'_d = (u'_d, v'_d)^\top = (u_d - u_0, v_d - v_0)^\top$ and $\rho_{\mathbf{x}_{m_d}} = \sqrt{(K_u u'_d)^2 + (K_v v'_d)^2}$ ($\mathbf{x}_{m_d}$ for the coordinates of a point in the distorted metric image plane as in [10]), to express undistorted coordinates:

$$\mathbf{u} = \mathbf{u}_d + \frac{k_{G_0}\rho_{\mathbf{x}_{m_d}} + k_{G_1}\rho^3_{\mathbf{x}_{m_d}} + k_{G_2}\rho^5_{\mathbf{x}_{m_d}}}{\rho_{\mathbf{x}_{m_d}}}\mathbf{u}'_d. \tag{4}$$

2.1.4. RPM: Rational Polynomial Model

CAHVOR and photogrammetric model formulations ensure accurate modeling for cameras with moderate FoV, typically up to 100°, beyond which more advanced radial distortion models, such as RPM [23], are required.

The RPM [23] considers distortions applied to a 2D point $\mathbf{x}$ of norm $\rho_{\mathbf{x}} = \sqrt{x^2 + y^2}$ in the normalized image plane, to obtain its distorted counterpart $\mathbf{x}_R = (x_R, y_R)^\top \in \mathbb{R}^2$:

$$\begin{cases} x_R = d_R(\rho_{\mathbf{x}})x + 2p_1xy + p_2(\rho^2_{\mathbf{x}} + 2x^2) \\ y_R = d_R(\rho_{\mathbf{x}})y + p_1(\rho^2_{\mathbf{x}} + 2y^2) + 2p_2xy \end{cases}. \tag{5}$$

The radial distortion coefficient $d_R(\rho_{\mathbf{x}}) \in \mathbb{R}$ in (5) is represented as a polynomial in terms of $\rho_{\mathbf{x}}$, extending the distortion models to a rational polynomial form:

$$d_R(\rho_{\mathbf{x}}) = \frac{1 + k_{R_1}\rho^2_{\mathbf{x}} + k_{R_2}\rho^4_{\mathbf{x}} + \cdots + k_{R_{N'_R}}\rho^{2N'_R}_{\mathbf{x}}}{1 + k_{R_{N'_R+1}}\rho^2_{\mathbf{x}} + k_{R_{N'_R+2}}\rho^4_{\mathbf{x}} + \ldots k_{R_{2N'_R}}\rho^{2N'_R}_{\mathbf{x}}},$$

with $N'_R \in \mathbb{N}$ being the number of first terms of each polynomial (in practice, N'_R is often equal to 3, that we follow in this article). The expression for $\mathbf{x}_R$ in (5) also involves tangential distortion parameters $p_1 \in \mathbb{R}$ and $p_2 \in \mathbb{R}$. One gets the digital image plan coordinates $\mathbf{u}_R = (u_R, v_R)^\top \in \mathbb{R}^2$ with the principal point and scale factors $\beta_u \in \mathbb{R}_+ \setminus \{0\}$ and $\beta_v \in \mathbb{R}_+ \setminus \{0\}$:

$$\begin{cases} u_R = \beta_u x_R + u_0 \\ v_R = \beta_v y_R + v_0 \end{cases}. \tag{6}$$

2.1.5. EFM: Equidistant Fisheye model

For fisheye cameras closer and around FoV of 180°, the EFM [6] considers an ideal constant angular increment per pixel, which is particularly useful for panoramic imaging and robotic vision [9]. This model assumes symmetric radial distortions in the image and implements a regular radial resolution of polar angle $\phi \in \mathbb{R}$, between the 3D point $\mathbf{X}$ and the camera optical axis $\mathbf{Z}_c \in \mathbb{R}^3$. Writing $\Psi \in \mathbb{R}$ the azimuth angle and setting $\rho_{\mathbf{X}} = \sqrt{X^2 + Y^2 + Z^2}$, these angles can be expressed from Cartesian coordinates $\phi = \arccos(Z/\rho_{\mathbf{X}})$ and $\Psi = \arctan(Y/X)$. This model maps the azimuth and elevation angles to normalized image plane coordinates by: $x_E = \phi\cos(\Psi)$ and $y_E = \phi\sin(\Psi)$. With $f \in \mathbb{R}_+ \setminus \{0\}$ the focal length of the fisheye lens and $n \in \mathbb{R}_+$ the pixel pitch (square pixels assumed), the point $\mathbf{u}_E = [u_E, v_E]^\top \in \mathbb{R}^2$ coordinates in the digital image are:

$$u_E = \frac{f}{n}x_E + u_0 \quad \text{and} \quad v_E = \frac{f}{n}y_E + v_0. \tag{7}$$

2.1.6. OFM: OpenCV's Fisheye Model

For the fisheye lenses not really implementing the EFM, OFM is specifically designed to handle the radial distortion pattern typical of fisheye lenses. In this model, the coordinates (u_F, v_F) in the digital image plane are obtained by scaling the distorted coordinates (x_F, y_F) with the magnification factors $f_x \in \mathbb{R} \setminus \{0\}$ and $f_y \in \mathbb{R} \setminus \{0\}$, then offset by the principal point (u_0, v_0):

$$\begin{cases} u_F = f_x x_F + u_0 \\ v_F = f_y y_F + v_0 \end{cases}. \tag{8}$$

The distorted coordinates (x_F, y_F) are derived from the pinhole normalized image plane coordinates $\mathbf{x}$ by introducing a radial factor $d_F(\theta)$ that accounts for the fisheye distortion. Recalling $\rho_{\mathbf{x}}$ is the norm of $\mathbf{x}$, the distorted coordinates are then given with:

$$\theta = \arctan(\rho_{\mathbf{x}}),$$

by:

$$\begin{cases} x_F = \frac{d_F(\theta)}{\rho_{\mathbf{x}}}x \\ y_F = \frac{d_F(\theta)}{\rho_{\mathbf{x}}}y \end{cases}, \tag{9}$$

where $d_F(\theta)$ is expressed with distortion coefficients $k_{F_j} \in \mathbb{R}, j \in \{1, 2, 3, 4\}$ that are specific to the fisheye lens calibration as:

$$d_F(\theta) = \theta + k_{F_1}\theta^3 + k_{F_2}\theta^5 + \cdots + k_{F_{N'_F}}\theta^{2N'_F+1},$$

with $N'_F \in \mathbb{N}$ being the number of first terms of the polynomial (in practice, N'_F rarely exceeds 4).

2.1.7. CPM: Cartesian Polynomial Model

Alternatively, the CPM [15] was designed for fisheye and catadioptric lens to simultaneously account for both mirror geometry and optical distortions. It models the line of sight from digital image plane coordinates $\mathbf{u}_C \in \mathbb{R}^2$ centered at the principal point $\mathbf{u_C}' = \mathbf{u_C} - [u_0, v_0]^\top$ to the 3D point $\mathbf{X}$ (which norm writes $\rho_\mathbf{X}$) and with $\rho_C = ||\mathbf{u_C}'||$ as:

$$\rho_\mathbf{X}[u'_C, v'_C, d_C(\rho_C)] = \mathbf{X}, \tag{10}$$

where the radial distortion is modeled by the fourth order polynomial function defined like:

$$d_C(\rho_C) = k_{C_0} + k_{C_2}\rho_C^2 + k_{C_3}\rho_C^3 + \cdots + k_{C_{N'_C}}\rho_C^{N'_C}, \tag{11}$$

with $N'_C \in \mathbb{N}$ being the number of first terms of the polynomial, ignoring the term in ρ_C (in practice, N'_C is often in the range from 2 to 4).

2.1.8. UCM: Unified Central Model

However, for certain fisheye lenses and mirror shapes, fewer parameters are required, as demonstrated by the UCM [13][14] for single-viewpoint cameras. Considering intrinsic parameters $\alpha_u \in \mathbb{R} \setminus \{0\}$, $\alpha_v \in \mathbb{R} \setminus \{0\}$ as the generalized focal length, $u_0 \in \mathbb{R}$, $v_0 \in \mathbb{R}$ as the principal point coordinates and $\xi \in \mathbb{R}$ a parameter associated to the lens (or mirror shape), the UCM projects a 3D point $\mathbf{X}$ to a digital image point $\mathbf{u}_U = [u_U, v_U]^\top \in \mathbb{R}^2$ with three steps. First, $\mathbf{X}$ is projected as $\mathbf{X}_S = [X_S, Y_S, Z_S]^\top \in \mathbb{R}^3$ on a unit sphere ($||\mathbf{X}_S|| = 1$) centered at the camera origin such that:

$$X_S = \frac{X}{\rho_\mathbf{X}} \quad , \quad Y_S = \frac{Y}{\rho_\mathbf{X}} \quad \text{and} \quad Z_S = \frac{Z}{\rho_\mathbf{X}}. \tag{12}$$

Second, $\mathbf{X}_S$ is projected as $\mathbf{x}_U = [x_U, y_U]^\top \in \mathbb{R}^2$ on the normalized image plane thanks to a second projection center distant of ξ from the sphere center as $x_U = X_S/(Z_S + \xi)$ and $y_U = Y_S/(Z_S + \xi)$. Third, $\mathbf{x}$ is transformed to the digital image plane as $\mathbf{u}_U$ with:

$$u_U = \alpha_u x_U + u_0 \quad \text{and} \quad v_U = \alpha_v y_U + v_0. \tag{13}$$

Table 1 summarizes the notations for the camera models UCM, EFM, CPM, RPM, and OFM.

Table 1: Summary of the notations for the various camera models considered in this article. From the left to right column: **Model** shows acronyms of the camera models and between parentheses the article section to find their details; $\boldsymbol{\rho}$ specifies the way a distance $\boldsymbol{\rho}$ found in every model but computed differently or from different inputs is noted; **Parameters** lists the intrinsic parameters per model; $\boldsymbol{x}, \boldsymbol{y}$ shows the way to note point coordinates expressed in the normalized image plane per model; $\boldsymbol{u}, \boldsymbol{v}$ is the equivalent but in the digital image plane; $\boldsymbol{d}$ shows the various ways to note the radial distorsion factor, when existing, per model.

Model	$\boldsymbol{\rho}$	**Parameters**	$\boldsymbol{x}, \boldsymbol{y}$	$\boldsymbol{u}, \boldsymbol{v}$	$\boldsymbol{d}$
RPM (2.1.4)	$\rho_{\mathbf{x}}$	$\beta_u, \beta_v, k_{R_1}, k_{R_2}, \ldots, p_1, p_2$	x_R, y_R	u_R, v_R	$d_R(\rho_{\mathbf{x}})$
CPM (2.1.7)	ρ_C	$k_{C_0}, k_{C_2}, \ldots \ (k_{C_1} = 0)$	x_C, y_C	u_C, v_C	$d_C(\rho_C)$
OFM (2.1.6)	$\rho_{\mathbf{x}}$	$f_x, f_y, k_{F_1}, k_{F_2}, \ldots$	x_F, y_F	u_F, v_F	$d_F(\theta)$
UCM (2.1.8)	ρ_U	α_u, α_v, ξ	x_U, y_U	u_U, v_U	–
EFM (2.1.5)	ρ_E	f, n	x_E, y_E	u_E, v_E	–

2.2. Conversion Between Camera Models

The conversion between different camera projection models remains a relatively underexplored area of research.

One of the pioneering contributions in this field focused on remote sensing applications [10], proposing a method to convert the CAHVOR model (Sec. 2.1) into the photogrammetric model [22]. This conversion consists of expressing CAHVOR's parameters in the target model's formalism. Specifically, the focal length f is derived from the horizontal and vertical components of vectors $\mathbf{H}$ and $\mathbf{V}$, while the principal point (u_0, v_0) is obtained from their respective coordinates. For the distortion models, one may note the similar models in (3) and (4) from which, after few manipulations, [10] could identify the distortion coefficients k_{G_j} of the photogrammetric model with those of the CAHVOR model as:

$$k_{G_0} = r_0, \quad k_{G_1} = \frac{r_1}{f^2} \quad \text{and} \quad k_{G_2} = \frac{r_2}{f^4}. \tag{14}$$

This identification is possible because the radial distortion models in CAHVOR and the photogrammetric model are equivalent, despite the latter is applied

to images coordinates whereas the former is applied to 3D coordinates, allowing for a direct conversion of all their parameters.

When two camera models are not equivalent, converting one to the other cannot be done with direct identification of the parameters. This is the case of camera models used in a broad range of robotic applications. Recent research [17] has thus elaborated new projection model conversions as linear conversion systems involving virtual point coordinates for several projection and distortion models: Equidistant Fisheye to Unified Central model conversion (EF2UC), Unified Central to Cartesian Polynomial model conversion (UC2CP), and Rational Polynomial radial distortion to Polynomial radial distortion model conversion (RP2P). Since, building on this foundation, the present work further explores the use of the former two in visual servoing and visual odometry (Sec. 5), we begin by recalling the details of EF2UC and UC2CP.

EF2UC focuses on coordinates along the horizontal axis ($\Psi = 0$), exploiting the radial nature of both EFM and UCM. UCM's intrinsics α_u and ξ are obtained by solving:

$$\begin{bmatrix} \vdots & \vdots \\ \frac{\text{sinc}(\phi_i)}{f/n} & -1 \\ \vdots & \vdots \end{bmatrix} \begin{bmatrix} \alpha_u \\ \xi \end{bmatrix} = \begin{bmatrix} \vdots \\ \cos(\phi_i) \\ \vdots \end{bmatrix}, \tag{15}$$

where the $\phi_i, i \in [1, N]$ are $N \in \mathbb{N}$ polar angles sampling the camera FoV (with a 1-degree step according to [17]). Given a set of points $\left[X_{S_i}, Y_{S_i}, Z_{S_i}\right]^\top$, Equation (15) forms an overdetermined linear system when more than two points are available. It is solved using the matrix inverse for the minimal set of points, or the pseudo-inverse for larger sets, ensuring robustness in the estimation of the polynomial coefficients in the least-squares sense.

CPM is also a radial model so UC2CP is obtained by simplifying (10) with $Y = 0$ again and substituting u'_C with the expression of $u'_U = u_U - u_0$ (12), leading to the following linear matricial equation:

$$\begin{bmatrix} \vdots & \vdots \\ 1 & \left(\alpha_u \frac{X_{S_i}}{Z_{S_i}+\xi}\right)^2 \\ \vdots & \vdots \end{bmatrix} \begin{bmatrix} k_{C_0} \\ k_{C_2} \end{bmatrix} = \begin{bmatrix} \vdots \\ \alpha_u \frac{Z_{S_i}}{Z_{S_i}+\xi} \\ \vdots \end{bmatrix}. \tag{16}$$

That allows to solve for the parameters of the CPM at order 2. The coordinates X_{S_i} and Z_{S_i} are computed for each ϕ_i based on the polar to Cartesian

coordinates conversion:

$$X_{S_i} = \sin(\phi_i) \text{ and } Z_{S_i} = \cos(\phi_i). \tag{17}$$

As for the previous conversion it is solved using the matrix inverse for the minimal set of points, or the matrix pseudo-inverse for more points.

Inspired by the approaches reported in (15) and (16), the next Section 3 generalizes them and develops new conversions of camera models, carefully evaluated in Section 4 and applied to various robotic vision tasks in Section 5.

3. CONVERSIONS

This section generalizes the conversion methods of [17] to compute parameters of one non-linear camera model into another (Sec. 3.1) by solving a linear system inspired by (15) and (16). This conversion method is then developed for new conversions as, first, Cartesian Polynomial to OpenCV's Fisheye model conversion (CP2OF) (Sec. 3.4.1) to compute OFM's parameters (Sec. 2.1), from the CPM's ones. Second, we develop the Rational Polynomial radial distortion (Sec. 2.1) to OpenCV's Fisheye model conversion (RP2OF) in Section 3.4.2.

3.1. General camera models conversion method

This section describes the general approach for computing the parameters of a given non-linear camera model into another target model without requiring a new calibration stage.

First of all, not all conversions would make sense. For instance, converting the CPM accurately describing a given camera which FoV exceeds 180 degrees to the pinhole model would obviously lead to very large modeling errors. Hence, we develop the conversion method for which pixel coordinates of the observation of a 3D point remain approximately identical between the two models, despite the latter are not equivalent. Considering each model characterized by the parameter sets μ and γ, with respectively $N_\mu \in \mathbb{N}$ and $N_\gamma \in \mathbb{N}$ parameters (μ and γ among U,E,C,R and F at least, Tab. 1), we have thus, ideally:

$$(u_\mu, v_\mu)^\top = (u_\gamma, v_\gamma)^\top. \tag{18}$$

After that, the parameters of the output model are structured in a linear matrix equation:

$$\mathbf{D} \cdot \boldsymbol{\gamma} = \mathbf{b}, \tag{19}$$

where $\mathbf{D} \in \mathbb{R}^{N \times N_\gamma}$ is a coefficient matrix, $\boldsymbol{\gamma} \in \mathbb{R}^{N_\gamma}$ is the vector of target model parameters to be estimated, and $\mathbf{b} \in \mathbb{R}^N$ is a vector depending on points coordinates or their lines of sight. Here, N_γ represents the number of parameters in the output model, while N is the number of points defined to compute the elements of $\mathbf{D}$ and $\mathbf{b}$.

To define the latter points, we generalize the method of [17] to compute points that are not necessarily on the horizontal axis of the camera coordinate system (ψ in Sec. 2.1.5 not always equal to zero, contrary to the EF2UC conversion and the following ones reported in Sec. 2.2) in order to deal with camera model conversions that are not only radial. Hence, the N points are defined from a discrete set of lines of sight spanning the camera FoV which angle is noted $\omega \in [0, 2\pi]$ on one axis in the camera coordinate system that is a rotation of angle $\delta \in [-\pi, \pi]$ of the camera horizontal axis $\mathbf{X_c}$ around the camera optical axis $\mathbf{Z_c}$. Thus, considering polar angles $\phi_i \in \Omega \subset [-\omega/2, \omega/2]$, the points are expressed in 3D in the camera coordinate system as:

$$X_i = \cos(\delta)\sin(\phi_i)\ ,\ Y_i = \sin(\delta)\sin(\phi_i) \text{ and } Z_i = \cos(\phi_i). \tag{20}$$

Then, considering an angular step $\Delta \in \mathbb{R}_+ \backslash \{0\}$, we define the set $\Omega = \{-\frac{\omega}{2}, -\frac{\omega}{2} + \Delta, \ldots, \frac{\omega}{2} - \Delta, \frac{\omega}{2}\} \setminus \{0\}$. In this work, we adopt the configuration ($\Delta = 1°, \delta = 0$), following the protocol introduced in [17]. This setting is used for all conversions in the paper, unless explicitly stated otherwise.

To solve (19), we use the matrix inverse if $N = N_\gamma$ and $\mathbf{D}$ is full-rank. When $N > N_\gamma$, we compute the Moore-Penrose pseudo-inverse, denoted $\mathbf{D}^\dagger$:

$$\boldsymbol{\gamma} = \mathbf{D}^\dagger \mathbf{b}. \tag{21}$$

The pseudo-inverse provides a least-squares estimation of $\boldsymbol{\gamma}$ in the overdetermined case. This method generalizes the specific conversions EF2UC, UC2CP and RP2P introduced in [17].

To evaluate the geometric error made in such conversion, we compute the approximation errors thanks to the 3D points defined for the conversion (20) using the Euclidean distance between their projections $(u_{\mu_i}, v_{\mu_i})^\top$ in the image plane (18) with the source model μ and those $(\hat{u}_{\gamma_i}, \hat{v}_{\gamma_i})^\top$ with the output model γ (the $\hat{\cdot}$ denoting the projection of 3D point $(X_i, Y_i, Z_i)^\top$ (20) thanks to the converted parameters). Thus, the approximation error vector $\mathbf{E}_\gamma \in \mathbb{R}_+^N$

is defined as follows:

$$\mathbf{E}_\gamma = \begin{bmatrix} \cdots \\ \sqrt{(u_{\mu_i} - \hat{u}_{\gamma_i})^2 + (v_{\mu_i} - \hat{v}_{\gamma_i})^2} \\ \cdots \end{bmatrix}, \tag{22}$$

and the average approximation error $e_\gamma \in \mathbb{R}_+$ is:

$$e_\gamma = \frac{1}{N} \sum_{i \in [1,2,\ldots,N]} \sqrt{(u_{\mu_i} - \hat{u}_{\gamma_i})^2 + (v_{\mu_i} - \hat{v}_{\gamma_i})^2}. \tag{23}$$

Finally, since to solve (21) we need the full rank of the system, in Section 3.2.1, we develop the linear independence study for the EF2UC in addition to developing an estimation of the converted intrinsic parameters' uncertainty. Then, we generalize UC2CP to polynomials of any degree for the source UCM. After that, Section 3.4, will thoroughly develop this general method to two completely new conversions to the OFM.

3.2. Linear independence in EF2UC and converted intrinsic parameters uncertainty

3.2.1. Linear independence in EF2UC

To analyze under which conditions the EF2UC camera conversion is solvable, a minimum of two rows in (15) must be linearly independent since there are two unknowns in the linear matrix equation (15). So, one must express the conditions under which there exist two different vectors:

$$\mathbf{V}_i = \left[\mathrm{sinc}(\phi_i)\frac{n}{f}, -1, -\cos(\phi_i)\right] \text{ and } \mathbf{V}_j = \left[\mathrm{sinc}(\phi_j)\frac{n}{f}, -1, -\cos(\phi_j)\right], \tag{24}$$

such that we cannot find any real $a \in \mathbb{R}$ leading to:

$$\mathbf{V}_j - a\mathbf{V}_i = 0, \tag{25}$$

where we identify easily $a = 1$ implying that $\mathrm{sinc}(\phi_j)n/f = \mathrm{sinc}(\phi_i)n/f$ and $\cos(\phi_j) = \cos(\phi_i)$. Since polar angle $\phi_i \neq \phi_j$, the two previous equations in sinc and cos, that are even functions, are only true for $\phi_j = -\phi_i$.

So, for EF2UC, generating the point set for the conversion must follow, for a minimum of one pair of polar angles: $|\phi_j| \neq |\phi_i|$.

This theoretical result confirms that generating a point per degree sampling the whole FoV (except $\phi \neq 0$) to solve for the EF2UC conversion [17] ensures linear independence of the minimum (or more) rows in (15).

3.2.2. EF2UC converted intrinsic parameters uncertainty

Among priors works on camera calibration, [2] shipped with an Omnidirectional Calibration Toolbox for Matlab, provided a way to compute the calibration uncertainty of UCM's intrinsic parameters. In a few words, they considered the standard deviation of point reprojection errors (from Euclidean distances between chessboard detected corners and the projection of their theoretical coordinates to the digital image plane using the calibrated parameters) to be propagated to the intrinsic (and possibly extrinsic) parameters.

In our case, we can propagate to the converted intrinsic parameters the standard deviation $\sigma_U \in \mathbb{R}_+$ of the elements in the conversion approximation error vector (22). Following [2] that leveraged the Jacobian of UCM's projection function but focusing on the parameters of the EF2UC conversion (15), *i.e.* α_u and ξ, we need to compute the following matrix (using our notations and only for the horizontal coordinate in the image):

$$\mathbf{J} = \sum_{i \in [1,2,\dots,N]} \begin{bmatrix} \frac{\partial \hat{u}_{U_i}}{\partial \alpha_u} & \frac{\partial \hat{u}_{U_i}}{\partial \xi} \end{bmatrix}^\top \begin{bmatrix} \frac{\partial \hat{u}_{U_i}}{\partial \alpha_u} & \frac{\partial \hat{u}_{U_i}}{\partial \xi} \end{bmatrix}, \tag{26}$$

for which the $\hat{u}_{U_i}$ are those point coordinates computed from the polar angles ϕ_i (defined for the conversion) using the converted parameters. And then, the uncertainties $\hat{\sigma}_{\alpha_u}$ on α_u and $\hat{\sigma}_\xi$ on ξ are obtained following the same seminal work as follows:

$$\begin{bmatrix} \hat{\sigma}_{\alpha_u} & \hat{\sigma}_\xi \end{bmatrix} = \sqrt{\mathrm{diag}(\mathbf{J}^{-1})}\sigma_U, \tag{27}$$

where the $\mathrm{diag}(\cdot)$ operator extracts the diagonal of a matrix as a row vector on the elements of which the $\sqrt{\cdot}$ operator is applied. [2] then highlights the uncertainties on intrinsic parameters with three times their estimated standard deviation, *i.e.* in our case $3\hat{\sigma}_{\alpha_u}$ and $3\hat{\sigma}_\xi$.

3.3. Conversion of the UCM to the Cartesian Polynomial Model of any order (generalized UC2CP)

3.3.1. Generalization of UC2CP

The original UC2CP in matrix equation (16) is a rewriting of the CPM of order 2, fed with point coordinates expressed with the UCM [17]. To consider higher order terms for the CPM in the UC2CP conversion, we use (11) for extending the equation (23) of [17] to:

$$k_{C_0} + |u'_U|^2 k_{C_2} + \cdots + |u'_U|^{N'_C} k_{C_{N'_C}} \quad = \quad \alpha_u \frac{Z_S}{Z_S + \xi} \ ,$$

leading to:

$$\begin{bmatrix} \vdots & \vdots & \vdots & \vdots \\ 1 & \left(\alpha_u \frac{X_{S_i}}{Z_{S_i}+\xi}\right)^2 & \dots & \left(\alpha_u \frac{X_{S_i}}{Z_{S_i}+\xi}\right)^{N'_C} \\ \vdots & \vdots & \vdots & \vdots \end{bmatrix} \begin{bmatrix} k_{C_0} \\ k_{C_2} \\ \vdots \\ k_{C_{N'_C}} \end{bmatrix} = \begin{bmatrix} \vdots \\ \alpha_u \frac{Z_{S_i}}{Z_{S_i}+\xi} \\ \vdots \end{bmatrix}, \tag{28}$$

instead of (16), that can be solved with the matrix inverse or pseudo-inverse if $N > N'_C$ and where coordinates X_{S_i} and Z_{S_i} are computed as described in (17).

3.3.2. Linear independence in solving for UC2CP

To ensure there is at least the minimum number of independent rows in (28) to solve the UC2CP conversion, we develop the case of $N'_C = 2$ (for conciseness) to express the conditions under which there exist two different vectors, noting $Z_{\xi_i} = Z_{S_i} + \xi$ and $Z_{\xi_j} = Z_{S_j} + \xi$:

$$\mathbf{V}_i = \left[1, \left(\alpha_u \frac{X_{S_i}}{Z_{\xi_i}}\right)^2, -\alpha_u \frac{Z_{S_i}}{Z_{\xi_i}}\right] \text{ and } \mathbf{V}_j = \left[1, \left(\alpha_u \frac{X_{S_j}}{Z_{\xi_j}}\right)^2, -\alpha_u \frac{Z_{S_j}}{Z_{\xi_j}}\right], \tag{29}$$

hence $Z_{S_i} \neq -\xi, \forall i$, such that we cannot find any real $a \in \mathbb{R}$ leading to verify (25). Substituting (29) in (25), we identify easily $a = -1$ implying that $X_{S_i} = 0$ and $Z_{S_j} = Z_{S_i}$.

So, for UC2CP with $N'_C = 2$, generating the point set for the conversion must follow $X_{S_i} \neq 0$ and $Z_{S_i} \neq -\xi, \forall i$ and for a minimum of one pair of points $Z_{S_j} \neq Z_{S_i}$.

This theoretical result confirms that generating a point per degree sampling the whole FoV (except $X_{S_i} = 0$ and $Z_{S_i} = -\xi$) to solve for the UC2CP conversion with $N'_C = 2$ [17] ensures linear independence of the minimum (or more) rows in (28). The conditions for cases where $N'_C > 2$ are left to the reader.

3.4. Conversions to OFM

In this section, we develop two new conversions to the OFM: in Section 3.4.1, the conversion from the CPM to the OFM and in Section 3.4.2, the conversion from the RPM to the OFM.

3.4.1. Cartesian Polynomial to OpenCV's Fisheye model conversion (CP2OF)

Since both CPM and OFM are radial, the conversion can be derived with $Y = 0$, thus $\rho_{\mathbf{x}}$ in (9) simplifies to:

$$\rho_x = \sqrt{x^2} = |x|. \tag{30}$$

Furthermore, setting $u'_F = u_F - u_0$ simplifies the OFM (8) to:

$$u'_F = f_x x_F = f_x \frac{d_F(\theta_x)}{\rho_x} x, \tag{31}$$

with:

$$\theta_x = \arctan(\rho_x) \geq 0. \tag{32}$$

For the conversion between CPM and OFM, they should yield the same image coordinates for the same 3D point to ensure consistency across models, thus we impose that $u_C = u_F$. From the latter equality the principal point coordinates are directly identified, so the condition simplifies to $u'_C = u'_F$, allowing us to rewrite (31) as:

$$u'_C = f_x \frac{x}{|x|}(\theta_x + k_{F_1}\theta_x^3 + k_{F_2}\theta_x^5 + \cdots + k_{F_{N'_F}}\theta_x^{2N'_F+1}). \tag{33}$$

Setting $f'_x = \frac{1}{f_x}$ and rearranging the terms, we obtain:

$$-u'_C \frac{|x|}{x} f'_x + \theta_x^3 k_{F_1} + \theta_x^5 k_{F_2} + \cdots + k_{F_{N'_F}}\theta_x^{2N'_F+1} = -\theta_x. \tag{34}$$

We rewrite the above equation with matrices for $i \in [1, N] \subset \mathbb{N}$ as:

$$\begin{bmatrix} \vdots & \vdots & \vdots & \vdots & \vdots \\ -u'_{C_i}\frac{|x_i|}{x_i} & \theta_{x_i}^3 & \theta_{x_i}^5 & \dots & \theta_{x_i}^{2N'_F+1} \\ \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix} \begin{bmatrix} f'_x \\ k_{F_1} \\ k_{F_2} \\ \vdots \\ k_{F_{N'_F}} \end{bmatrix} = \begin{bmatrix} \vdots \\ -\theta_{x_i} \\ \vdots \end{bmatrix}, \tag{35}$$

which can be solved with the matricial inverse, or pseudo-inverse if $N > N'_F + 1$. In (35), the x_i coordinates are computed from X_i and Z_i as defined in (20) ($Y_i = 0$), substituted in the perspective projection equations (1). However, contrary to all the other conversions, the generation of the set Ω of line-of-sight polar angles ϕ_i is different. This is due to the fact that the

CPM is an inverse model, unprojecting digital image plane coordinates to 3D lines of sight (10). Indeed, contrary to generating Ω from a regular angular sampling of the camera field-of-view, the CP2OF conversion samples first the image plane on the horizontal line of width $W \in \mathbb{N}$ around the principal point, *i.e.* input coordinates $u'_{C_i} \in \mathcal{U} = [-W/2, -W/2+1, \ldots, W/2-1, W/2] \subset \mathbb{Z}$, leading to:

$$\left\{\begin{array}{rcl} \rho_{\mathbf{x}_i} & = & 1 \\ X_i & = & u'_{C_i} \\ Z_i & = & k_{C_0} + k_{C_2}|u'_{C_i}|^2 + k_{C_3}|u'_{C_i}|^3 + \cdots + k_{C_{N'_C}}|u'_{C_i}|^{N'_C} \\ x_i & = & \frac{X_i}{Z_i} \\ \phi_i & = & \arctan(x_i). \end{array}\right. \tag{36}$$

In (36), the expression of ϕ_i is only relevant for the theoretical expression of Ω (Sec. 3.1) for the CP2OF conversion because x_i that is used in all terms of (35) (except u'_{C_i}) is already expressed. Nonetheless, as a side fact, Ω is hence not regularly sampling the camera FoV for CP2OF.

Then, to analyze under which conditions the CP2OF conversion is solvable, a minimum of $N'_F + 1$ rows in (35) must be linearly independent since there are $N'_F + 1$ unknowns in the linear matrix equation (35). Focusing on the case of $N'_F = 1$ for the sake of conciseness, one must hence express the conditions under which there exist two different vectors, *i.e.* $u'_{C_i} \neq u'_{C_j}$:

$$\mathbf{V}_i = \left[-u'_{C_i}\frac{|x_i|}{x_i}, \theta^3_{x_i}, \theta_{x_i}\right] \text{ and } \mathbf{V}_j = \left[-u'_{C_j}\frac{|x_j|}{x_j}, \theta^3_{x_j}, \theta_{x_j}\right], \tag{37}$$

hence coordinate $x_i \neq x_j \neq 0$, such that we cannot find any real $a \in \mathbb{R}\backslash\{0\}$ leading to verify (25). Substituting (37) in (25), we identify easily that:

$$a = \theta_{x_j}/\theta_{x_i}, \tag{38}$$

which is possible since $x_i \neq 0 \Rightarrow \theta_{x_i} \neq 0$ (see (32)), implying that:

$$\begin{array}{rcl} \theta^3_{x_j} - \dfrac{\theta_{x_j}}{\theta_{x_i}}\theta^3_{x_i} = 0 \\ \Leftrightarrow \quad \theta_{x_j}(\theta^2_{x_j} - \theta^2_{x_i}) = 0. \end{array} \tag{39}$$

Since $\theta_{x_j} \neq 0 \Rightarrow \theta_{x_i} \neq 0$ (see (32)), and the other term rewrites as:

$$\begin{array}{rcl} \theta^2_{x_j} - \theta^2_{x_i} = 0 & \Leftrightarrow & \theta^2_{x_j} = \theta^2_{x_i} \\ & \Leftrightarrow & |\theta_{x_j}| = |\theta_{x_i}| \end{array} \tag{40}$$

which due to (30) and (32) making θ_x always positive, simplifies as:

$$\theta_{x_j} = \theta_{x_i}. \tag{41}$$

Due to (30), $u'_{C_j} = -u'_{C_i}$ verifies (41).

Finally, from (37) and (38) we develop the third and last term from (25):

$$-u'_{C_j}\frac{|x_j|}{x_j} + \frac{\theta_{x_j}}{\theta_{x_i}}u'_{C_i}\frac{|x_i|}{x_i} = 0. \tag{42}$$

Let us first develop the first term of (42):

$$-u'_{C_j}\frac{|x_j|}{x_j} = -u'_{C_j}\frac{\left|\frac{u'_{C_j}}{a_0+a_2|u'_{C_j}|^2+\ldots}\right|}{\frac{u'_{C_j}}{a_0+a_2|u'_{C_j}|^2+\ldots}} = -\left|\frac{u'_{C_j}}{a_0+a_2|u'_{C_j}|^2+\ldots}\right|(a_0+a_2|u'_{C_j}|^2+\ldots), \tag{43}$$

where there are two solutions:

- either
$$a_0 + a_2|u'_{C_j}|^2 + \cdots > 0 \Rightarrow -u'_{C_j}\frac{|x_j|}{x_j} = -|u'_{C_j}|; \tag{44}$$
- or
$$a_0 + a_2|u'_{C_j}|^2 + \cdots < 0 \Rightarrow -u'_{C_j}\frac{|x_j|}{x_j} = |u'_{C_j}|. \tag{45}$$

Since $\theta_{x_i} > 0$ and $\theta_{x_j} > 0$ (see (38) and (32)) and proceeding similarly for $u'_{C_i}\frac{|x_i|}{x_i}$ than for $u'_{C_j}\frac{|x_j|}{x_j}$, we deduce that (42) has four solutions that are:

$$|u'_{C_j}| + \frac{\theta_{x_j}}{\theta_{x_i}}|u'_{C_i}| = 0 \Leftrightarrow \frac{|u'_{C_j}|}{\theta_{x_j}} = -\frac{|u'_{C_i}|}{\theta_{x_i}}, \tag{46}$$

for the first one, which is impossible since the left side of the equation is always positive but the right side is always negative and $u'_{C_j} \neq u'_{C_i} \neq 0$. Then, the second solution is:

$$-|u'_{C_j}| - \frac{\theta_{x_j}}{\theta_{x_i}}|u'_{C_i}| = 0 \Leftrightarrow -\frac{|u'_{C_j}|}{\theta_{x_j}} = \frac{|u'_{C_i}|}{\theta_{x_i}}, \tag{47}$$

leading to the same conclusion of impossibility as the first solution. The third and fourth solutions are respectively to:

$$-|u'_{C_j}| + \frac{\theta_{x_j}}{\theta_{x_i}}|u'_{C_i}| = 0 \Leftrightarrow -\frac{|u'_{C_j}|}{\theta_{x_j}} = -\frac{|u'_{C_i}|}{\theta_{x_i}}, \tag{48}$$

and:

$$|u'_{C_j}| - \frac{\theta_{x_j}}{\theta_{x_i}}|u'_{C_i}| = 0 \Leftrightarrow \frac{|u'_{C_j}|}{\theta_{x_j}} = \frac{|u'_{C_i}|}{\theta_{x_i}}, \tag{49}$$

both only possible when $u'_{C_j} = -u'_{C_i}$ that was already verifying (41).

To summarize, any pair of source points $(u'_{C_i}, u'_{C_j}) \in \mathcal{U}^2$ such that $u'_{C_j} \neq -u'_{C_i}$ ensures there are at least two linearly independent rows in (35). This theoretical result shows that generating $\mathcal{U}$ with $W > 4$ such that $0 \notin \mathcal{U}$ to solve for the CP2OF is enough, hence exceeded by far when setting W to the width of the camera FoV in the digital image plane to proceed on a similar principle to Section 3.1 as we propose.

3.4.2. Rational Polynomial radial distortion to OpenCV's Fisheye model conversion (RP2OF)

The conversion from RPM to OFM follows the general conversion method introduced in Section 3.1 where the pixel coordinates of both models are made equal, *i.e.* $[u_R, v_R]^\top = [u_F, v_F]^\top$, which is detailed to:

$$\begin{cases} \beta_u x_R + u_0 = f_x x_F + u_0 \\ \beta_v y_R + v_0 = f_y y_F + v_0 \end{cases}. \tag{50}$$

The key challenge in this conversion lies in properly translating the distortion components from the RPM to the OFM, as each model represents them differently.

To make expressions more compact, we start by expressing the tangential distortion terms of the RPM (5) as follows:

$$t_x = 2p_1xy + p_2(\rho_{\mathbf{x}}^2 + 2x^2), \quad t_y = 2p_2xy + p_1(\rho_{\mathbf{x}}^2 + 2y^2). \tag{51}$$

Then, after identifying u_0 and v_0, (50) becomes in detail:

$$\begin{cases} \beta_u(d_R(\rho_{\mathbf{x}})x + t_x) = f_x\frac{x}{\rho_{\mathbf{x}}}(\theta + k_{F_1}\theta^3 + k_{F_2}\theta^5 + \cdots + k_{F_{N'_F}}\theta^{2N'_F+1}) \\ \beta_v(d_R(\rho_{\mathbf{x}})y + t_y) = f_y\frac{y}{\rho_{\mathbf{x}}}(\theta + k_{F_1}\theta^3 + k_{F_2}\theta^5 + \cdots + k_{F_{N'_F}}\theta^{2N'_F+1}) \end{cases}. \tag{52}$$

Posing $f'_x = \frac{1}{f_x}$ and $f'_y = \frac{1}{f_y}$, we re-arrange the system as:

$$\begin{cases} \frac{-\beta_u \rho_{\mathbf{x}}}{x}(d_R(\rho_{\mathbf{x}})x + t_x)f'_x + k_{F_1}\theta^3 + k_{F_2}\theta^5 + \cdots + k_{F_{N'_F}}\theta^{2N'_F+1} = -\theta \\ \frac{-\beta_v \rho_{\mathbf{x}}}{y}(d_R(\rho_{\mathbf{x}})y + t_y)f'_y + k_{F_1}\theta^3 + k_{F_2}\theta^5 + \cdots + k_{F_{N'_F}}\theta^{2N'_F+1} = -\theta \end{cases}. \tag{53}$$

Then, posing:

$$A_x = -\beta_u \rho_{\mathbf{x}} \left(d_R(\rho_{\mathbf{x}}) + \frac{t_x}{x} \right) \quad \text{and} \quad A_y = -\beta_v \rho_{\mathbf{x}} \left(d_R(\rho_{\mathbf{x}}) + \frac{t_y}{y} \right),$$

these terms capture the relationship between the distortion parameters of the RPM and those of the OFM and we deduce that $x \neq 0$ and $y \neq 0$, hence $\phi \neq 0$ (Sec. 3.1) and $\theta \neq 0$ (Sec. 2.1.6).

Finally, arranging (53) into a matrix equation provides a compact representation, for $i \in [1, N] \subset \mathbb{N}$:

$$\begin{bmatrix} \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ A_{x_i} & 0 & \theta_i^3 & \theta_i^5 & \dots & \theta_i^{2N'_F+1} \\ 0 & A_{y_i} & \theta_i^3 & \theta_i^5 & \dots & \theta_i^{2N'_F+1} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \end{bmatrix} \begin{bmatrix} f'_x \\ f'_y \\ k_{F_1} \\ k_{F_2} \\ \vdots \\ k_{F_{N'_F}} \end{bmatrix} = \begin{bmatrix} \vdots \\ -\theta_i \\ -\theta_i \\ \vdots \end{bmatrix}, \tag{54}$$

that can be solved for the OFM parameters $f'_x = 1/f_x$, $f'_y = 1/f_y$, k_{F_1}, k_{F_2}, $\dots$, $k_{F_{N'_F}}$ again using the matrix inverse or pseudo-inverse if $N > (N'_F+2)/2$. For this RP2OF conversion, we need that both point coordinates to be never equal to zero since A_{x_i} and A_{y_i} involve not only both x_i and y_i but as divisors. Hence, x_i and y_i are computed from X_i, Y_i and Z_i defined using (20) with $\delta \neq 0$, set in practice to $\delta = \pi/4$ to excite both x_i and y_i with equity. Thus, in the case of this conversion, with ϕ_i defined as in Section 3.1, we have $X_i = \sin(\phi_i)\sqrt{2}/2$ and $Y_i = \sin(\phi_i)\sqrt{2}/2$.

Let analyze under which conditions the RP2OF conversion is solvable. Since there are $N'_F + 2$ unknowns in the linear system (54), a minimum of $N'_F + 2$ rows in (54) must be linearly independent. Focusing on the case $N'_F = 1$ for the sake of conciseness, one must hence express the conditions under which there exist three different vectors:

$$\mathbf{V}_i = \left[A_{x_i}, 0, \theta_i^3, \theta_i\right], \quad \mathbf{V}'_i = \left[0, A_{y_i}, \theta_i^3, \theta_i\right], \quad \mathbf{V}_j = \left[A_{x_j}, 0, \theta_j^3, \theta_j\right], \tag{55}$$

such that no non-trivial tuple $(a, b, c) \in \mathbb{R}^3 \setminus \{(0, 0, 0)\}$ satisfies

$$a\mathbf{V}_j - b\,\mathbf{V}_i - c\,\mathbf{V}'_i = \mathbf{0}. \tag{56}$$

Then, component-wise, this condition yields the following four equations:

$$aA_{x_j} - bA_{x_i} = 0, \tag{57}$$

$$-cA_{y_i} = 0, \tag{58}$$

$$a\theta_j^3 - (b + c)\theta_i^3 = 0, \tag{59}$$

$$a\theta_j - (b + c)\theta_i = 0. \tag{60}$$

From (60), we have two possibilities: either $a = 0$ or $a \neq 0$.

If $a \neq 0$ we obtain:

$$\theta_j = \frac{b + c}{a}\,\theta_i,$$

which we substitute into (59), giving:

$$a\left(\frac{b + c}{a}\,\theta_i\right)^3 - (b + c)\theta_i^3 = (b + c)\left(\frac{(b + c)^2}{a^2} - 1\right)\theta_i^3 = 0.$$

Since $\theta_i \neq 0$ (see above within this section), this implies either $b + c = 0$ or $(b + c)^2 = a^2$.

Case $b + c = 0$. Substituting in (60) gives $\theta_j = 0$, which contradicts that $\theta_j \neq 0$. Hence, $b + c \neq 0$.

Case $(b + c)^2 = a^2$. There are two possibilities:
• If $b + c = -a$, then $\theta_j = -\theta_i$, imposing that $|\theta_j| \neq |\theta_i|$ to allow linear independence;
• Otherwise, if $b + c = a$, then $\theta_j = \theta_i$, which is impossible since θ_i and θ_j are respectively computed from two different normalized image plane points (x_i, x_i) and (x_j, x_j) (because $\delta = \pi/4$, see above within this section) and the $\arctan(\cdot)$ function is bijective. Hence $b + c \neq a$.
Hence all possibilities with $a \neq 0$ lead to the sole condition that $|\theta_j| \neq |\theta_i|$, which is common with the other conversions.

Now let consider $a = 0$, leading (60) to simplify to:

$$-(b + c)\theta_i = 0 \quad \Longrightarrow \quad b = -c,$$

since $\theta_i \neq 0$, which combined with the difference between (57) and (58) yields:

$$b(A_{y_i} - A_{x_i}) = 0.$$

$a = 0$ and $b = -c$ imply that $b \neq c \neq 0$, otherwise we fall down to the trivial tuple $(0, 0, 0)$. It implies that the above equation simplifies to $A_{x_i} = A_{y_i}$, and since $y_i = x_i$, $\rho_{\mathbf{x}_i} = \sqrt{2}\,|x_i|$, leading to:

$$\frac{t_{x_i}}{x_i} = 2x_i(p_1 + 2p_2), \qquad \frac{t_{y_i}}{y_i} = 2x_i(p_2 + 2p_1).$$

Assuming that $\beta_u = \beta_v$ (which is reasonable since we always have $\beta_u \approx \beta_v$ in our experiments):

$$A_{x_i} = -\beta_u \rho_{\mathbf{x_i}}(d_R(\rho_{\mathbf{x_i}}) + 2x_i(p_1 + 2p_2)) \text{ and } A_{y_i} = -\beta_u \rho_{\mathbf{x_i}}(d_R(\rho_{\mathbf{x_i}}) + 2x_i(p_2 + 2p_1)),$$

simplifies to:

$$2x_i(p_1 + 2p_2) = 2x_i(2p_1 + p_2) \quad \Longrightarrow \quad p_1 = p_2,$$

imposing that $p_1 \neq p_2$ to allow linear independence (which is always the case in practice in our experiments).

Therefore, any pair of points of indexes i and j in $\mathcal{U}$ such that $|\theta_i| \neq |\theta_j|$ and $p_1 \neq p_2$ ensures that the three vectors $\{\mathbf{V}_i, \mathbf{V}'_i, \mathbf{V}_j\}$ are linearly independent.

This theoretical result guarantees that selecting $\mathcal{U}$ with $W > 3$ when $N'_F = 1$ and $\delta = \pi/4$ is sufficient to solve for the RP2OF conversion, a condition systematically satisfied in practice when W is set to the width of the camera FoV in the digital image plane, following the same principle as in Section 3.1.

4. Evaluations of the conversions

In this section, we evaluate the accuracy and reliability of our mathematical model conversion approach. The goal is to demonstrate that the parameters obtained through our conversion method allow for image undistortion with a quality comparable to that achieved using explicit calibration. This validation is crucial, as it highlights the ability of our approach to bypass the need for complex or even infeasible calibration procedures, making it particularly useful in challenging imaging conditions. We first explain the

evaluation protocol in Section 4.1 and then apply this protocol to various conversions. We first assess two conversions from [17]: the transformation from the EFM to the UCM for a 185° fisheye camera (EF2UC, Sec. 4.2), and for a 210° catadioptric camera, the conversion from the UCM to the CPM (UC2CP, Sec. 4.3). We then evaluate the two new conversions, first using the same catadioptric camera from the CPM to the OFM (CP2OF, Sec. 4.4), and then for 120° FoV Infrared (IR) camera, the conversion from the RPM to the OFM (RP2OF, Sec. 4.5). These evaluations validate the capability of our approach to produce reliable parameters, without explicit calibration for the target model. All methods are implemented in the C++ `libPeR` library [24].

4.1. Evaluation Protocol

To assess the effectiveness of our camera model conversions, we follow a structured protocol. First, we compare the parameters obtained through our model conversion with those from direct calibration. To quantify the accuracy of each conversion, we compute approximation errors per 3D point defined for the conversion as Euclidean distances between their projections in the image plane with the source model and those with the output model (22).

Second, we analyze images undistorted with [25] on both sets of parameters to determine if our method achieves comparable or superior results. To this end, we compute the Structural Similarity Index Measure (SSIM) between undistorted images obtained via calibration and those generated by our converted parameters. The SSIM is used to measure the perceptual similarity between two images, accounting for luminance, contrast, and structural information. A score close to 1 indicates a nearly identical image, whereas lower values highlight significant differences.

For calibration with the UCM, we utilize the latest version of MIXEDVISION [26] [27]. The CPM calibration is performed using the Omnidirectional Camera Calibration toolbox (OCamCalib) [28]. For cameras using the OFM, calibration is carried out using the OpenCV library [11] and Kalibr toolbox [29].

4.2. Evaluation of Equidistant Fisheye to Unified Central model conversion (EF2UC)

This conversion has been previously introduced and evaluated in terms of residual error and parameter values obtained in [17]. Here, we further validate the approach using a different camera and calibration dataset, demonstrating its applicability across various imaging conditions. Furthermore, the result of

the EF2UC conversion is used in Section 5.1 for a visual servoing application for the first time. For this conversion, we used a Fujinon 185-degree fisheye lens attached to a Flir GS3-U3-41C6C-C camera (Fig. 2a). According to the manufacturer's datasheet, the fisheye lens has a 185-degree field-of-view with a focal length $f_s = 2.7$ mm, and the camera pixel pitch in binned mode is $n_s = 11$ μm, with an image resolution of 1024×1024 pixels.

Using the MIXEDVISION [26] calibration software, we obtain the baseline parameters of the UCM. These baseline parameters are: $\hat{\alpha}_u = 633.446$, $\hat{\alpha}_v = 633.775$, $\hat{u}_0 = 503.83$, $\hat{v}_0 = 492.238$, and $\hat{\xi} = 1.635$, with an average chessboard corners reprojection error of 0.269 pixels.

Converting the EFM for this camera to the UCM (Sec. 4.2), using $N = 185$ elevation angles from -92 degrees to 92 degrees with a step of one degree (ignoring 0) leads to $\alpha_u = \alpha_v = 681.086$ and $\xi = 1.7841$, with uncertainties (27) $3\hat{\sigma}_{\alpha_u} = 3\hat{\sigma}_{\alpha_v} = 6.665$ and $3\hat{\sigma}_{\xi} = 0.0202$, the image circle center is in $u_0 = 506$ and $v_0 = 490$ pixels.

We visualize the conversion approximation error across the image plane (22) in Figure 1. The error remains reasonable in the center of the image but becomes significant at the edges. This is consistent with the typical behavior of fisheye distortion, which is usually more pronounced at the image boundaries.

To further evaluate the effectiveness of our method, we undistorted the images using the cv2.omnidir.undistort function from the OpenCV library [11]. Figure 2 compares the original image of the chessboard (Fig. 2b) with the undistorted images obtained using either the calibrated parameters (Fig. 2c) or the converted parameters (Fig. 2d).

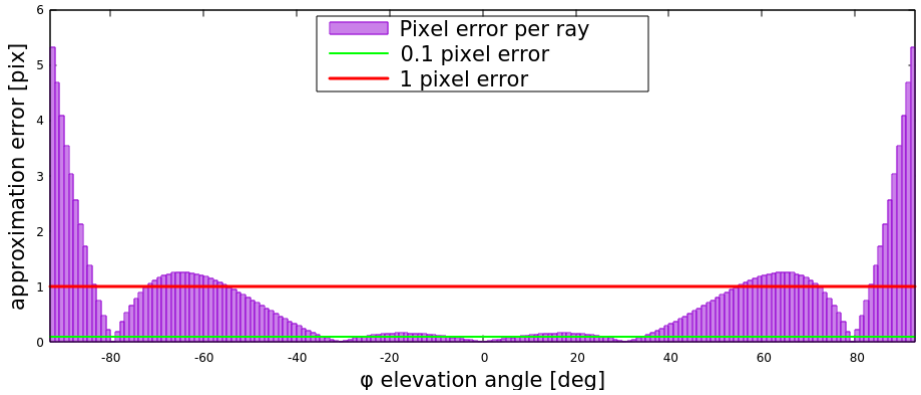
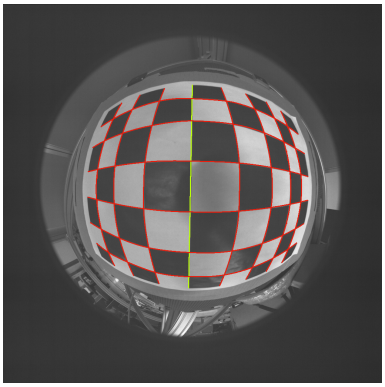
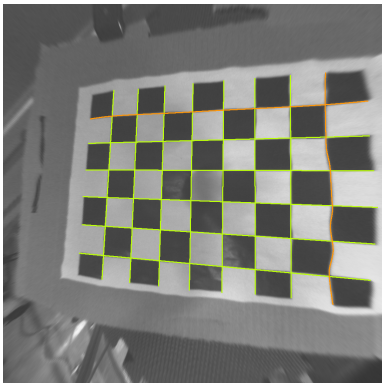


Figure 1: EF2UC conversion: in digital image plane approximation errors.

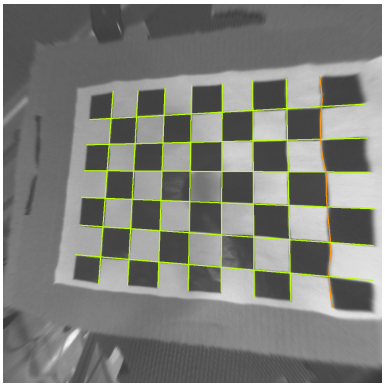
(a) Camera with Fujinon lens



(b) Original Image



(c) Undistorted image from calibrated parameters.



(d) Undistorted image from converted parameters.

Figure 2: Comparison of Original and Undistorted Images. Fig. 2b shows the original fisheye image, while Fig. 2c and Fig. 2d display undistorted versions. Red lines highlight the most pronounced curved lines, orange lines indicate weaker curves, and green lines represent the straight lines. Although not every line is perfectly corrected, in both cases, the results of the conversion are similar to those of the calibration.

To quantitatively evaluate the difference between undistorted images obtained using the calibration and conversion, we use a reference image undistorted using the EFM, i.e., the camera's initial model based on manufacturer data. We obtain an SSIM of 0.81 for the image undistorted from the calibration and an SSIM of 0.74 using the converted parameters. The calibration-based image is therefore closer to the reference, but the conversion still provides comparable results. While differences exist, they remain moderate. These conversion results make it easy to use the camera with the EFM in vision software that relies on the UCM, such as the Direct Visual Servoing

(DVS) framework achieving accurate robot positionning (Sec. 5.1).

4.3. Unified Central to Cartesian Polynomial model conversion (UC2CP)

For the evaluation of the camera model conversion, we utilized the image set provided in Sequence 6 of the PanoraMIS dataset [30]. This dataset contains eight calibration images captured using a catadioptric lens, where a checkerboard pattern was moved to cover the entire FoV of 210°. We employed the OCamCalib Toolbox [28] for Matlab to compute the camera parameters in the CPM for both second-order and fourth-order polynomials. These computed parameters were used as a baseline for comparison with the converted parameters. The fourth-order model is recommended in the original paper [15], while the second-order model reflects the capabilities of UC2CP [17]. The reference parameters for the second-order model are thus as follows: $\hat{a}_0 = 117.073$, $\hat{a}_2 = -0.002$, $\hat{u}_0 = 321.364$, $\hat{v}_0 = 311.875$, with an average reprojection error of 0.45 pixels. For the fourth-order model, the reference parameters are: $a_0^* = 121.186$, $a_2^* = -0.003$, $a_3^* = 4.566 \times 10^{-6}$, $a_4^* = -7.412 \times 10^{-9}$, $u_0^* = 321.502$, $v_0^* = 311.665$, with an average reprojection error of 0.43 pixels.

Additionally, we used the UCM parameters from the PanoraMIS dataset, $\alpha_u = 231.462$, $\alpha_v = 232.422$, $u_0 = 319.704$, $v_0 = 310.944$, and $\xi = 0.958$ as input into UC2CP. Subsequently, we applied our conversion method from UCM to CPM, resulting in the following parameters: $u_0 = 319.704$, $v_0 = 310.944$, $a_0 = 118.012$, and $a_2 = -0.002$, with an average approximation error (23) of 0.48 pixels. We visualize the approximation error across the image area processed during the conversion in Figure 3. The error is less than one pixel error over the interval [-101°,101°].

To undistort the image based on the CPM parameters, we used the py-OCamCalib software [31] based on the work of [15, 32]. As a reference, we used one of the calibration images (Fig. 4b) undistorted using the UCM (Fig. 4c). We use it together, with the second-order (Fig. 4e) and fourth-order (Fig. 4d) CPM's undistorted images to evaluate our conversion method (Fig. 4f). In all cases, the lines of the chessboard are obviously straightened. To further assess the quality of the undistortion process, we computed the SSIM between the reference undistorted image and the other undistorted images. The SSIM between the reference image and the fourth-order CPM is 0.641, 0.647 for the second-order CPM, and 0.796 for the undistorted image obtained using our conversion method (UC2CP).

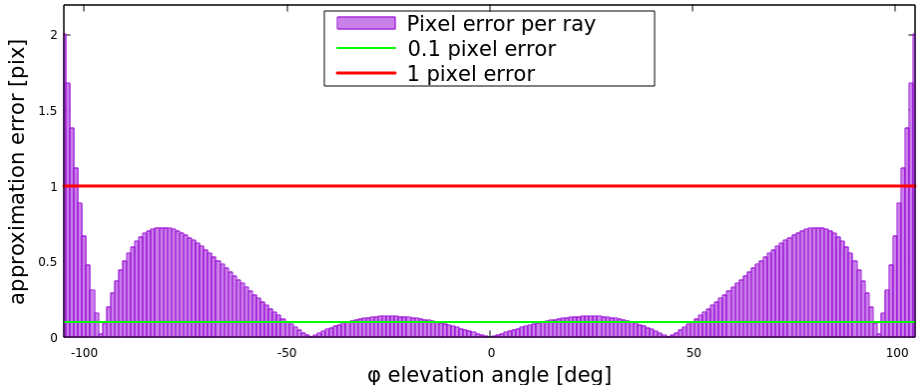


Figure 3: UC2CP conversion: in digital image plane approximation errors.

Our conversion method's higher SSIM score compared to both the second-order and fourth-order CPM's, suggests a better alignment with the reference undistorted image, suitable for practical applications.

The conversion results enable the straightforward use of the camera calibrated with the UCM in vision software that uses the CPM, such as the Semi-direct Visual Odometry (SVO) framework [5] (see Sec. 5.2).

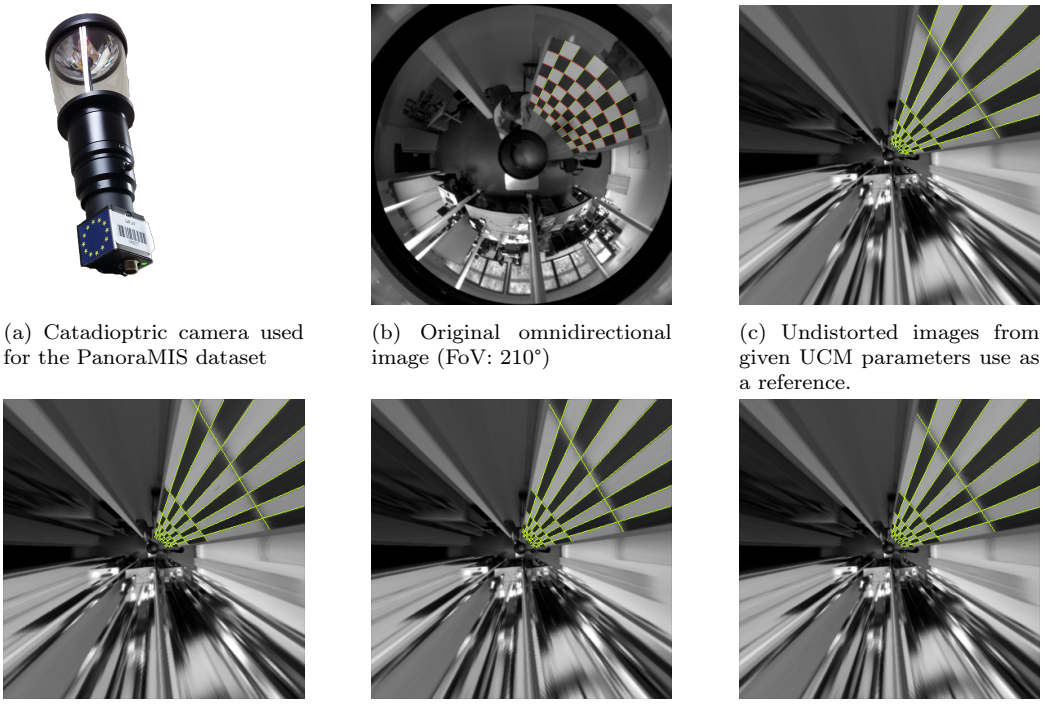
4.4. Cartesian Polynomial to OpenCV's Fisheye model conversion (CP2OF)

For this conversion, we again use the PanoraMIS dataset (Sec. 4.3), now focusing on converting the CPM to the OFM. We utilize the fourth-order parameters of the CPM, obtained from the calibration process described in Section 4.3, as input for our conversion algorithm. Since the OFM relies on the pinhole projection model (see Sec. 2.1), we restrict parameter calculations to regions where $Z > 0$ in the image.

The conversion algorithm yields the following OFM parameters: $f_x = 120.984$, $f_y = 120.984$, $k_{F_1} = 0.026$, $k_{F_2} = 0.044$, $k_{F_3} = -0.007$, and $k_{F_4} = 0.001$, with an average approximation error (23) of only 0.012 pixels.

Figure 5 shows the reprojection error distribution over the image region used during the conversion. The error remains minimal across the entire region of interest, indicating a consistent fit of the converted model.

Using these parameters, we transform an omnidirectional image (Fig. 6a) into an undistorted image (Fig. 6c). For reference, the same image is undistorted using the original UCM parameters (Fig. 6b). As shown in Figs. 6b and 6c, the curved lines in the original image become straight in the undis-



(a) Catadioptric camera used for the PanoraMIS dataset

(b) Original omnidirectional image (FoV: 210°)

(c) Undistorted images from given UCM parameters use as a reference.

(d) Undistorted images from 4th order CPM calibrated parameters.

(e) Undistorted images from given 2nd order CPM calibrated parameters..

(f) Undistorted images from CPM converted parameters.

Figure 4: Comparison of Original and Undistorted Images. Fig. 4b shows the original catadioptric image. Fig. 4c shows the undistorted image taken as a reference, while Fig. 4d, Fig. 4e and Fig. 4f display the undistorted versions. Red lines highlight the curved lines and green lines represent the straight lines. The conversion gives a similar result than the reference.

torted versions, demonstrating that the conversion preserves the global geometry of the scene.

To benchmark our approach, we compared it against two calibration methods for the OFM: Kalibr [29], a widely used open-source calibration toolbox known for its robustness and ability to optimize multiple sensors jointly, and a custom OpenCV-based fisheye calibration framework [25] tailored for the PanoraMIS dataset, integrating advanced corner detection strategies (adaptive thresholding, Contrast Limited Adaptive Histogram Equalization, Gaussian filtering) and an iterative calibration process with outlier rejection.

The Kalibr calibration produced the parameters $f_x = 144.657$, $f_y = 144.754$, $u_0 = 315.233$, $v_0 = 314.071$, $k_{F_1} = -0.434$, $k_{F_2} = 0.508$,

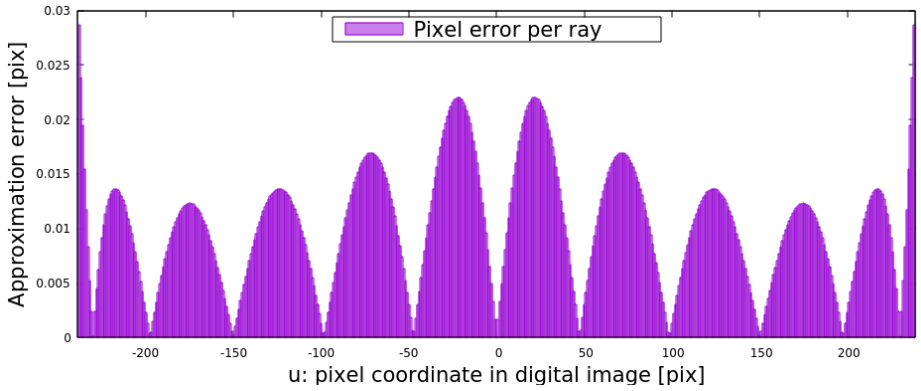
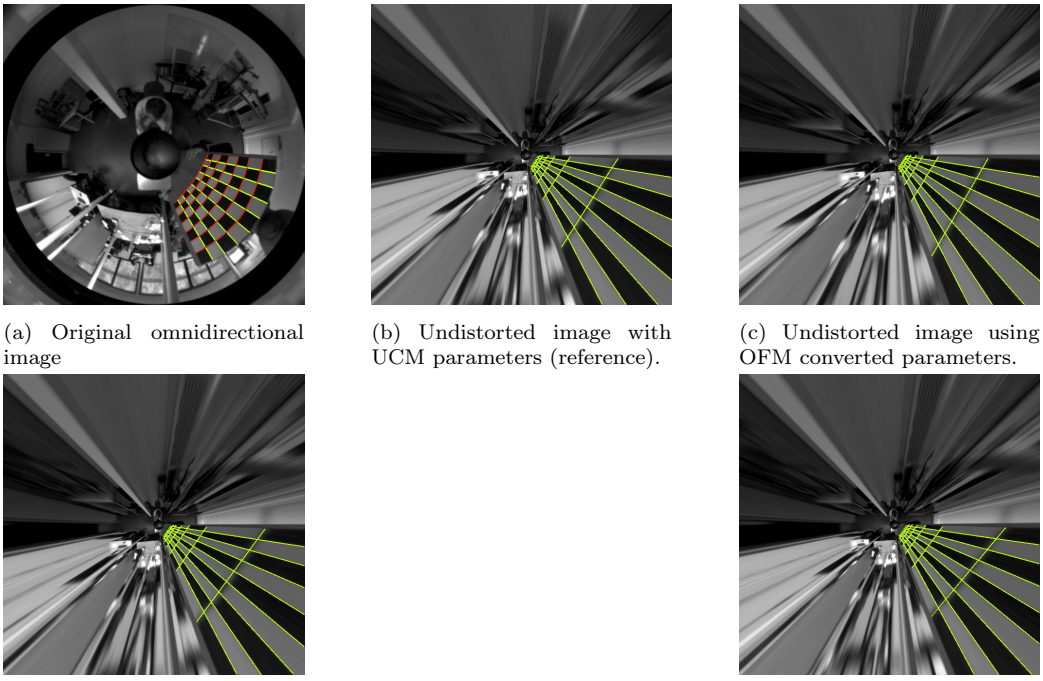


Figure 5: CP2OF conversion: approximation errors in the digital image plane.

$k_{F_3} = -0.211$, $k_{F_4} = 0.033$, with an average reprojection error of 0.370 pixels. Although 8 calibration images were provided as input, Kalibr automatically discarded 5 of them, whichever the settings, thus concretely using extracted calibration target corners from only 3 images to obtain the intrinsic parameters. The OpenCV calibration yielded $f_x = 129.3854$, $f_y = 130.279$, $u_0 = 317.651$, $v_0 = 311.899$, $k_{F_1} = -0.168$, $k_{F_2} = 0.253$, $k_{F_3} = -0.106$, $k_{F_4} = 0.018$, with a lower average reprojection error of 0.293 pixels. Similarly to Kalibr, the OpenCV calibration automatically neglected 5 images among the 8 provided, whichever the settings, but not the same than those ignored by Kalibr. Beyond the core implementation differences between Kalibr and OpenCV, this is one key factor of the significant differences between the intrinsic parameters obtained by the two software.

Visual inspection of the undistorted images (Figs. 6d and 6e) reveals that all three methods straighten the main scene lines, but subtle differences remain, especially near the periphery. Quantitatively, the SSIM with respect to the UCM-undistorted reference image is 0.770 for the converted parameters, 0.813 for OpenCV, and 0.615 for Kalibr. While OpenCV achieves the highest SSIM in this case, Kalibr performs less well.

These results illustrate that the proposed conversion achieves geometric accuracy comparable to calibration methods, and in some aspects outperforms them, while avoiding the pitfalls of poor dataset compatibility or limited retained data. By enabling reliable OFM parameters without requiring any additional calibration, our method allows cameras to be reused across experiments and integrated directly into vision-based algorithms such as StellaVSLAM (Sec. 5.3.1) without the need for repeated calibration.



(a) Original omnidirectional image

(b) Undistorted image with UCM parameters (reference).

(c) Undistorted image using OFM converted parameters.

(d) Undistorted image with OpenCV calibrated parameters.

(e) Undistorted image with Kalibr calibrated parameters.

Figure 6: Comparison of original and undistorted images. Fig. 6a shows the original catadioptric image. Figs. 6b to 6e display undistorted versions using different parameter sources. Red lines highlight curved structures and green lines indicate straight lines.

4.5. Rational Polynomial radial distortion to OpenCV's Fisheye model conversion (RP2OF)

For this last conversion, we consider the IR camera of the Microsoft Azure Kinect DK (AKDK, Fig. 8a), which features a wide-angle lens with a 120° of FoV and comes calibrated from the factory in the RPM with $\beta_u = 503.877$, $\beta_v = 504.145$, $u_0 = 509.078$, $v_0 = 510.833$, $k_{R_1} = 0.445$, $k_{R_2} = -0.027$, $k_{R_3} = -0.002$, $p1 = 1.189{\times}10^{-4}$, $p2 = 2.884{\times}10^{-5}$, $k_{R_4} = 0.786$, $k_{R_5} = 0.049$, $k_{R_6} = -0.012$.

To compare our converted parameters, we calibrated the camera using the OpenCV library [11]. Calibration was performed using checkerboard images captured in the camera's IR passive mode. These images enabled the estimation of OFM's parameters through OpenCV's calibration function. One of the eleven IR checkerboard image used for calibration is shown in

Figure 8b. The resulting intrinsic parameters are as follows: $\hat{f}_x = 530.113$, $\hat{f}_y = 526.699$, $\hat{u}_0 = 511.052$, and $\hat{v}_0 = 512.578$. The distortion coefficients obtained are $\hat{k}_{F_1} = -0.012$, $\hat{k}_{F_2} = -0.062$, $\hat{k}_{F_3} = 0.066$ and $\hat{k}_{F_4} = -0.035$.

Next, we convert from the RPM to the OFM, applying the method described in Section 3.4.2, but without any restrictions on the computation due to the FoV of the AKDK being approximately 120° in IR mode. The resulting intrinsic parameters are: $f_x = 503.916$, $f_y = 504.185$, $u_0 = 509.078$ and $v_0 = 510.833$, with distortion coefficients $k_{F_1} = -0.009$, $k_{F_2} = -0.011$, $k_{F_3} = -0.007$ and $k_{F_4} = -5.12 \times 10^{-5}$. On the 120°, the average approximation error in the digital image plane (23) is 0.14 pixels, with the largest error observed at the extremities of the image FoV (Fig. 7).

To validate the conversion, we undistort the original fisheye image (Fig. 8b) using the calibrated OFM parameters (Fig. 8d) and the converted OFM parameters (Fig. 8e). As a reference, we use the undistorted image obtained with the factory RPM parameters (Fig 8c).

The undistorted image obtained with our converted parameters closely resembles the reference, as confirmed by a high SSIM score of 0.9996. The higher SSIM for the conversion than for the calibration confirms the low approximation error of the RP2OF conversion for this camera. Thus, the intrinsic parameters of the OFM obtained by conversion can be set to use the AKDK with fisheye mode of StellaVSLAM (as detailed in Sec. 5.3.3).

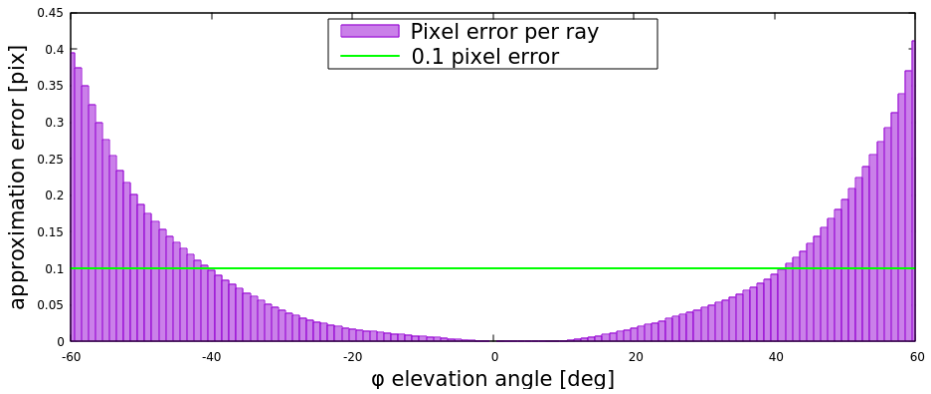
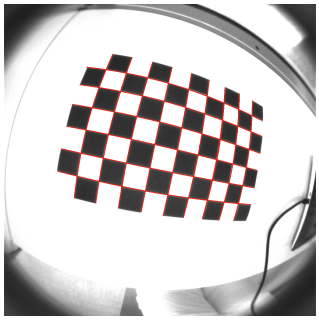
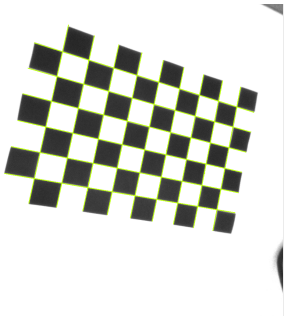


Figure 7: RP2OF conversion: in digital image plane approximation errors.

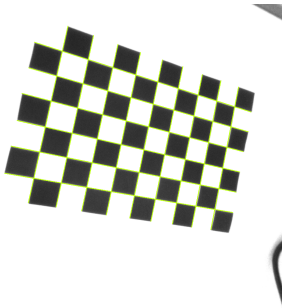
(a) AKDK camera



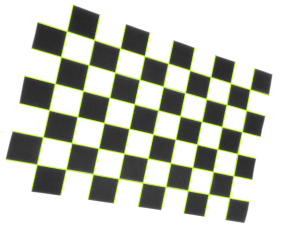
(b) Original fisheye IR image from the AKDK



(c) Undistorted image in RPM for reference



(d) Undistorted image in OFM from calibrated parameter



(e) Undistorted image in OFM from converted parameter.

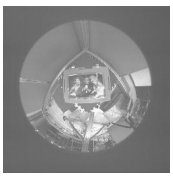
Figure 8: Comparison of Original and Undistorted Images. Fig. 8b shows the distorted image. Fig. 8c shows the undistorted image in RPM with the camera's factory parameters while the Fig. 8d and 8e display the undistorted versions. Red lines highlight the curved lines and green lines represent the straight lines. The conversion gives similar results than the calibration.

5. Applications to visual servoing, visual odometry and SLAM

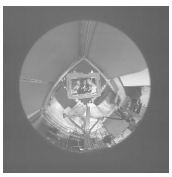
5.1. Direct Visual Servoing (DVS) with fisheye camera (EF2UC)

To assess the impact of the fisheye camera UCM parameters (see Sec. 4.2), we compare the trajectories obtained through omnidirectional photometric visual servoing with a robotic arm [33] that uses the pixel intensities of the whole images. The experimental setup, illustrated in Figure 9f, shows a fisheye camera mounted on the wrist of a Universal Robots 10 arm. The omnidirectional photometric visual servoing code is open source [34]. In this experiment, the desired image is captured with the camera's optical axis perpendicular to the ground, positioned approximately 50 cm above the ground, as shown in Figure 9a. The initial image captured is shown in Figure 9b,

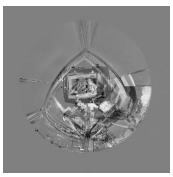
and Figure 9c depicts the error between the initial and desired images (the median gray meens the error is zero). The visual servoing algorithm moves the camera (the trajectory is shown in Figure 9g), and the residual error decreases monotonically, converging as expected. The two trajectories are nearly identical. Figure 9d illustrates the final error in the image space using calibrated parameters, which corresponds to a final pose error of 0.05 mm and 0.1 degree. Conversely, Figure 9e presents the final error obtained using converted parameters, with a final pose error of 0.28 mm and 0.1 degree. These results demonstrate that omnidirectional photometric visual servoing, using EF2UC conversion, achieves the expected submillimeter precision.



(a) Desired image



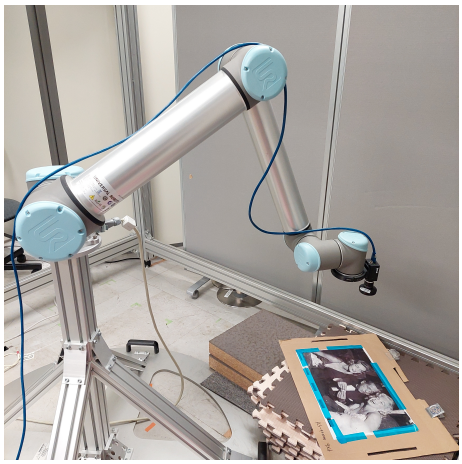
(b) Initial image



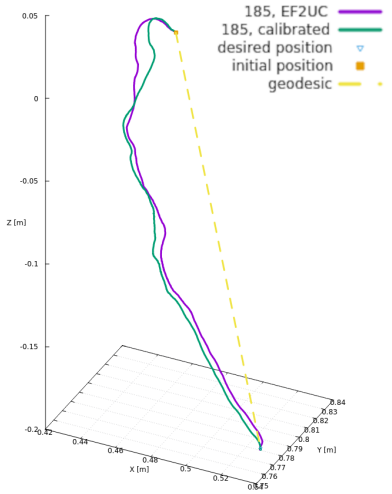
(c) Initial error

(d) Final error with calibrated parameters

(e) Final error with converted parameters



(f) Setup



(g) Camera trajectory

Figure 9: Experimental Setup and Results of Omnidirectional Visual Servoing.

5.2. Semi-direct Visual Odometry (SVO) with Catadioptric Camera (UC2CP)

The sequence 6 of the PanoraMIS dataset [30] consists of 318 images captured using the catadioptric camera shown in Figure 10b, mounted on a Pioneer 3-AT robot performing a figure-of-eight trajectory outdoors over a distance of 25.27 meters. Figure 10a illustrates the experimental setup of the PanoraMIS dataset.

To evaluate the impact of various sets of catadioptric camera CPM parameters, we analyzed the trajectories computed using SVO [5]. SVO solves the visual odometry problem by combining both direct (i.e. photometric) and indirect (i.e. feature-based) methods. It employs direct techniques for tracking and triangulating pixels with high image gradients, while using indirect feature-based methods for optimizing both the structure and motion.

We evaluate the performance of three different parameter sets: (i) CPM_Calibrated_order_2, (ii) CPM_Calibrated_order_4, and (iii) CPM_Converted, which applies the conversion from the UCM to the CPM with $N'_C = 2$ (see Sec. 3.3). The Ground Truth (GT) is provided by the dataset, indicating the relative pose of the images in the sequence. Since [15] stated that a fourth-order calibration is necessary to accurately represent the system, we display the results for both second (CPM_Calibrated_order_2) and fourth-order (CPM_Calibrated_order_4) CPM calibration leading to the baseline reference trajectories used for comparison.

To ensure a fair comparison, we performed an alignment optimization to correct the misalignment between the visual odometry body frame associated with the camera and the GT body frame. This optimization minimized

(a) Experimental setup (b) Image from the dataset

Figure 10: PanoraMIS dataset setup, showing images captured by the catadioptric camera.

the orientation difference before evaluating the trajectories with EVO [35], following the approach described in the paper [36]. Without this step, large Relative Pose Error (RPE) errors could occur due to frame misalignment.

As observed in the trajectory plot (Fig. 11), the CPM_Calibrated_order_4 and CPM_Converted trajectories closely follow the GT, whereas the CPM_Calibrated_order_2 trajectory is significantly less accurate.

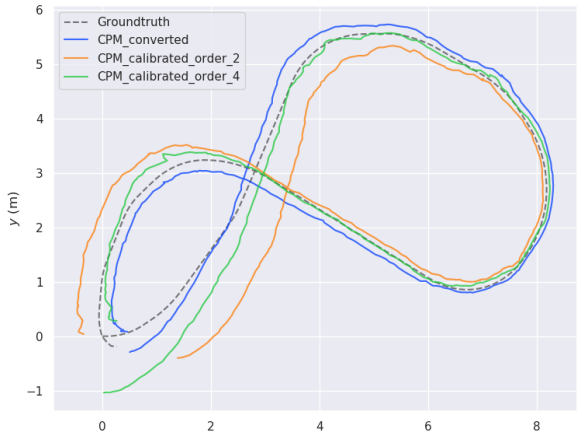


Figure 11: Trajectory results calculated with SVO according to different parameter sets.

This observation is further confirmed in Table 2, where the Absolute Pose Error (APE) and RPE errors with CPM_Converted are comparable to using CPM_Calibrated_order_4 in translation and lower in rotation, both better overall than when using CPM_Calibrated_order_2.

Table 2: Relative and absolute error values for translation (in mm) and rotation (in degrees) of trajectory calculated with different parameter sets. In bold the minimum error.

		CPM converted	CPM Calibrated Order 2	CPM Calibrated Order 4
Translation	APE	305	488	**292**
	RPE	33	**32**	37
Rotation	APE	**8**	10	9
	RPE	**1.3**	1.4	**1.3**

5.3. Visual SLAM

To evaluate camera model conversion results of RP2OF and CP2OF, we use the respectively obtained parameters for Simultaneous Localization And Mapping (SLAM) with the StellaVSLAM software [37]. It is is a visual SLAM algorithm designed to operate with various camera models, including RPM and OFM. It leverages feature-based tracking and keyframe-based optimization to provide efficient mapping and localization. In this section, we apply the OFM parameters obtained with our method to two different camera types: a Red Blue Green (RGB) catadioptric camera (Sec. 5.3.1), an RGB fisheye camera (Sec. 5.3.2) and an IR wide-angle camera (Sec. 5.3.3).

The primary goal of this section is to study if the parameters obtained from model conversions provide results comparable to those of camera calibration.

Since StellaVSLAM relies on Random Sample Consensus (RANSAC) for essential matrix estimation during initialization and loop closure, the trajectories generated from different runs on the same dataset may vary slightly. This randomness is inherent to the RANSAC algorithm, which iteratively selects random subsets of points to estimate the model parameters while dealing with outliers. To mitigate this variability, we run StellaVSLAM four times and analyze averaged results after alignment, ensuring that the observed differences in the trajectories do not affect our comparison of parameter sets.

In the following Section 5.3.1, Section 5.3.2 and Section 5.3.3, we present the results obtained from running StellaVSLAM with respectively catadioptric camera, RGB and IR fisheye camera.

5.3.1. SLAM with a Catadioptric Camera CP2OF

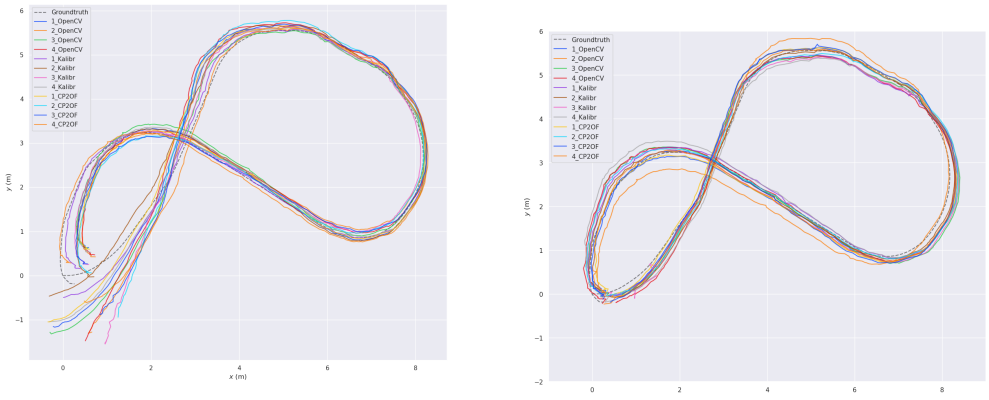
To demonstrate the effectiveness of our conversion method CP2OF in a SLAM context, we compare the trajectories obtained with our converted parameters to those obtained with calibration using OpenCV and Kalibr, as reported in Section 4.4. We use the PanoraMIS dataset's Sequence 6, introduced in Section 5.2. The dataset was recorded at 2 Hz, and StellaVSLAM was run at the same frequency.

This sequence is particularly challenging due to the predominance of the ground in the images, mainly grass with few discriminative features. A mask is applied to exclude the fixed parts of the robot and the regions outside the mirror of the catadioptric system. The resulting trajectories, aligned and scaled to the GT, are shown in Figure 12a. As in Section 5.2, the GT was optimized before error computation to ensure fair comparison. This

alignment step corrects any offset between the visual odometry body frame and the ground-truth body frame, which is crucial to avoid artificially inflated RPE errors. The APE and RPE results are reported in Table 3.

Since the original dataset was designed primarily for odometry, loop closures are not detected. To enable their detection, we extended the dataset by appending the initial frames to the end. Running StellaVSLAM four times on this extended dataset yielded trajectories with successful loop closure detection (Figure 12b), leading to improved absolute accuracy. As before, the same APE computation pipeline was used, and the results in Table 3 show that loop closure significantly reduces the error.

On the original dataset, the parameters calibrated with Kalibr lead StellaVSLAM to the lowest APE and RPE in both translation and rotation. Our converted parameters lead StellaVSLAM to perform weaker (126 mm increase of mean translational APE over the 25.27 m trajectory) but very similarly than using the parameters obtained with OpenCV calibration (lower translation APE by 6 mm but higher rotational APE by 1.1 degree; RPE are identical). On the extended dataset, the enabled loop closure reduces significantly the APE for all sets of intrinsic parameters. In this context, the CP2OF-converted parameters even lead StellaVSLAM to achieve the lowest APE and RPE (Table 3) compared to using the intrinsic parameters of



(a) Original dataset (no loop closure detected)

(b) Extended dataset with loop closure

Figure 12: Trajectories obtained with StellaVSLAM for four runs on the PanoraMIS dataset using CP2OF-converted, OpenCV-calibrated, and Kalibr-calibrated parameters, compared with the GT. (a) Original dataset without loop closure. (b) Extended dataset with loop closure, improving trajectory accuracy.

Table 3: Mean absolute and relative errors for translation (Tra., in mm) and rotation (Rot., in degrees), averaged over four trajectories. Results are shown for both the *Original* dataset and the *Extended PanoraMIS dataset*, using three methods (CP2OF, Kalibr, and OpenCV).

		Original dataset			Extended PanoraMIS dataset		
		CP2OF	Kalibr	OpenCV	CP2OF	Kalibr	OpenCV
Tra. (mm)	APE	443	**317**	449	**156**	199	225
	RPE	35	**33**	35	**35**	38	36
Rot. (deg.)	APE	10.1	**7.0**	9.0	**3.6**	4.1	4.1
	RPE	1.2	**1.1**	1.2	**1.0**	1.1	**1.0**

the baseline calibration methods (43 mm decrease of the translation APE with respect to the second best that uses Kalibr calibration), confirming the relevance of the conversion.

These findings confirm that the CP2OF conversion method is not only a viable alternative to calibration but also capable of producing accurate SLAM trajectories without the need for recalibration. This is particularly advantageous in long-term deployments or scenarios where recalibration is impractical, enabling the seamless reuse of cameras with consistent intrinsic parameters across datasets and experiments.

5.3.2. SLAM on synthetic fisheye dataset CP2OF

To further validate the proposed conversion, we evaluate CP2OF on a synthetic fisheye dataset [38]. In this work, we use the urban fisheye sequence, where the image distortion is modeled with the CPM. The dataset is distributed with reference intrinsic parameters: $a_0^* = 179.472$, $a_2^* = 2.317 \times 10^{-3}$, $a_3^* = -3.636 \times 10^{-6}$, $a_4^* = 2.055 \times 10^{-8}$, and principal point $(u_0^*, v_0^*) = (320, 240)$.

The CP2OF conversion yields the following OFM parameters: $f_x = 179.028$, $f_y = 179.028$, $k_{F_1} = -3.041 \times 10^{-2}$, $k_{F_2} = -6.511 \times 10^{-3}$, $k_{F_3} = 8.819 \times 10^{-4}$, $k_{F_4} = 3.433 \times 10^{-4}$, with an average approximation error of 0.025 pixels. This very low error demonstrates that the conversion preserves the geometric accuracy of the CPM representation.

Since the dataset does not provide calibration targets, an OFM calibration is not available. To address this, we leverage the C++ implementation of the CPM model [15]. Using this tool, synthetic checkerboards are generated with the specified CPM distortion and stored in a rosbag. The

rosbag is then processed with Kalibr [29] to perform a reference calibration in the OFM. The resulting parameters are: $f_x = 179.150$, $f_y = 179.263$, $k_{F_1} = -5.268 \times 10^{-2}$, $k_{F_2} = 5.048 \times 10^{-2}$, $k_{F_3} = -5.251 \times 10^{-2}$, $k_{F_4} = 1.717 \times 10^{-2}$, $(u_0, v_0) = (318.519, 239.045)$, with an average reprojection error of 0.320 pixels.

Following the same protocol as in previous experiments, four independent runs are executed with StellaVSLAM using both the converted and calibrated parameters. Since the dataset contains no loop closures along its trajectory, StellaVSLAM is operated in odometry-only mode. The resulting trajectories, aligned and rescaled to the ground truth, are shown in Fig. 13. In both cases, a drift in translation is observed between the start and the end of the sequence, which is consistent with the absence of loop closure.

Nevertheless, a quantitative comparison reveals that both sets of parameters yield similar accuracy. For the converted parameters, the average APE is 19.023 m in translation and 9.820° in rotation, while the RPE is 0.105 m (translation) and 0.253° (rotation). For the calibrated parameters, the corresponding values are 18.912 m and 9.210° (APE), and 0.107 m and 0.248° (RPE). These results confirm that the CP2OF conversion provides performance comparable to calibration, with errors of the same order of magnitude across all metrics.

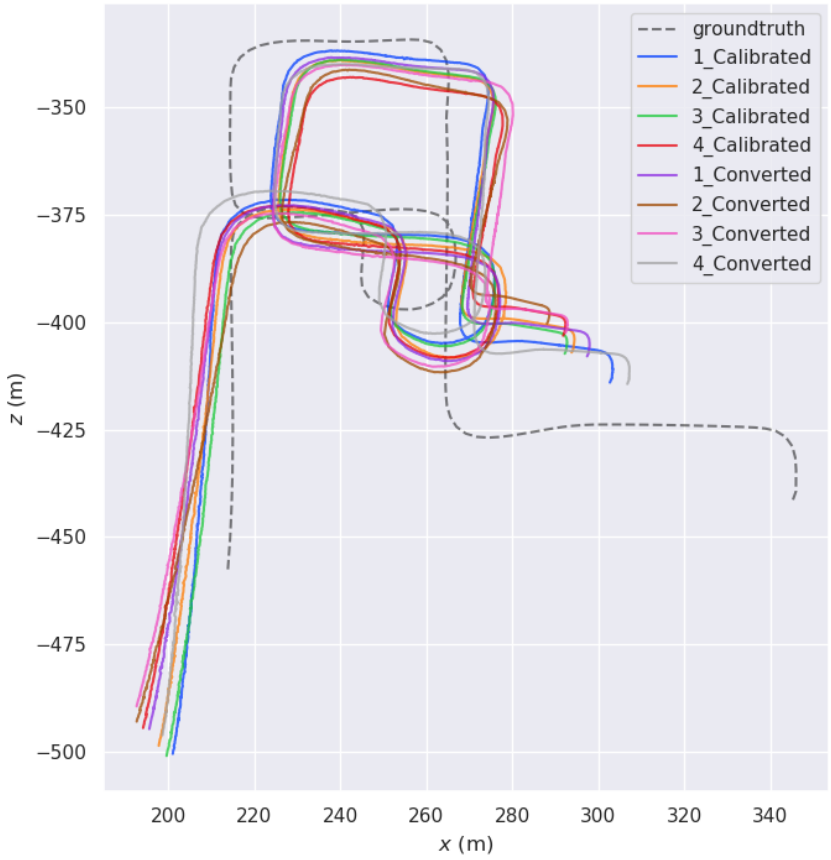


Figure 13: Comparison of four SLAM trajectories on Zhang's dataset using parameters from CP2OF conversion and Kalibr calibration.

5.3.3. SLAM with a wide-angle IR Camera (RP2OF)

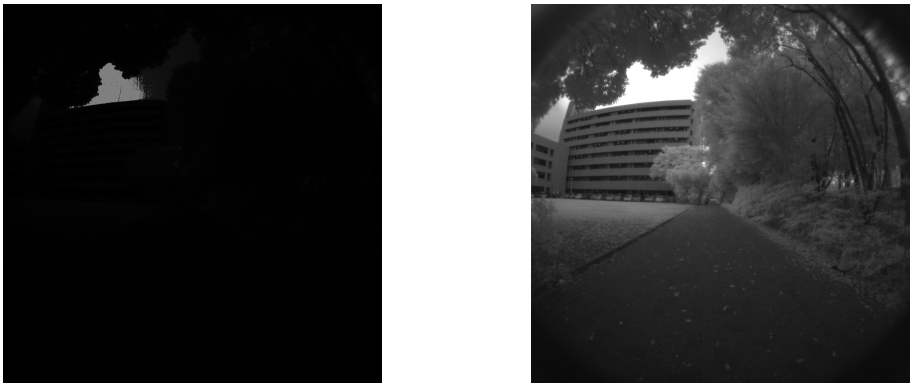
When using the AKDK camera, it is evident that the FoV of the color camera is significantly smaller than that of the IR camera. The goal in this section is to leverage the quasi-fisheye capabilities of the IR camera to assess the performance of StellaVSLAM RP2OF's converted parameters.

The IR image stream captured by the AKDK is initially in 16-bit grayscale. Figure 14a illustrates the normalized 16-bit image, where each pixel value is divided by 256 to scale the data and display it in 8-bit grayscale, ranging from 0 to 255. While this allows for simple visualization, it does not yield an image suitable for use in StellaVSLAM. To make the image compatible with

StellaVSLAM in monocular mode, we apply a custom process. This process involves normalizing the pixel values by detecting the minimum and maximum values, followed by gamma correction to enhance the dynamic range. The processed images are then published as a stream, ready for use by the SLAM algorithm, as shown in Figure 14b [39].

The data were recorded outdoors using the passive IR mode of the camera. Two trajectories were captured: one on a parking lot near a building, approximately 130 meters in length and 149.2 seconds in duration, referred to as Parking, and the other on a tree-lined path about 207 meters long and 176 seconds in duration, named Trail. The Parking trajectory is characterized by high redundancy, with limited visual features, mostly consisting of asphalt and the side of a building, which are repeatedly observed. In contrast, the Trail trajectory provides much richer visual information, with trees and scattered leaves on the ground offering a variety of textures and depth cues. Both trajectories were recorded outdoors to ensure the proper IR lighting conditions necessary for data acquisition. The dataset named WAIr-JaM for Wide Angle Infrared JRL and MIS dataset is open source [40].

To ensure a comprehensive analysis of the SLAM results, we perform four runs of StellaVSLAM on each dataset using the converted RP2OF camera parameters obtained in Section 3.4.2. For comparison, we also run StellaVSLAM four times using the calibrated parameters for each dataset. In the absence of ground-truth data, we employ a cross-comparison approach: for each trajectory generated with the converted parameters, we compute its APE and RPE averages—both for rotation and translation—against all



(a) 8-bit grayscale IR after a basic image processing. (b) 8-bit grayscale IR after our image processing.

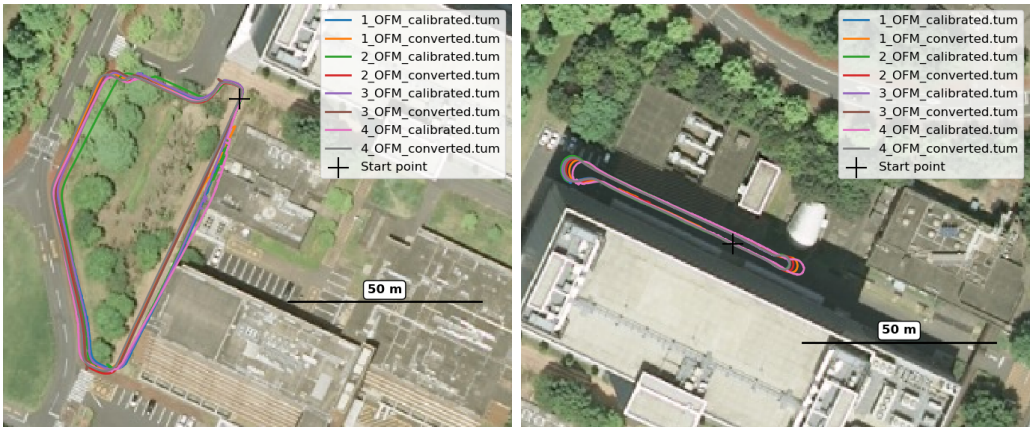
Figure 14: Comparison of two 8-bit grayscale IR images from the Trail experiment.

trajectories produced using the calibrated parameters. We then calculate the mean error for each converted trajectory relative to all reference trajectories, followed by an overall average across all comparisons. This method allows us to quantify the error between trajectories generated with the converted parameters and those using the calibrated parameters while accounting for the variability introduced by RANSAC.

The obtained trajectories are aligned with the first trajectory generated using the calibrated parameters and are presented on the satellite map in Figure 15a and Figure 15b. The satellite map data is provided by the Geospatial Information Authority of Japan (GSI) and is licensed under CC BY 4.0 [41].

The alignment, scaling of the trajectories, and calculation of the APE were performed using the EVO toolkit [35]. To account for the inherent scale ambiguity in monocular SLAM, the translational APE was normalized by dividing it by the unit-less length of the reference trajectory and then multiplying by 100 to get a percentage. This normalized error, expressed as a percentage of the reference trajectory length, ensures comparability between trajectories of varying scales.

A comparison is conducted for each trajectory, as explained earlier. For each reference trajectory, the normalized translational APE percentages are averaged across the four derived trajectories. These averages are then com-



(a) Trail trajectory

(b) Parking trajectory

Figure 15: Superposition of the obtained trajectories aligned and resized using the EVO toolkit with the satellite map. The satellite imagery is provided by the Geospatial Information Authority of Japan (GSI) [41].

bined across all reference trajectories to provide an overall percentage error for each experiment.

For the Parking experiment, the mean normalized translational APE is 0.27%, indicating a high degree of similarity between the trajectories generated using the converted parameters or those generated using the calibrated ones and the original model of the camera. The rotational APE remains consistent at 3.5°, showing minimal variation in angular estimation.

In the Trail experiment, the mean normalized translational APE increases to 0.51%, reflecting a slightly larger deviation due to the more complex nature of the trajectory, which includes richer features such as trees and scattered leaves. However, the rotational APE of 2.9° demonstrates stable orientation estimation, despite the environmental challenges.

These results suggest that the trajectories obtained with the converted parameters exhibit a strong similarity to those obtained using the calibrated parameters, validating the effectiveness of the RP2OF conversion method. The increase in translational APE in the Trail experiment underscores the influence of environmental complexity, while the overall low percentage errors confirm the robustness and applicability of the converted parameters across different scenes.

The use of converted parameters in StellaVSLAM, originally designed for monocular fisheye mode with classically calibrated color images, produces trajectories and maps that closely align with those obtained using directly calibrated parameters. This demonstrates that our conversion approach maintains the accuracy and reliability expected of SLAM systems.

6. Discussion of the article

The proposed conversion framework offers a computationally efficient alternative to full calibration. Once the source camera parameters are available, the conversion requires only a few milliseconds to compute, whereas calibration involves image acquisition, feature detection, and nonlinear optimization over both intrinsic and extrinsic parameters, resulting in a computational cost that is several orders of magnitude higher. In practice, this efficiency makes the conversions attractive for large-scale dataset processing or scenarios where rapid deployment is needed.

Regarding robustness, the accuracy of the conversion naturally depends on the reliability of the input parameters. When the manufacturer or factory calibration parameters are accurate, the converted models can be as precise

as, or even more consistent than, direct re-calibration. Conversely, if the source parameters are imprecise, the resulting conversions will inherit and propagate these inaccuracies.

Nevertheless, the method also has limitations and potential failure modes. The validity of the approach relies on the assumption that the camera model used adequately captures the physical geometry of the real sensor. If the selected model diverges significantly from the true optical behavior, the converted parameters, although mathematically consistent, will not provide physically meaningful projections. In cases where no reliable initial model with its parameters is available, the conversion cannot be applied directly.

Overall, these considerations underline the complementary role of conversions and calibrations: conversions provide a lightweight and efficient tool for interoperability across data, models and robot vision software, while calibration remains a necessary fallback when no reliable source parameters exist or when extreme optical distortions cannot be captured by the input camera model.

7. CONCLUSION

In this work, we introduced a mathematical conversion framework for camera models, enabling accurate computation of intrinsic parameters without requiring an explicit recalibration. Our approach was extensively evaluated for multiple conversions: Equidistant Fisheye Model to Unified Central Model, Unified Central Model to Cartesian Polynomial Model, Cartesian Polynomial Model to OpenCV's Fisheye Model, and Rational Polynomial radial distortion Model to OpenCV's Fisheye Model. Despite the nonlinearities of the latter camera models, our approach computes the conversion in an elegant and light-weight linear manner.

Results of image undistortion show that the converted parameters lead to an accuracy comparable to calibration thanks to evaluation criteria such as the Structural Similarity Index Measure, reprojection errors, and qualitative analysis for various cameras ranging for the wide-angle infrared to the fisheye and catadioptric ones offering a panoramic field-of-view.

These findings and the implementation of these new conversion methods within the open-source `libPeR` library are going to make easier the integration of a camera provided with parameters for a different camera model than the one considered robot vision tasks such as visual servoing, visual odometry and visual Simultaneous Localization And Mapping. This highlights the

practical and scientific relevance of our approach.

Future works will apply the general framework introduced in this article for camera model conversions to other camera models and the formal error bound analyses of the conversions.

References

[1] P. Sturm, S. Ramalingam, J.-P. Tardif, S. Gasparini, J. Barreto, Camera Models and Fundamental Concepts Used in Geometric Computer Vision, Foundations and Trends in Computer Graphics and Vision 6 (2011) 1–183. doi:10.1561/0600000023.

[2] C. Mei, P. Rives, Single View Point Omnidirectional Camera Calibration from Planar Grids, in: Proc. of IEEE Int. Conf. on Robotics and Automation, 2007, pp. 3945–3950. doi:10.1109/ROBOT.2007.364084.

[3] Y. Chen, M. Q.-H. Meng, L. Liu, Direct visual servoing based on discrete orthogonal moments, IEEE Trans. on Robotics 40 (2024) 1795–1812. doi:10.1109/TRO.2024.3360954.

[4] D. Sharafutdinov, M. Griguletskii, P. Kopanev, M. Kurenkov, G. Ferrer, A. Burkov, A. Gonnochenko, D. Tsetserukou, Comparison of modern open-source Visual SLAM approaches, J. of Intelligent and Robotic Systems 107 (2023) 43. doi:10.1007/s10846-023-01812-7.

[5] C. Forster, M. Pizzoli, D. Scaramuzza, SVO: Fast semi-direct monocular Visual Odometry, in: Proc. of IEEE Int. Conf. on Robotics and Automation, 2014, pp. 15–22. doi:10.1109/ICRA.2014.6906584.

[6] K. Miyamoto, Fish Eye Lens, J. of the Optical Society of America 54 (8) (1964) 1060–1061. doi:10.1364/JOSA.54.001060.

[7] S. Gao, K. Yang, H. Shi, K. Wang, J. Bai, Review on Panoramic Imaging and Its Applications in Scene Understanding, IEEE Trans. on Instrumentation and Measurement 71 (2022) 1–34. doi:10.1109/TIM.2022.3216675.

[8] G. Caron, Models and Calibration Methods, in: Omnidirectional Vision: From Theory to Applications, ISTE, 2023, pp. 39–63.

[9] J. Kannala, S. S. Brandt, A Generic Camera Model and Calibration Method for Conventional, Wide-Angle, and Fish-Eye Lenses, IEEE Trans. on Pattern Analysis and Machine Intelligence 28 (8) (2006) 1335–1340. doi:10.1109/TPAMI.2006.153.

[10] K. Di, R. Li, CAHVOR camera model and its photogrammetric conversion for planetary applications, J. of Geophysical Research: Planets 109 (E4) (2004) E04004. doi:10.1029/2003JE002199.

[11] OpenCV contributors, OpenCV Library, https://docs.opencv.org/3.4/db/d58/group__calib3d__fisheye.html, [Library] (1999).

[12] S. Baker, S. K. Nayar, A Theory of Single-Viewpoint Catadioptric Image Formation, Proc. of International Conference on Computer Vision 35 (2) (1999) 175–196. doi:10.1109/ICCV.1998.710698.

[13] C. Geyer, K. Daniilidis, A Unifying Theory for Central Panoramic Systems and Practical Applications, in: Proc. of European Conference on Computer Vision, Vol. 1843, 2003, p. 445–461. doi:10.1007/3-540-45053-X_29.

[14] J. P. Barreto, F. Martin, R. Horaud, Visual Servoing/Tracking Using Central Catadioptric Images, in: Experimental Robotics VIII, 2003, pp. 245–254. doi:10.1007/3-540-36268-1_21.

[15] D. Scaramuzza, A. Martinelli, R. Siegwart, A Toolbox for Easily Calibrating Omnidirectional Cameras., in: Proc. of IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, 2006, pp. 5695–5701. doi:10.1109/IROS.2006.282372.

[16] X. Ying, Z. Hu, Can We Consider Central Catadioptric Cameras and Fisheye Cameras within a Unified Imaging Model, in: T. Pajdla, J. Matas (Eds.), Proc. of European Conference on Computer Vision, Springer, 2004, pp. 442–455. doi:10.1007/978-3-540-24670-1_34.

[17] E. Goichon, G. Caron, P. Vasseur, F. Kanehiro, On camera model conversions, in: Proc. of IEEE Int. Conf. on Robotics and Automation, 2024, pp. 1–8. doi:10.1109/ICRA57147.2024.10610009.

[18] S. Lee, Fisheye-Calib-Adapter: An Easy Tool for Fisheye Camera Model Conversion (2024). doi:10.48550/arXiv.2407.12405.

[19] B. Khomutenko, G. Garcia, P. Martinet, An enhanced unified camera model, IEEE Robotics and Automation Letters 1 (1) (2016) 137–144. doi:10.1109/LRA.2015.2502921.

[20] V. Usenko, N. Demmel, D. Cremers, The Double Sphere Camera Model, in: 2018 International Conference on 3D Vision (3DV), 2018, pp. 552–560. doi:10.1109/3DV.2018.00069.

[21] D. B. Gennery, Least-Squares Camera Calibration Including Lens Distortion and Automatic Editing of Calibration Points, Vol. 34, Springer Berlin Heidelberg, 2001. doi:10.1007/978-3-662-04567-1_5.

[22] P. R. Wolf, Elements of Photogrammetry: With Air Photo Interpretation and Remote Sensing, Civil Engineering Series, McGraw-Hill, 1983.

[23] V. Larsson, T. Sattler, Z. Kukelova, M. Pollefeys, Revisiting Radial Distortion Absolute Pose, in: Proc. of International Conference on Computer Vision, 2019, pp. 1062–1071. doi:10.1109/ICCV.2019.00115.

[24] G. Caron, A. Andre, libPeR library, https://github.com/PerceptionRobotique/libPeR_base, version 0.6.0 [software].

[25] E. Goichon, Utils library, https://github.com/isri-aist/Utils, [Library] (2025).

[26] G. Caron, D. Eynard, Multiple camera types simultaneous stereo calibration, in: Proc. of IEEE Int. Conf. on Robotics and Automation, 2011, pp. 2933–2938. doi:10.1109/ICRA.2011.5979975.

[27] G. Caron, MIXEDVISION library, https://github.com/PerceptionRobotique/MIXEDVISION, [Library] (2025).

[28] D. Scaramuzza, OcamCalib: Omnidirectional Camera Calibration Toolbox for MATLAB, https://sites.google.com/site/scarabotix/ocamcalib-omnidirectional-camera-calibration-toolbox-for-matlab, [software] (2025).

[29] P. Furgale, J. Rehder, R. Siegwart, Kalibr - a toolbox for camera and imu calibration, https://github.com/ethz-asl/kalibr, [Computer software] (2013).

[30] H.-E. Benseddik, F. Morbidi, G. Caron, PanoraMIS: An ultra-wide field of view image dataset for vision-based robot-motion estimation, The Int. J. of Robotics Research 39 (9) (2020) 1037–1051, [dataset]. doi:10.1177/0278364920915248.

[31] H. Vazquez, Py-OCamCalib, https://github.com/jakarto3d/py-OCamCalib, [software].

[32] S. Urban, J. Leitloff, S. Hinz, Improved wide-angle, fisheye and omnidirectional camera calibration, ISPRS J. of Photogrammetry and Remote Sensing 108 (2015) 72–79. doi:10.1016/j.isprsjprs.2015.06.005.

[33] G. Caron, E. Marchand, E. Mouaddib, Photometric visual servoing for omnidirectional cameras, Autonomous Robots 35 (2-3) (2013) 177–193. doi:10.1007/s10514-013-9342-3.

[34] G. Caron, PhotometricOmnidirectionalVisualServoing program, https://github.com/PerceptionRobotique/PhotometricOmnidirectionalVisualServoing, [software].

[35] M. Grupp, evo: Python package for the evaluation of odometry and SLAM., https://github.com/MichaelGrupp/evo, [software] (2017).

[36] B. Li, D. Zou, Y. Huang, X. Niu, L. Pei, W. Yu, TextSLAM: Visual SLAM With Semantic Planar Text Features, IEEE Trans. on Pattern Analysis and Machine Intelligence 46 (1) (2024) 593–610. doi:10.1109/TPAMI.2023.3324320.

[37] S. Sumikura, M. Shibuya, K. Sakurada, OpenVSLAM: A Versatile Visual SLAM Framework, in: Proc. of ACM Int. Conf. on Multimedia, 2019, pp. 2292–2295. doi:10.1145/3530839.353084.

[38] Z. Zhang, H. Rebecq, C. Forster, D. Scaramuzza, Benefit of large field-of-view cameras for visual odometry, in: 2016 IEEE International Conference on Robotics and Automation (ICRA), 2016, pp. 801–808. doi:10.1109/ICRA.2016.7487210.

[39] A. Caillot, E. Goichon, Azure Kinect IR Image Streamer, https://github.com/isri-aist/AKIRS, [software] (2025).

[40] E. Goichon, WAIr-JaM dataset, https://extra.u-picardie.fr/nextcloud/index.php/s/Ltj322skRmkTmQC, [dataset] (2025).

[41] Geospatial Information Authority of Japan, Standard and Ortho Map Layers, https://maps.gsi.go.jp, (accessed March 2025) (2024).

Author Biographies

Eva Goichon earned a degree in Electrical Engineering from the National Institute of Applied Sciences de Strasbourg (INSA Strasbourg) in 2022, along with a master's in Robotic Imaging and Life Engineering from Telecom Physique Strasbourg (TPS). She completed final-year projects at INSA Rennes and the Joint Robotics Laboratory (JRL) in Japan on SLAM for multi-exposure vision. Currently, she is a CNRS-AIST JRL Ph.D. student at the National Institute of Advanced Industrial Science and Technology (AIST) in Japan and the MIS laboratory in France, working on camera projection model conversion for computer vision compatibility and on infrared and RGB-D SLAM.

Guillaume Caron (Senior Member, IEEE) received the Ph.D. degree in robotics and the Habilitation degree from the University of Picardie Jules Verne (UPJV), France, in 2010 and 2019, respectively. He has been an Associate Professor with the UPJV since 2011, and a CNRS delegate with CNRS-AIST Joint Robotics Laboratory (JRL), IRL, Japan, since 2019. He has been the co-director of JRL since 2022. He has been serving as Associate Editor for the IEEE ROBOTICS AND AUTOMATION LETTERS since 2023. His research interests include artificial vision for robotics, real-time visual tracking and servoing, and digital heritage.

Pascal Vasseur received his MSc degree in System Control from the Technological University of Compiègne in 1995 and his PhD degree from the University of Picardie Jules Verne in 1998 for works related to Computer Vision and Image Analysis. From 2010 to 2020, he was Full Professor at the Université de Rouen Normandie within the STI team (Intelligent Transportation System) at the LITIS Lab (IT Laboratory, Information Processing and Systems). Since 2020, he is Full Professor at the Université de Picardie Jules Verne and is a member of the MIS laboratory (Modelization, Information and Systems Lab). His research interests are computer vision and perception for applications to intelligent transportation, mobile and aerial robots.

Fumio Kanehiro (Member, IEEE) received his Ph.D. in engineering from The University of Tokyo in 1999. He was a Research Fellow at the Japan Society for the Promotion of Science (1998–1999) before joining the National Institute of Advanced Industrial Science and Technology in 2000. He was a Visiting Researcher at the Laboratory for Analysis and Architecture of Systems, CNRS (2007–2008). He is currently the Director of the CNRS-AIST JRL(Joint Robotics Laboratory), an International Research Laboratory at AIST. His research focuses on software frameworks and whole-body motion planning for humanoid robots.