University of Padova

Department of Mathematics "Tullio Levi-Civita"

*Master Thesis in Data Science*

# Mathematical models for geometric calibration of a hyperemispheric camera for planetary exploration

*Supervisor*
Wolfgang Erb
University of Padova

*Co-supervisors*
Emanuele Simioni
Claudio Pernechele
INAF

*Master Candidate*
Andrea Marchini

*Student ID*
2044711

*Academic Year*
2022/2023

“Mathematics is not about numbers, equations, computations, or algorithms. It is about understanding.”
— William Paul Thurston

# Abstract

Hyper-Hemispheric lenses belong to the ultra-wide field-of-view optical objectives. The lens considered is named PANCAM and it was firstly introduced in 2016. This camera is particularly interesting for planetary exploration purposes, since its capability to acquire large Field-of-View images. Its Field of View is $360°$ on the azimuth and $135°$ for the off-boresight angle.
The calibration of the lens consists in computing the extrinsic and intrinsic parameters of a model whose aim is to define the relationship between a pixel of the image and the real-world points reprojected on it. Davide Scaramuzza introduced in 2006 a useful model for the calibration of lenses with wide FoV, which assumes the single effective viewpoint property for the camera. Together with this model, also a MATLAB toolbox got implemented, which follows the latter model. This toolbox performs the calibration procedure starting from images containing black-and-white chessboards, which are portrayed at different positions and orientations with respect of the camera. Their internal vertices are used as benchmarks to create the geometric vinculums.
Since we are not dealing with a central camera, new models come up to better describe the PANCAM lens. All of these models feature a function, $z_0$, which associates a pixel of the image to a point on the Z axis of the coordinate system of the camera, such that light rays which are directed towards this point will be mapped into the corresponding pixel.
Each model owns a different version of the $z_0$ function: linear, quadratic and piecewise.
The MATLAB toolbox is then modified, integrating the changes carried by new models.
In order to test the models, images were acquired using the PANCAM lens. Performances are evaluated by computing the residuals of the errors in the reprojection of the corners onto the image and by analyzing the distributions of such errors with respect of the zenith and azimuth angles related to the vertices.
Piecewise-AM models, which feature a piecewise $z_0$ function, manage to obtain lower reprojection errors than Scaramuzza's model, especially when dealing with high zenith angles, where the former model had its main difficulties.

# Contents

# Listing of figures

# Listing of tables

# Listing of acronyms

| | |
|---|---|
| **FTC** | Fundamental Theorem of Calculus |
| **FoV** | Field of View |
| **SVD** | Single Value Decomposition |
| **MLE** | Maximum Likelihood Estimate |
| **MSE** | Mean Squared Error |
| **HH** | Hyper-Hemispherical |
| **SSRE** | Sum of Squared Reprojection Errors |
| **AM** | A-central Model |

# 1 Introduction

Very wide-angle lenses, characterized by a field of view exceeding 100 degrees, have the capability to project objects embedding a space as vast as a hemisphere onto their focal plane [1]. These lenses operate as inverted telephoto lenses, leading to considerable image distortion in the focal plane. The distortion arises from the fact that chief ray angles on the object side undergo alteration as they pass through the optics preceding the aperture stop.
Consequently, the extent of distortion can be so substantial that it is often not explicitly acknowledged by optical designers, rendering the focal length less meaningful. Certain researchers assert that lenses with extreme field of view, such as fisheye lenses, no longer fit within the conventional very-wide-angle-lens category but instead form an independent class [2].
Nevertheless, despite their drawbacks, very wide angle lenses are gaining popularity, largely attributed to the availability of cost-effective large-area digital sensors.
These sensors enable the manipulation of the inherently distorted ”native” wide field image to produce a comprehensible output for the lens user, even in real-time, making them highly appealing for numerous applications.
The most well-known very wide angle lens, the fisheye lens, possesses the ability to capture a hemispheric space (360° in azimuth angle and approximately 180° in zenith angle). Authors have also conceived fisheye lenses with extreme field of view in both the visible and thermal infrared ranges [3]. In addition to the fisheye, another contemporary lens type is the omnidirectional lens, capable of recording images at 360° in azimuth and tens of degrees above and below the horizon. Nevertheless, conventional omnidirectional lenses have a blind spot in the middle of field of view, around the boresight (which correspond to low zenith angles), yielding a ”donut-shaped” image.
To address this limitations INAF Padova observatory designed the PANCAM lens based on hyper hemispheric model [1]. Its design merges the capabilities of fish-eye and omnidirectional lenses to create a lens system able to capture a space of 360° in azimuth and 130° in zenith angle. The geometrical calibration of this specific lens is the core of the thesis.
Lenses with similar capabilities where historically calibrated thanks to a model proposed by Davide Scaramuzza, which represents the starting point of this work. This model designed for generic hyper-hemisheric lenses [4]

turned out to be not applicable to PANCAM lens, as assessed by Monica Beghini [5]. She tested this model over a set of images each containing a chessboard, of known size, position and orientation with respect of the camera. Through her work, she found that this model struggles when handling points with high zenith angles, which are subject to significant reprojection errors.
The models introduced throughout this discussion assume that not all light rays converge towards a single point, defined by Scaramuzza as the centre of the coordinate system of the camera, given that the camera under consideration is a non-central one.
The relevance of the models is evaluated by computing the residuals of the errors committed by the reprojection of the corners onto the images.
The results obtained by the piecewise-AM models significantly decrease the reprojection errors, showing that these models manage to properly handle corners associated to high zenith angles, too.
The discussion follows this subdivision:

- chapter 2) Hyper-hemispheric cameras are introduced, with a particular focus on PANCAM lens.
- chapter 3) Pinhole projection camera model and Scaramuzza's model are described, as they are the basis on which future camera models are defined. A clear explanation of all the passages involved in the calibration procedure for Scaramuzza's model is provided.
- chapter 4) The toolbox created by Scaramuzza for the calibration of hyper-hemispheric cameras is explained in detail, together with the most important MATLAB functions involved.
- chapter 5) New models represent the core of the work. They are discussed together with the modifications needed in the MATLAB codes to implement theses changes.
  These models are then evaluated by computing the residuals of the errors in the reprojection of the points onto the images.

.

# 2
# PANCAM camera

The content of this chapter regards the PANCAM camera, that is the lens used to acquire images and whose calibration is the core of the discussion. The chapter starts introducing the idea beyond hyper-hemispheric lenses, then follows the explanation of how the lens works, with figures to help the understanding.
The final part of the chapter focuses on applications of the lens, regarding the DAEDALUS project.

## 2.1 Hyper-hemispheric lenses

Panoramic omnidirectional lenses work as depicted in 2.1. Let's define the "horizon" as a flat plane that intersects the lens and is perpendicular to the zenith axis. The Zenith angle, denoted as Z, is measured from the zenith downward to the maximum field of view, Zmax. The lens is designed to capture a panoramic view extending $360^\circ$ around the azimuth axis, along with a range of degrees both above and below the horizon (as shown in panel a). Typically, an omnidirectional lens operates effectively within the Zmin range of $30^\circ$ to Zmax range of $135^\circ$.

Moving to the focal plane image (as displayed in panel b), it takes on the distinctive donut shape, with its inner rim corresponding to Zmin and the outer rim aligning with Zmax. The physical dimensions of this "donut" are determined by the lens's focal length in the paraxial region and the lens mapping function, which we will discuss in more detail later in this paper. Notably, the area surrounding the zenith, referred to as the frontal field, is projected onto the focal plane as a blind spot, occupying the central part of the donut.

It's important to recognize that the central hole in the "donut" shown in figure 2.1 has a disadvantage because it means that a portion of the sensor remains unused. As a point of reference, it's worth mentioning that a fish-eye lens covers the range from Z = 0○ up to approximately Z = $90^\circ$, encompassing the entire horizon.

For generic narrow or wide cameras the pupil (which can me considered the projection center ) can be considered unique for all the field of view. Different is the case of the Hyper hemispheric lenses where pupil position and projection orientation chance with the zenith angle. As the field angle Z widens, the entrance pupil starts to

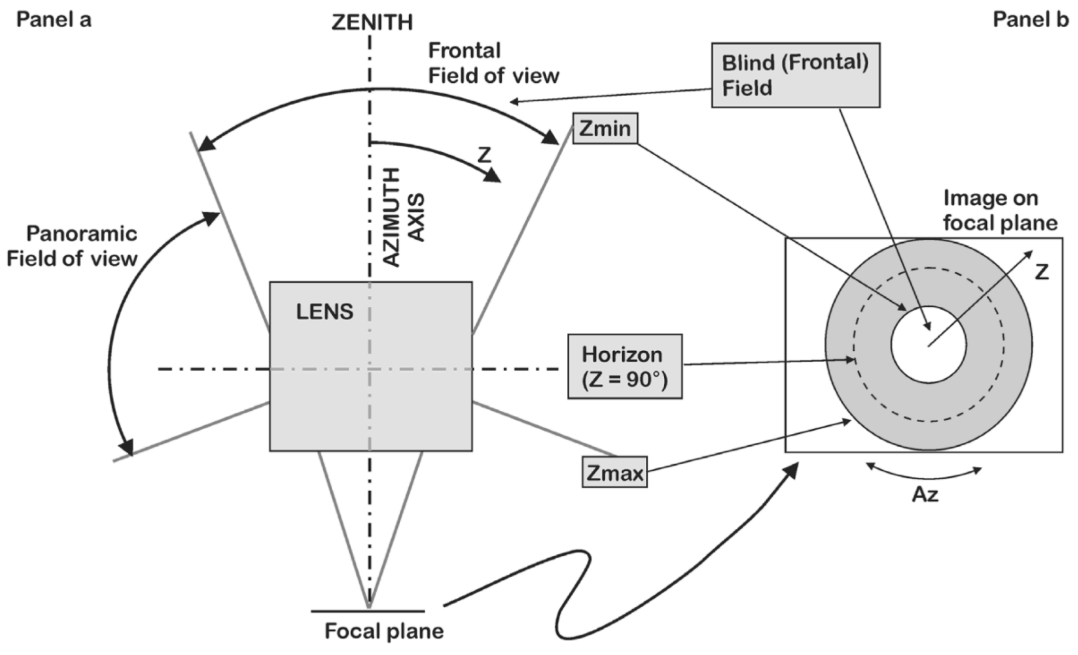


**Figure 2.1:** Omnidirectional lens and how images are made.

appear tilted, and at the extreme angle of 90°, it becomes entirely obscured. To enable the observation of objects in wide Z fields, the optical designer must adjust the orientation of the entrance pupil. This adjustment can be achieved by shifting and compressing the pupil, although it comes at an associated cost.
The HH lens described in [1] is able to portray on the image both the panoramic and the frontal field, as shown in figure 2.2. The lens comprises three distinct logical segments: a fore-optics section (located before the aperture stop, AS), a lens responsible for capturing the frontal field (FO), and an objective lens positioned after the aperture stop (OBJ), which is responsible for forming the field image on the focal plane.

The fore-optics section consists of a catadiopter (C) featuring a reflective concave surface and a lens designed to reduce the speed of incoming light beams (SD) to ensure they enter the OBJ through the AS at an appropriate speed. Within the panoramic field, which is defined by the chief rays 2 and 3, these rays enter the catadiopter. They refract at its first surface, then reflect from the concave surface, and subsequently pass through the SD optics. One surface of the SD lens is partially reflective, causing half of the light to redirect back into the catadiopter before crossing the AS and entering the OBJ for imaging on the focal plane.

On the other hand, the frontal field takes the form of a cone with its axis pointing toward the zenith and extending downward to encompass chief ray 1. This frontal field enters the FO, then passes through the SD (similarly, only half of the light passes through SD, with the rest being lost), C, AS, and OBJ, ultimately forming an image on the focal plane, precisely within the central hole of the donut. In the right panel, the primed numbers indicate the chief rays (1, 2, 3), along with reference points for objects 4 and 5.

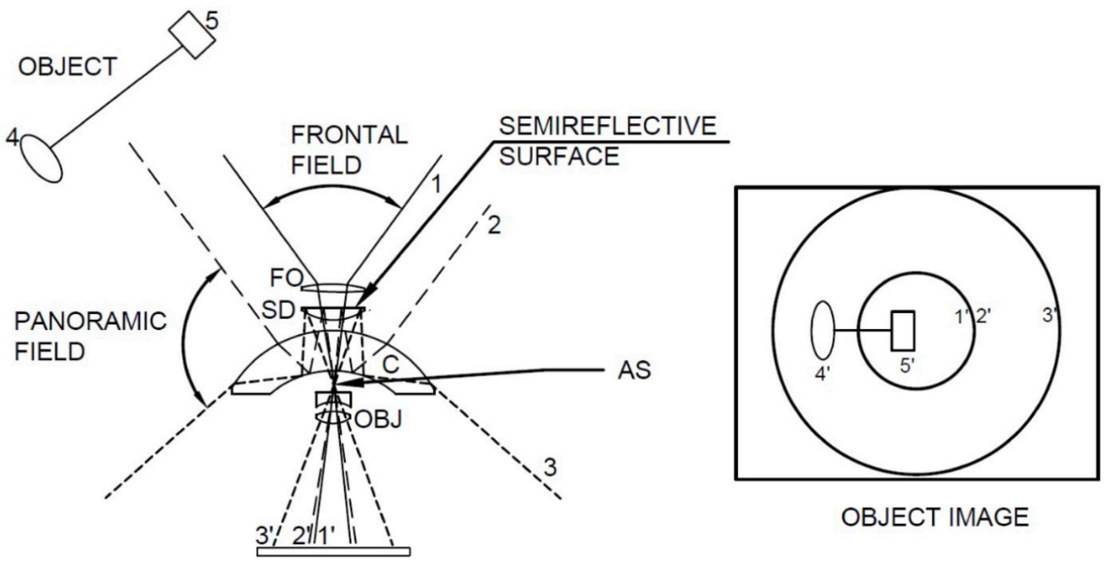


**Figure 2.2:** HH lens and how images are made

## 2.2 An application for HH lens

In August 2019, the European Space Agency (ESA) initiated the Sysnova Lunar Caves Challenge, with the aim of soliciting innovative proposals for addressing the exploration, mapping, and documentation of lunar caves or tubes. Among the selected proposals, one developed through collaboration between the University of Wurzburg, Jacobs University Bremen, the University of Padua, INAF Padova, and VIGEA stood out. This proposal introduced the DAEDALUS Sphere, which stands for "Descent And Exploration in Deep Autonomy of Lava Underground Structures." Lave tubes are large underground voids which represent a source of interest for the science community since Kaguya, LRO and GRAIL missions have unequivocally underlined the presence of deep voids below the Lunar surface [6]. These subsurface voids could be the base for long-term presence in human exploration framework because:

- represent a natural shield from possible impacts of micrometeorites and cosmic rays (here the equivalent radiation is the $0.3\%$ of the surface one);
- are characterized by stable temperature environments;
- might provide access to resources such as possible sources of water.

DAEDALUS mission have as aim to enter and explore the skylight named Marius Hills Hole in Oceanus Procellarum ($303.3E, 14.2N$), which is believed to be an access point to one of the lunar lava tubes [7].

The DAEDALUS Sphere is a spherical probe comprised of two primary components. The outer structure is constructed from polyimide and serves to shield the instruments and electronics from dust contamination while allowing relevant wavelengths for the instruments to pass through. The inner structure houses various instruments, including four LIDARs and the DAEDALUS-CAM, which incorporates four bifocal panoramic lenses

(BLPs). These BLPs are a specialized application of the hyper-hemispheric lens (HH lens) and were already applied in the design of a stereo system for another lunar rove, the PANROVER. In that case the system was based on a vertical baseline hosting two BPLs. In the DAEDAULS case the BPLs are four and the stereo acquisition can be performed during the descending phase e moving the rover sphere. As described in [8], the two ultra-wide field of view optical objectives are both composed of an objective with two optical components:

- a catadioptric element (with a reflective concave surface) for the panoramic fields (PF);
- a secondary lens (fore optics) for the frontal field (FF).

# 3 Pinhole projection and Scaramuzza's model

The calibration of a camera is an important passage to derive useful pieces of information, starting from 2D images. In this chapter the topic presented is the theory concerning central cameras and lenses with a very high FoV (field of view). This concepts are the building blocks of the model presented by Davide Scaramuzza in 2006, which deals with the geometric calibration of lenses with 360° angle on the azimuth and 90° or more on the zenith.
To sum up the contents of this chapter, we will start the discussion introducing central omnidirectional cameras. After that, the focus will shift to the *Pinhole Camera Projection Model*, the basic theoretical model for the calibration of cameras, in which all light rays converge towards a single point.
Sections continue with a description of the model introduced by Davide Scaramuzza in [4], which consists in an evolution of the classic pinhole camera model which comprehends a radially symmetric mirror. This addition makes the model more suitable for hyper-hemispheric lens calibration. All the calibration steps proposed by Scaramuzza are then analyzed and commented.

## 3.1 Central Omnidirectional Cameras

A vision system is characterized as having a central configuration when the optical rays originating from observed objects intersect at a singular 3D point referred to as the projection center or single effective viewpoint.

This property is termed the *single effective viewpoint* property. The perspective camera serves as an exemplar of a central projection system, wherein all optical rays converge at a solitary point, specifically the camera's projection center.

In the domain of contemporary fish-eye cameras, the attribute of being central holds true, thereby satisfying the single effective viewpoint property. In contrast, the realization of central catadioptric cameras hinges on the deliberate selection of mirror shape and and the extrinsic mounting of the camera respect with the mirror. The subset of mirrors conforming to the single viewpoint property encompasses the category of rotated (swept) conic sections: hyperbolic, parabolic, and elliptical mirrors. In instances involving hyperbolic and elliptical mirrors, the attainment of the single viewpoint property is contingent upon aligning the camera center (i.e., the pinhole or lens center) with one of the foci of the hyperbola or ellipse. In the context of parabolic mirrors, the insertion of an orthographic lens between the camera and mirror is requisite. This arrangement facilitates the convergence of parallel rays, reflected by the parabolic mirror, toward the camera center.

The paramount desirability of a single effective viewpoint emanates from its capacity to facilitate the generation of geometrically accurate perspective images from omnidirectional camera captures. This feasibility arises due to the imposition of the single view point constraint, wherein each pixel within the sensed image assesses the luminous flux traversing the viewpoint in a distinct direction. In cases of calibrated omnidirectional cameras, wherein the camera's geometry is well-defined, the direction of the versor representing the chief ray associated to each pixel can be precomputed.

Consequently, the luminance value measured by each pixel can be projected onto a plane positioned at any distance from the viewpoint, yielding a local plane perspective image or , more adequately, the image can be mapped onto a sphere centered around the singular viewpoint, thus resulting in a spherical projection.
Two different planes are considered in the models:

- image plane: an imaginary plane that represents the actual display screen through which a user views a virtual 3D scene;
- sensor plane: the plane associated to the digital sensor, which is made by pixels.

In addition to these entities, it is also useful the concept of *homogeneous coordinates*. Let $a \in \mathbb{R}^n$ such that $a = [a_1, \dots, a_n]^T$. The expression of a in homogeneous coordinates is given by the (n+1)-tuple of elements $[a'_1, \dots, a'_n, a'_{n+1}]^T$ which satisfies the following relation:

$$a_i = \frac{a'_i}{a'_{n+1}}, \ \forall i \in [1 \dots n].$$

## 3.2 Pinhole Camera Projection Model

A pinhole camera is a basic camera that lacks a lens and features a singular small aperture. Light rays traverse this aperture and form an inverted image on the opposite end of the camera. To obtain an upright image, we posit the placement of the image plane before the optical center C, which functions as the origin of the camera's coordinate system. To be precise, the image plane is positioned at a distance f from the optical center, oriented perpendicularly to the optical axis, where f is the focal length.
Figure 3.1 shows the simple behaviour of the projection of a world point (**X**) onto the image plane, according to the pinhole model.
Let's define a world coordinate system. Now a transformation is needed, in order to determine the position of a 3D point in the camera coordinate system.
Let M a world point, in homogeneous coordinates, such that $M = [M_x, M_y, M_z, 1]^T$. Then, the relationship

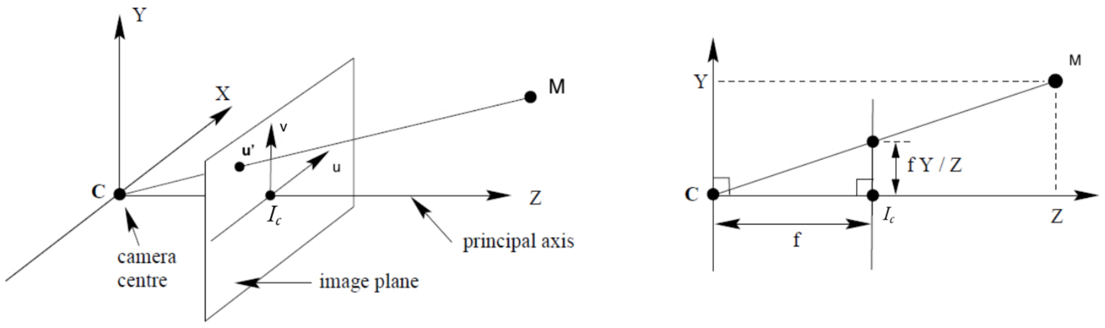


**Figure 3.1:** Example of how projection onto the image plane works

between the world and the camera coordinate systems is found by this equation:

$$\begin{bmatrix} M'_x \\ M'_y \\ M'_z \\ 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} M_x \\ M_y \\ M_z \\ 1 \end{bmatrix} \tag{3.1}$$

with $M' = [M'_x, M'_y, M'_z, 1]^T$ the expression of M in the camera coordinate system.
The matrix in the equation 3.1 defines a roto-translation and its elements are called *extrinsic parameters*. In this way, we can define the orientation and the location of the camera, with respect to a known reference system.
At this point, we need another transformation, the one which describes the projection of M onto the image plane. This coordinate system associated to the image plane has its center in the intersection between the plane and the Z axis ($I_c$) and two axis (u,v) to whom two coordinates are related. Let $\mathbf{u}' = [u', v']^T$ the projection of M onto the image plane.
From the figure on the right side of 3.1, it is easy to get that $v'$ =f$\cdot\frac{M'_y}{M'_z}$, since triangles C-$I_c$-v' and C-$M'_z$-$M'_y$ are similar. Reasoning in the same way, we get that $u'$ =f$\cdot\frac{M'_x}{M'_z}$. By expressing **u**' in homogeneous coordinates, the projection can be seen as a linear function:

$$\mathbf{u}' = \begin{bmatrix} u' \\ v' \\ 1 \end{bmatrix} = \begin{bmatrix} f \cdot M'_x \\ f \cdot M'_y \\ M'_z \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} M'_x \\ M'_y \\ M'_z \\ 1 \end{bmatrix} = P_f \cdot \begin{bmatrix} M'_x \\ M'_y \\ M'_z \\ 1 \end{bmatrix} \tag{3.2}$$

It is possible to merge equations 3.1 and 3.2, in order to get the relationship between a point and its projection:

$$\mathbf{u}' = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} M'_x \\ M'_y \\ M'_z \\ 1 \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} M_x \\ M_y \\ M_z \\ 1. \end{bmatrix} \tag{3.3}$$

By imposing $f = 1$, 3.3 can be rewritten as:

$$\mathbf{u}' = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \\ 0 & 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} M_x \\ M_y \\ M_z \\ 1 \end{bmatrix} = \begin{bmatrix} r_{11} & r_{12} & r_{13} & t_x \\ r_{21} & r_{22} & r_{23} & t_y \\ r_{31} & r_{32} & r_{33} & t_z \end{bmatrix} \begin{bmatrix} M_x \\ M_y \\ M_z \\ 1 \end{bmatrix} = [\mathbf{R}|\mathbf{t}] \cdot M. \qquad (3.4)$$

After having defined the passage from a 3D point to its projection on the image plane, the following step is to define the the transformation which maps a point of the image plane onto a pixel, belonging to the sensor plane. As exposed in [9], let **u**" the pixel associated to **u**'. Then consider:

- The sensor scales $\alpha$ and $\beta$, which may be distinct in the x- and y-directions, respectively.
- The location of the center of the image $O_c = (u''_c, v''_c)$ with respect to the image coordinate system (i.e., the optical axis). $O_c$ is the so called "principal point" (intersection between the image plane and the principal axis ,Ic in figure 3.1)
- The presence of diagonal distortion, denoted as $\gamma$, within the image plane. This aspect is typically minimal or negligible [10].

Given these premises, we can write:

$$\mathbf{u}'' = \begin{bmatrix} u'' \\ v'' \end{bmatrix} = \begin{bmatrix} \alpha & \gamma & u''_c \\ 0 & \beta & v''_c \end{bmatrix} \begin{bmatrix} u' \\ v' \\ 1 \end{bmatrix} = A \begin{bmatrix} u' \\ v' \end{bmatrix} + \begin{bmatrix} u''_c \\ v''_c \end{bmatrix} \qquad (3.5)$$

The elements $\alpha$, $\beta$, $\gamma$, $u''_c$ and $v''_c$ are called *intrinsic parameters* [11]. Generally speaking, intrinsic parameters encompass the optical, geometric, and digital attributes of the camera (including focal length and pitch dimension), elucidating the manner in which a spatial point gets projected into a pixel on the image plane, that is, the imaging process.

## 3.3 Imaging function in Scaramuzza's camera model

In this section, the model proposed is the one introduced by Davide Scaramuzza, which starts from the pinhole projection model, previously discussed. This model introduces a new function ($g$), called imaging function, which represents a mirror.

Figure 3.2 shows how the projection works under Scaramuzza's model: the point M is projected onto the center of the camera coordinate system, through vector **p**. This vector intersects the mirror (the curve in the figure) in a point. Starting from this point, in a natural way we get the projection of M onto the image plane and the projection on to the sensor plane, too. As we will discuss in the next lines, the shape of the curve is defined by the function g itself.

As it was previously discussed, the relationship between a point on the image plane and its projection on the sensor plane is described by an affine transformation, i.e.:

$$\mathbf{u}'' = A\mathbf{u}' + \mathbf{b}\,, \qquad (3.6)$$

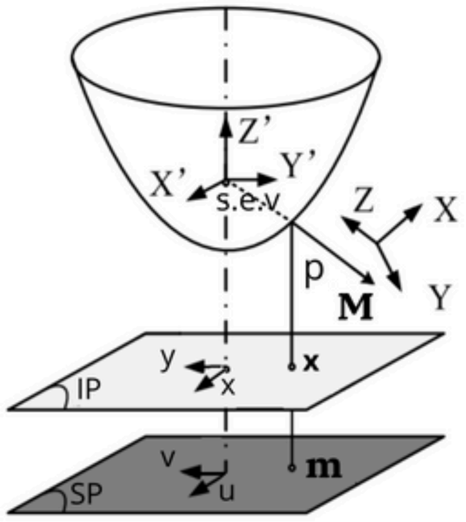


**Figure 3.2:** Camera coordinate system

where $A \in \mathbb{R}^{2x2}$ and $\mathbf{b} \in \mathbb{R}^2$. This process aims to fix some mis-alignments due to errors in the manufacturing of the camera, such as the non perfect orthogonality between the sensor plane and the z-axis, which could introduce an homography here approximated as an affine trasformation, and also incorporates the creation of the digital 2D image.
Now let M a point in the external world. M can be expressed using a real-world homogeneous coordinate system, such that $M = [x_M, y_M, z_M, 1]^T$. Let **u'** the projection of M onto the camera image plane and **u**" its projection onto the sensor plane. The process of associating a point of the sensor plane to a vector going from the pinhole to the 3D point can be summed up with a function $g$, and it holds:

$$\lambda \cdot \mathbf{p} = \lambda \cdot g(\mathbf{u}'') = \lambda \cdot g(A\mathbf{u'} + \mathbf{b}) = P \cdot M \ = [\mathbf{R}|\mathbf{t}] \cdot M, \tag{3.7}$$

with $P \in \mathbb{R}^{3x4}$ and $\lambda > 0$. The coefficient $\lambda$ lets all the point belonging to the straight line of direction $(M - O)$ to be mapped onto the same pixel. This relation is fundamental since we are representing 3D entities on a 2D limited space. Last equality states that the $P$ matrix represents a roto-translation.
The process known as calibration consists in defining the matrix $A$, the vector $b$ and the nature of imaging function (g), such that for each point the equation 3.7 holds.
In order to better understand the meaning and the usage of the $g$ function, figure 3.3 resumes the behaviour of the projection from what was explained regarding figure 3.2. Here we have two different systems, which differ in the shape of the mirror, the blue curve.
The two curves are different parabolas, and it is easy to observe that the reprojections onto the image plane of the points $M_1$ and $M_2$ change is different in the two pictures.

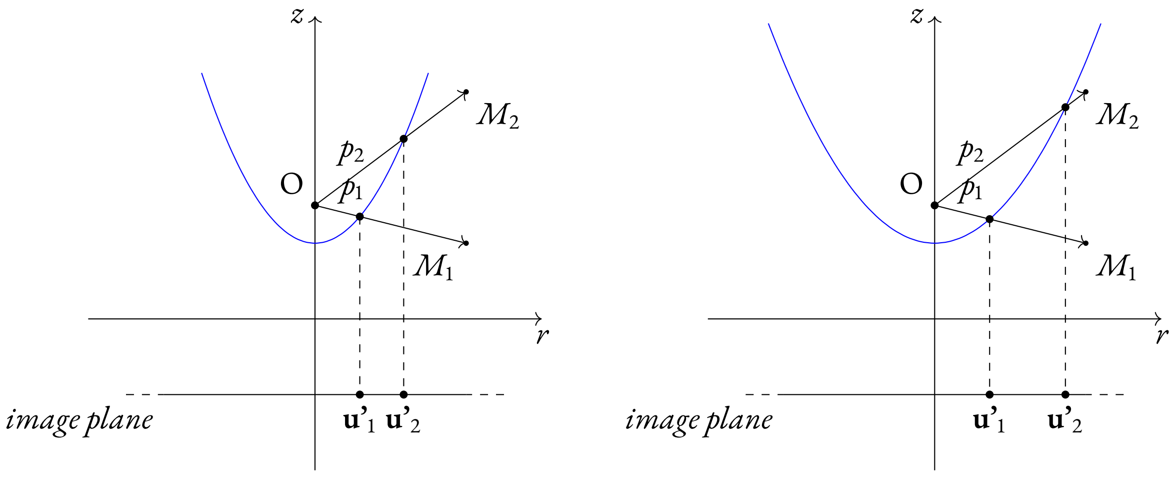


**Figure 3.3:** Pinhole projection of a point M onto the image plane

How can we relate the imaging function with the shape of the mirror? The function $g$ is defined such that:

$$\begin{aligned} g: \quad & \mathbb{R}^2 \longrightarrow \mathbb{R}^3, \\ & (u'', v'') \mapsto (u'', v'', f(u'', v'')) \end{aligned} \tag{3.8}$$

where $f$ is radially symmetric with respect to the optical axis $z$. This means that $f = f(\rho)$, with $\rho = \sqrt{u''^2 + v''^2} = \|\mathbf{u}''\|$. Moreover, the function $f(\rho)$ should be monotonic increasing. The function $f$ wants to describe the blue shape of 3.3.

According to [4], $f$ is defined as a polynomial of grade N, whose coefficients are additional intrinsic parameters for the calibration. This polynomial can be seen as a truncated form of the Taylor series which would define the ideal f, the one which perfectly describes the mirror.

Given these premises, we can rewrite equation 3.7 as:

$$\lambda \cdot \mathbf{p} = \lambda \cdot g(A\mathbf{u}'' + \mathbf{b}) = \lambda \cdot \begin{bmatrix} A\mathbf{u}' + \mathbf{b} \\ f(\mathbf{u}'') \end{bmatrix} = P \cdot M. \tag{3.9}$$

## 3.4 Calibration procedure

In this section, we are going to describe:

- The first initialization (through linear estimation) of both intrinsic and extrinsic parameters (section 3.4.1);
- refinement of extrinsic parameter followed by the improvements of the intrinsic one including the center of the image not consider previously (section 3.4.2);
- algorithm LM applied for the final definition of all the parameters to minimize the residual lined to the reprojections of the tie points detected (section 3.4.3).

### 3.4.1 Initialization of the parameters

As we have previously stated, with the term calibration we mean the extraction of the parameters appearing in equation 3.7.
In order to simplify the parameter estimation process, we undertake the computation of matrix A and **b**, adjusted by a scale factor $\alpha$. This is achieved by transforming the ellipse defining the view field into a centered circle. The process of transformation is automated and facilitated by an ellipse detector when the circular outer boundary of the sensor is discernible within the image. Subsequent to the execution of the affine transformation, an image point **u'** finds its connection to the corresponding point on the sensor plane **u**" via the relation $\mathbf{u}'' = \alpha \cdot$ **u'**. Now equation 3.7 can be rewritten as

$$\lambda \cdot \mathbf{p} = \lambda \cdot g(\alpha \cdot \mathbf{u}'') = \lambda \cdot \begin{bmatrix} \alpha u' \\ \alpha v' \\ f(\alpha \cdot \rho') \end{bmatrix} = \lambda \cdot \alpha \cdot \begin{bmatrix} u' \\ v' \\ \sum_{k=0}^{N} a_k \cdot \rho^k \end{bmatrix} = P \cdot M. \tag{3.10}$$

Moreover, the factor $\alpha$ can be incorporated to $\lambda$, so that only the coefficients $\{a_k\}_{k=0}^{N}$ of the polynomial are considered as intrinsic parameters.
It is also possible to reduce the extrinsic parameters, i.e. the elements of the matrix P = $[R|\mathbf{t}]$. If generally *extrinsic* parameters represent the position of the camera in the global system the chessboard pattern calibration procedures are based on the acquisition of chessboard target covering all the field of view, as shown in pictures 3.4. Each calibration chessboard defines a precise orientation and position respect with the camera.
Hereafter we always will refer to "extrinsic parameters" the rotation matrices $[R|\mathbf{t}]$ which define the chessboard reference system respect with the camera . In fact, data involved in the calibration procedure are images which contain 2-dimensional patterns, in the specific case of this project they have been considered black-and-white chessboards, whose internal vertices made up the pattern of interest. Consider the set of $n$ images considered for

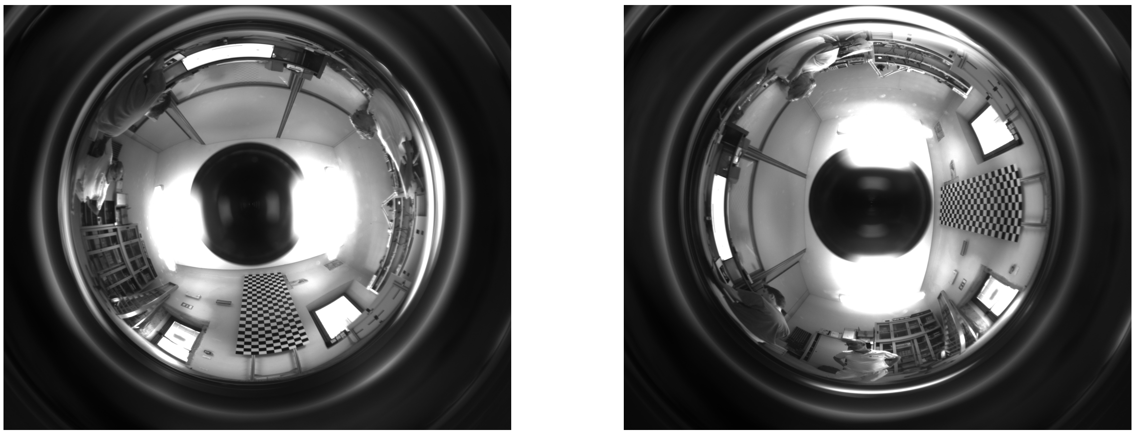


**Figure 3.4:** Examples of images acquired for the calibration procedure, here with chessboards 28 squares long and 10 squares wide.

the calibration $\{I_i : i = 1...n\}$. Let each image contain $m$ points which make up the pattern. Then the j-th corner of the i-th image can be written as $M_{ij} = [X_{ij}, Y_{ij}, Z_{ij}]$ in the pattern coordinate system. One of the reason of the use of chessboard targets in the great part [12] of the calibration methods of projective system is the simplification introduced by the fact that the pattern is planar and so we have $Z_{ij} = 0$. Then equation 3.10 turns into

$$\lambda_{ij} \cdot p_{ij} = \lambda_{ij} \cdot \begin{bmatrix} u \\ v \\ \sum_{i=0}^{N} a_i \cdot \rho_{ij}^i \end{bmatrix} = P^i \cdot M = [\mathbf{r}_1^i\, \mathbf{r}_2^i\, \mathbf{r}_3^i\, \mathbf{t}^i] \cdot \begin{bmatrix} X_{ij} \\ Y_{ij} \\ 0 \\ 1 \end{bmatrix} = [\mathbf{r}_1^i\, \mathbf{r}_2^i\, \mathbf{t}^i] \cdot \begin{bmatrix} X_{ij} \\ Y_{ij} \\ 1 \end{bmatrix}. \tag{3.11}$$

Each image owns a set of extrinsic parameters, given that all of them have a different pattern coordinate system. The first step necessary to solve the equation is to get rid of the $\lambda$ parameter, the depth scale. So both sides of the equation 3.11 are multiplied vectorially by $\mathbf{p}_{ij}$:

$$\lambda_{ij} \cdot \mathbf{p}_{ij} \wedge \mathbf{p}_{ij} = \mathbf{p}_{ij} \wedge [\mathbf{r}_1^i\, \mathbf{r}_2^i\, \mathbf{t}^i] \cdot \begin{bmatrix} X_{ij} \\ Y_{ij} \\ 1 \end{bmatrix}. \tag{3.12}$$

In this last equation, the first term is 0, which brings the equation to the form

$$\underline{0} = \begin{bmatrix} u_{ij} \\ v_{ij} \\ \sum_{k=0}^{N} a_k \cdot \rho^k \end{bmatrix} \wedge [\mathbf{r}_1^i\, \mathbf{r}_2^i\, \mathbf{t}^i] \cdot \begin{bmatrix} X_{ij} \\ Y_{ij} \\ 1 \end{bmatrix}. \tag{3.13}$$

This equation can be split into 3 different homogeneous equations:

$$v_{ij} \cdot (r_{31}X_{ij} + r_{32}Y_{ij} + t_3) - f(\rho_{ij}) \cdot (r_{21}X_{ij} + r_{22}Y_{ij} + t_2) = 0 \tag{3.14}$$

$$f(\rho_{ij}) \cdot (r_{11}X_{ij} + r_{12}Y_{ij} + t_1) - u_{ij}(r_{31}X_{ij} + r_{32}Y_{ij} + t_3) = 0 \tag{3.15}$$

$$u_{ij}(r_{21}X_{ij} + r_{22}Y_{ij} + t_2) - v_{ij}(r_{11}X_{ij} + r_{12}Y_{ij} + t_1) = 0 \tag{3.16}$$

Coordinates of the planar pattern points ($[X_{ij}, Y_{ij}]^T$) and coordinates on the sensor plane ($[u_{ij}, v_{ij}]^T$)are known and it is possible to notice that the last equation only considers extrinsic parameters.
By stacking $m$-times equation 3.16, one for each point belonging to the pattern in the i-th image, we get the homogeneous linear system

$$\mathbf{U} \cdot \mathbf{h} = 0, \tag{3.17}$$

with

$$\mathbf{U} = \begin{bmatrix} -v_{i1}X_{i1} & -v_{i1}Y_{i1} & u_{i1}X_{i1} & u_{i1}Y_{i1} & -v_{i1} & u_{i1} \\ \vdots & \vdots & \vdots & \vdots & \vdots & \vdots \\ -v_{im}X_{im} & -v_{im}Y_{im} & u_{im}X_{im} & u_{im}Y_{im} & -v_{im} & u_{im} \end{bmatrix}$$

and

$$\mathbf{h} = [r_{11}, r_{12}, r_{21}, r_{22}, t_1, t_2]^T.$$

Equation 3.17 can be approximately solved by minimizing $||\mathbf{U} \cdot \mathbf{h}||^2$, with $||\mathbf{h}|| = 1$. This constraint avoids to fall in the case $\mathbf{h} = \underline{0}$.
The approach used to get the solution for $\mathbf{h}$ is applying an SVD decomposition. First of all, matrix U is decomposed into its Single-Value-Decomposition form, which means that $\mathbf{U} = \mathbf{WSV}^T$, with $\mathbf{W} \in \mathbb{R}^{mxm}$, $\mathbf{S} \in \mathbb{R}^{mx6}$ and $\mathbf{V} \in \mathbb{R}^{6x6}$. W and V are unitary orthogonal matrices and S is a rectangular diagonal matrix, whose non-0 elements are called *singular values*.
The solution turns out to be the right-singular vector of U corresponding to the smallest singular value, which corresponds to the last row of matrix $V$.
Given the constraint stated above, the solution of 3.17 is known up to a scale factor, which can be determined by including that $\mathbf{r}_i 1$ and $\mathbf{r}_2$ must be orthonormal. We can also computed $\mathbf{r}_3$, always because of orthonormality.
With this passage, all the extrinsic parameters are found, with the exception of $t_3$, for all the calibration images. The following step focuses on intrinsic parameters, i.e the coefficients $a_0, \ldots, a_N$ that define the shape of the imaging function.
By rewriting the function $f(\rho)$ using its polynomial form, equations 3.14 and 3.15 become:

$$v_{ij} \cdot (r_{31}X_{ij} + r_{32}Y_{ij} + t_3) - \left(\sum_{k=0}^{N} a_k \cdot \rho^k\right) \cdot (r_{21}X_{ij} + r_{22}Y_{ij} + t_2) = 0 \tag{3.18}$$

$$\left(\sum_{k=0}^{N} a_k \cdot \rho^k\right) \cdot (r_{11}X_{ij} + r_{12}Y_{ij} + t_1) - u_{ij}(r_{31}X_{ij} + r_{32}Y_{ij} + t_3) = 0 \tag{3.19}$$

Following the same procedure as described earlier, we gather all the unspecified elements from equations 3.18 and 3.19 and arrange them into a vector, which allows us to rephrase the equations as a linear system. However, in this iteration, we extend this approach to encompass all n images. in addition to the simplifications implemented at the beginning of this section, [13] also worries about the shape of the imaging function. In fact, if we desire to have a function $f(\rho) \in C^1$, it is needed the constraint

$$\left.\frac{df(\rho)}{d\rho}\right|_{\rho=0} = 0 \tag{3.20}$$

which implies that $a_1 = 0$. The resultant system takes the form:

$$\begin{bmatrix} A_1 & A_1\rho_1^2 & \ldots & A_1\rho_1^N & -v_1 & 0 & \ldots & 0 \\ C_1 & C_1\rho_1^2 & \ldots & C_1\rho_1^N & -u_1 & 0 & \ldots & 0 \\ \vdots & \vdots & \ldots & \vdots & \vdots & \vdots & \ldots & \vdots \\ A_n & A_n\rho_n^2 & \ldots & A_n\rho_n^N & 0 & 0 & \ldots & -v_k \\ C_n & C_n\rho_n^2 & \ldots & C_n\rho_n^N & 0 & 0 & \ldots & -u_k \end{bmatrix} \cdot \begin{bmatrix} a_0 \\ a_2 \\ \vdots \\ a_N \\ t_3^1 \\ \vdots \\ t_3^n \end{bmatrix} = \begin{bmatrix} B_1 \\ D_1 \\ \vdots \\ B_n \\ D_n \end{bmatrix} \tag{3.21}$$

with

$$A_{ij} = r_{21}^i X_{ij} + r_{22}^i Y_{ij} + t_2^i$$

$$B_{ij} = v_{ij} \cdot (r^i_{31} X_{ij} + r^i_{32} Y_{ij})$$

$$C_{ij} = r^i_{11} X_{ij} + r^i_{12} Y_{ij} + t^i_1$$

$$D_{ij} = u_{ij} \cdot (r^i_{32} X_{ij} + r^i_{32} Y_{ij}).$$

System 3.21 does not show $j$ indices, referring to the points in a specific image, for sake of easily reading. In fact, the system is made up by $2mn$ equations, 2 for all the points belonging to the calibration pattern.
The least-squares solution of the previous system can be reached by using the pseudoinverse of the matrix.
The pseudoinverse (also known as Moore–Penrose inverse) of a matrix A ($A^\dagger$) is the generalization of the concept of inverse matrix for non-squared matrices and it can be easily computed starting from the SVD decomposition. Given $A = USV^T, A^+ = VS^\dagger U^T$, with

$$S = \begin{bmatrix} \sigma_1 & 0 & 0 & \dots & 0 & 0 & \dots & 0 \\ 0 & \sigma_2 & 0 & \dots & 0 & 0 & \dots & 0 \\ 0 & 0 & \sigma_3 & \dots & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots & \vdots & \dots & \vdots \\ 0 & 0 & 0 & \dots & \sigma_k & 0 & \dots & 0 \end{bmatrix} \qquad S^\dagger = \begin{bmatrix} \sigma_1 & 0 & 0 & \dots & 0 \\ 0 & \sigma_2 & 0 & \dots & 0 \\ 0 & 0 & \sigma_3 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & \sigma_k \\ 0 & 0 & 0 & \dots & 0 \\ \vdots & \vdots & \vdots & \ddots & \vdots \\ 0 & 0 & 0 & \dots & 0 \end{bmatrix}.$$

These steps provide a starting point for the choice of the calibration parameters, both extrinsic and intrinsic.

### 3.4.2 Parameters refinement

In the previous steps we have defined an initial estimate for our calibration parameters, following equations 3.14, 3.15 and 3.16. As reported by Davide Scaramuzza in [13], now parameters are refined through linear estimation. This first refinement passage is made up by 2 steps:

- Given the first approximation of intrinsic parameters $a_0, a_2, ..., a_N$, extrinsic parameters are computed by solving equations 3.14, 3.15 and 3.16. The challenge results in a linear homogeneous system, solvable through the application of Singular Value Decomposition (SVD), with a scale factor being present. Subsequently, the scale factor is exclusively ascertained by leveraging the inherent orthonormality between vectors $\mathbf{r}_1$ and $\mathbf{r}_2$.
- Given these new values for extrinsic parameters, coefficients of the polynomial are re-computed with the same procedure used for the their first estimation (now without focusing on $t_3$, which is refined together with the other extrinsic parameters).

As highlighted in the introductory part of section 3.4.1, we are assuming that the center of the sensor plane coincide with the center of the image plane. Consequently, we aspire to possess the ability to accurately determine the center of the omnidirectional image ($O_c$), even when the external boundary of the sensor isn't directly visible within the image.

To achieve this objective, we note that our calibration procedure effectively estimates the intrinsic parametric model when $O_c$ is treated as the origin of the image coordinates. Deviating from this, if $O_c$ is not considered the origin, and instead, the 3D points of the calibration pattern are backprojected into the image, a significant reprojection error would be observed concerning the calibration points. Drawing inspiration from this observation, we undertook multiple trials of our calibration procedure, varying the center locations. For each trial, we computed the Sum of Squared Reprojection Errors (SSRE). Notably, our findings consistently affirmed that the SSRE consistently attains a global minimum at the precise correct center location.

This outcome guides us towards an iterative search for the center $O_c$, which halts when the discrepancy between two potential center locations becomes smaller than a certain fraction of a pixel $\varepsilon$ (we reasonably set $\varepsilon = 0.5$ pixels):

1. At each iteration of this iterative search, a specific image region is uniformly sampled at a certain number of points.
2. For each of these points, calibration is executed, employing that point as a potential center location, and the Sum of Squared Reprojection Errors (SSRE) is calculated.
3. The point yielding the minimum SSRE is considered as a potential center.
4. The search continues by improving the sampling in the vicinity of that point, and steps 1, 2, and 3 are reiterated until the termination condition is met.

The last step of the refinement phase is the one which focuses more on the training data, that in this case are the images of the chessboards. In fact, our pattern points could be corrupted by independent and identically distributed noise. As we will better describe in following chapters, the detection of the vertices on the chessboards may be prone to errors, and even little mistakes in the labelling of the corners would move our situation away from the theory introduced in this chapter.
To tackle this issue, we can look for *Maximum Likelihood Estimate*, which can be obtained by minimizing the MSE:

$$MSE = \sum_{i=1}^{n}\sum_{j=1}^{m} \left|\left| \mathbf{u}''_{ij} - u''(\mathbf{R}^i, \mathbf{T}^i, a_0, a_2, ..., a_N, \mathbf{M}_{ij}) \right|\right|^2 \tag{3.22}$$

where $u''(\mathbf{R}^i, \mathbf{T}^i, a_0, a_2, \dots, a_N, \mathbf{M}_{ij})$ is the projection the i-th image of the 3D point $M_{ij}$, according to equation 3.10 and $\mathbf{u}''_{ij}$ is the location of the j-th point of the i-th image during the corner extraction phase.
The algorithm used to perform this is the *Levenberg-Marquadt*.

## 3.4.3 Levenberg-Marquadt algorithm

The algorithm chosen for the final optimization of both extrinsic and intrinsic parameters is the Levenberg-Marquadt method, which is described in a very precise form in [14].
The Levenberg-Marquardt algorithm is an iterative optimization technique used to solve non-linear least squares problems. It represents a meeting point between the classic *Gradient descent* and the *Gauss-Newton method*. The first one behaves better when the starting point is far from the local optimum, while the second speeds up the

achievement of the solution if starting from a point close to it.
Given the function general MSE function:

$$MSE = \sum_{i=1}^{n} ||y_i - \hat{y}(x_i, \mathbf{p})||^2 = (\mathbf{y} - \hat{\mathbf{y}}(\mathbf{p}))^T \cdot (\mathbf{y} - \hat{\mathbf{y}}(\mathbf{p})) \tag{3.23}$$

we get $\hat{\mathbf{y}} : \mathbb{R}^m \longrightarrow \mathbb{R}^n$, which takes in input a vector of parameter of size m ($\mathbf{p}$)and outputs a vector of size n (one for each data point). Let $\frac{d\hat{\mathbf{y}}(\mathbf{p})}{d\mathbf{p}} = \mathbf{J}$, the Jacobian of the function.
According to the Gradient Descent algorithm, in this case the parameter update $p_{GD}$ that moves the parameters in the direction of steepest descent is given by:

$$p_{GD} = -\alpha \cdot \mathbf{J}^T(\mathbf{y} - \hat{\mathbf{y}}) \tag{3.24}$$

In a similar manner, the parameter update given by Gauss-Newton method is the value solving the equation:

$$\left(\mathbf{J}^T\mathbf{J}\right) p_{GN} = \mathbf{J}^T(\mathbf{y} - \hat{\mathbf{y}}) \tag{3.25}$$

The Levenberg-Marquardt algorithm dynamically adjusts the parameter updates by alternating between the gradient descent update and the Gauss-Newton update:

$$\left(\mathbf{J}^T\mathbf{J} + \lambda\mathbf{I}\right) p_{LM} = \mathbf{J}^T(\mathbf{y} - \hat{\mathbf{y}}) \tag{3.26}$$

whereas small values of the damping parameter $\lambda$ lead to a Gauss-Newton update, larger values of $\lambda$ lead to a gradient descent update (in this case $\lambda$ would take the role of $\alpha$). The algorithm proceeds as follows:

1. Initialize an initial guess for the solution vector $\mathbf{p}$.
2. Compute the Jacobian matrix $J$ of at the current $\mathbf{p}$ point.
3. Solve the system: $\left(\mathbf{J}^T\mathbf{J} + \lambda\mathbf{I}\right) \Delta\mathbf{p} = \mathbf{J}^T(\mathbf{y} - \hat{\mathbf{y}})$, where $\Delta\mathbf{p}$ is the update to $\mathbf{p}$.
4. Compute the gain ratio $\rho$. The gain ratio is used to determine whether the proposed update is accepted or not. It's calculated as the ratio of the reduction in the objective function to the reduction predicted by a quadratic approximation.
5. If $\rho > \tau$, where $\tau$ is a threshold, accept the update ($\mathbf{p} = \mathbf{p} + \Delta\mathbf{p}$) and decrease the damping parameter. The damping parameter balances between the Gauss-Newton step and the Gradient Descent step, helping to handle ill-conditioned problems.
6. If $\rho \leq \tau$, reject the update and increase the damping parameter. This step prevents aggressive updates that could lead to divergence.
7. Repeat steps 2-7 until convergence or a maximum number of iterations is reached. Convergence is typically achieved when the changes in $\mathbf{p}$ become small or the objective function reaches a satisfactory value.

# 4
# Omnidirectional Camera Calibration Toolbox for Matlab

Davide Scaramuzza developed a useful Matlab toolbox for the calibration of an omnidirectional lens, which can downloaded for free at [15]. In this chapter, we are going to provide a comprehensive explanation of all the steps that make up the calibration procedure. The images involved portray black-and-white chessboards, whose internal corners are the points building up the pattern useful for the calibration procedure. Example of these images are 4.1 and 4.2, where we can notice the donut shape, with the blank space in the middle, which is a peculiarity given by the fact that our camera has a blind spot on the top.

The calibration procedure consists in these 5 passages:

1. loading images: pictures are captured and charged into the toolbox;
2. corners extraction: the internal vertices' locations are marked, for each chessboard;
3. first estimate of the parameters: intrinsic and extrinsic parameters are find by solving the equations in section;
4. find center: determine the location of the center of the image;
5. parameter refinement: parameters are fine-tuned through a non-linear optimization procedure. This passage is required since points might be subject to noise.

The toolbox and its functions are user-friendly and they are based on the class *C_calib_data*, which stores neatly all the parameters involved in the calibration procedure.
The following list comprehends the parameters which will be necessary to understand all the steps and the functions that will be discussed in this chapter:

- **ima_proc**: indices of images to be processed and not erased;

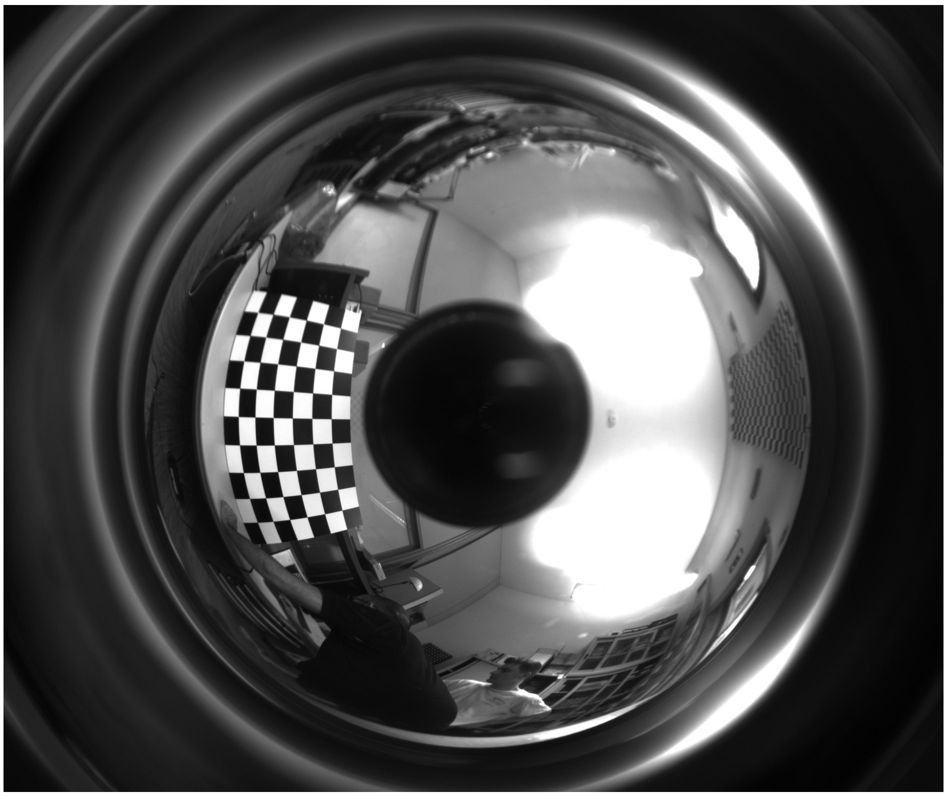


**Figure 4.1:** 7x10 chessboard, picture acquired by PANCAM lens

- **RRFin**: a tensor of dimensions 3x3xima_proc containing all the roto-translation matrices, which contain the extrinsic parameters for each grid;
- **Xt**, **Yt**: vectors containing metric coordinates of the corners;
- **Xp_abs**, **Yp_abs**: vectors containing the pixel coordinates of the corners;
- **dX**, **dY**: size of chessboard squares;
- **taylor_order**: grade of the polynomial $f(\rho)$, also known as $N$;
- **ocam_model**: an internal struct containing
  - ss: vector containing the coefficients of the polynomial;
  - xc, yc: pixel coordinates of the image center

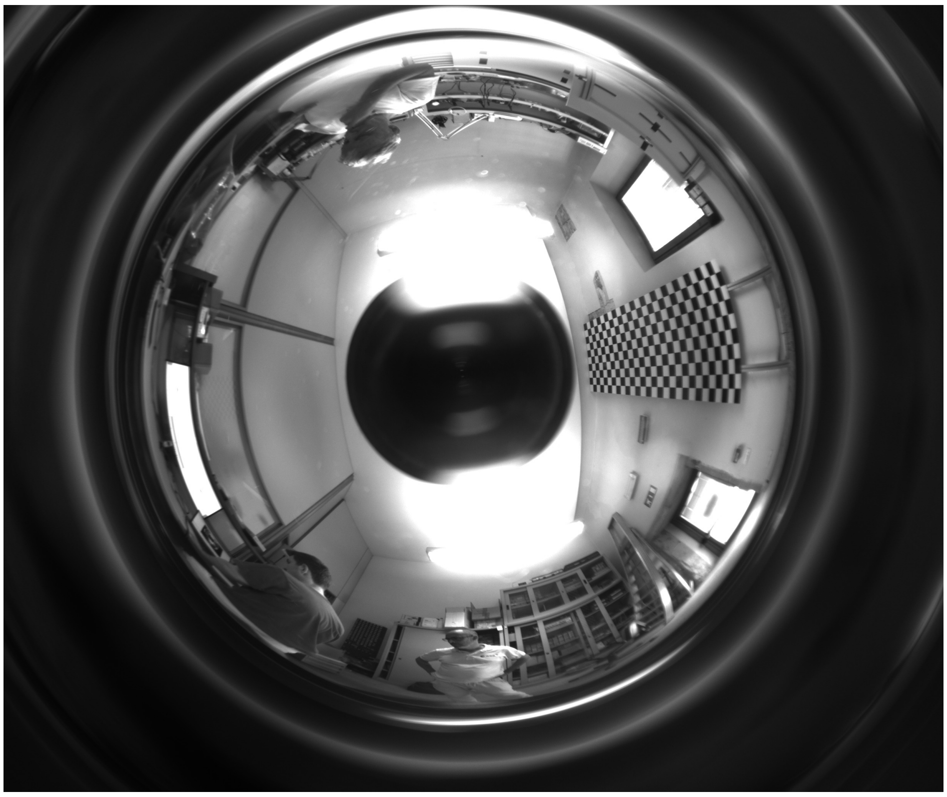


**Figure 4.2:** 28x10 chessboard, picture acquired by PANCAM lens

## 4.1 Loading images

Images are the data on which is based our model, and they directly influence the calibration coefficients. To attain accurate calibration results, it is recommended to position the checkerboard in close proximity to either the mirror or the fish-eye lens. This strategy not only enhances calibration accuracy but also boosts the chances of successful corner detection through the Automatic Checkerboard Extraction tool. It is imperative that every corner of the checkerboard remains visible in each image, and the presence of a white border encompassing the pattern is essential for optimal performance of the Automatic Checkerboard Extraction tool.
Furthermore, capturing images of the checkerboard from various perspectives is recommended to cover the entire visible area of the camera. This approach facilitates the compensation for potential misalignment between the camera and mirror axes during the calibration process. Additionally, capturing images from different angles aids in the automatic identification of the center of the omnidirectional image, a pivotal step in the calibration procedure.

## 4.2 Corners extraction

The extraction phase consists in identifying the location of the internal corners of the grids, automatically or manually. The internal corners are intended as the vertices of the squares, without considering the ones on the border. So given a checkerboards of dimension $ax \cdot b$, the considered squares are $(a - 2) \cdot (b - 2)$ and the internal corners are $(a - 1) \cdot (b - 1)$. To speed up the computations, the code incorporates an automated corner detection process based on the Harris corner detector [16], a widely used operator in computer vision algorithms for extracting corners and inferring image features. This algorithm [17] operates by analyzing the eigenvalues of the $2D$ discrete structure tensor matrix at each pixel in the image. A pixel is identified as a corner when the eigenvalues of its structure tensor exceed a certain threshold, ensuring a high-quality corner detection.
The software itself recommends to perform the manual extraction of the vertices, but using an automatic corner detector. This choice allows the user to check and also modify the locations of the corners identified by the toolbox. It is also possible to erase a whole picture, in the case of having particular difficulties in the corners detection

## 4.3 First estimate of the parameters

The calibration process executed by the code follows the procedures outlined in section 3.4. It initiates by estimating all extrinsic parameters, excluding $\mathbf{t}_3$, through an iterative approach across all images. The solution is derived using SVD on the matrix U defined in 3.4.1 while leveraging the inherent orthogonality of the grid.

Similar to the second step, the code defines the variables $A_{ij}, B_{ij}, C_{ij}, and\ D_{ij}$, and then determines the polynomial coefficients and $\mathbf{t}_3$ by employing a linear least-squares approach on the system 3.21. The degree of the polynomial can be selected by the user; Scaramuzza proposes employing a fourth-degree polynomial equation based on empirical experiments with various camera models, demonstrating its superior performance.

The parameters obtained from this swift calibration step offer an initial estimation, subject to further refinement following the determination of the omnidirectional image center, $O_C$.

There exist three primary functions pivotal to this calibration step:

- [RRfin,ss] = calibrate(Xt, Yt, Xp_abs, Yp_abs, xc, yc, taylor_order_default, ima_proc)
  This function yields an initial estimation of extrinsic parameters and polynomial coefficients, elucidated in the preceding section. Specifically, it establishes all extrinsic parameters except for $t_3$, with the subsequent function entrusted to ascertain the remaining factors.

- [RRfin,ss] = omni_find_parameters_fun(Xt, Yt, Xp_abs, Yp_abs, xc, yc, RRfin, taylor_order_default,ima_proc)
  Employed by the preceding function, this one concludes the estimation of *RRfin* and *ss*. The linear optimization of all parameters is effectuated via the Matlab Optimization Toolbox, specifically the function *lsqlin*.

- reprojectpoints(calib_data)
  The outputs comprise:

  - The average reprojection error, computed for each checkerboard, which represents the mean of the reprojection errors across all corners of the checkerboard.
  - The average error, signifying the mean of all the aforementioned errors.
  - The Sum of Squared Reprojection Errors (SSRE).

This function employs the estimated parameters and the *omni3d2pixel* function to project the corners and determine the distance from the extracted ones.

## 4.4 Find center

As discussed in chapter 3, the centre of the image plane $O_C$ is made to coincide with the centre of the sensor plane, $I_C$. The iterative search for the location of $O_C$ avoids great reprojection errors. In order to avoid the algorithm to take a lot of time to get the result, it is recommended to choose a starting point close to the center of the images, otherwise it would take hours for the algorithm to end.
The function that deals with finding the Image center is named **findcenter(calib_data)**, which takes in input the information of the class coming from the first estimate step and, after finding the proper center, it performs both the functions *calibrate* and *reprojectpoints*, to display the new SSRE.

## 4.5 Refinement phase

As presented in the subsection 3.4.2, last step of the algorithm is the refinement of the parameters, both intrinsic and extrinsic. This phase will be examined more in detail in the next chapter, where the the new models and the theory behind them will be introduced.
The function responsible for this last step is named **optimizefunction**(calib_data).
At first, the user is asked to specify the maximum number of iterations that we want the algorithm to perform. Then a *while* cycle starts, with as stopping condition the reaching of the maximum number of iterations or a MSE value that isn't much smaller than the previous one (MSE_old - MSE_new < tolerance).
The alternating refinement of intrinsic and extrinsic parameters is governed by two functions, which could be described as the most important of the entire toolbox, which allow the calculation of reprojections:

- [x, y] = omni3d2pixel(ss, xx, width, height)
  The parameters provided to this function include the polynomial coefficients (ss), the corners 3D coordinates in the camera's reference system (xx), and the pixel dimensions of the image (width and height, with values of 2448 and 2048 pixel respectively). The resulting output is the projection of these corners onto the image plane, specified in pixels. This function calculates the pixel coordinates of the projected points using the distance $\rho$ from the image center. This function will be the focus of next chapter.
- m=world2cam(M, ocam_model)
  The provided inputs encompass the corners 3D coordinates in the camera's reference system (M) and the ocam_model, which holds the intrinsic parameters of the model as well as the width and height of the image. The resulting output is the projection of these corners onto the sensor plane. This function initiates by invoking omni3d2pixel(ss, xx, width, height) and subsequently employs an affine transformation (the matrix A defined in 3.6) to rectify the pixel coordinates of the projected points.

# 5 Adaptation of Scaramuzza's model to a non central projection model

This chapter is going to show how Scaramuzza's model has been modified, in order to fit the case of the PANCAM lens described in chapter 2.
The core idea behind the new models developed is that the lens can't be approximated with a central projection (as assumed by Scaramuzza) but we can model it with a non central model, that can also be renamed as A-central Model (AM). The order in which the models are presented follows the chronological order on which they were thought and implemented in Matlab, whose codes are reported in every section.
Every model is then tested with for the calibration of the camera over 2 different sets of images:

1. Small chessboards: a set of 54 images, each containing a 7x10 black-and-white chessboard. Square's sizes are 10*cm* and grids cover the whole azimuth FoV, with a special attention in placing our patterns also in the hyper-hemispherical side.
2. Big chessboards: a set of 50 images, now regarding 28x10 chessboards, with the same square size of previous data bench . This new grids can cover almost the whole zenithal FoV, but at the cost of being way difficult to work with during corner extraction.

Both sets of images are caught in **bmp** format, with 2048x2448 as pixel dimensions.
The degree of the polynomial (N) is set to 4, since it is the best value, as assessed by Scaramuzza.

Figures 5.1 is an example taken from the first set of images, which is peculiar since it is located close to the hyper-hemispherical part. The grey ellipse is used just to cover the long chessboard, which was fixed in the laboratory. Anyway, it doesn't effect the calibration process at all.
Image 5.2 shows the long chessboard, which was fixed onto the wall during image acquisition.

Figures 5.3 and 5.4 show the positions of the chessboards, with respect to the camera. These plots are computed by the toolbox, through the button **Show extrinsic**, which runs the function **create_simulation_points**(calib_data).

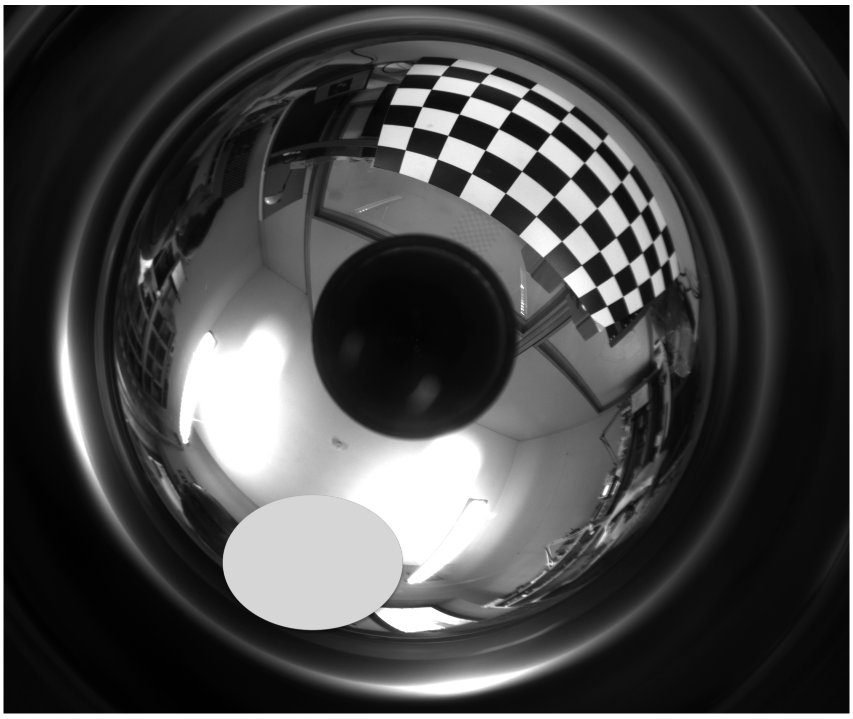


**Figure 5.1:** Hyper-hemispherical picture taken by the lens, small grid

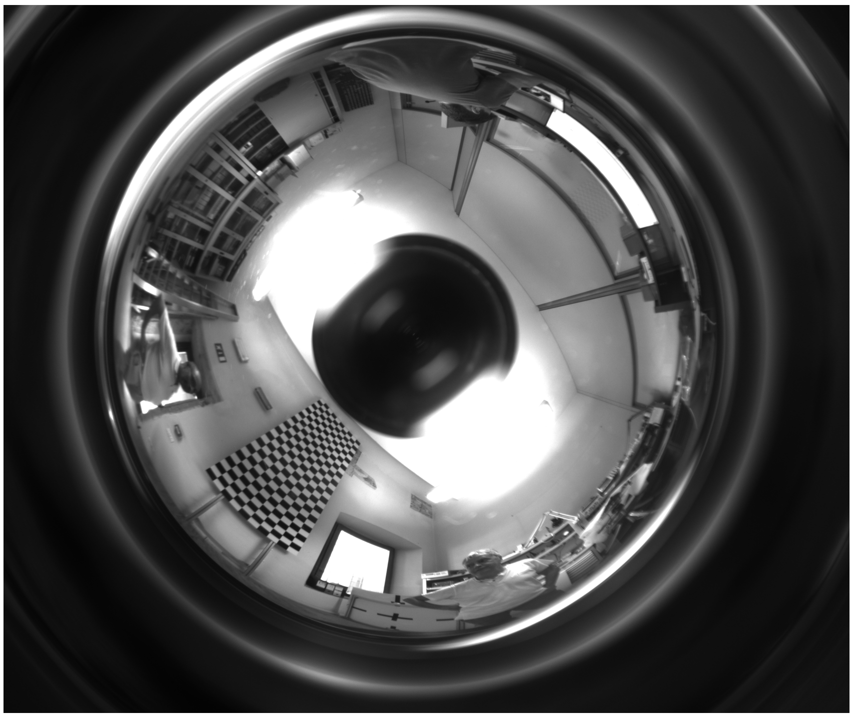


**Figure 5.2:** Hyper-hemispherical picture taken by the lens, large grid

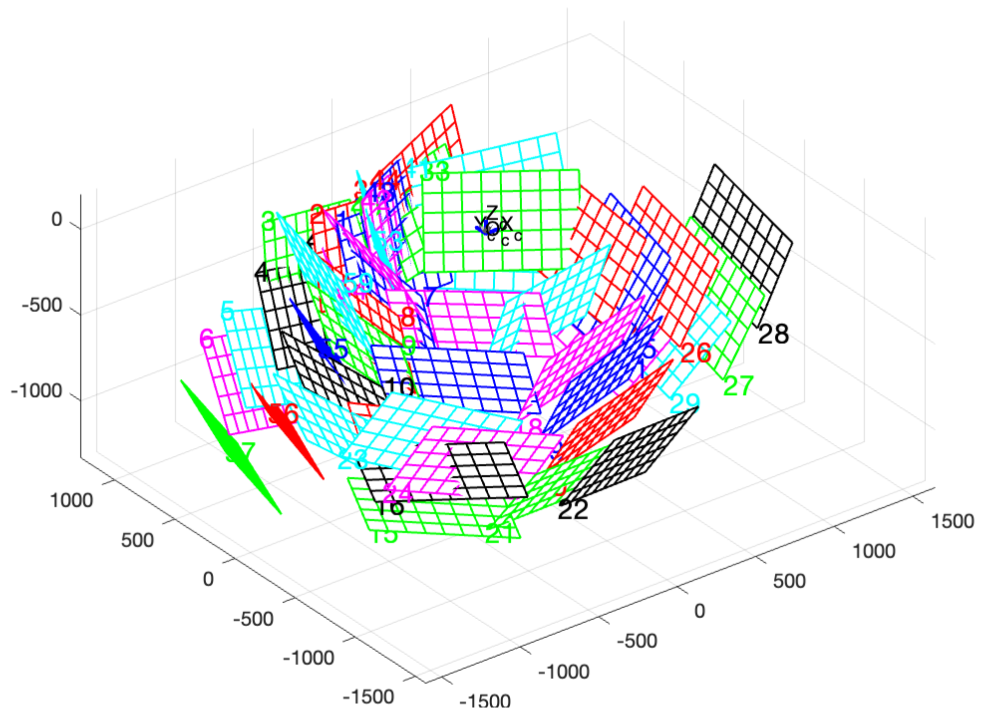


**Figure 5.3:** Positions of small chessboards. Unit of measurements is millimeter.

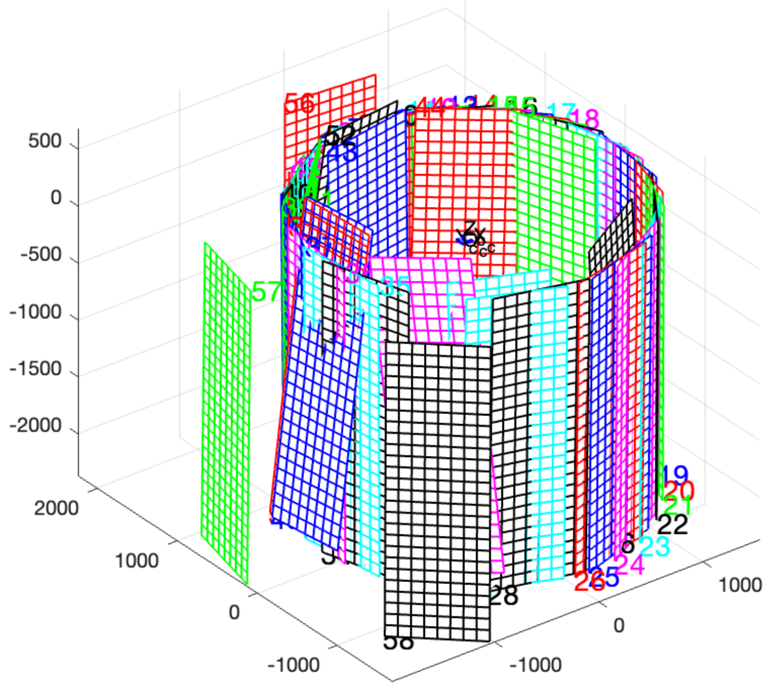


**Figure 5.4:** Positions of big chessboards. Unit of measurements is millimeter.

Grids are drawn following the extrinsic parameters, that are the roto-translation matrices associated to the images.
This chapter represents the core part of the discussion.
We will begin with the results and the issues of the calibration procedure following Scaramuzza's model.
After that, the latter will be modified, moving away from the *single effective viewpoint* property. This new model, called **AM model**, will be analyzed together with its basic assumptions and its calibration results commented.
AM model will be further modified, in order to tackle problems of reprojection on the hyper-hemispheric part, in the so called **piecewise-AM model**.
The last model presented is named **AM-RZ**, which extends the previous models by assuming that the points towards whom the light rays are directed are not constrained anymore to lie on the Z axis of the camera coordinate system.
The performances of these models are assessed through these indicators:

- Average reprojection error: the mean distance between a location of a corner and its reprojection on the image;
- Standard deviation error, on both axis: for each image, computed the difference between the coordinates of the true location of the corners and the coordinates of their reprojections on to the image. For both coordinates it is computed the standard deviation and then these value are averaged over all the pictures.

## 5.1 Scaramuzza's Camera model

In the preceding chapter, we conducted an extensive examination of the projection model that underpins OCamCalib. However, for the purpose of our current analysis, it becomes imperative to introduce a revision of equation 3.7 that relies on the zenith angle.
From the definition of the $g$ function, called imaging function, each pixel $\mathbf{u}''$ on the image plane is associated to a point in the 3D world, with coordinates $[u'', v'', f(\rho)]^T$. Then all the points lying on the line intersecting the latter point and the center of the camera coordinate system will be mapped onto $\mathbf{u}''$.
So, when we are projecting a world point onto the image plane, the following equation holds:

$$tan(\theta) = \frac{f(\rho)}{\rho}\ ,\ \theta = \arctan(\frac{z_M}{\sqrt{x_M^2 + y_M^2}}) \tag{5.1}$$

where $\theta$ is the zenith angle (with respect to the horizon plane), $[x_M, y_M, z_M]^T$ are the corners 3d coordinates in mm and $\rho$ the distance in pixel of the reprojection from the centre of the image. . The smallest $\rho \in \mathbb{R}$ which solves this equation, which can be easily rewritten as

$$a_0 - tan(\theta)\rho + a_2\rho^2 + \cdots + a_N\rho^N = 0. \tag{5.2}$$

This equation is the one solved by function **omni3d2pixel**. In particular, this function looks for the smallest real solution of equation 5.2, by performing the **roots** function, a built-in tool useful to compute the solutions of a polynomial. It works by creating the companion matrix associated to the left term of 5.2, which shall be previously rewritten as a monic polynomial. The eigenvalues of this matrix are the roots of the former polynomial.

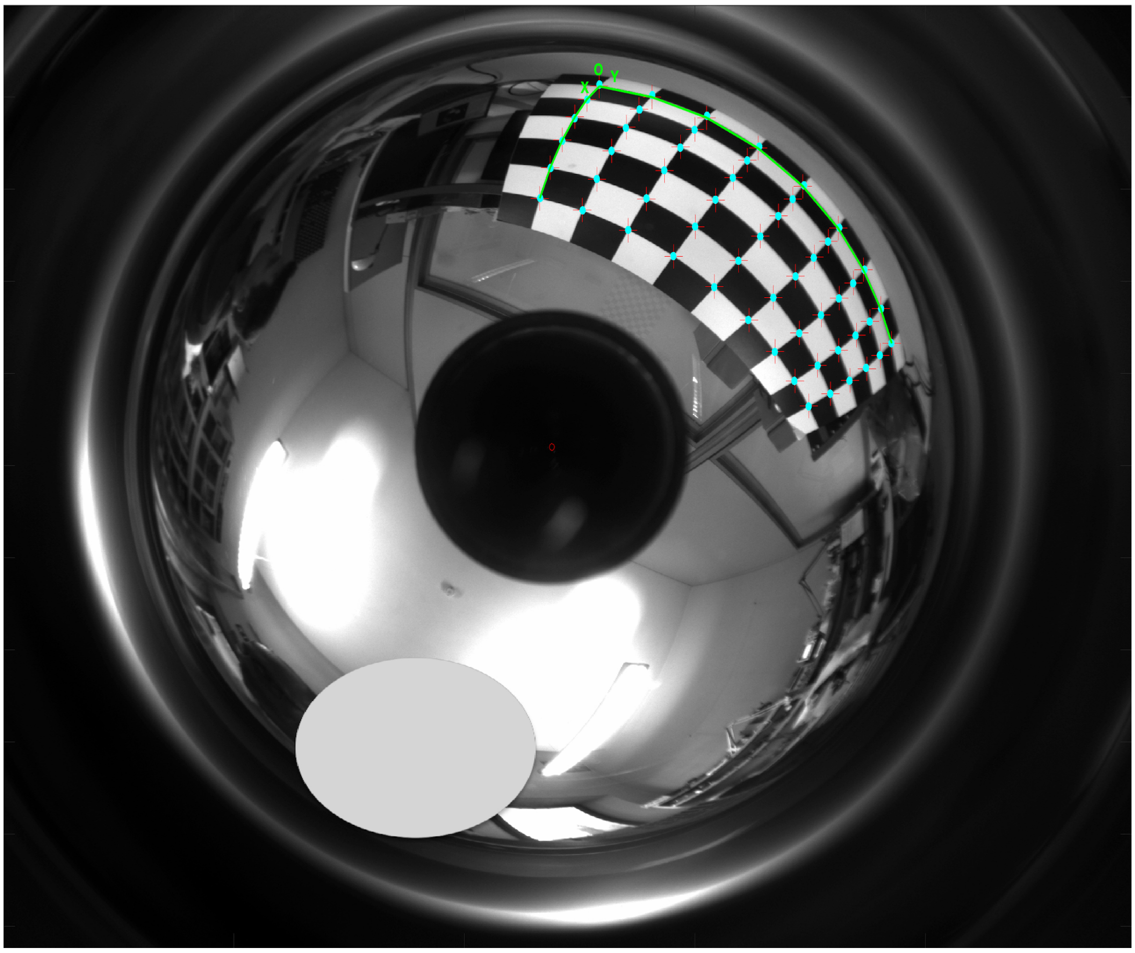


**Figure 5.5:** Reprojection of corners onto the image

As it was presented in chapter 2, the camera can't be properly approximated with a pinhole projection model. This is due to the fact that the FoV comprehends 50° under the equator, making the PANCAM an hyper-hemispherical lens.
To verify the relevance of our idea of having to change the nature of the model, two calibration test are performed, one for each set of images.

In 5.5, red crosses represent where the points were located at at the beginning of the calibration, while blue circles show the reprojections of the corners using the relation defined in 3.10. As we can see in the picture 5.5, Scaramuzza's model performs well, as crosses are really close to circles. This does not hold true fore the hyper-hemispheric part.

As we can see in the zoom displayed in 5.6, reprojections of corners associated to real-world points which have high zenithal angles in the camera coordinate system are remarkably distant from their true locations in the image. It is possible to quantify the error committed by Scaramuzza's model in the reprojections of the corners with these

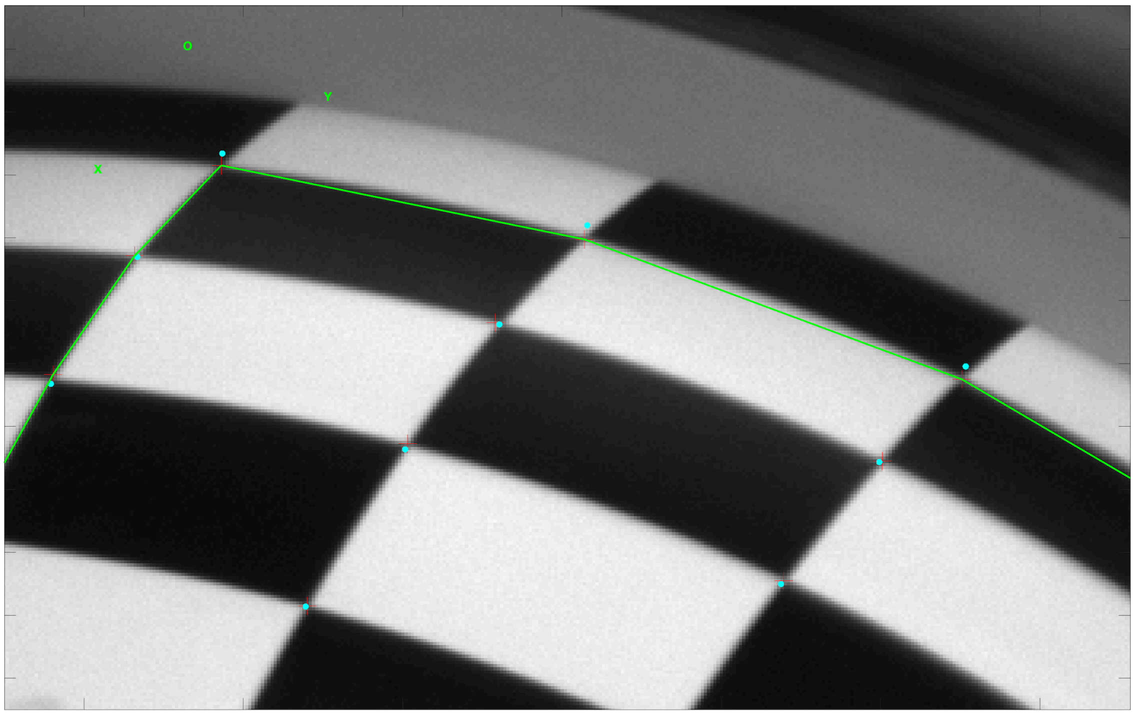


**Figure 5.6:** Zoom of figure 5.5 on the HH part

indicators:

- average reprojection error: 0.823725 px;
- std error on x axis: 0.622204 px;
- std error on y axis: 0.576497 px.

Regarding the second set of images, the one including pictures of long chessboards, we can anticipate that now the errors are going to be greater, since the corner extraction is less efficient.
The grid is 2.80m long, which means that the points on top and at the bottom of the chessboard are significantly distant from the camera. So, we can state that these points are more prone to noise and they are more difficult to exactly identify during the corner extraction procedure. While for small chessboards the automatic corner extractor manages to correctly identify every corner in almost all the images, for long chessboards, on average, 80% of the corners are found, and so user intervention is needed to get accurate locations.
The errors committed during this calibration procedure are:

- average reprojection error: 2.215582 px;
- std error on x axis: 1.519904 px
- std error on y axis: 1.589742 px

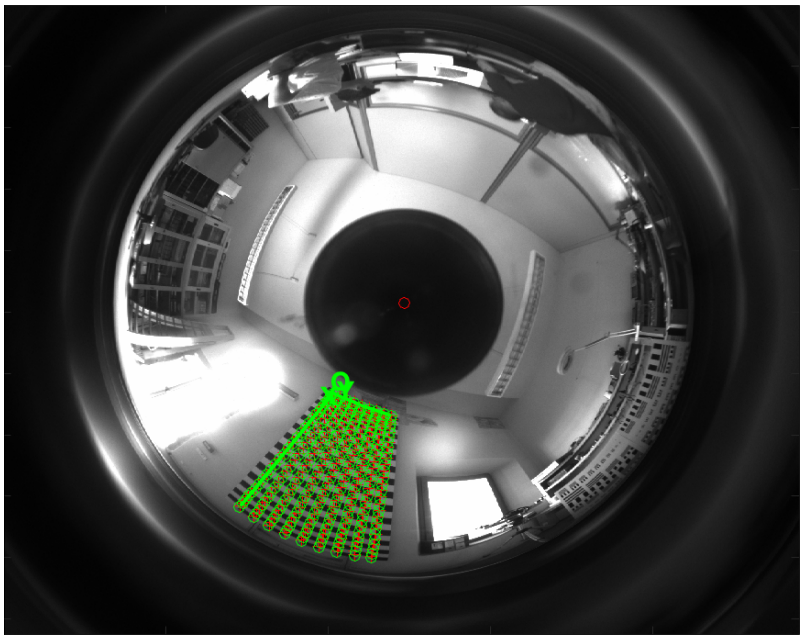


**Figure 5.7:** Reprojection of corners onto the image, long chessboard

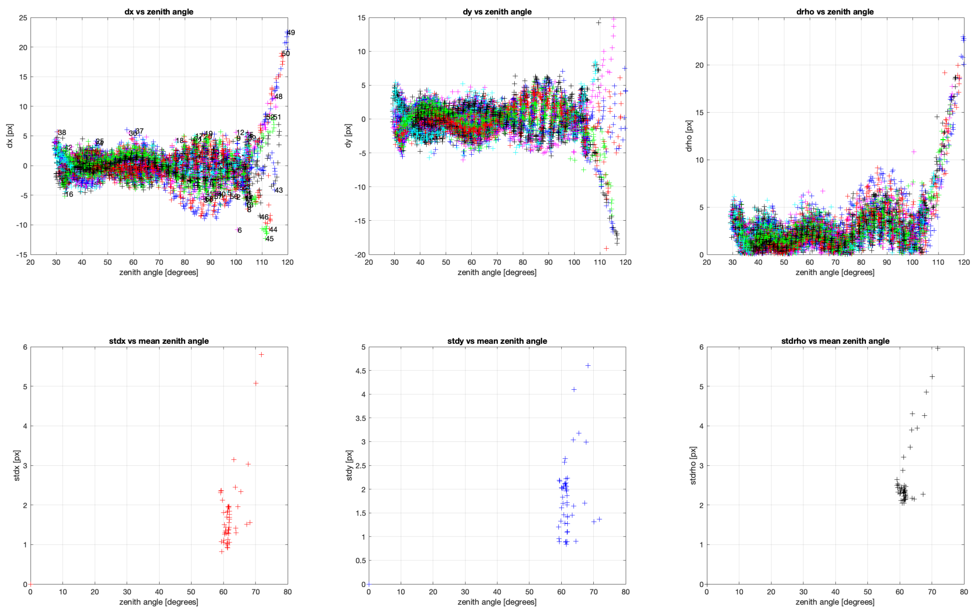


**Figure 5.8:** Zoom on the HH part

Figure 5.7 shows the reprojections of the corners onto an image, while plots in 5.8 highlight the behaviour of the lens on the hyper-hemispherical side, where reprojections are not so close the exact locations, as we would.

## 5.2 First models with non central projection

The idea of a non central projection comes from a thoughtful analysis of the behaviour of the lens, and the difficulties faced by Scaramuzza's model during its calibration.

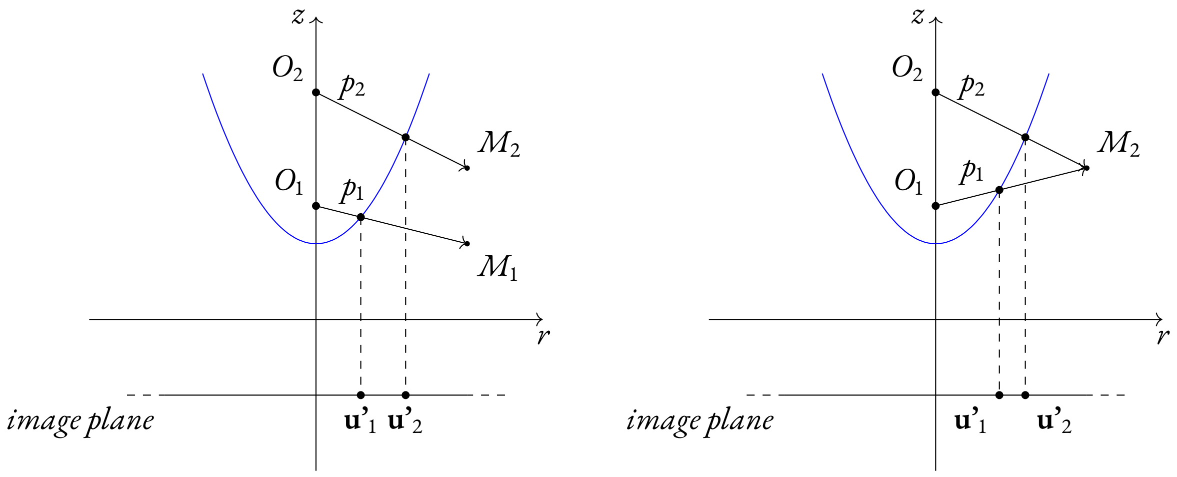


**Figure 5.9:** Left: projections of two points. Right: differences in the projections of a point according to pinhole camera model and the new model.

The concept is simple: we suppose that not all the points are projected to a unique point, as the case of the pinhole projection.
Plots in 5.9 show how the new projections work. Here the situation focuses only on one plane, but the situation can be easily generalized to the whole 3D space, since the mirror (here represented through the blue curve) is radially symmetric.
The left plot shows how each point on the image plane (**u**') is associated to a point on the Z axis (O). The vector (p) starting from O and passing through the projection of **u**' on the mirror defines a specific direction. All the points belonging to this direction will be mapped onto **u**'. The figure on the left provides to examples of projections, for the points $M_1$ and $M_2$. The right figure, instead, remarks how the difference between the points associated to $M_2$, with $\mathbf{u}'_1$ found according to the classic pinhole projection, with $O_1$ as the center of the camera, while $\mathbf{u}'_2$ is found with this new procedure.
The relationship between the point on the image plane and the point on the Z axis can be expressed with a function of the variable $\rho$, that we are going to name $z_0 = z_0(\rho)$. This function is meant to be monotonous, to avoid chaos in the images. Its dependence from $\rho$ makes it easy to integrate this new term into our equations.
In fact, we just have to modify 5.1 such that:

$$\frac{(z_M - z_0(\rho))}{\sqrt{x_M^2 + y_M^2}} = \frac{f(\rho)}{\rho}\,, \tag{5.3}$$

By introducing a non central model, we have that points associated to vectors $p$ with same direction are not mapped onto the same pixel anymore. In fact, in the case of a central camera we had a "privileged" point, the one considered as the center of the camera coordinate system. Let M a point in the camera coordinates system and rewrite its coordinate in spherical polar coordinates, that is

$$M = (r,\ \theta,\ \varphi),\ with\, r \geq 0,\ \theta \in [0,\ 2\pi)\ and\, \varphi \in [0,\ \pi)$$

Then points associated to the same $\theta$ and $\varphi$ angles used to be mapped onto the same pixel. Since in the non central model the directions (p) do not start from a unique point, this relation falls.
How can we define the $z_0$ function? At the beginning, a first guess is to focus on a linear function:

$$z_0 = b_0 + b_1 \cdot \rho \quad . \tag{5.4}$$

By doing this, we are introducing 2 new parameters, $b_0$ and $b_1$, which are to be determined during the calibration process. In order to optimize these two variables, they are considered as new intrinsic parameters, along with the coefficients of $f(\rho)$.
In this way , we can rewrite equation 5.3 as:

$$\frac{(z_M - (b_0 + b_1 \cdot \rho))}{\sqrt{x_M^2 + y_M^2}} = \frac{f(\rho)}{\rho} \tag{5.5}$$

Having $z_O$ in a polynomial form allows us to rewrite the latter equation as:

$$a_0 - \frac{(z_M - b_0)}{\sqrt{x_M^2 + y_M^2}}\rho + \left(a_2 + \frac{b_1}{\sqrt{x_M^2 + y_M^2}}\right)\rho^2 + \cdots + a_N\rho^N = 0\ . \tag{5.6}$$

Given how easily this modification can be introduced in our equations, it is also possible to define $z_0$ as a quadratic function. We can sum up everything as:

$$z_0 = b_0 + b_1 \cdot \rho + b_2 \cdot \rho^2 \tag{5.7}$$

Equations 5.5 and 5.6 can be rewritten as:

$$\frac{(z_M - (b_0 + b_1 \cdot \rho + b_2 \cdot \rho^2))}{\sqrt{x_M^2 + y_M^2}} = \frac{f(\rho)}{\rho} \tag{5.8}$$

and

$$a_0 - \frac{(z_M - b_0)}{\sqrt{x_M^2 + y_M^2}}\rho + \left(a_2 + \frac{b_1}{\sqrt{x_M^2 + y_M^2}}\right)\rho^2 + \left(a_3 + \frac{b_2}{\sqrt{x_M^2 + y_M^2}}\right)\rho^3 + \cdots + a_N\rho^N = 0\ . \tag{5.9}$$

In order to evaluate these new models, two different calibrations are performed, over the datasets previously described: the one with 7x10 chessboards and the one with 28x10 chessboards.
The performances of the the two models following a non central projection are very similar. Moreover, they do not out-preform the results obtained by Scaramuzza's model. In tables 5.2, we can see that the errors committed

by the models are almost the same.

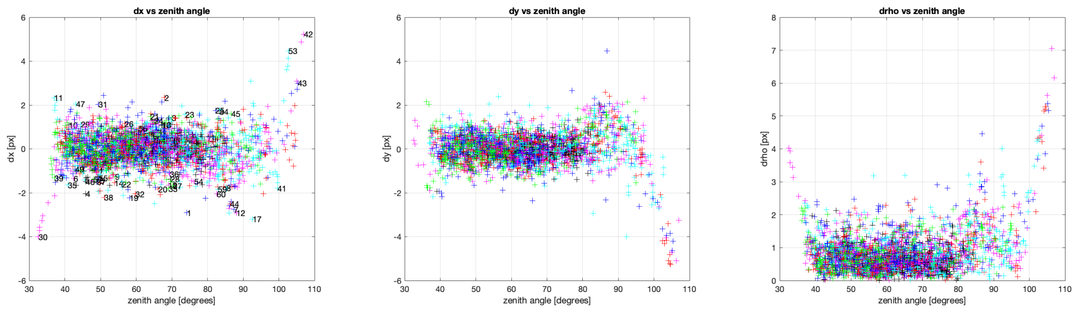
| | Avg error (px) | Std error on x (px) | Std error on y (px) |
|---|---|---|---|
| Scaramuzza's model | 0.823725 | 0.622204 | 0.576497 |
| Linear $z_0$ | 0.818188 | 0.620968 | 0.572519 |
| Quadratic $z_0$ | 0.814733 | 0.617055 | 0.570284 |

**Table 5.1:** Comparison between performances of the Scaramuzza's model and the two versions of the AM model, over 7x10 chessboards

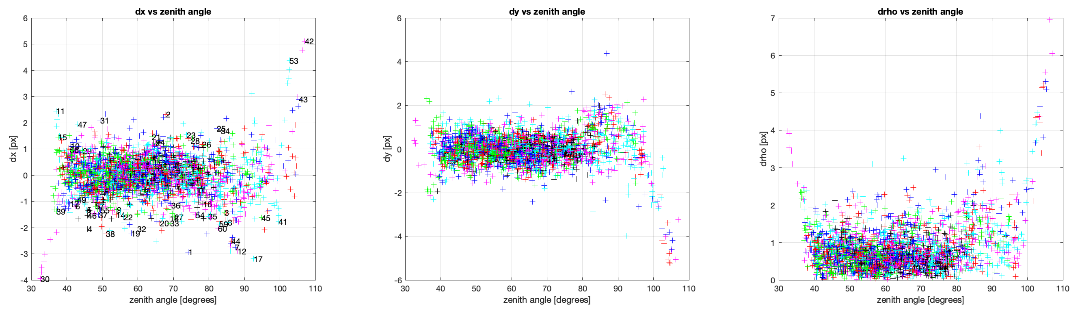
| | Avg error (px) | Std error on x (px) | Std error on y (px) |
|---|---|---|---|
| Scaramuzza's model | 2.215582 | 1.519904 | 1.589742 |
| Linear $z_0$ | 2.173670 | 1.509782 | 1.567191 |
| Quadratic $z_0$ | 2.177170 | 1.510186 | 1.572896 |

**Table 5.2:** Comparison between performances of the Scaramuzza's model and the two versions of the AM model, over 28x10 chessboards

New models reach slightly smaller errors than the Scaramuzza's one. This is natural since we are adding new parameters and in the worst case they would just be ignored (put to 0) during the computations of the algorithm. AM models introduce new parameters ($b0$, $b1$ and $b2$) to Scaramuzza's model, which means that they increase its complexity. Given that there aren't considerable improvements in the performances, we shall review the underlying theory, with new ideas and new equations.
Moreover, we can see from the plots in 5.10a and 5.12a that the distributions of the errors do not change, always with high reprojection errors when dealing with corners with high zenith angles.



**(a)** Distribution of the errors with respect to the zenithal angle, following Scaramuzza's model over small chessboards



**(b)** Distribution of the errors with respect to the zenithal angle, following AM model over small chessboards (quadratic $z_0$)

As far as the Matlab toolbox is concerned, the function to be modified is **optimizefunction**, the one responsible for the refinement phase. Here a new *field* is added, named **lin_coeff**, a vector containing the coefficients related to the $z_0$ function, $[b_0, b_1]$ in the case of $z_0$ as a linear function and $[b_0, b_1, b_2]$ in the case of $z_0$ as a quadratic function. This vector is merged with the one containing the coefficients of the polynomial (ss), creating a vector state whose aim is to keep together all the intrinsic parameters. This new vector is now the replacement of ss in the function **omni3d2pixel**. The updates of omni3d2pixel, renamed as omni3d2pixel_am, can be found in the tables 5.3 and 5.4, respectively for the cases of $z_0$ as a linear function or a quadratic one.

```
function [x,y] = omni3d2pixel_am_linear(vs, xx)
% z_0 = b0 + b1*x, linear function

b0 = vs(end-1);
b1 = vs(end);

m = (xx(3,:)-b0)./sqrt(xx(1,:).^2+xx(2,:).^2);
m1 = b1./sqrt(xx(1,:).^2+xx(2,:).^2);

rho=[];

%coefficients of the polynomial do not comprehend the last 2 elements,
%that are the coefficients of z_0
poly_coef = vs(end-2:-1:1); %inverted polynomial
poly_coef_tmp = poly_coef;

for j = 1:length(m)
    %sum the coefficents of the polynomial to the contributions given
        by z_0
    poly_coef_tmp(end-1) = poly_coef(end-1)-m(j);
    poly_coef_tmp(end-2) = poly_coef(end-2)+m1(j);
    rhoTmp = roots(poly_coef_tmp); %find all the solutions

    %filter and keep only the real and positive solutions
    res = rhoTmp(find(imag(rhoTmp)==0 & rhoTmp>0));

    %pick the smallest solution
    if isempty(res) %| length(res)>1
        rho(j) = NaN;
    elseif length(res)>1
        rho(j) = min(res);
    else
        rho(j) = res;
    end
end

%reproject the points onto the sensor plane
x = xx(1,:)./sqrt(xx(1,:).^2+xx(2,:).^2).*rho ;
y = xx(2,:)./sqrt(xx(1,:).^2+xx(2,:).^2).*rho ;
```

**Table 5.3:** MATLAB code for the solution of the equation 5.5

```
function [x,y] = omni3d2pixel_am_quadratic(vs, xx)
% z_0 = b0 + b1*x + b2*x^2, quadratic function
b0 = vs(end-2);
b1 = vs(end-1);
b2 = vs(end);

m = (xx(3,:)-b0)./sqrt(xx(1,:).^2+xx(2,:).^2);
m1 = b1./sqrt(xx(1,:).^2+xx(2,:).^2);
m2 = b2./sqrt(xx(1,:).^2+xx(2,:).^2);

rho=[];
%coefficients of the polynomial do not comprehend the last 3 elements,
%that are the coefficients of z_0
poly_coef = vs(end-3:-1:1); %inverted polynomial
poly_coef_tmp = poly_coef;

for j = 1:length(m)
    %sum the coefficents of the polynomial to the contributions given
       by z_0
    poly_coef_tmp(end-1) = poly_coef(end-1)-m(j);
    poly_coef_tmp(end-2) = poly_coef(end-2)+m1(j);
    poly_coef_tmp(end-3) = poly_coef(end-3)+m2(j);
    rhoTmp = roots(poly_coef_tmp); %find all the solutions

    %filter and keep only the real and positive solutions
    res = rhoTmp(find(imag(rhoTmp)==0 & rhoTmp>0));
    %pick the smallest solution
    if isempty(res) %| length(res)>1
        rho(j) = NaN;
    elseif length(res)>1
        rho(j) = min(res);
    else
        rho(j) = res;
    end
end

%reproject the points onto the sensor plane
x = xx(1,:)./sqrt(xx(1,:).^2+xx(2,:).^2).*rho ;
y = xx(2,:)./sqrt(xx(1,:).^2+xx(2,:).^2).*rho ;
```

**Table 5.4:** MATLAB code for the solution of the equation 5.8

## 5.3 Non central projection models performing piecewise functions

In the previous section it was stated that AM models do not solve our problems, since they do not manage to obtain better results and, in particular, they do not succeed in the task of reducing the reprojection errors for corners associated with high zenithal angles.
In order to tackle this last issue, the idea that we are going to introduce in the following lines is to insert a modification in AM model, by defining $z_0$ as a piecewise function, with 2 pieces.
First of all, let's introduce R, an hyper-parameter whose function is to split the space of an image in 2 distinct parts:

- the hyper-hemispherical part, i.e. the portion the image including all the pixels whose distance from the image center is bigger than R;
- the "narrow" part, i.e. the portion of the image closer to the image center, where the pixels are distant from it less than R.

In the new definition of $z_0$ as a piecewise function, the value R defines where these two pieces meet, so that the function can be written as:

$$z_0(\rho) = \begin{cases} z_0^1(\rho), & \text{se}\, \rho \leq R \\ z_0^2(\rho), & \text{se}\, \rho > R \end{cases} \tag{5.10}$$

Given that pixels which lie further from the center of the image are associated to points in the real world, which, in the camera coordinate system, have high zenith angles, this new reformulation of $z_0$ substantially implies the creation of two different models, one for the narrow part and another for the hyper-hemispheric portion. This last model is going to be specific for the points highly misclassified by previous models, that are the points associated to high zenith angles.
Picture 5.11 displays a picture belonging to one of the datasets used to test the calibration models, the one with 28x10 chessboards. The green point in the middle is the center of the image, while the red circumference marks all the pixels whose distance from the center is R. In this specific example R is set to 720 pixels. The red curve clearly splits the image in two parts, dividing in two the chessboards as well.
The solutions of equation 5.3 will consider the function $z_0^1(\rho)$ for corners in the *narrow* portion and $z_0^2(\rho)$ for corners in the hyper-hemispheric part.

In order to have a smooth $z_0$ function, at first we have to impose $z_0^1, z_0^2 \in C^0(\mathbb{R}^+)$ and $z_0^1, z_0^2 \in C^1(\mathbb{R}^+)$. In order to have the whole $z_0$ function continuous and with continuous derivative, we have to simply introduce the following constraints:

$$\begin{aligned} z_0^1(R) &= z_0^2(R) \\ \frac{d}{d\rho} z_0^1(\rho)\bigg|_{\rho=R} &= \frac{d}{d\rho} z_0^2(\rho)\bigg|_{\rho=R} \end{aligned} \tag{5.11}$$

Obviously we can put constraints involving also derivative of higher orders, but for the models that are going to be introduced these two constraints are enough.

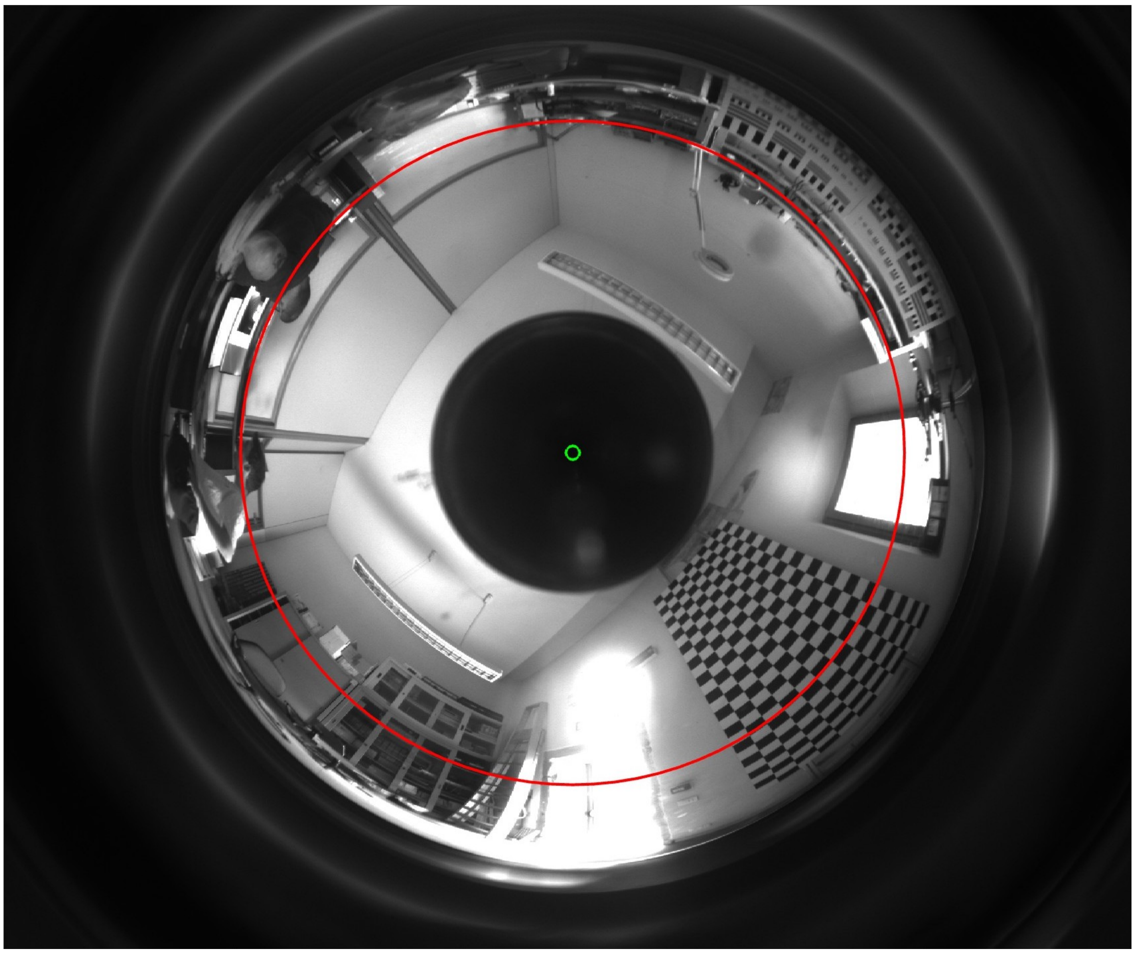


**Figure 5.11:** Example image to see practically the meaning of R

Models following the idea discussed in this section will be named **piecewise-AM**.
The first piecewise-AM model presented draws inspiration from the AM models described in the previous section. In fact, the function $z_0$ is taken as a linear function for corners in the narrow part and as a quadratic function for corners belonging to the hyper-hemispheric, such that 5.12 can be expressed as:

$$z_0(\rho) = \begin{cases} a\rho + b, & \text{se}\,\rho \leq R \\ c\rho^2 + d\rho + e, & \text{se}\,\rho > R \end{cases} \tag{5.12}$$

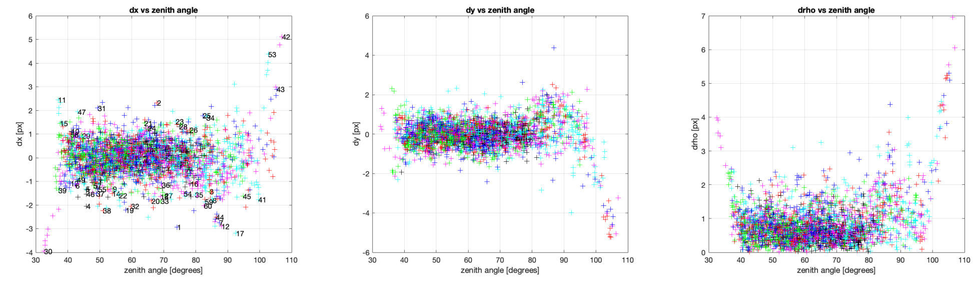
The constraints defined in 5.11 can be rewritten as

$$\begin{aligned} aR + b &= cR^2 + dR + e \\ a &= 2cR + d \quad . \end{aligned} \tag{5.13}$$

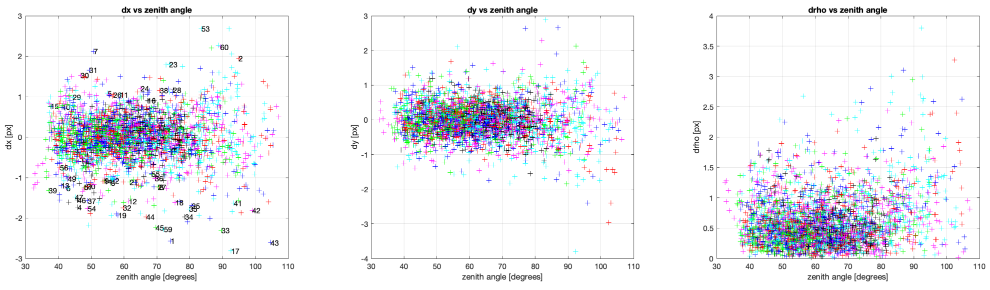
Solutions of these equations are $d = a - 2cR$ and $e = b + cR^2$. In this way, the number of parameters is reduced from 5 to 3, that are coefficients $a$, $b$ and $c$. This model can be compared to the quadratic version of former AM model, since they share the same number of parameters. Therefore, we can say that the new model does not add complexity to the previous one.
After having defined the idea of a piecewise function and its integration in the equations, we are ready to perform a calibration test, either on the dataset with the small chessboards or on the dataset with the large ones. The R hyper-parameter is set to 700, since it is the one which lets the model achieve the best results.
Firstly, the calibration is performed using as data the small grids, which is made up by 54 images. In order to evaluate the performance of the model, we compare its indicators about the reprojection errors with previous models and the results are in table 5.5.

| | Avg error (px) | Std error on x (px) | Std error on y (px) |
|---|---|---|---|
| Scaramuzza's model | 0.823725 | 0.622204 | 0.576497 |
| Quadratic-AM | 0.814733 | 0.617055 | 0.570284 |
| Piecewise-AM | 0.615650 | 0.481807 | 0.421362 |

**Table 5.5:** Comparison between Scaramuzza's model, Quadratic-AM model and piecewise-AM model. The dataset involved is the one regarding small chessboards.



**(a)** Distribution of the errors with respect to the zenithal angle, following AM model (quadratic $z_0$).



**(b)** Distribution of the errors with respect to the zenithal angle, following piecewise-AM model.

**Figure 5.12:** Comparison between AM model and piecewise-AM. Dataset used for the calibration is the one involving 7x10 chessboards.

The new model clearly obtains better results than its predecessors, both in the average reprojection errors and in the standard error on each axis. Given these positive results, let's give a look at the distribution of the reprojection error with respect to the zenith angles. Plots in 5.12 display the comparison between a version of the AM model and the piecewise-AM model. In particular they show how for high zenith angles Scaramuzza model generates high residuals (i.e. chessboard 42) while AM model fits correctly the same FoV region.
This new model clearly manages to get almost uniform distributions of the errors, without a significant increase for high zenith angles.
The images 5.13 present a comparison between the last two models. The pictures highlight a particular zone of one of the images, that is the hyper-hemispheric part and some corners belonging to it.
Now, it is interesting to examine the pictures, by looking at how far the reprojections of the corners are from the real locations. The locations of the vertices identified in the *corners extraction phase* are marked with red crosses, while the reprojections computed by the models are marked with blue circles.
The lower image shows the results due to the new piecewise-AM model. There the circles are clearly closer to the red crosses than in the upper image, where blue circles are the result of the calibration following AM model.

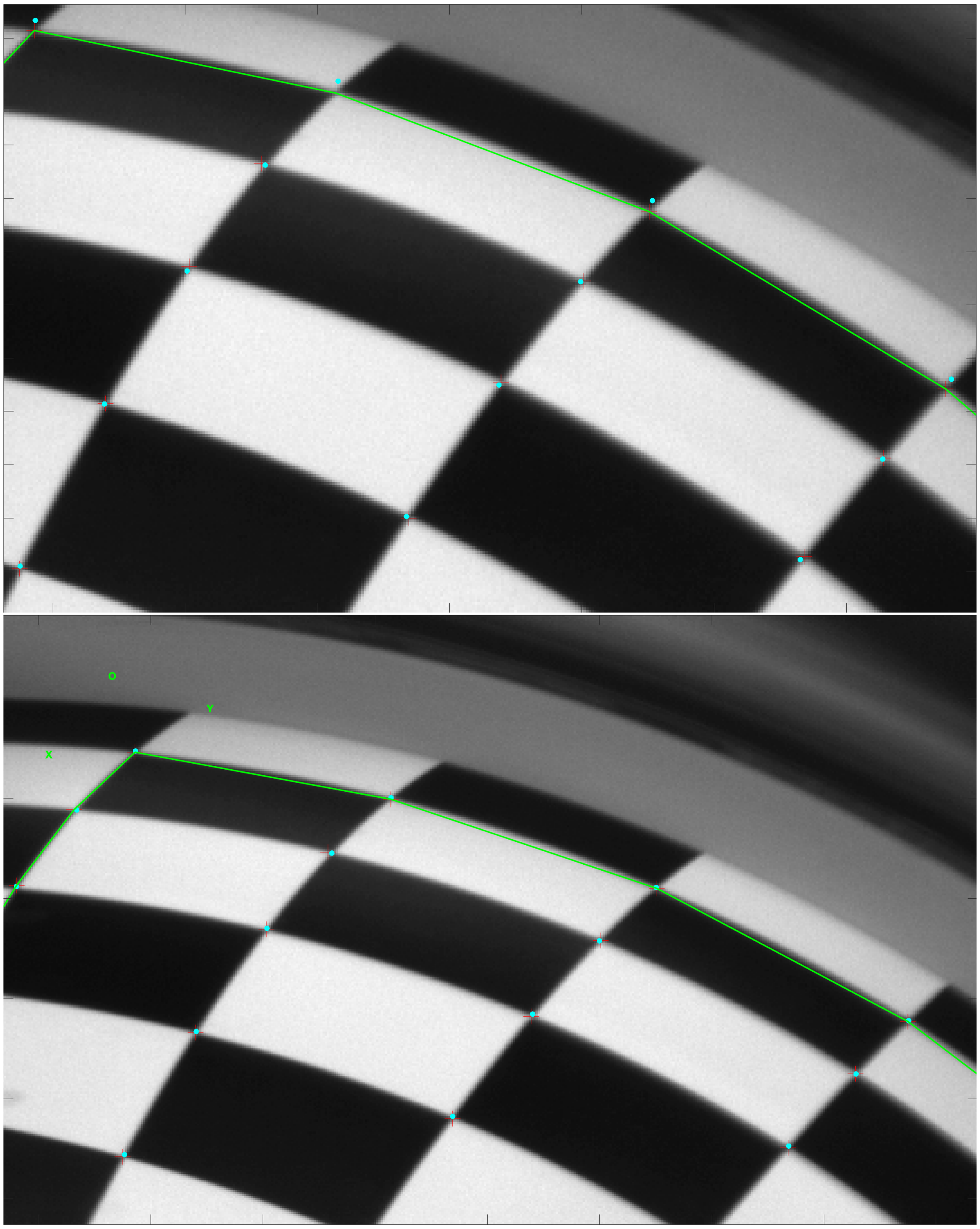
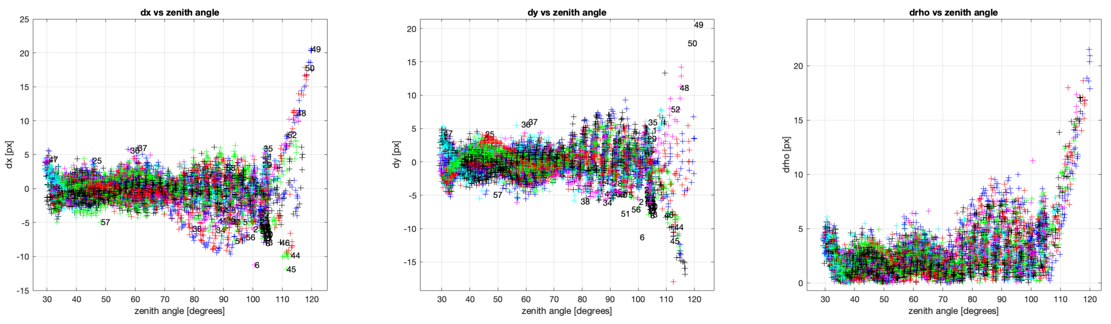


**Figure 5.13:** Differences in the reprojection of the corners between AM model (above) and piecewise-AM model (below). Piecewise-AM model behaves better especially on the corners belonging to the upper row.

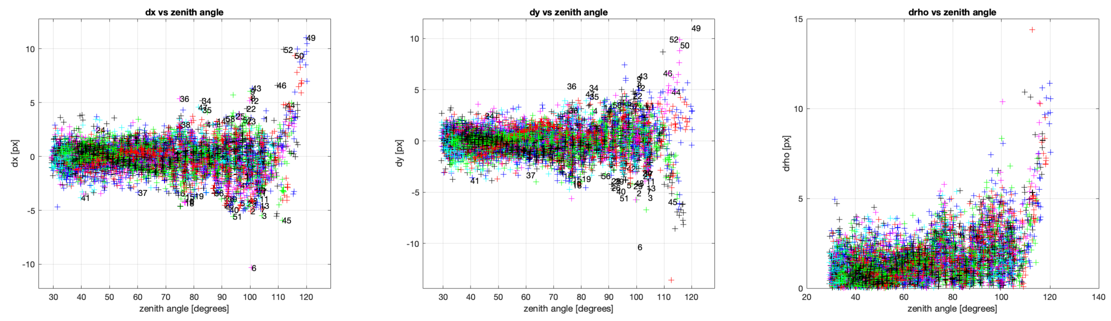
| | Avg error (px) | Std error on x (px) | Std error on y (px) |
|---|---|---|---|
| Scaramuzza's model | 2.215582 | 1.519904 | 1.589742 |
| Quadratic-AM | 2.177170 | 1.510186 | 1.572896 |
| Piecewise-AM | 1.250087 | 0.922428 | 0.919204 |

**Table 5.6:** Comparison between Scaramuzza's model, Quadratic-AM model and piecewise-AM model. The dataset involved is the one regarding long chessboards.

After having dealt with the dataset of images of 7x10 chessboards, now the calibration following piecewise-AM model is performed over the dataset containing long chessboards.
The results of this last calibration are displayed in the table 5.6.
These results prove that even in this case the new piecewise-AM model behaves way better, since the average error is almost halved and the mean standard errors on both axis decrease significantly. As we have done before, let's visualize the distributions of these error indicators with respect to the zenith angle.
The last three plots in 5.14 suggest that the piecewise-AM model reduces errors for high zenith corners, even if in this case the residuals are still very high, if compared with the other corners.



**(a)** Distribution of the errors with respect to the zenithal angle, following AM model (quadratic $z_0$).



**(b)** Distribution of the errors with respect to the zenithal angle, following piecewise-AM model.

**Figure 5.14:** Comparison between AM model and piecewise-AM. Dataset used for the calibration is the one involving 28x10 chessboards.

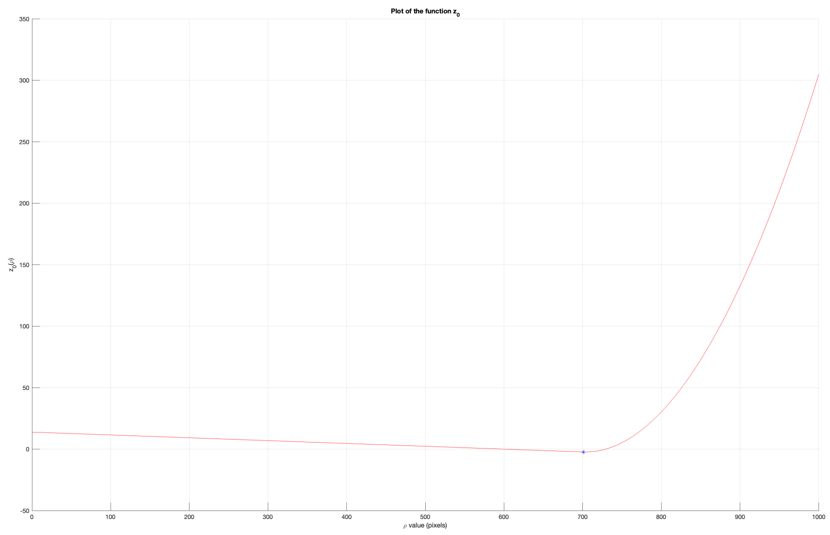


**Figure 5.15:** Plot of $z_0$ according to its coefficients. The blue dot marks the beginning of the second trait.

In order to further deepen the analysis of this new model, let's give a look at the function $z_0$. In the case of the calibration procedure involving $7x10$ chessboards, the coefficients $a, b, c$ of the $z_0$ function design the curve plotted in 5.15. The plot shows that the linear trait is almost a constant function. Therefore, a possible idea is to define a new model, a variant of the previous piecewise-AM model, now with a constant function instead of a linear one. The core idea behind this new model is that we may get a similar model to the former one, but with one less parameter. This modification even prevents the function to be not monotonic as the one displayed in the plot (which is something quite strange). For this new variant, the equation 5.12 and the respcetive constraints can be rewritten as :

$$z_0(\rho) = \begin{cases} a & \text{se} \rho \leq R \\ b + c\rho + d\rho^2, & \text{se} \rho > R \end{cases} \tag{5.14}$$

and

$$\begin{aligned} a =& b + cR + dR^2 \\ 0 =& c + 2Rd. \end{aligned} \tag{5.15}$$

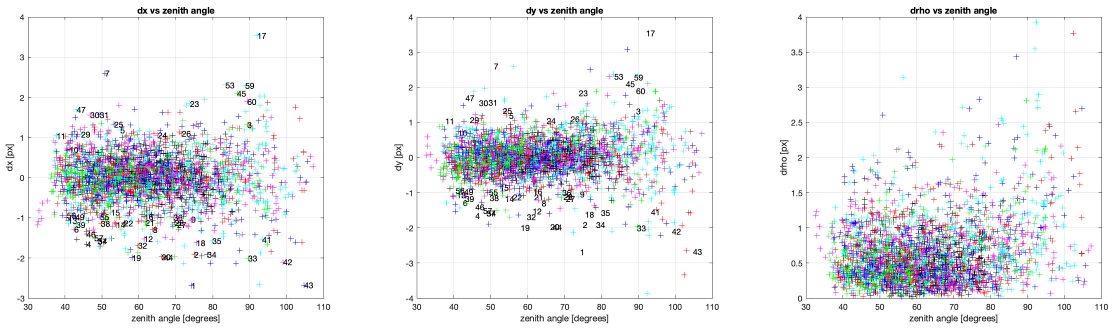
Solutions to the constraints expressed in 5.15 are $b = a + dR^2$ and $c = -2Rd$. In this way only 2 parameters define the function $z_0$, instead of 3 as the former piecewise-AM model, thus reducing the complexity.
In order to verify whether this new piecewise-AM model with one less parameter performs well or not, we can compare the results obtained after the calibration procedure with the previous model's ones.

Table 5.7 reports the calibration results computed starting from images containing small chessboards. It can be immediately seen that the two piecewise-AM models perform in a similar way, despite that the second one has one less parameter. The distributions of the error indicators (average error and mean standard deviations for both axis)

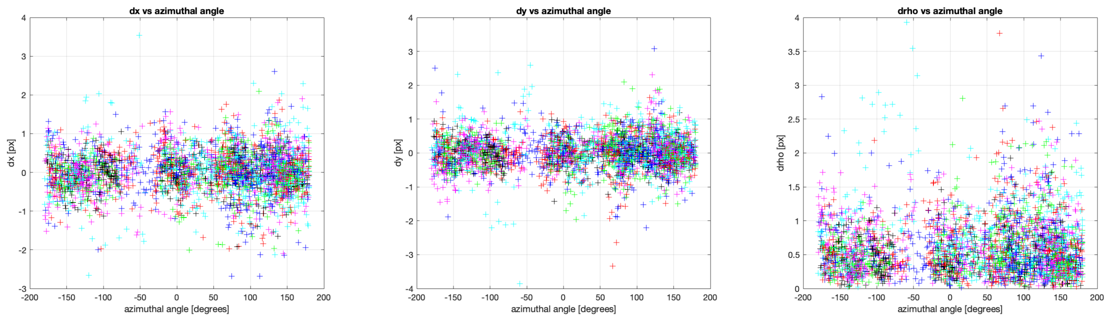
|  | Avg error (px) | Std error on x (px) | Std error on y (px) |
|---|---|---|---|
| Scaramuzza's model | 0.823725 | 0.622204 | 0.576497 |
| Quadratic-AM | 0.814733 | 0.617055 | 0.570284 |
| Piecewise-AM (1st trait: linear) | 0.615650 | 0.481807 | 0.421362 |
| Piecewise-AM (1st trait: constant) | 0.624592 | 0.484726 | 0.428303 |

**Table 5.7:** Comparison between Scaramuzza's model, Quadratic-AM model and piecewise-AM models. The dataset involved is the one regarding small chessboards.

with respect to the zenith angle are really quite identical with the ones regarding the first piecewise-AM model, which ensure that even in this case the issue encountered by Scaramuzza's model and AM models (high errors for high zenith corners) is well tackled (figures 5.16a).



**(a)** Distribution of the errors with respect to the zenithal angle, following the second version of piecewise-AM model.



**(b)** Distribution of the errors with respect to the azimuthal angle, following the second version of piecewise-AM model.

**Figure 5.16:** Distributions of the errors for zenithal (a) and azimuthal (b) angles, involving $2^{nd}$ piecewise-AM model. Dataset used for this calibration is the one involving 7x10 chessboards.

For this specific case, it is also plotted analogous figures regarding the distributions with respect to the azimuthal angle. In figures 5.16b it is possible to notice that the distributions are quite uniform, which implies that the radial model is well represented in the camera and there are no mounting or defocus errors.

| | Avg error (px) | Std error on x (px) | Std error on y (px) |
|---|---|---|---|
| Scaramuzza's model | 2.215582 | 1.519904 | 1.589742 |
| Quadratic-AM | 2.177170 | 1.510186 | 1.572896 |
| Piecewise-AM (1st trait: linear) | 1.250087 | 0.922428 | 0.919204 |
| Piecewise-AM (1st trait: constant) | 1.248232 | 0.937039 | 0.929807 |

**Table 5.8:** Comparison between Scaramuzza's model, Quadratic-AM model and piecewise-AM models. The dataset involved is the one regarding long chessboards.

Analogous results occur if we base the calibration over the dataset containing long chessboards, as shown in table 5.8. For what concerns the Matlab implementation of piecewise-AM models, the main function considered is always the one named **optimizefunction**. In addition to the changes introduced in the previous section, the one regarding AM models, there must be introduced some lines of code to individuate which points belong to the hyper-hemispheric part. Script in table 5.9 is the modification introduced in *optimizefunction* to solve this issue. It also adds two new fields to the class *calib_data*, both Boolean objects that mark with a 1 a corner on the hyper-hemispheric zone and with a 0 a corner who is not. Scripts in tables 5.10 and 5.11 show how the function *omni3d2pixel* gets modified for the case of piecewise-AM model.

```
%% Hyper-hemispheric structure
%Rewrite the coordinates with respect to the center of the images
Xp_centered = calib_data.Xp_abs - calib_data.ocam_model.xc;
Yp_centered = calib_data.Yp_abs - calib_data.ocam_model.yc;

iper_emi = [];
%load a file containig the value of the hyper-parameter
load("R_parameter.mat");
for i=calib_data.ima_proc      %cycle over images
    %compute the distance between corners and the center of the
        image
    raggi = sqrt(Xp_centered(:,1,i).^2 + Yp_centered(:,1,i).^2);
    Iper_Emi{i}=(raggi>R_parameter)';
    iper_emi = [iper_emi, (raggi>R_parameter)'];
end
calib_data.ocam_model.Iper_Emi = Iper_Emi;
calib_data.ocam_model.iper_emi = iper_emi;
```

**Table 5.9:** MATLAB code for the identification of hyper-hemispheric points

```
function [x,y] = omni3d2pixel_am_piecewise_linear(vs, xx, iper_emi)
%define hyper-hemisferic corners
load("R_parameter.mat"); % hyper-parameter which gives us the
    distinction between narrow and hyper-hemispheric
isi=find(iper_emi); % define indices of iper-emisferic corners
% function divided into 2 pieces, before linear and then quadratic
b0=vs(end-2); % constant term : b  in  z_0^1 = ax+b
b1=vs(end-1); % slope term: a in y = ax+b
b2=vs(end); % coefficient of the parabolic trait: c in y = e+dx+cx^2
% define m's
m = (xx(3,:)-b0)./sqrt(xx(1,:).^2+xx(2,:).^2);
m1 = b1./sqrt(xx(1,:).^2+xx(2,:).^2); m2 = 0*m1;
% define m's for hyper-hemispheric points
m(isi) = (xx(3,(isi))-b0-b2*(R_parameter^2))./sqrt(xx(1,(isi)).^2+xx
    (2,(isi)).^2);
m1(isi) = (b1-2*b2*R_parameter)./sqrt(xx(1,(isi)).^2+xx(2,(isi)).^2);
m2(isi) = b2./sqrt(xx(1,isi).^2+xx(2,isi).^2);
rho=[];
poly_coef = vs(end-3:-1:1);   %inverted polynomio
poly_coef_tmp = poly_coef;
for j = 1:length(m)
     poly_coef_tmp(end-1) = poly_coef(end-1)-m(j);
     poly_coef_tmp(end-2) = poly_coef(end-2)+m1(j);
     poly_coef_tmp(end-3) = poly_coef(end-3)+m2(j);
     rhoTmp = roots(poly_coef_tmp);
     %filter and keep only the real and positive solutions
     res = rhoTmp(find(imag(rhoTmp)==0 & rhoTmp>0));
     %pick the smallest solution
     if isempty(res) %| length(res)>1
         rho(j) = NaN;
     elseif length(res)>1
         rho(j) = min(res);
     else
         rho(j) = res;
     end
end
%reproject the points onto the sensor plane
x = xx(1,:)./sqrt(xx(1,:).^2+xx(2,:).^2).*rho ;
y = xx(2,:)./sqrt(xx(1,:).^2+xx(2,:).^2).*rho ;
```

**Table 5.10:** MATLAB code for the solution of the equation 5.5

```
function [x,y] = omni3d2pixel_am_piecewise_constant(vs, xx, iper_emi)
%define hyper-hemisferic corners
load("R_parameter.mat"); % hyper-parameter which gives us the
    distinction between narrow and hyper-hemispheric
isi=find(iper_emi); % define indices of iper-emisferic corners
% function divided into 2 pieces, before constant and then quadratic
a = vs(end-1); % constant parameter of the first trait
d = vs(end); % coefficient of the parabolic trait: d in y = b+cx+dx^2
%define m's
m = (xx(3,:)-a)./sqrt(xx(1,:).^2+xx(2,:).^2);
m1 = 0*m; m2 = 0*m1;
%define m's for hyper-hemispheric points
m(isi) = (xx(3,(isi))-a-d*(R_parameter^2))./sqrt(xx(1,(isi)).^2+xx(2,(
    isi)).^2); %zeta-b
m1(isi) = (-2*d*R_parameter)./sqrt(xx(1,(isi)).^2+xx(2,(isi)).^2);
m2(isi) = d./sqrt(xx(1,isi).^2+xx(2,isi).^2);
rho=[];
poly_coef = vs(end-3:-1:1);   %inverted polynomio
poly_coef_tmp = poly_coef;
for j = 1:length(m)
     poly_coef_tmp(end-1) = poly_coef(end-1)-m(j);
     poly_coef_tmp(end-2) = poly_coef(end-2)+m1(j);
     poly_coef_tmp(end-3) = poly_coef(end-3)+m2(j);
     rhoTmp = roots(poly_coef_tmp);
     %filter and keep only the real and positive solutions
     res = rhoTmp(find(imag(rhoTmp)==0 & rhoTmp>0));
     %pick the smallest solution
     if isempty(res) %| length(res)>1
          rho(j) = NaN;
     elseif length(res)>1
          rho(j) = min(res);
     else
          rho(j) = res;
     end
end
%reproject the points onto the sensor plane
x = xx(1,:)./sqrt(xx(1,:).^2+xx(2,:).^2).*rho ;
y = xx(2,:)./sqrt(xx(1,:).^2+xx(2,:).^2).*rho ;
```

**Table 5.11:** MATLAB code for the solution of the equation 5.5

## 5.4 AM-RZ MODEL

The models discussed in previous chapters describe the mathematical relationship between the coordinates of a point in the real world and its projection onto the image plane. Starting from the model introduced by Scaramuzza, **the contribution given by this thesis is to evolve the latter in the case of hyper-hemispheric cameras, considering that light rays do not converge towards a single point. This is more in agreement with the physical model which assumes the entrance pupil starting to appear tilted for low zenith angles**.
This change was performed by taking as an assumption that each light ray points to a point on a line, identified as the $Z$ axis of the camera coordinate system, so that we can imagine the $z_0$ function as a shift of the focus of the projection away from the point nominated to be the .
From direct measurements on the behaviour of the lens, what transpires is that the projection can be better described if we take into account that this particular shift occurs both on the Z axis and on the $\{X, Y\}$ plane.
The plots in 5.17 are examples of what is meant with this new concept of projecting a point outside the Z axis.

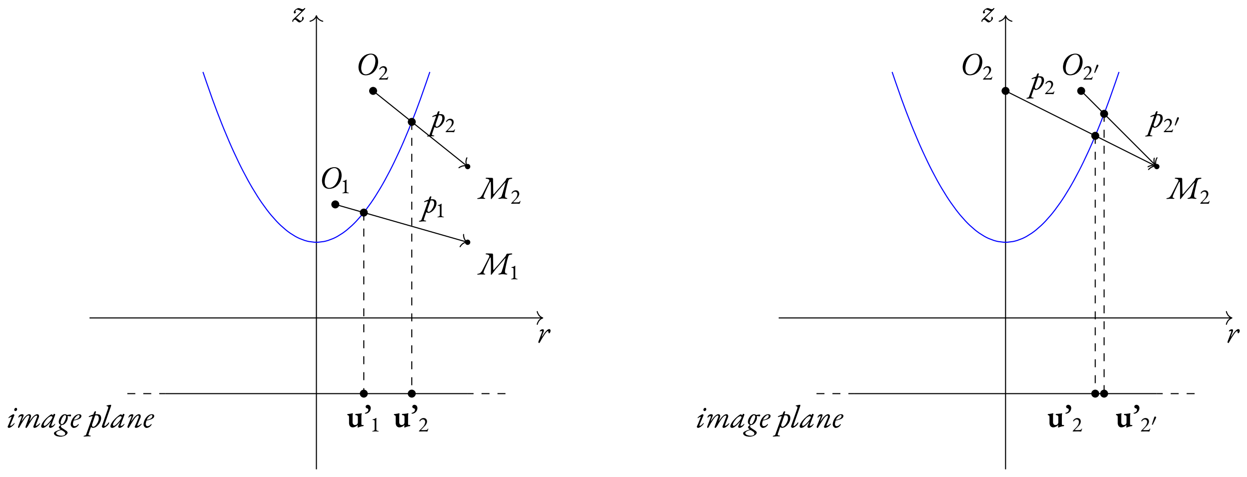


**Figure 5.17:** Left: projections of two points with the shift also on the $\rho$ axis.
Right: comparison between projections of a point according AM model and this new theory.

The plot on the left side shows the meaning of this new idea, with the Z axis that is no more involved in the projection. On the right, it is shown the difference between the projection according to the previous models, AM and piecewise-AM, and the projection which also adds along axis $\rho$, a vector lying in the $\{X, Y\}$ plane.
Obviously, one could argue that it would be possible to extend $p$ vectors towards the Z axis, thus ending up in the case of the previously mentioned AM and piecewise-AM models.
Despite that, a model which replies in a more accurate form the nature of the lens can perform interesting results.
Given these new initial premises, the path is ready for an additional edit to the equations and the toolbox code.
We can proceed similarly to what we did in section 5.3 and introduce a new function, whose name is going to be $\mathbf{r}_0(\rho)$, to rule the shift on the $\{X, Y\}$ plane. Equation 5.3 can be updated in its following version:

$$\frac{z_M - z_0(\rho)}{r_M - r_0(\rho)} = \frac{f(\rho)}{\rho}, \tag{5.16}$$

where $r_M = \sqrt{x_M^2 + y_M^2}$.
The nature of $r_0$ is defined as a function of the parameter $\rho$ and it follows from the same reasoning as for $z_0$: considering an ideal point towards whom every light ray shall converge, according to pinhole camera model and Scaramuzza's model, both functions $z_0$ and $r_0$ define two shifts from it in the projections, respectively on the Z axis and in the $\{X, Y\}$ plane.
Similarly to what we did for $z_0$, the function $r_0$ is chosen to be a piecewise function, with two traits: the first one as the null function, just to keep the model similar to all the previous ones, while a second trait follows a quadratic function.
The value that determines the switch form a trait to the other is chosen to be the hyper-parameter R, because we are assuming that the quadratic trait of $r_0$ is needed only for points related to high zenith angles.
Under these assumptions, equation 5.16 can be rewritten in the two forms:

$$\frac{z_M - a}{r_M} = \frac{f(\rho)}{\rho}\ , \rho \leq R \quad \text{and} \quad \frac{z_M - (b_0 + b_1 \cdot \rho + b_2 \cdot \rho^2)}{r_M - (c_0 + c_1 \cdot \rho + c_2 \cdot \rho^2)} = \frac{f(\rho)}{\rho}\ , \rho > R\ . \tag{5.17}$$

Equations 5.17 take $z_0$ from the second version of the piecewise-AM model, defined in 5.12 (a constant trait and a quadratic one). It is interesting to notice that, given $f(\rho)$ to be a polynomial of the fourth order, solving the second equation of 5.17 would mean to find the roots of a polynomial of the sixth order.
Similarly to what we did in the case of the $z_0$ function, we shall add simple constraints in order to get $r_0$ to be continuous and with continuous derivative. We can set up the following system:

$$\begin{cases} 0 & = c_0 + c_1 \cdot R + c_2 \cdot R^2 \\ 0 & = c_1 + 2 \cdot c_2 \cdot R\ . \end{cases} \tag{5.18}$$

This system's solutions are $c_1 = -2 \cdot c_2 \cdot R$ and $c_0 = c_2 \cdot R^2$, and, from a total of 7 parameters that we can find in 5.17, with these conditions of continuity, we end up with only 3 parameters:

- a, the constant value indicating the $1^{st}$ trait of $z_0$;
- $b_2$, the quadratic coefficient of the parabola in the second trait of $z_0$,
- $c_2$, the quadratic coefficient of the parabola in the second trait of $r_0$.

From now on, we can name this model as **AM-RZ**, reminding at the functions $r_0$ and $z_0$.
In order to evaluate the robustness of this model, calibrations are performed, on both datasets we have come to know in this discussion.
Starting from the dataset containing 7x10 chessboards, table 5.12 shows a comparison among different models regarding the errors committed in the reprojection of the corners on the images.

| | Avg error (px) | Std error on x (px) | Std error on y (px) |
|---|---|---|---|
| Scaramuzza's model | 0.823725 | 0.622204 | 0.576497 |
| Quadratic-AM | 0.814733 | 0.617055 | 0.570284 |
| Piecewise-AM (1$^{st}$ trait: constant) | 0.624592 | 0.484726 | 0.428303 |
| AM-RZ model | 0.630071 | 0.490191 | 0.427288 |

**Table 5.12:** Comparison among AM-RZ model and previous ones. The dataset involved is the one regarding small chessboards.

The new model behaves similarly to the previous piecewise-AM model meaning, outperforming the models introduced at the beginning of the discussion, Scaramuzza's one and AM models.
From the plots regarding the distributions of the error with respect to the zenith angle, plots 5.18, we get that even this model manages to reduce the error in the hyper-hemispheric side, which is comparable with the residuals of the errors regarding the narrow zone.

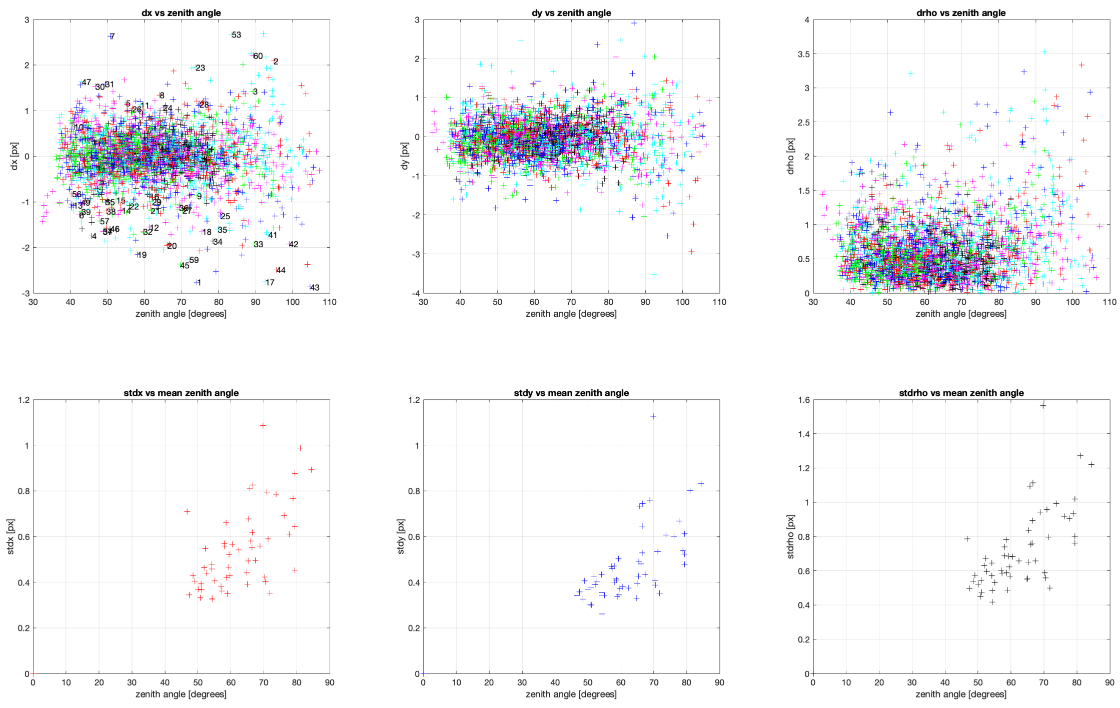


**Figure 5.18:** Distributions of the errors for zenithal angles, involving AM-RZ model. Dataset used for this calibration is the one involving 7x10 chessboards.

Analogous results occur if we focus on 28x10 chessboards dataset. Table 5.13 shows the residuals of the errors in the reprojection of the corners onto the images.
Along with these results, pictures 5.19 and 5.20 portray the image 49 of this dataset. The first one contains the

reprojection of the corners according to Scaramuzza's model. The second one, instead, performs AM-RZ model. It is immediate to notice how in the second pictures the reprojections (blue circles) are closer to the corner locations (red crosses), which means that the last model is way more accurate. For what concerns the implementation in Matlab of this last model, the function to be modified is always the one named omni3d2pixel, whose new code is reported in 5.14. The great mole of calculations present is due to the solution of the right equation of 5.17, which also increases the degree of the polynomial $f$ by 2.
Images acquired by PANCAM camera, as previously stated, are fish-eye pictures, with a black hole in the middle, which gives them the classical donut shape. In this images, the FoV is represented in a circle of radius 900 pixels. Now we can compare $r_0$ parabola to the shift that physically occurs in the lens, to investigate how much accurate this model is. Despite the good results achieved through this work, we don't have any matching between values assumed by the parabolic traits of $r_0$ and $z_0$ and the shift of the pupil measured.

| | Avg error (px) | Std error on x (px) | Std error on y (px) |
|---|---|---|---|
| Scaramuzza's model | 2.215582 | 1.519904 | 1.589742 |
| Quadratic-AM | 2.177170 | 1.510186 | 1.572896 |
| Piecewise-AM ($1^{st}$ trait: constant) | 1.248232 | 0.937039 | 0.929807 |
| AM-RZ | 1.250087 | 0.922428 | 0.919204 |

**Table 5.13:** Comparison between Scaramuzza's model, Quadratic-AM model and piecewise-AM models. The dataset involved is the one regarding long chessboards.

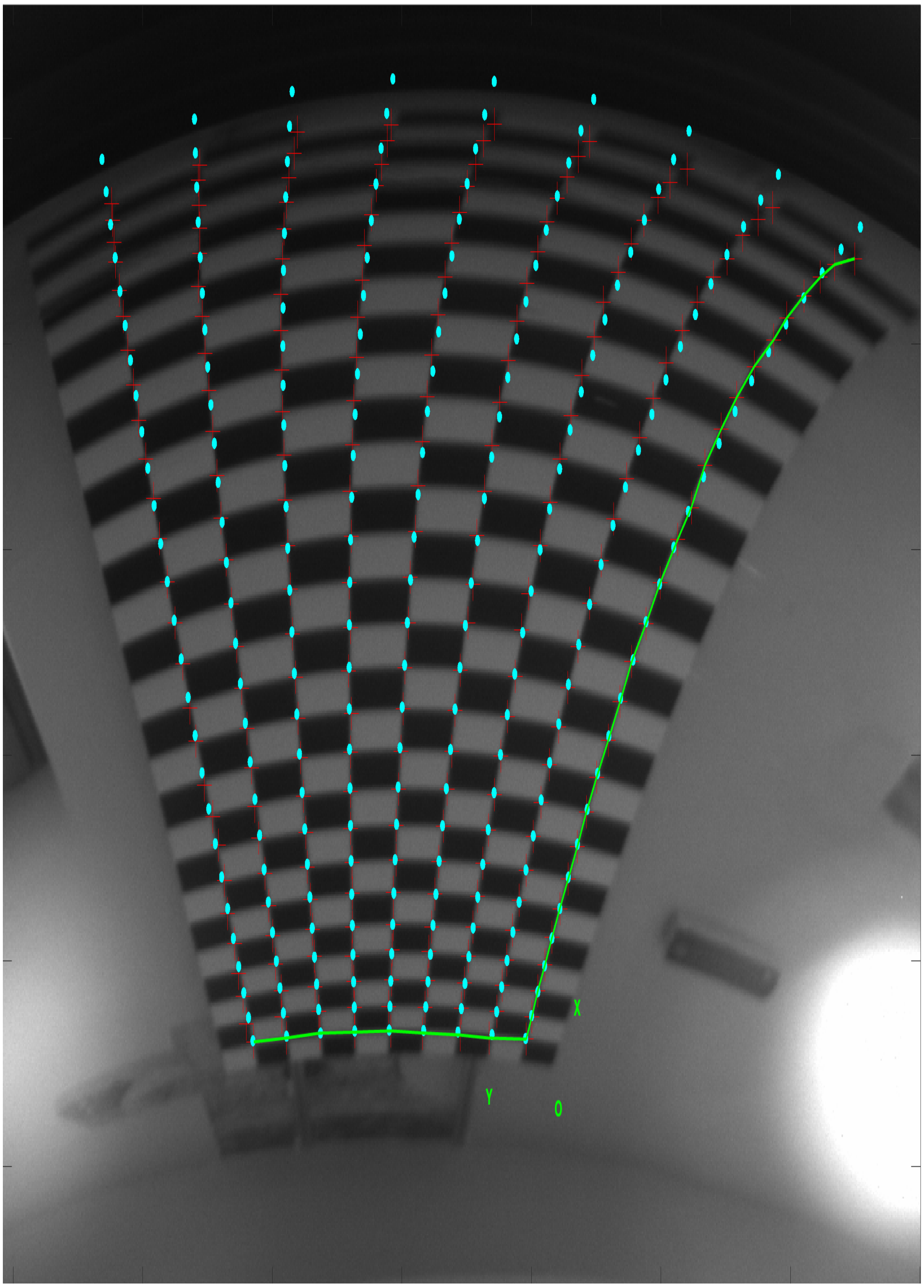


**Figure 5.19:** Reprojection of the corners for the $49^{th}$ picture, according to Scaramuzza's model

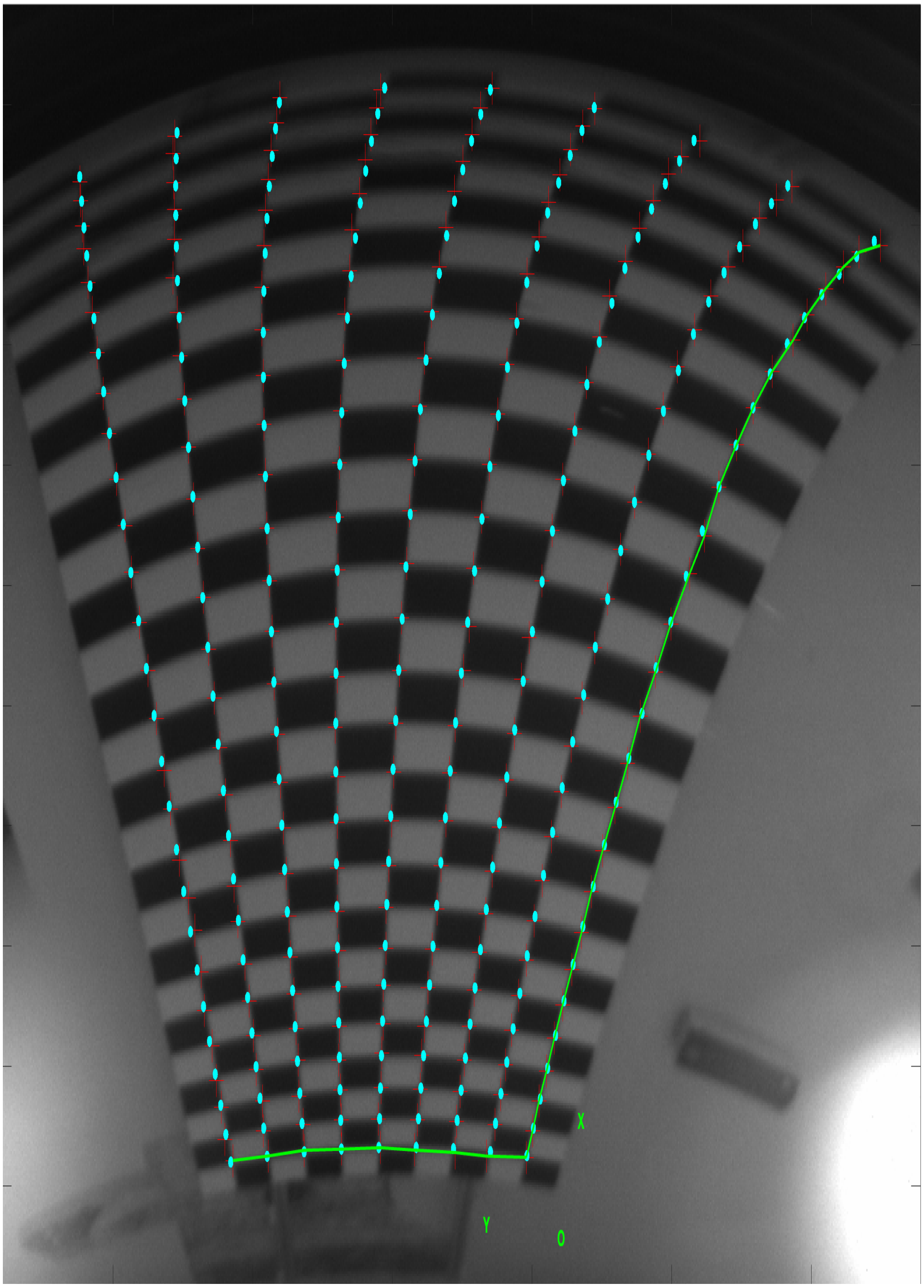


**Figure 5.20:** Reprojection of the corners for the $49^{th}$ picture, according to AM-RZ model.

```
function [x,y] = omni3d2pixel_am_rz(vs, xx, iper_emi)
%define iper-emisferic corners
load("R_parameter.mat"); % define hyper-parameter
isi=find(iper_emi); % define indices of iper-emisferic corners

%extract the three parameters of the model
a = vs(end-2); % first trait of z_0 : z_0 = a
b2 = vs(end-1); % z_0 = b0 + b1*x +b2 *x^2
c2 = vs(end);   % r_0 = c0 + c1*x + c2*X^2

%define parameters constrained by continuity equations
c0 = c2*R_parameter^2;
c1 = -2*c2*R_parameter;
b0 = b2*R_parameter^2 + a;
b1 = -2*b2*R_parameter;

r = sqrt(xx(1,:).^2+xx(2,:).^2);
poly_coef = vs(end-3:-1:1); %inverted polynomio
poly_coef = [0; 0; poly_coef']; %polynomial to solve has a degree
    which is N+2

%explicitly name coefficients of the polynomial
a0 = poly_coef(end);
a1 = poly_coef(end-1);
a2 = poly_coef(end-2);
a3 = poly_coef(end-3);
a4 = poly_coef(end-4);
a5 = poly_coef(end-5);
a6 = poly_coef(end-6);

%modify coefficients of the polynomial, for narrow part
m1 = a1-(xx(3,:)-a)./sqrt(xx(1,:).^2+xx(2,:).^2);
m0 = a0-0*m1;
m2 = a2-0*m1;
m3 = a3-0*m1;
m4 = a4-0*m1;
m5 = a5-0*m1;
m6 = a6-0*m1;
```

```
%modify coefficients of the polynomial, HH part
r(isi) = sqrt(xx(1,(isi)).^2+xx(2,(isi)).^2);

m0(isi) = a0*r(isi) - a0*c0 ;
m1(isi) = a1*r(isi) - a1*c0 - a0*c1 - xx(3, isi) +b0;
m2(isi) = a2*r(isi) - a2*c0 - a1*c1 - a0*c2 + b1;
m3(isi) = a3*r(isi) - a3*c0 - a2*c1 - a1*c2 +b2;
m4(isi) = a4*r(isi) - a4*c0 - a3*c1 - a2*c2;
m5(isi) = -a4*c1 - a3*c2;
m6(isi) = -a4*c2;

rho=[];
poly_coef_tmp = poly_coef;
for j = 1:length(m1)

    poly_coef_tmp(end)   = m0(j);
    poly_coef_tmp(end-1) = m1(j);
    poly_coef_tmp(end-2) = m2(j);
    poly_coef_tmp(end-3) = m3(j);
    poly_coef_tmp(end-4) = m4(j);
    poly_coef_tmp(end-5) = m5(j);
    poly_coef_tmp(end-6) = m6(j);

    rhoTmp = roots(poly_coef_tmp);
    %filter and keep only the real and positive solutions
    res = rhoTmp(find(imag(rhoTmp)==0 & rhoTmp>0));
    %pick the smallest solution
    if isempty(res) %| length(res)>1
        rho(j) = NaN;
    elseif length(res)>1
        rho(j) = min(res);
    else
        rho(j) = res;
    end
end

%reproject the points onto the sensor plane
x = xx(1,:)./sqrt(xx(1,:).^2+xx(2,:).^2).*rho ;
y = xx(2,:)./sqrt(xx(1,:).^2+xx(2,:).^2).*rho ;
```

**Table 5.14:** MATLAB code for the solution of the equation 5.5

# 6 Conclusion

Throughout this work, we introduced a very peculiar lens, called PANCAM, which belongs to the group of the hyper-hemispheric lenses, which historically have no precise geometrical model for their characterisation. The focus of the discussion has been to define a proper mathematical model for the geometrical calibration of this specific lens, an initial but indispensable step in image processing and possible applications in photogrammetry, where the chef ray direction associate to each pixel must be known as accurately as possible.
At first, the pinhole camera projection model was introduced, a basic model which assumes that all light rays converge towards a single point. This model represents the basis of the one introduced by **Scaramuzza**, whose aim is to calibrate hyper-hemispheric cameras. Scaramuzza is Professor of Robotics and Perception at the University of Zurich, where he directs the Robotics and Perception Group. It research lies at the intersection of **robotics**, **computer vision**, and **machine learning** and he is involved in projects with NASA, PHILIPS, BOSCH, DAIMLER.
Despite this, Scaramuzza's model did not adapt very well to PANCAM lens, since it struggled in the reprojections of high zenith points While the algorithm is strong for fish eye cameras (which can be considered central cameras ) the PANCAM (which has the advantage to cover with their field of view more than one hemisphere) is a non-central camera. This thesis proposed new models which abandon the single effective viewpoint assumption, by introducing a function, named $z_0$, which associates each pixel to a point on the Z axis of the coordinate system of the camera, such that all the light rays directed towards this point are then mapped onto the given pixel.
These **A-central models (AM)** were the first brand new models of this category and they feature a linear or quadratic $z_0$ function. Despite these new assumptions, these models still don't manage to overcome the results obtained by Scaramuzza's model. Moreover, they did not tackle the issue regarding points on the image associated to high zenith real-world points.
A second category of models is proposed hereafter, named piecewise-AM. They differ from the previous model by defining a different $z_0$ function (which defines the pupil position). As the name suggests, this function is defined with two different traits, one of which is specific for the hyper-hemispheric part.

The results provided by piecewise-AM models are way better than the previous ones, either in the errors of projection and in the capability of correctly managing high-zenithal points.
The last model proposed is named AM-RZ and it introduces a new function, named $r_0$. The aim of this function is to move the point towards which a light ray is directed out of the Z axis of the camera coordinate system .
Differently with the Scaramuzza model or the other proposed the AM-RZ model is the closer representation of the pupilla phenomenology in an hyper-hemispheric lens. The projection center in fact must be not considered fix or moving only in an axis but tilting himself even radially. In this manner, this modification wants to create a model closer to the physical description of the lens. Results of this model are comparable with the ones occurred for piecewise-AM models.
What inputs would be useful to extend the analysis that were provided and continue the work defended in this discussion?
First of all, we can state that refusing the single viewpoint property, stated at the beginning of chapter 3, proved to be crucial in order to obtain better results. Splitting the $z_0$ function, allowed to build up a sort of different model for the hyper-hemispheric side, which solves the problems regarding points belonging to this part of the image. This idea is the core of the work and the results obtained by piecewise-AM model are enough to state that this is the concept that new researches shall follow.
Also the architecture can be improved, for example in the choice of the hyper-parameter R, which can be included in the set of intrinsic parameters and refined during the refinement phase.
Moreover, it was certainly useful to build a model which follows the characteristics of the lens.
Including in our models the fact that we are dealing with a non-central camera allowed us to obtain better results, since the models are now closer to reality. This is an example suggesting that a new model capable of describing in a better way some aspects of the lens would certainly lead to interesting results.

# References

[1] C. Pernechele, "Hyper hemispheric lens," *Optics Express*, vol. 24, p. 5014, 03 2016.

[2] K. Miyamoto, "Fish eye lens," *J. Opt. Soc. Am.*, vol. 54, no. 8, pp. 1060–1061, Aug 1964. [Online]. Available: https://opg.optica.org/abstract.cfm?URI=josa-54-8-1060

[3] M. Laikin, "Wide Angle Lens Systems," in *1980 International Lens Design Conference*, R. E. Fischer, Ed., vol. 0237, International Society for Optics and Photonics. SPIE, 1980, pp. 530 – 533. [Online]. Available: https://doi.org/10.1117/12.959125

[4] D. Scaramuzza, A. Martinelli, and R. Siegwart, "A flexible technique for accurate omnidirectional camera calibration and structure from motion," in *Fourth IEEE International Conference on Computer Vision Systems (ICVS'06)*, 01 2006, p. 45.

[5] M. Beghini, "Mathematical models for camera calibration and application to the pancam," Bachelor's thesis, University of Padua, Padua, December 2022.

[6] J. Haruyama, T. Morota, S. Kobayashi, S. Sawai, P. Lucey, M. Shirao, and M. Nishino, *Lunar Holes and Lava Tubes as Resources for Lunar Science and Exploration*, 01 2012, pp. 139–163.

[7] E. Simioni, C. Pernechele, R. Pozzobon, M. Massironi, V. Della Corte, M. Landoni, B. Saggin, and D. Scaccabarozzi, "The daedalus cam: the immersive and stereoscopic way for lunar lava tubes esploration," *4th International Planetary Caves Conference 2023 (LPI Contrib. No. 2697)*, 2023.

[8] E. Simioni, C. Pernechele, C. Re, L. Lessio, and G. Cremonese, "Geometrical calibration for the panrover: A stereo omnidirectional system for planetary rover," *The International Archives of the Photogrammetry, Remote Sensing and Spatial Information Sciences*, vol. XLIII-B3-2020, pp. 1151–1158, 2020. [Online]. Available: https://isprs-archives.copernicus.org/articles/XLIII-B3-2020/1151/2020/

[9] W. Burger, "Zhang's camera calibration algorithm: In-depth tutorial and implementation," 2016. [Online]. Available: https://www.researchgate.net/publication/303233579_Zhang's_Camera_Calibration_Algorithm_In-Depth_Tutorial_and_Implementation

[10] E. Simioni, V. Da Deppo, C. Re, G. Naletto, E. Martellato, D. Borrelli, M. Dami, G. Aroldi, I. Ficai-Veltroni, and G. Cremonese, "Geometrical distortion calibration of the stereo camera for the bepicolombo mission to mercury," 06 2016.

[11] R. Hartley and A. Zisserman, *Multiple view geometry in computer vision*. Cambridge university press, 2003.

[12] Z. Zhang, "A flexible new technique for camera calibration," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 22, no. 11, pp. 1330–1334, 2000.

[13] D. Scaramuzza, A. Martinelli, and R. Siegwart, "A toolbox for easily calibrating omnidirectional cameras," *IEEE International Conference on Intelligent Robots and Systems*, 10 2006.

[14] H. P. Gavin, "The levenberg-marquardt method for nonlinear least squares curve-fitting problems c ©," 2013. [Online]. Available: https://api.semanticscholar.org/CorpusID:5708656

[15] D. Scaramuzza, "Ocamcalib: Omnidirectional camera calibration toolbox for matlab," 2006. [Online]. Available: https : / / sites . google . com / site / scarabotix / ocamcalib-omnidirectional-camera-calibration-toolbox-for-matlab?authuser=0

[16] C. Harris and M. Stephens, "A combined corner and edge detector," in *Procedings of the Alvey Vision Conference 1988*. Alvey Vision Club, 1988. [Online]. Available: https://doi.org/10.5244%2Fc.2.23

[17] M. Rufli, D. Scaramuzza, and R. Siegwart, "Automatic detection of checkerboards on blurred and distorted images," *2008 IEEE/RSJ International Conference on Intelligent Robots and Systems, IROS*, pp. 3121–3126, 09 2008.