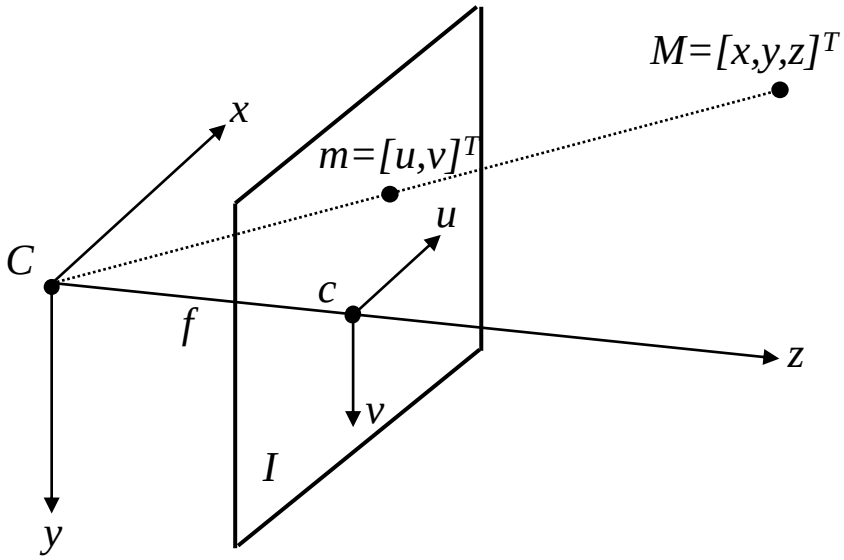
Image Processing and Computer Vision

Image Formation and Acquisition – Camera Calibration

Perspective Projection

- Let us consider
  - A point in the 3D space, $M=[x,y,z]^T$, with coordinates given in the Camera Reference Frame (CRF).
  - Its projection onto the image plane I, denoted as $m=[u,v]^T$
- The **non-linear** equations providing image coordinates as a function of the 3D coordinates in the CRF are as follows:



$$\begin{cases} u = \dfrac{f}{z} x \\ v = \dfrac{f}{z} y \end{cases}$$

Projective Space

- The physical space is a 3D Euclidean Space ($\mathbf{R}^3$) whose points can be represented as 3D vectors in a given reference frame.
  - In this space parallel lines do not intersect, or intersect " at infinity ".
  - Points at infinity cannot be represented in this vector space.
- Let's now append one more coordinate to our Euclidean triples, so that e.g.

$$\begin{pmatrix} x & y & z \end{pmatrix} \text{ becomes } \begin{pmatrix} x & y & z & 1 \end{pmatrix}$$

and assume that both vectors are correct representations of the same 3D point.

- Moreover, we do not constrain the 4th coordinate to be 1 but instead assume

$$\begin{pmatrix} x & y & z & 1 \end{pmatrix} \equiv \begin{pmatrix} 2x & 2y & 2z & 2 \end{pmatrix} \equiv \begin{pmatrix} kx & ky & kz & k \end{pmatrix} \forall k \neq 0$$

- In this representation a point in space is represented by an **equivalence class** of **quadruples**, wherein equivalent quadruples differ just by a multiplicative factor.
- This is the so called **homogeneous coordinates** (a.k.a. projective coordinates) representation of the 3D point having Euclidean coordinates (X, Y, Z). The space associated with the homogeneous coordinates representation is called **Projective Space**, denoted as $\mathbf{P}^3$.
- Extension to Euclidean spaces of any other dimension is straightforward ($\mathbf{R}^n \rightarrow \mathbf{P}^n$)

Point at infinity of a 3D line

Let's consider the parametric equation of a 3D line:

$$\mathrm{M} = M_0 + \lambda D = \begin{bmatrix} x_0 \\ y_0 \\ z_0 \end{bmatrix} + \lambda \begin{bmatrix} a \\ b \\ c \end{bmatrix} = \begin{bmatrix} x_0 + \lambda a \\ y_0 + \lambda b \\ z_0 + \lambda c \end{bmatrix}$$

and represent the generic point along the line in projective coordinates:

$$\widetilde{M} = \begin{bmatrix} M \\ 1 \end{bmatrix} = \begin{bmatrix} x_0 + \lambda a \\ y_0 + \lambda b \\ z_0 + \lambda c \\ 1 \end{bmatrix} = \begin{bmatrix} \frac{x_0}{\lambda} + a \\ \frac{y_0}{\lambda} + b \\ \frac{z_0}{\lambda} + c \\ \frac{1}{\lambda} \end{bmatrix}$$

by taking the limit with $\lambda \to \infty$ we obtain the projective coordinates of the point at infinity of the given line:

$$\widetilde{M}_\infty = \lim_{\lambda \to \infty} \widetilde{M} = \begin{bmatrix} a \\ b \\ c \\ 0 \end{bmatrix}$$

The projective coordinates of the point at infinity of a 3D line are obtained by taking any Euclidean vector parallel to the line and appending a 0 as fourth coordinate. There exist infinitely many points at infinity in $\mathbf{P^3}$, as many as the directions of the 3D lines.

Points at infinity

- The points of the 3D Projective Space **having the fourth coordinate equal to 0** , e.g. (x, y, z, 0), are the points at infinity of the 3D lines. These points cannot be represented in the 3D Euclidean Space.
- Indeed, to map such points into the Euclidean Space we would divide by the –null- fourth coordinate, so as to get (x/0, y/0, z/0), i.e. infinite coordinates, which is not a valid representation in the Euclidean Space.
- By the homogenous coordinates it is therefore possible to represent and process seamlessly both ordinary points as well as *points at infinity*.
- Point (0, 0, 0, 0) is undefined.
  - indeed, the above point is NOT the origin of the Euclidean Space (0, 0, 0), for such point is represented in homogeneous coordinates as (0, 0, 0, k), $k \neq 0$.
- It can be shown that all points at ininity of $\mathbf{P}^3$ lie on a plane, which is called the **plane at infinity**.

Recap

- Any Euclidean Space $\mathbf{R}^n$ can be extended to a corresponding **Projective Space $\mathbf{P}^n$** by representing points in homogeneous coordinates.
- The projective representation features one additional coordinate, referred to here as k, wrt the Euclidean representation:
  - $k \neq 0$ denotes point existing in $\mathbf{R}^n$, their coordinates given by x_i/k, $i = 1 \ldots n$
  - $k = 0$ denotes points at infinity (a.k.a. ideal points) in $\mathbf{R}^n$, which do not admit a representation via Euclidean coordinates.
- The Projective Space allows then to represent and process homogeneously (i.e. without introduction of exceptions or special cases) both the ordinary and the ideal points of the Euclidean Space.
- Why are we interested in Projective Spaces ?

Because Perspective Projection is more conveniently dealt with using projective coordinates !

Perspective Projection in projective coordinates (1)

- We already know that there exist a non-linear transformation between 3D coordinates and image coordinates:

$$u = \frac{f}{z}x \quad v = \frac{f}{z}y$$

- Let's now go back to our 3D point *M* (with coordinates expressed in the CRF) and its projection onto the image plane, *m*:

$$\boldsymbol{M} = [x,y,z]^T \qquad \boldsymbol{m} = [u,v]^T .$$

and represent both points in homogenous coordinates (denoted by symbol ~)

$$\tilde{\mathbf{m}} = \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} \qquad \tilde{\mathbf{M}} = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

Perspective Projection in projective coordinates (2)

- In homogeneous coordinates (hence considering the mapping between projective spaces) the prespective projection becomes a **linear transformation**:

$$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \begin{bmatrix} f\frac{x}{z} \\ f\frac{y}{z} \\ 1 \end{bmatrix} = \begin{bmatrix} fx \\ fy \\ z \end{bmatrix} = \begin{bmatrix} f & 0 & 0 & 0 \\ 0 & f & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix}$$

- In matrix notation

$$k\tilde{\mathbf{m}} = \tilde{\mathbf{P}}\tilde{\mathbf{M}}$$

Show examples:
- VP generic line
- VP lines // z axis
- VP lines // image plane

often expressed also as below, where $\approx$ means "equal up to an arbitrary scale factor"

$$\tilde{\mathbf{m}} \approx \tilde{\mathbf{P}}\tilde{\mathbf{M}}$$

Perspective Projection Matrix (PPM)

- Matrix $\tilde{\mathbf{P}}$ represents the geometric camera model, and is known as **Perspective Projection Matrix (PPM).**
- If we assume distances to be measured in focal lenght units ($f = 1$), the PPM becomes

$$\tilde{\mathbf{P}} = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} = \left[\mathbf{I} \mid \mathbf{0}\right]$$

- This form is useful to understand the core operation carried out by perspective projection, which is indeed scaling lateral coordinates (i.e. x,y) according to the distance from the camera (z). The actual focal length just introduces an additional, fixed (i.e. independent of z) scaling factor of projected coordinates. The above form is usually referred to as *canonical* or *standard* PPM.

A more comprehensive camera model

- To come up with a really useful camera model we need to take into account two additional issues:
  - Image Digitization;
  - The 6 DOF (3D rotation and translation) rigid motion between the Camera Reference Frame (CRF) and the World Reference Frame (WRF).

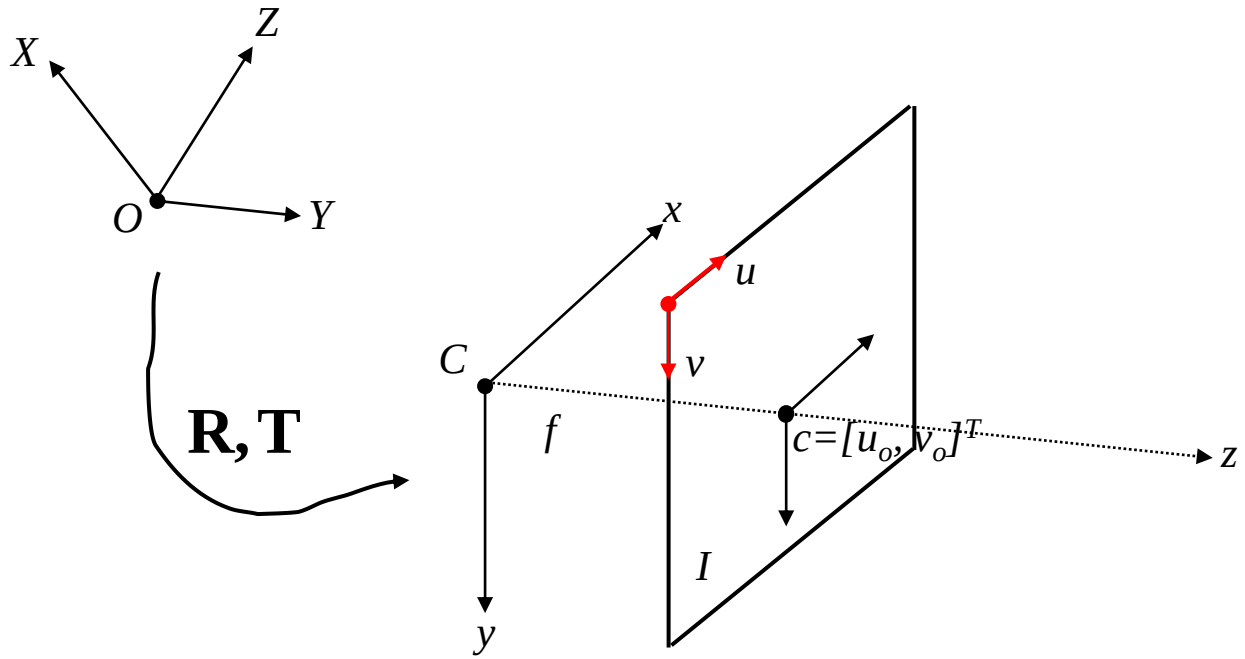
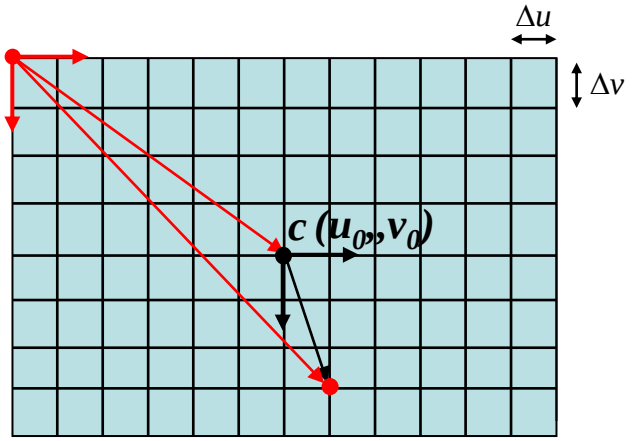


Image Digitization



Δu= horizontal pixel size
Δv= vertical pixel size

$$u = \frac{f}{z}x \qquad \rightarrow u = \frac{1}{\Delta u}\frac{f}{z}x = k_u \frac{f}{z}x + u_0$$

$$v = \frac{f}{z}y \qquad \rightarrow v = \frac{1}{\Delta v}\frac{f}{z}y = k_v \frac{f}{z}y + v_0$$

Digitization can be accounted for by including into the projection equations the scaling factors along the two axis due to the quantization associated with the horizontal and vertical pixel size. Morever, we need to model the translation of the piercing point (intesection between the optical axis and the image plane) wrt the origin of the image coordinate system (top-left corner of the image).

Intrinsic Parameter Matrix

- Based on the equations in the previous slide, the PPM can be written as

$$\tilde{\mathbf{P}} = \begin{bmatrix} fk_u & 0 & u_0 & 0 \\ 0 & fk_v & v_0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} = \begin{bmatrix} fk_u & 0 & u_0 \\ 0 & fk_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} = \mathbf{A}\left[\mathbf{I} \mid \mathbf{0}\right]$$

- Matrix **A**, which models the characteristics of the image sensing device, is called **Intrinsic Parameter Matrix**.
- Intrinsic parameters can be reduced in number by setting $\alpha_u = fk_u$, $\alpha_v = fk_v$, such quantities representing, respectively, the focal length expressed in horizontal and vertical pixel sizes. The smallest number of intrinsic parameters is thus 4.
- A more general model would include a 5^{th} parameter , known as *skew*, to account for possible non orthogonality between the axis of the image sensor. The *skew* would be A[1,2], but it is usually 0 (= ctg(π/2)) in practice.

Rigid motion between CRF and WRF (1)

- So far we have assumed 3D coordinates to be measured into the CRF, though this is hardly feasible in practice.
- More generally, 3D coordinates are measured into a World Reference Frame (WRF) external to the camera. The WRF will be related to the CRF by:
  - A rotation around the optical centre (e,g. expressed by a 3x3 rotation matrix **R**)
  - A translation (expressed by a 3x1 translation vector **T**)
- Therefore, the relation between the coordinates of a point in the two RFs is:

$$\mathbf{W} = \begin{bmatrix} X \\ Y \\ Z \end{bmatrix}, \mathbf{M} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} \Rightarrow \mathbf{M} = \mathbf{R}\mathbf{W} + \mathbf{T}$$

Which can be rewritten in homogeneous coordinates as follows:

$$\tilde{\mathbf{W}} = \begin{bmatrix} X \\ Y \\ Z \\ 1 \end{bmatrix}, \tilde{\mathbf{M}} = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} \Rightarrow \tilde{\mathbf{M}} = \begin{bmatrix} \mathbf{R} & \mathbf{T} \\ \mathbf{0} & 1 \end{bmatrix} \tilde{\mathbf{W}} = \mathbf{G}\tilde{\mathbf{W}}$$

Rigid motion between CRF and WRF (2)

So far we have seen how to map a 3D point expressed in the CRF

$$k\tilde{\mathbf{m}} = \mathbf{A}\left[\mathbf{I}\,|\,\mathbf{0}\right]\tilde{\mathbf{M}}$$

- We need now to consider also the rigid motion between the WRF and the CRF :

$$\tilde{\mathbf{M}} = \mathbf{G}\tilde{\mathbf{W}} \longrightarrow k\tilde{\mathbf{m}} = \mathbf{A}\left[\mathbf{I}\,|\,\mathbf{0}\right]\mathbf{G}\tilde{\mathbf{W}}$$

$$k\tilde{\mathbf{m}} = \mathbf{A}\left[\mathbf{I}\,|\,\mathbf{0}\right]\begin{bmatrix}\mathbf{R} & \mathbf{T}\\ \mathbf{0} & 1\end{bmatrix}\tilde{\mathbf{W}}$$

- Accordingly, the general form of the PPM can be expressed as follows:

$$\tilde{\mathbf{P}} = \mathbf{A}\left[\mathbf{I}\,|\,\mathbf{0}\right]\mathbf{G} \quad \text{or also} \quad \tilde{\mathbf{P}} = \mathbf{A}\left[\mathbf{R}\,|\,\mathbf{T}\right]$$

Extrinsic Parameters

- Matrix **G**, which encodes the position and orientation of the camera with respect to the WRF, is called **Extrinsic Parameter Matrix**.

- As a rotation matrix (3x3=9 entries) has indeed only 3 independent parameters (DOF), which correspond to the rotation angles around the axis of the RF, the total number of extrinsic parameter is 6 (3 translation parameters, 3 rotation parameters).

- Hence, the general form of the PPM can be thought of as encoding the position of the camera wrt the world into **G,** the perspective projection carried out by a pinhole camera into the canonical PPM $[\,\mathbf{I}\,|\,\mathbf{0}\,]$ and, finally, the actual characteristics of the sensing device into **A**.

Lens Distortion

- However, to explain observed images we often need to model also the effects due to the optical distortion induced by lenses, which indeed renders the pure pinhole not accurate enough a model in many applications. Lens distortion is modelled through additional parameters that do not alter the form of the PPM.

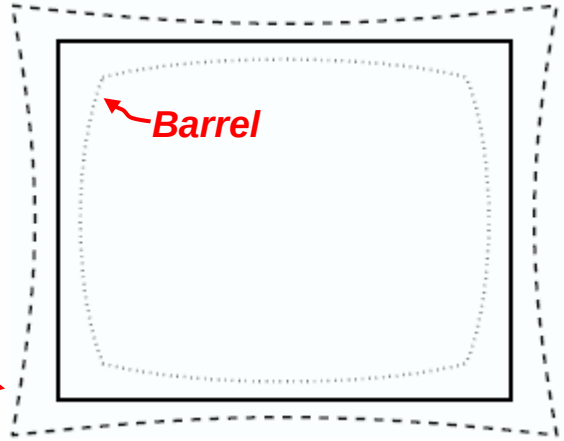
Modelling Lens Distortion

- The PPM is based on the pinhole camera model.
- However, real lenses introduce distortions wrt to the pure pinhole model, this being true especially for cheap and/or short focal length lenses. The most significant deviation from the ideal pinhole model is known as **radial distortion** (lens "curvature"). Second order effects are induced by **tangential distortion** ("misalignement" of optical components and/or defects).
- Lens distortion is modeled through a non-linear transformation which maps ideal (i,e. *undistorted*) image coordinates into the observered (i.e. *distorted*) image coordinates.

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = L(r)\begin{pmatrix} \tilde{x} \\ \tilde{y} \end{pmatrix} + \begin{pmatrix} d\tilde{x} \\ d\tilde{y} \end{pmatrix}$$

depending on the distance r from the distortion centre $(\tilde{x}_c, \tilde{y}_c)$

$$r = \sqrt{(\tilde{x} - \tilde{x}_c)^2 + (\tilde{y} - \tilde{y}_c)^2}$$



Lens Distortion Parameters

- The radial distortion function *L(r)* is defined for positive *r* only and such as *L*(0) = 1. This non-linear function is typically approximated by its Taylor series (up to the a certain approximation order)

$$L(r) = 1 + k_1 r^2 + k_2 r^4 + k_3 r^6 + \ldots$$

- The tangential distortion vector is instead approximated as follows

$$\begin{pmatrix} d\tilde{x} \\ d\tilde{y} \end{pmatrix} = \begin{pmatrix} 2p_1\tilde{x}\tilde{y} + p_2\left(r^2 + 2\tilde{x}^2\right) \\ p_1\left(r^2 + 2\tilde{y}^2\right) + 2p_2\tilde{x}\tilde{y} \end{pmatrix}$$

- The radial distortion coefficients $k_1, k_2, \ldots, k_n$, together with the distortion centre $(\tilde{x}_c, \tilde{y}_c)$ and the two tangential distortion coefficients p_1 and p_2 form the set of the lens distortion pararmeters, which extends the set of parameter required to build a useful and realistic camera model. Typically, for the sake of simplicity, the distortions centre is taken to coincide with the image centre (i.e. the piercing point).

Lens distortion within the image formation flow

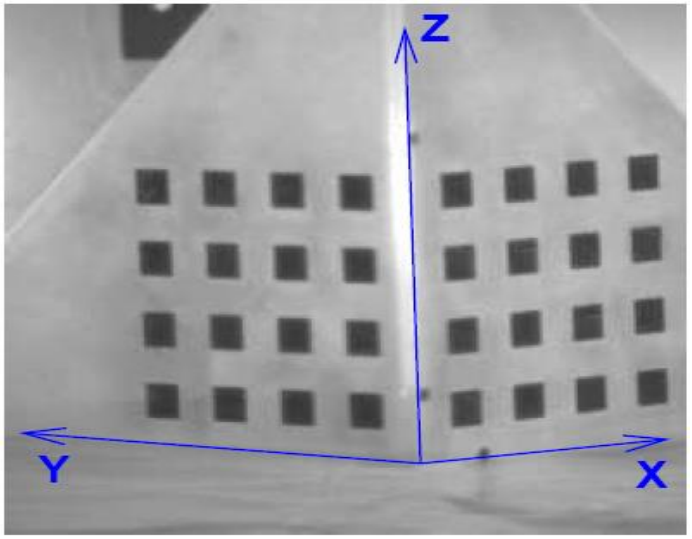
- Lens distortion is modelled as a non-linear mapping taking place after canonical perspective projection onto the image plane. Afterwards, the intrinsic parameter matrix allow applying an affine transformation which maps image coordinates into pixel coordinates.
- Accordingly, the image formation flow can be summarized as follows:
  1. Transformation of 3D points from the WRF to the CRF, according to extrinsic parameters:
  $$\mathbf{M} = \mathbf{RW} + \mathbf{T}$$
  2. Canonical pespective projection (i.e. scaling by the third coordinate):
  $$\tilde{x} = x / z \quad , \quad \tilde{y} = y / z$$
  3. Non linear mapping due to lens distortion:
  $$\begin{pmatrix} x' \\ y' \end{pmatrix} = L(r) \begin{pmatrix} \tilde{x} \\ \tilde{y} \end{pmatrix} + \begin{pmatrix} d\tilde{x} \\ d\tilde{y} \end{pmatrix}$$
  4. Mapping from image coordinates to pixels coordinates according to the intrinsic parameters:
  $$\mathbf{m} = \mathbf{A}(x' \; y' z)^T$$

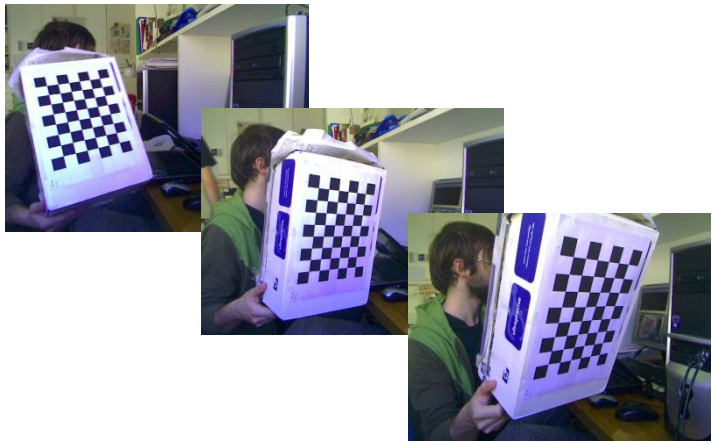
Calibration (1)

- We have now at disposal a camera model thoroughly explaining the image formation and digitization process.
- Such a model is the PPM, which in turn can be decomposed into 3 independent tokens: intrinsic parameter matrix (**A**), rotation matrix (**R)** and translation vector (**T)**.
- **Camera calibration** is the process whereby all parameters defining the camera model are - as accurately as possible- estimated for a specific camera device. Depending on the application, either the PMM only or also its independent components (**A**, **R**, **T)** need to be estimated.
- Many camera calibration algorithms do exist. The basic process, though, relies always on setting up a *linear* system of equations given a set of known 3D-2D correspondences, so as to then solve for the unknown camera parameters.
- To obtain the required correspondences specific physical objects (referred to as calibration targets) having easily detectable features (such as e.g. chessboard or dot patterns) are typically deployed.

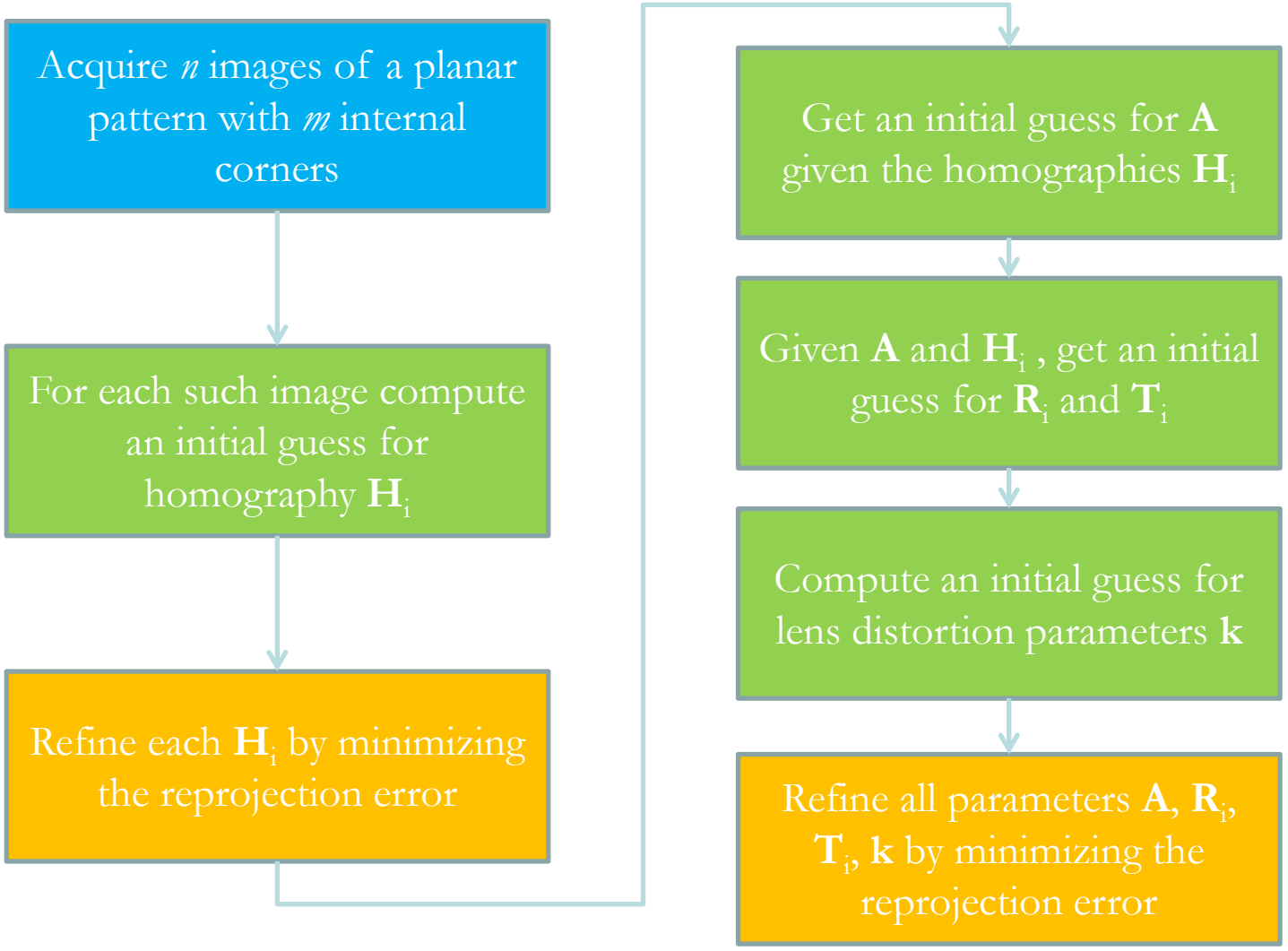
Calibration (2)

- Camera calibration approaches can be split into two main categories:
  - Those relying on a **single image** featuring **several** (at least 2) **planes** containing a known pattern**.**
  - Those relying on **several** (at least 3) different **images** of **one** given **planar pattern**.
- In practise, it is difficult to build accurate targets containing multiple planes, while an accurate planar target can be attained rather easily.
- Implementing a camera calibration software requires a significant effort. However, the main Computer Vision toolboxes include specific functions (OpenCV, Matlab CC Toolbox, Halcon.)





Zhang's Method (1)

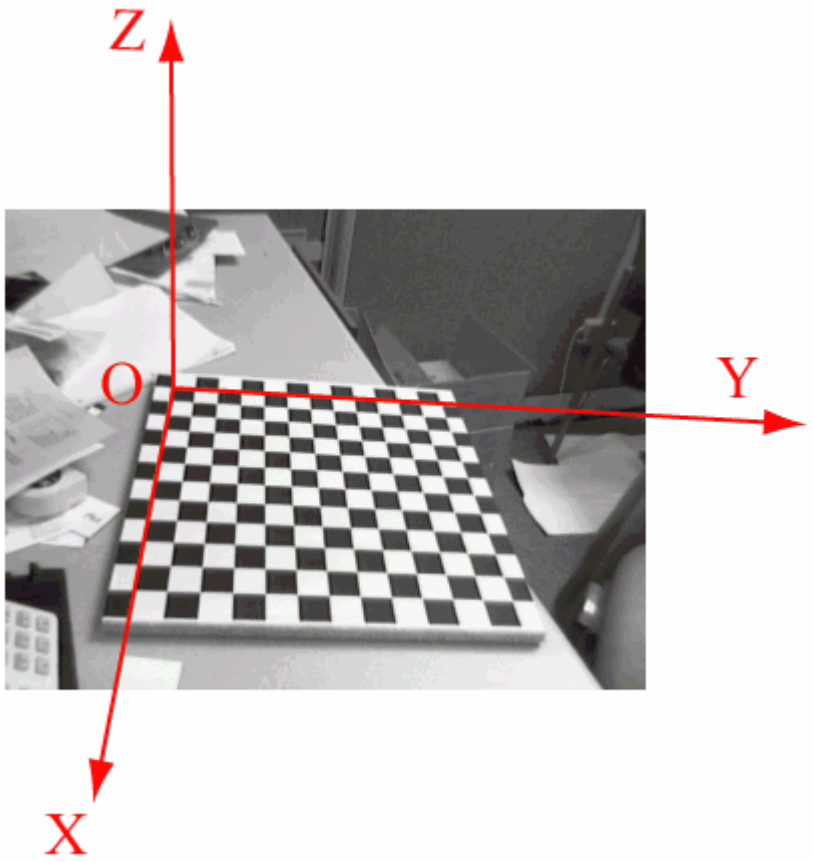


Zhang's Method (2)

- Given a planar chessboard pattern, known are:
  - The number of internal corners of the pattern, different along the two orthogonal directions for the sake of disambiguation (i.e. rows, columns).
  - The size of the squares which form the pattern.
- Internal corners can be detected easily by standard algorithms (e.g. the Harris corner detector, possibly with sub-pixel refinement for improved accuracy).
- In each image, the 3D WRF is taken at the top-left corner of the pattern, with plane z=0 given by the pattern itself and the x,y axis aligned to the two orthogonal directions, in particular so as to keep always the same association between axis and directions (e.g. x=rows, y = columns). Thus, as for 3D points:
  - The third coordinate is always 0.
  - x and y are determined by the known size of the squares forming the chessboard.
- To get camera parameters, two main steps are carried out:
  - Initial guess by a linear optimization (minimization of an algebraic error)
  - Refinement by a non-linear minimization (minimization of a geometric error)

Estrinsic Parameters

- It is worth pointing out that each image requires its own estimate of the extrinsic parameters, as they are different from one to the other.
  - A global WRF can be taken to coincide with that associated with one of the image, e.g. the first.



P as a Homography

- Due to the choice of the WRF associated with calibration images, in each of them we consider only 3D points with z=0. Accordingly, the PPM boils down to a simpler transformation defined by a 3x3 matrix:

$$k\tilde{\mathbf{m}} = \tilde{\mathbf{P}}\tilde{\mathbf{w}} = \begin{bmatrix} p_{1,1} & p_{1,2} & p_{1,3} & p_{1,4} \\ p_{2,1} & p_{2,2} & p_{2,3} & p_{2,4} \\ p_{3,1} & p_{3,2} & p_{3,3} & p_{3,4} \end{bmatrix} \begin{bmatrix} x \\ y \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} p_{1,1} & p_{1,2} & p_{1,4} \\ p_{2,1} & p_{2,2} & p_{2,4} \\ p_{3,1} & p_{3,2} & p_{3,4} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix} = \mathbf{H}\tilde{\mathbf{w}}'$$

- Such a transformation, denoted here as **H,** is known as **homography** and represents a general linear transformation between planes. Above, **$\tilde{\mathbf{w}}$'** represents vector (x; y; 1). **H** can be thought of as a simplification of **P** in case the imaged object is planar.
- Given a pattern with m corner, we can write m systems of 3 linear equations as above, wherein both 3D as well as 2D coordinates are known due to corners having been detected in the i^{th} image and the unknowns are thus the 9 elements in $\mathbf{H_i}$. However, as $\mathbf{H_i}$, and $\mathbf{P_i}$ alike, is known up to an arbitrary scale factor, the independent elements in $\mathbf{H_i}$ are indeed just 8.

Estimating $\mathbf{H}_i$ (1)

- Starting from the previous homography equation we can write

$$k\tilde{\mathbf{m}} = \mathbf{H}\tilde{\mathbf{w}}' \quad \Rightarrow \quad \tilde{\mathbf{m}} \times \mathbf{H}\tilde{\mathbf{w}}' = \mathbf{0} \quad \Rightarrow \quad \begin{bmatrix} v\mathbf{h}_3^{\mathbf{T}}\tilde{\mathbf{w}}' - \mathbf{h}_2^{\mathbf{T}}\tilde{\mathbf{w}}' \\ \mathbf{h}_1^{\mathbf{T}}\tilde{\mathbf{w}}' - u\mathbf{h}_3^{\mathbf{T}}\tilde{\mathbf{w}}' \\ u\mathbf{h}_2^{\mathbf{T}}\tilde{\mathbf{w}}' - v\mathbf{h}_1^{\mathbf{T}}\tilde{\mathbf{w}}' \end{bmatrix} = \mathbf{0}, \mathbf{H} = \begin{bmatrix} \mathbf{h}_1^{\mathbf{T}} \\ \mathbf{h}_2^{\mathbf{T}} \\ \mathbf{h}_3^{\mathbf{T}} \end{bmatrix}$$

i.e.

$$\begin{bmatrix} \mathbf{0}^{\mathbf{T}} & -\tilde{\mathbf{w}}'^{\mathbf{T}} & v\tilde{\mathbf{w}}'^{\mathbf{T}} \\ \tilde{\mathbf{w}}'^{\mathbf{T}} & \mathbf{0}^{\mathbf{T}} & -u\tilde{\mathbf{w}}'^{\mathbf{T}} \\ -v\tilde{\mathbf{w}}'^{\mathbf{T}} & u\tilde{\mathbf{w}}'^{\mathbf{T}} & \mathbf{0}^{\mathbf{T}} \end{bmatrix} \begin{bmatrix} \mathbf{h}_1 \\ \mathbf{h}_2 \\ \mathbf{h}_3 \end{bmatrix} = \mathbf{A}\mathbf{h} = \mathbf{0}$$

- Between the previous 3 equations in 9 unknowns, only 2 are linearly independent, so that typically only the first 2 of them are kept. By deploying all correspondences, for each image we can build a linear system of $2m$ equations in 9 unknowns. This allows for achieving an initial estimation of $\mathbf{H}_i$ by minimization of the "algebraic error" represented by the norm of vector $\mathbf{Ah}$ and enforcing the additional constraint $\|\mathbf{h}\| = 1$ (DLT Algorithm)
- It is well known that the solution of the above estimation problem can be obtained by the Singular Value decomposition (SVD) of matrix $\mathbf{A}$.

Estimating $\mathbf{H}_i$ (2)

- Given the previous initial estimation, $\mathbf{H}_i$ is later refined by the least-squares solution of the non-linear minimization problem:

$$\min_{\mathbf{H}_i} \sum_j \left\| \tilde{\mathbf{m}}_j - \mathbf{H}_i \tilde{\mathbf{w}}'_j \right\|^2, \quad j = 1.....m$$

which can be obtained in practice using by the Levenberg-Marquardt algorithm.

- This additional optimization steps corresponds to the minimization of the reprojection error measured for each of the 3D corners (with z=0) of the pattern by comparing the pixel coordinates predicted by the estimated homograpy to the pixel coordinates of the corresponding corner extracted in the image. The rationale is that the "best" homography would predict with the best accuracy the positions of the corner features actually found in the image. Such a reprojection error is typically refereed to as "geometric error".

DLT Algorithm – 4 points case (1)

- By considering 4 point pairs, we obtain a system of 8 equations in 9 unknowns :

$$\mathbf{Ah} = 0, \quad \mathbf{A} = \left[\mathbf{A}_1 \ \mathbf{A}_2 ... \mathbf{A}_9\right], \ \mathbf{h} = \begin{bmatrix} \mathbf{h}_1 \\ \mathbf{h}_2 \\ \mathbf{h}_3 \end{bmatrix} = \begin{bmatrix} h_1 \\ h_2 \\ \vdots \\ h_9 \end{bmatrix}$$

where **A** is a 8x9 matrix, $\mathbf{A_1 \ldots A_9}$ are 8x1 vectors, **h** is a 9x1 vector, $\mathbf{h_1, h_2, h_3}$ are 3x1 vectors representing the rows of the 3x3 matrix **H** defining the homography, $h_1 ... h_9$ are the 9 elements of such matrix.

As **H** is defined up to a scale factor, we can set $h_9 = 1$, so as to obtain a non homogeneous linear system with 8 equations and 8 unknowns

$$\tilde{\mathbf{A}}\tilde{\mathbf{h}} = \mathbf{b}, \quad \tilde{\mathbf{A}} = \left[\mathbf{A}_1 \ \mathbf{A}_2 ... \mathbf{A}_8\right], \ \tilde{\mathbf{h}} = \begin{bmatrix} h_1 \\ h_2 \\ \vdots \\ h_8 \end{bmatrix}, \ \mathbf{b} = -\mathbf{A}_9$$

which can be solved easily by standard methods (Cramer's rule, Inversion of the coefficient matrix, *Gaussian Elimination*)

DLT Algorithm – 4 points case (2)

- Alternatively, it is possible to constrain the nom of **h**, e.g. so as to render it equal to 1. Purposely, we can assume again h_9 as fixed, but no longer equal to one :

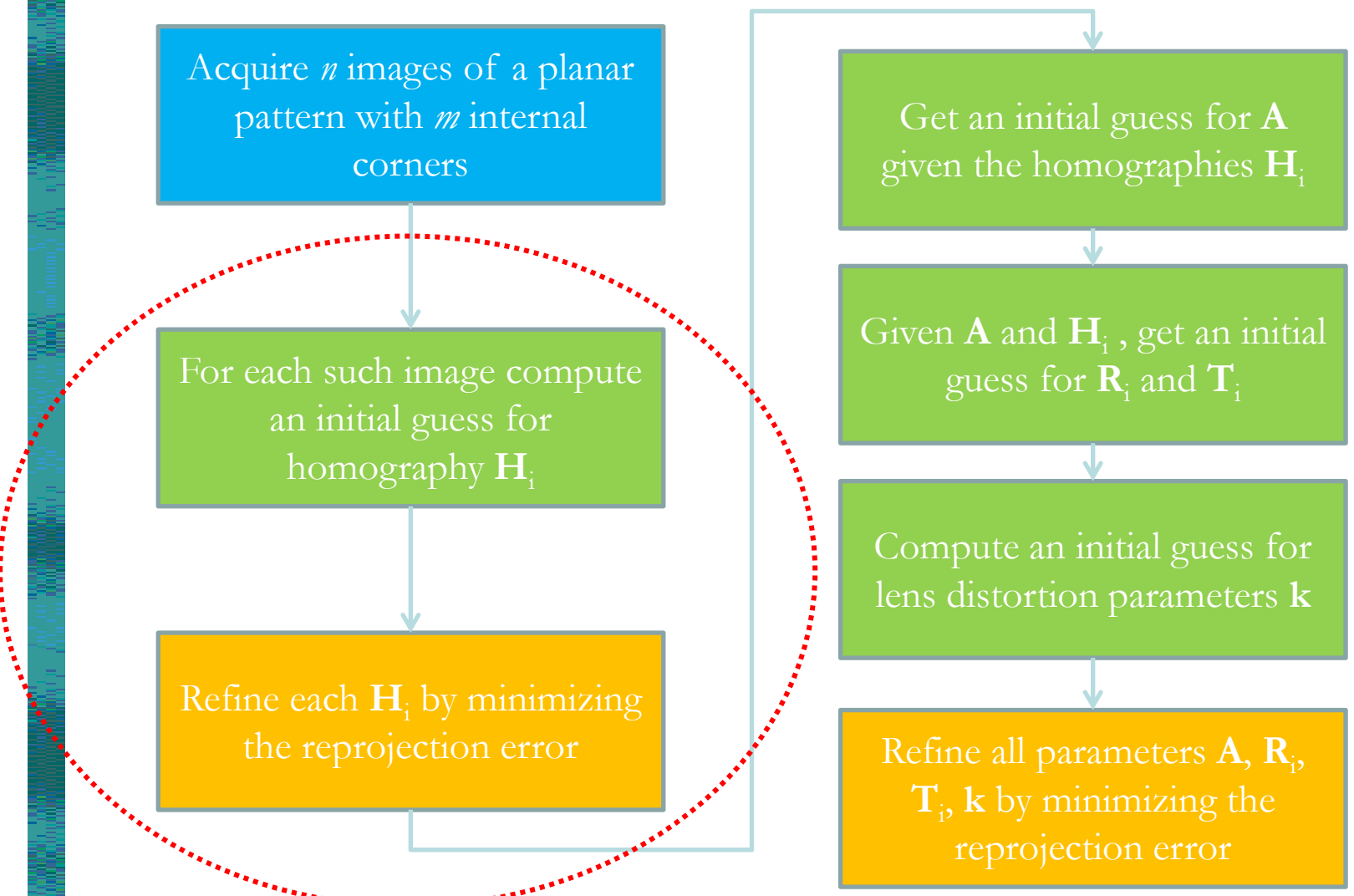
$$\tilde{\mathbf{A}}\tilde{\mathbf{h}} = \mathbf{b}, \quad \tilde{\mathbf{A}} = \left[\mathbf{A}_1 \ \mathbf{A}_2 \ldots \mathbf{A}_8\right], \ \tilde{\mathbf{h}} = \begin{bmatrix} h_1 \\ h_2 \\ \vdots \\ h_8 \end{bmatrix}, \ \mathbf{b} = -h_9\mathbf{A}_9$$

$$\tilde{\mathbf{h}} = -h_9\tilde{\mathbf{A}}^{-1}\mathbf{A}_9 \rightarrow \ \mathbf{h} = \begin{bmatrix} -h_9\tilde{\mathbf{A}}^{-1}\mathbf{A}_9 \\ h_9 \end{bmatrix} = h_9\begin{bmatrix} -\tilde{\mathbf{A}}^{-1}\mathbf{A}_9 \\ 1 \end{bmatrix}$$

$$\|\mathbf{h}\| = 1 \rightarrow h_9\sqrt{\left\|\tilde{\mathbf{A}}^{-1}\mathbf{A}_9\right\|^2 + 1} = 1 \rightarrow h_9 = \frac{1}{\sqrt{\left\|\tilde{\mathbf{A}}^{-1}\mathbf{A}_9\right\|^2 + 1}}$$

- According to this method, the coefficient matrix of the linear system needs is inverted and then, using the formulas shown above it is possible to compute h_9 first and then **h**

Zhang's Method



Estimation of the intrinsic parameters (1)

- As $\mathbf{H}_i$ is known up to a scale factor, we can establish the following relation between $\mathbf{H}_i$ and the PPM:

$$\mathbf{H} = \begin{bmatrix} \mathbf{h_1} & \mathbf{h_2} & \mathbf{h}_3 \end{bmatrix} = \begin{bmatrix} \mathbf{p_1} & \mathbf{p_2} & \mathbf{p_4} \end{bmatrix} = \lambda \mathbf{A} \begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 & \mathbf{T} \end{bmatrix}$$

- **R** is an orthogonal matrix, its vectors being orthonormal. This constrains the intrinsic parameters to obey to the following relations:

$$\mathbf{r}_1^T \mathbf{r}_2 = 0 \quad \Rightarrow$$

$$\mathbf{r}_1^T \mathbf{r}_1 = \mathbf{r}_2^T \mathbf{r}_2 \quad \Rightarrow$$

where the unknowns are the entries of $\mathbf{B}=\mathbf{A}^{-T}\mathbf{A}^{-1}$. As **A** is upper triangular, **B** turns out to be symmetric, so that the unknowns are just 6.

- By stacking together the above two equations provided by each calibration image, we obtain a $2n$ x 6 linear system, which can be solved in case at least 3 images are available (more details on next slide)

Estimation of the extrinsic parameters (2)

- By posing

$$\mathbf{B} = \mathbf{A}^{-\mathbf{T}}\mathbf{A}^{-1} = \begin{pmatrix} B_{11} & B_{12} & B_{13} \\ B_{12} & B_{22} & B_{23} \\ B_{13} & B_{23} & B_{33} \end{pmatrix}, \quad b = \left[B_{11}, B_{12}, B_{22}, B_{13}, B_{23}, B_{33}\right]^T$$

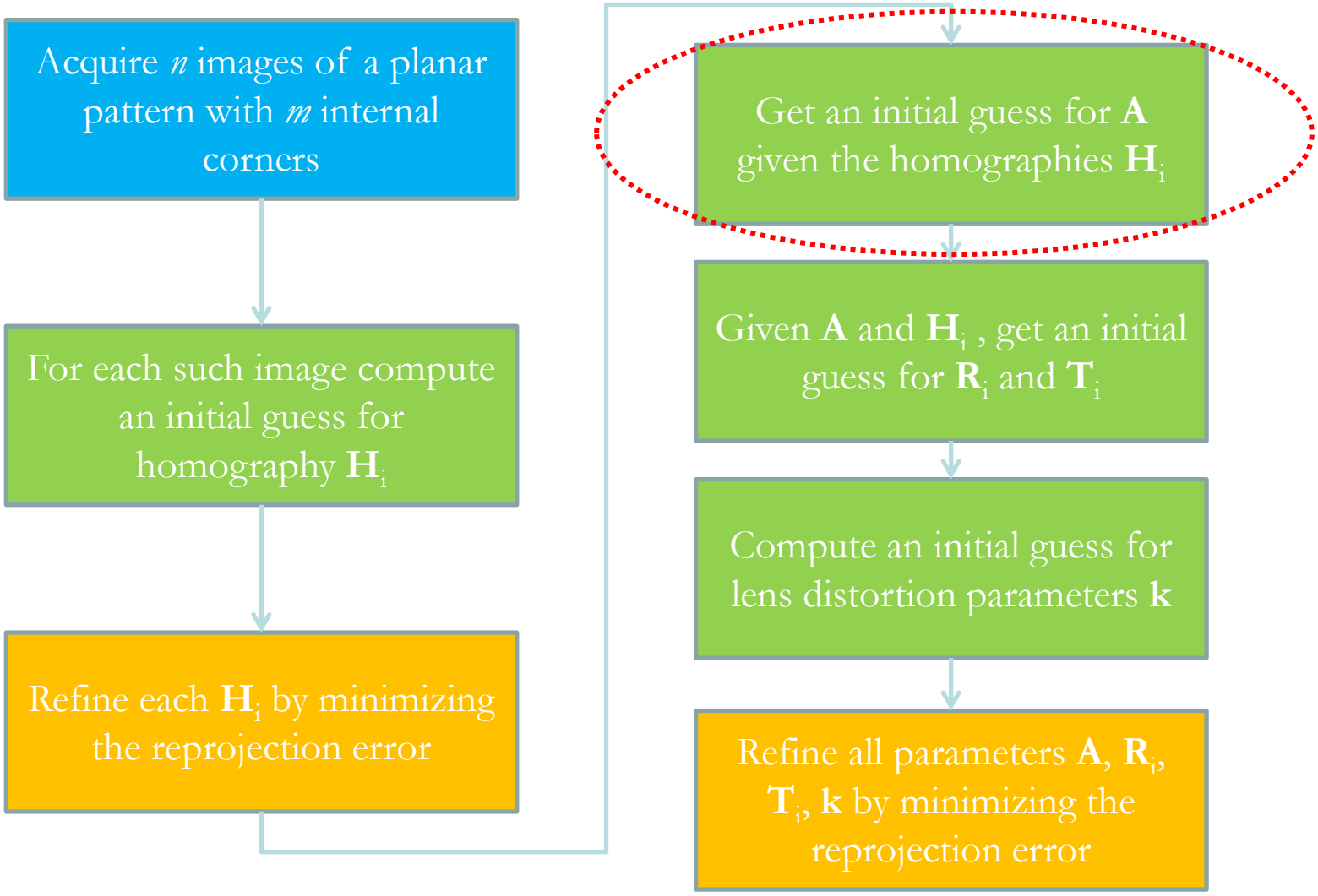
$$\mathbf{h}_i^T = \left[h_{i1}, h_{i2}, h_{i3}\right], \qquad \mathbf{v}_{ij}^T = \left[h_{i1}h_{j1},\ h_{i1}h_{j2} + h_{i2}h_{j1},\ h_{i2}h_{j2},\ h_{i3}h_{j1} + h_{i1}h_{j3},\ h_{i3}h_{j2} + h_{i2}h_{j3},\ h_{i3}h_{j3}\right]$$

it can be noticed that

$$\mathbf{h}_i^{\mathbf{T}}\mathbf{B}\,\mathbf{h}_j = \mathbf{v}_{ij}^{\mathbf{T}}\mathbf{b} \longrightarrow \begin{cases} \mathbf{h}_1^{\mathbf{T}}\mathbf{B}\mathbf{h}_2 = 0 \ \Rightarrow\ \mathbf{v}_{12}^{\mathbf{T}}\mathbf{b} = 0 \\ \mathbf{h}_1^{\mathbf{T}}\mathbf{B}\mathbf{h}_1 = \mathbf{h}_2^{\mathbf{T}}\mathbf{B}\mathbf{h}_2 \ \Rightarrow\ \mathbf{v}_{11}^{\mathbf{T}}\mathbf{b} = \mathbf{v}_{22}^{\mathbf{T}}\mathbf{b} \ \Rightarrow \left(\mathbf{v}_{11} - \mathbf{v}_{22}\right)^{\mathbf{T}}\mathbf{b} = 0 \end{cases}$$

- Therefore, each image provides 2 equations in the 6 independent unknowns in **B,** so that with con *n* calibration images we get a homogeneous linear system of equations in the form $\mathbf{Vb} = 0$, which can be solved in a least squares sense.
- Once **b** has been calculated, the intrinsic parameters (i.e. **A**) can be obtained in closed form.

Zhang's Method



Estimation of the extrinsic parameters

- Once **A** has been estimated, for each image it is possible to compute **R** and then **T** given the previously computed homography **H**:

$$\mathbf{H} = \begin{bmatrix} \mathbf{h}_1 & \mathbf{h}_2 & \mathbf{h}_3 \end{bmatrix} = \lambda \mathbf{A} \begin{bmatrix} \mathbf{r}_1 & \mathbf{r}_2 & \mathbf{T} \end{bmatrix}$$

$$\mathbf{h}_1 = \lambda \mathbf{A} \mathbf{r}_1 \quad \Rightarrow \quad \mathbf{r}_1 = \frac{1}{\lambda} \mathbf{A}^{-1} \mathbf{h}_1$$

- As $\mathbf{r}_1$ is a unit vector:

$$\lambda = \left\| \mathbf{A}^{-1} \mathbf{h}_1 \right\|, \quad \mathbf{r}_2 = \frac{1}{\lambda} \mathbf{A}^{-1} \mathbf{h}_2$$

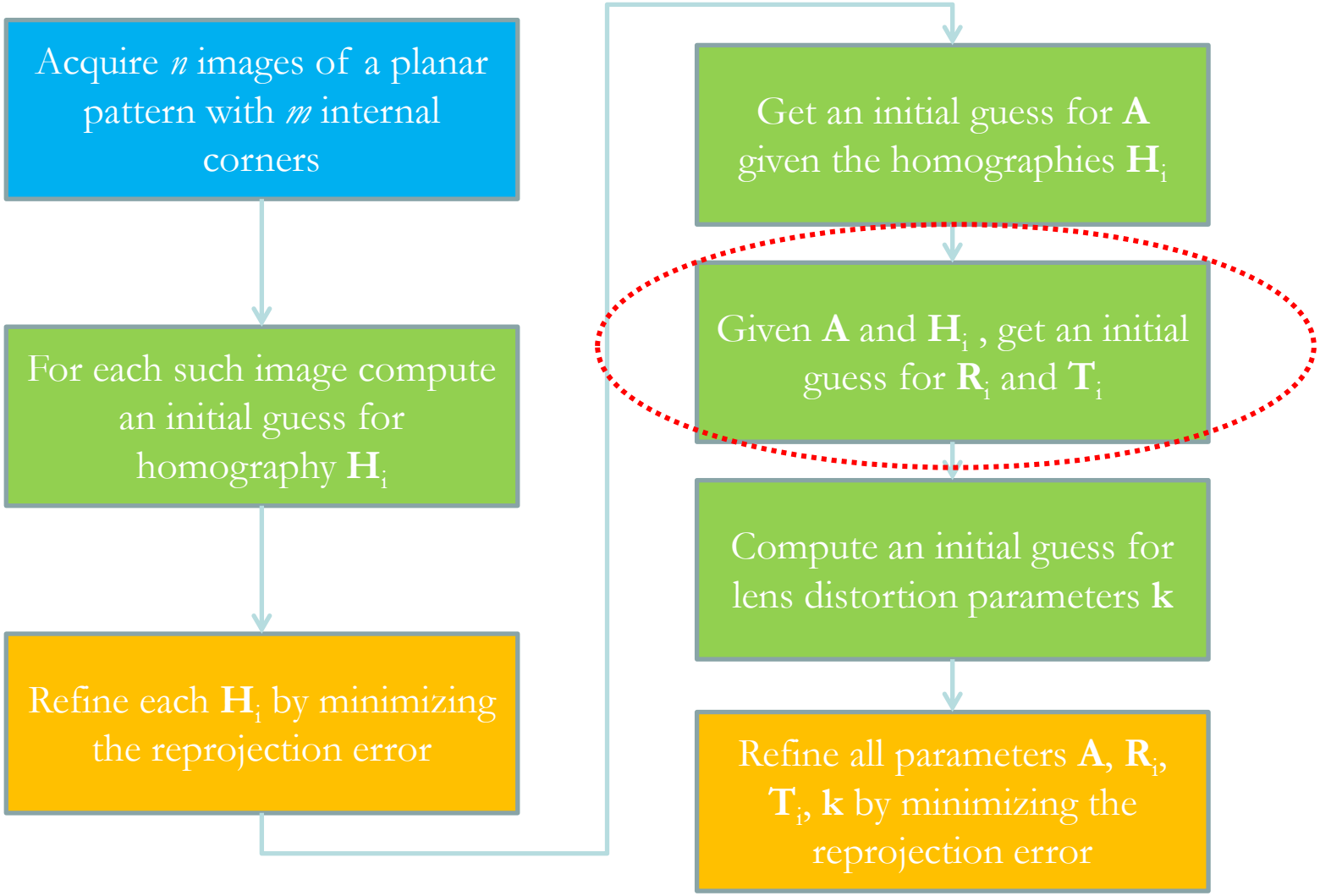
- $\mathbf{r}_3$ can be derived from $\mathbf{r}_1$ and $\mathbf{r}_2$ by exploiting orthonormality

$$\mathbf{r}_3 = \mathbf{r}_1 \times \mathbf{r}_2$$

- Finally we can get **T**

$$\mathbf{T} = \frac{1}{\lambda} \mathbf{A}^{-1} \mathbf{h}_3$$

Zhang's Method



Lens distortion coefficients (1)

- Given the homographies, we have both the real (distorted) coordinates of the corner features found in the images as well as the ideal (undistorted) coordinates predicted by the homographies . Zhang's method deploys such information to estimate coefficients k_1, k_2 of the radial distortion function.
- Given the already known intrinsic parameter matrix, **A**, and the lens distortion model reported below:

$$\begin{pmatrix} x' \\ y' \end{pmatrix} = L(r)\begin{pmatrix} \tilde{x} \\ \tilde{y} \end{pmatrix} = \left(1 + k_1 r^2 + k_2 r^4\right)\begin{pmatrix} \tilde{x} \\ \tilde{y} \end{pmatrix}$$

we need first to establish the relationship between distorted (u', v') and ideal $(\tilde{u}, \tilde{v})$ **pixel** coordinates:

$$\begin{bmatrix} u' \\ v' \\ 1 \end{bmatrix} = A\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = \begin{pmatrix} \alpha_u & 0 & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{pmatrix}\begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} \rightarrow \begin{cases} x' = \dfrac{u' - u_0}{\alpha_u} \\ y' = \dfrac{v' - v_0}{\alpha_v} \end{cases} \qquad \begin{bmatrix} \tilde{u} \\ \tilde{v} \\ 1 \end{bmatrix} = A\begin{bmatrix} \tilde{x} \\ \tilde{y} \\ 1 \end{bmatrix} \rightarrow \begin{cases} \tilde{x} = \dfrac{\tilde{u} - u_0}{\alpha_u} \\ \tilde{y} = \dfrac{\tilde{v} - v_0}{\alpha_v} \end{cases}$$

Lens distortion coefficients (2)

$$\begin{bmatrix} x' \\ y' \end{bmatrix} = \left(1 + k_1 r^2 + k_2 r^4\right) \begin{bmatrix} \tilde{x} \\ \tilde{y} \end{bmatrix} \rightarrow \begin{cases} \dfrac{u' - u_0}{\alpha_u} = \left(1 + k_1 r^2 + k_2 r^4\right)\left(\dfrac{\tilde{u} - u_0}{\alpha_u}\right) \\ \dfrac{v' - v_0}{\alpha_v} = \left(1 + k_1 r^2 + k_2 r^4\right)\left(\dfrac{\tilde{v} - v_0}{\alpha_v}\right) \end{cases}$$

$$\begin{cases} u' = \tilde{u} + \left(k_1 r^2 + k_2 r^4\right)\left(\tilde{u} - u_0\right) \\ v' = \tilde{v} + \left(k_1 r^2 + k_2 r^4\right)\left(\tilde{v} - v_0\right) \end{cases}$$

- It is therefore possible to set up a linear system where the unkowns are the distortion coefficients:

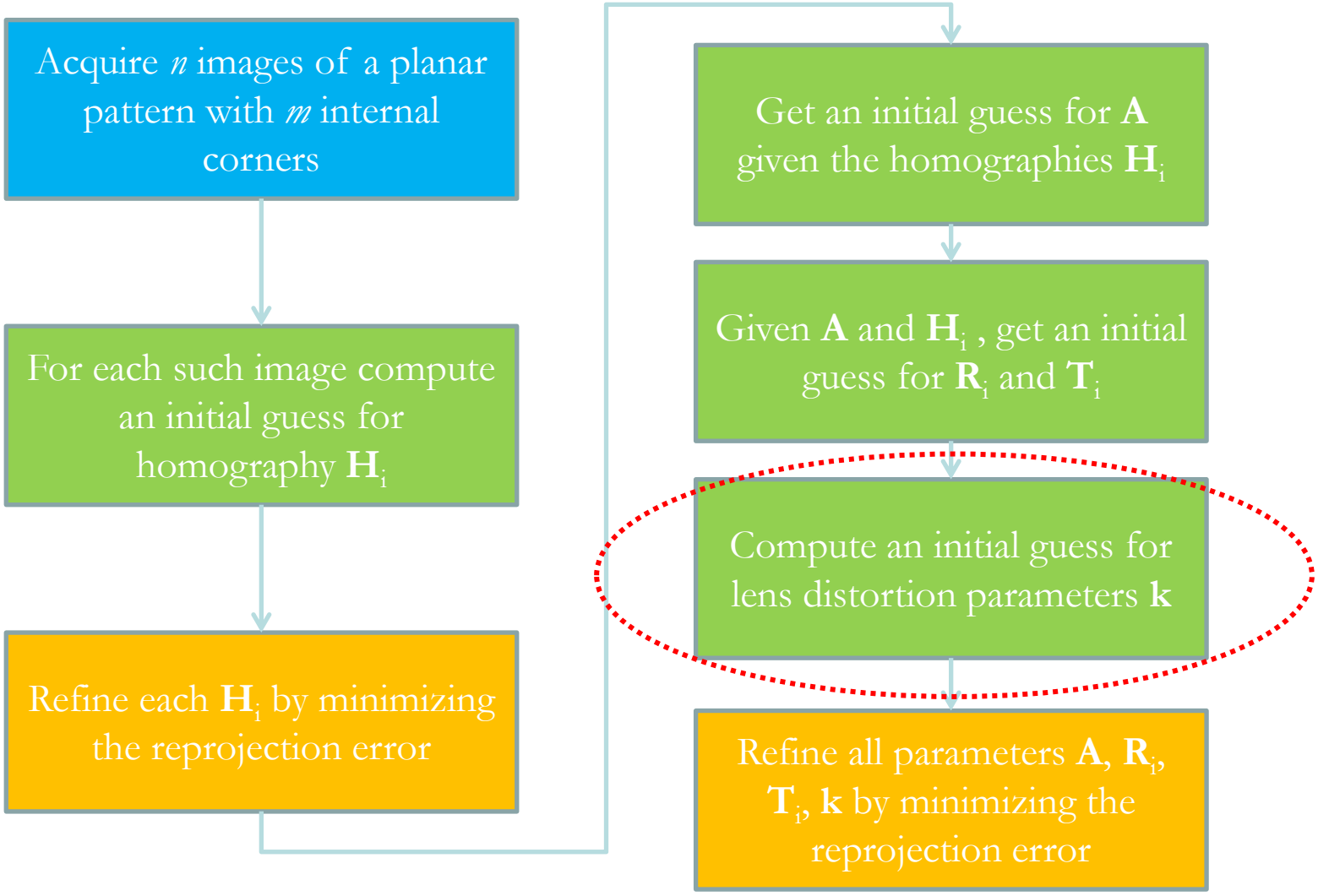
$$\begin{bmatrix} \left(\tilde{u} - u_0\right) r^2 & \left(\tilde{u} - u_0\right) r^4 \\ \left(\tilde{v} - v_0\right) r^2 & \left(\tilde{v} - v_0\right) r^4 \end{bmatrix} \begin{bmatrix} k_1 \\ k_2 \end{bmatrix} = \begin{bmatrix} u' - \tilde{u} \\ v' - \tilde{v} \end{bmatrix}$$

$$r^2 = \left(\frac{\tilde{u} - u_0}{\alpha_u}\right)^2 + \left(\frac{\tilde{v} - v_0}{\alpha_v}\right)^2$$

- With *m* corner features in *n* images, we get a linear system with *2mn* equations in *2* unknowns, which admits a least-squares solution:

$$\mathbf{Dk} = \mathbf{d} \rightarrow \mathbf{k} = \mathbf{D}^{+}\mathbf{d} = \left(\mathbf{D}^{\mathrm{T}}\mathbf{D}\right)^{-1}\mathbf{D}^{\mathrm{T}}\mathbf{d}$$

Zhang's Method



Refinement by non-linear optimization

- Again, the procedure highlighted so far seeks to minimize an algebraic error, without any real physical meaning.
- A more accurate solution can instead be found by a so called Maximum Likelihood Estimate (MLE) aimed at minimization of a geometric (i.e. reprojection) error.
- Under the hypothesis of *i.i.d.* (independent identically distributed) noise, the MLE for our models is obtained by minimization of the error

$$\sum_{i=1}^{n}\sum_{j=1}^{m}\left\|\mathbf{m}_{ij}-\hat{\mathbf{m}}\left(\mathbf{A},\mathbf{k},\mathbf{R}_i,\mathbf{T}_i,\mathbf{w}_j\right)\right\|^2$$

with respect to all the unknown camera parameters. .

- The solution of the above non-linear optimization problem is again provided by the Levenberg-Marquardt algorithm.

Decomposition of the PPM

- Zhang's method provides separately **A**, **R** e **T** . Other methods, though, compute directly the PPM (see, e.g. Hartley&Zisserman Chapter 7).
- It is however always possible to decompose any given PPM into its elementary components. Purposely, we can deploy the so called **QR factorization,** in particular to factorize a real square matrix into the product between an ***orthogonal*** and a ***upper triangular*** matrix.
- Let's then $\mathbf{P}^{-1} = \mathbf{UL}$ be the factorization of the inverse of the square matrix **P**, with **U** orthogonal and **L** upper triangular.
- Hence, to obtain the rotation matrix and the intrinsic parameter matrix:

$$\tilde{\mathbf{P}} = \left[\mathbf{P}\middle|\mathbf{p}_4\right] = \left[\mathbf{AR}\middle|\mathbf{AT}\right] \Rightarrow \mathbf{P}^{-1} = \mathbf{R}^{-1}\mathbf{A}^{-1} \Rightarrow$$

$$\mathbf{R} = \mathbf{U}^{-1}, \mathbf{A} = \mathbf{L}^{-1}$$

- Finally, the translation vector **T** is given by:

$$\mathbf{T} = \mathbf{A}^{-1}\mathbf{p}_4 = \mathbf{L}\mathbf{p}_4$$

Optical centre from the PPM

- It can be shown that a PPM is always a full rank (i.e. rank 3) matrix, and, alike, that every full-rank 3x4 matrix defines a perspective projection.
  - Intuitively: full rank is mandatory, as otherwise the projection of a 3D point will not be an image point but a higher dimensional subspace (such as a line or plane)
  - Recalling that (0,0,0) does not represent any valid point in homogeneous coordinates, we can observe that a point in space belonging to the matrix null space (or kernel, i.e. the solutions of the associated homogenous system) must be a point whose projection is undefined.
- Such a point is the optical centre, which is indeed the only point in space whose perspective projection onto the image plane is undefined (one cannot define a projection ray from the optical centre and itself). We can therefore write :

$$\tilde{\mathbf{P}}\begin{bmatrix}\mathbf{c}\\1\end{bmatrix}=\mathbf{0} \quad \text{and thus} \quad \mathbf{c}=-\mathbf{P}^{-1}\mathbf{p}_4 \quad \text{with} \quad \tilde{\mathbf{P}}=\left[\mathbf{P}\middle|\mathbf{p}_4\right]$$

3D coordinate of the optical center in the WRF

Optical ray from the PPM

- The optical ray of an image point, $\tilde{\mathbf{m}}$, is the 3D line between $\tilde{\mathbf{m}}$ and the optical centre, $\tilde{\mathbf{C}} = \begin{bmatrix} \mathbf{C} \\ 1 \end{bmatrix}$. Indeed, the optical ray is the locus of points, $\tilde{\mathbf{M}}$, satisfying the equation $k\tilde{\boldsymbol{m}} = \tilde{\mathbf{P}}\tilde{\mathbf{M}}$ for a given $\tilde{\mathbf{m}}$.

The equation of the optical ray (in Euclidean coordinates) can be determined as follows.

$$\mathbf{M} = \mathbf{C} + \lambda \mathbf{V}$$

optical centre

any vector parallel to the optical ray

$$\widetilde{M}_{\infty} = \begin{bmatrix} a \\ b \\ c \\ 0 \end{bmatrix} = \begin{bmatrix} \mathbf{V} \\ 0 \end{bmatrix}, \quad \tilde{\mathbf{P}} \begin{bmatrix} \mathbf{P}^{-1}\tilde{\mathbf{m}} \\ 0 \end{bmatrix} = [\mathbf{P} \ \boldsymbol{p}_4] \begin{bmatrix} \mathbf{P}^{-1}\tilde{\mathbf{m}} \\ 0 \end{bmatrix} = \tilde{\mathbf{m}}$$

is a point at infinity AND belongs to the optical ray

is the point at infinity of the optical ray

$$\mathbf{V} = \mathbf{P}^{-1}\tilde{\mathbf{m}}$$

$$\mathbf{M} = \mathbf{C} + \lambda \mathbf{P}^{-1}\tilde{\mathbf{m}}$$

$$\tilde{\mathbf{M}} = \tilde{\mathbf{C}} + \lambda \begin{bmatrix} \mathbf{P}^{-1}\tilde{\mathbf{m}} \\ 0 \end{bmatrix}$$ (in projective coordinates)

From pixels to 3D coordinates

If the camera is calibrated, i.e. matrix **A** is known, and depth is also known, as it is the case, e.g., for the reference image of a stereo camera or a TOF or Structured Light (e.g. Kinect) sensor, we can estimate the **3D coordinates of pixels in the CRF** (Camera Reference Frame) in order to obtain a so called ***point cloud*** , a representation widely used in **3D Computer Vision**.

$$\begin{bmatrix} \alpha_u x + u_0 z \\ \alpha_v y + v_0 z \\ z \end{bmatrix} = \begin{bmatrix} \alpha_u & 0 & u_0 \\ 0 & \alpha_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x \\ y \\ z \end{bmatrix}$$

$\mathbf{p}^*$ $\mathbf{A}$ $\mathbf{P}$

$\mathbf{p}^* = \mathbf{A}\,\mathbf{P}$

$\mathbf{P} = \mathbf{A}^{-1}\,\mathbf{p}^*$

$\mathbf{P} = z\,\mathbf{A}^{-1}\,\frac{\mathbf{p}^*}{z}$

$$\frac{\mathbf{p}^*}{z} = \begin{bmatrix} \alpha_u \frac{x}{z} + u_0 \\ \alpha_v \frac{y}{z} + v_0 \\ 1 \end{bmatrix} = \begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = \mathbf{p}$$

3D coordinates in the CRF — depth — inverse of A — pixel coordinates

Thus: $\mathbf{P} = z\,\mathbf{A}^{-1}\,\mathbf{p}$

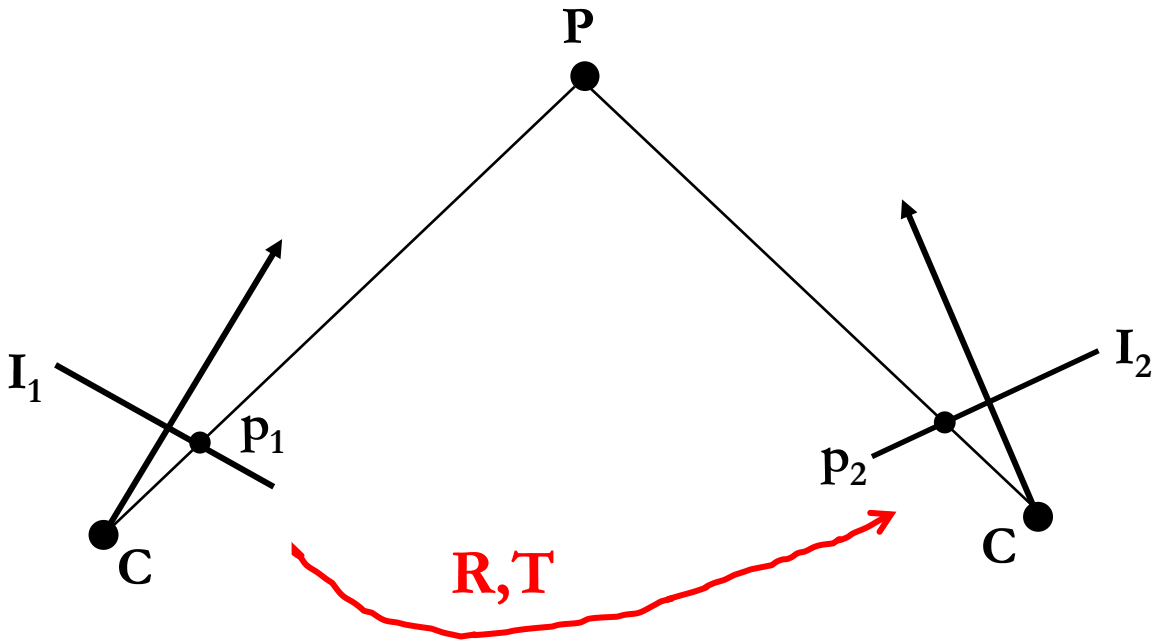
with:

$$A^{-1} = \begin{bmatrix} \frac{1}{\alpha_u} & 0 & -\frac{u_0}{\alpha_u} \\ 0 & \frac{1}{\alpha_v} & -\frac{v_0}{\alpha_v} \\ 0 & 0 & 1 \end{bmatrix}$$

Samuele Salti, Luigi Di Stefano

..and back to pixels

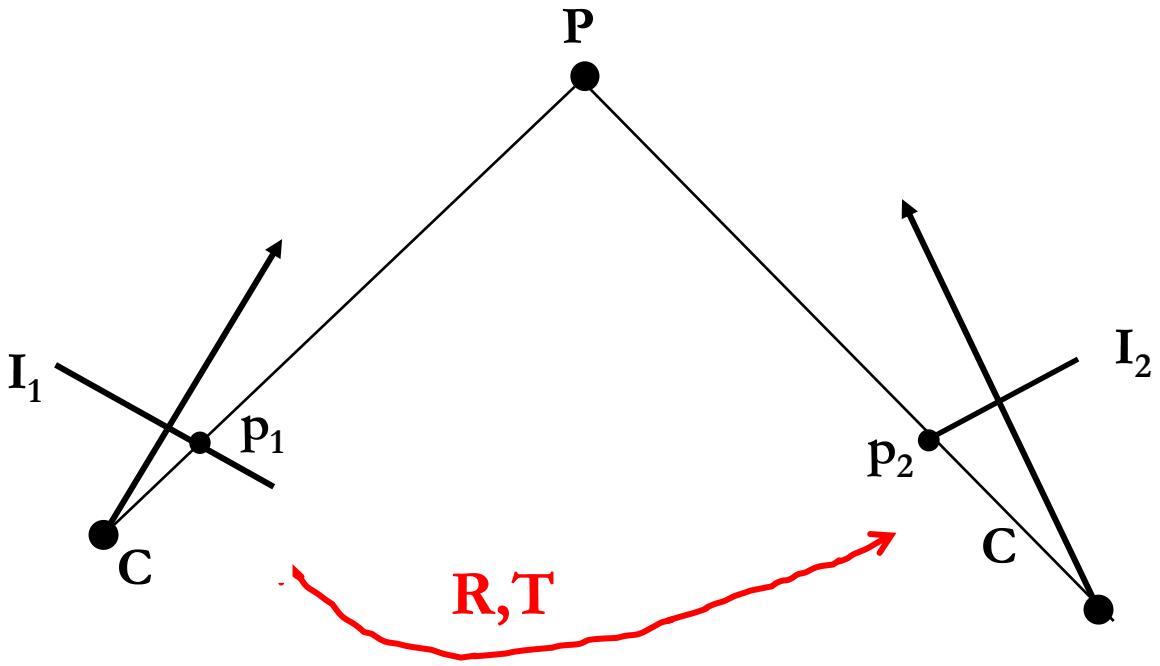
We may wish to find where a certain pixels does project into another image taken by the same camera, typically a moving camera used to scan a static scene. Purposely, alongside depth and camera intrinsics, we need to know the rigid motion (roto-translation) between the two views (i.e. CRFs).



$$p_2 = A\, T_{1\to 2}(zA^{-1}p_1) \quad \text{with:} \quad T_{1\to 2}(P_1) = RP_1 + T$$

Cameras may be different

Similarly, the other image may be taken by a different camera. We would then need to know the intrinsic parameters of the other camera too.

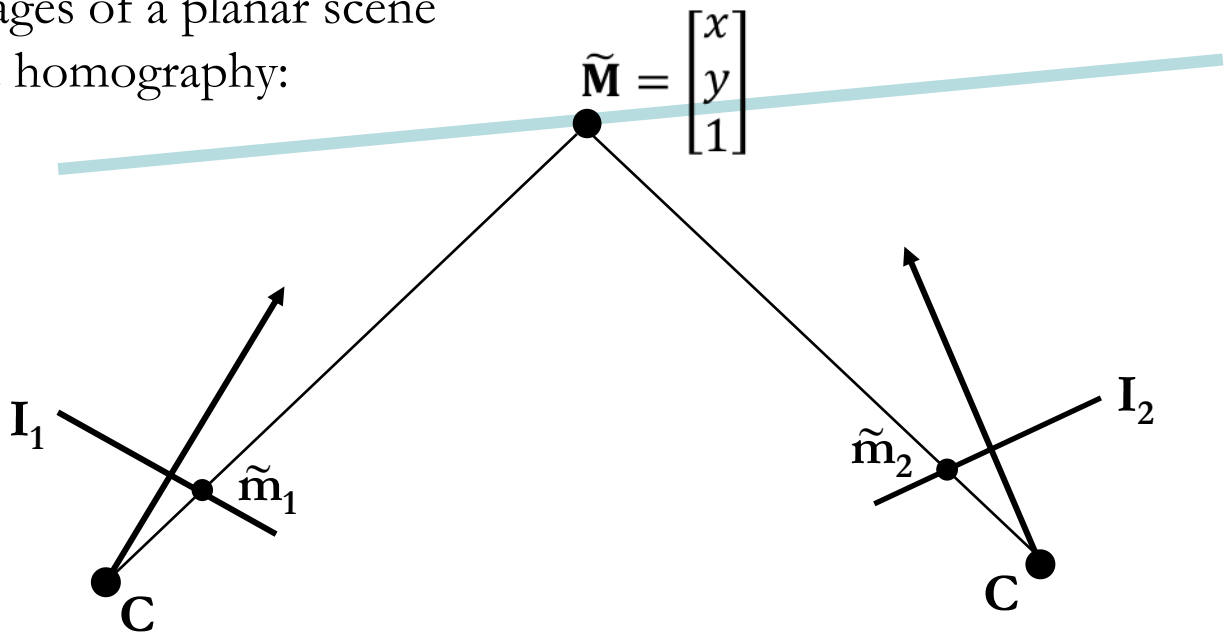


$p_2 = A_2\ T_{1\to2}(zA_1^{-1}p_1)$ with: $T_{1\to2}(P_1)=RP_1+T$

This is how an RGB-D image is obtained in many *depth cameras* (e.g. MS Kinect).

Homographies often occur (1)

1) Any two images of a planar scene are related by a homography:



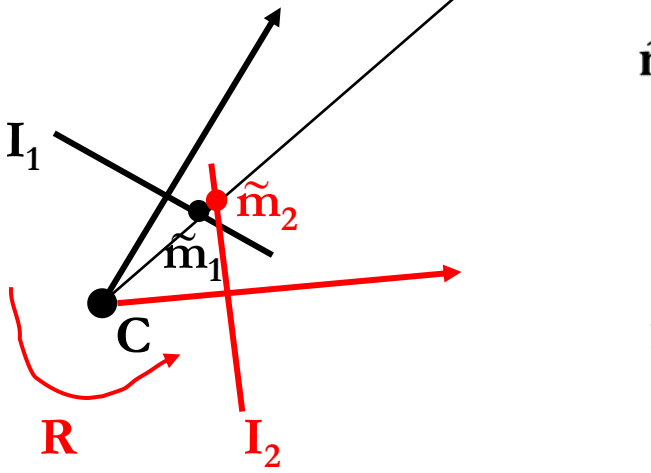
$$\widetilde{\mathbf{m}_1} = \mathbf{H}_1\tilde{\mathbf{M}} \qquad \widetilde{\mathbf{m}_2} = \mathbf{H}_2\tilde{\mathbf{M}}$$

$$\widetilde{\mathbf{m}_1} = \mathbf{H}_1\mathbf{H}_2^{-1}\widetilde{\mathbf{m}_2} \qquad \widetilde{\mathbf{m}_2} = \mathbf{H}_2\mathbf{H}_1^{-1}\widetilde{\mathbf{m}_1}$$

$\mathbf{H}_{21}$ ← Homography → $\mathbf{H}_{12}$

Homographies often occur (2)

1) Any two images taken by a camera rotating about the optical center are related by a homography:



$$\tilde{\mathbf{M}} = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{M} \\ 1 \end{bmatrix}$$

$$\tilde{m_1} = \mathbf{A}[\mathbf{I}\ \mathbf{0}]\tilde{\mathbf{M}} = \mathbf{AM} \qquad \tilde{m_2} = \mathbf{A}[\mathbf{R}\ \mathbf{0}]\tilde{\mathbf{M}} = \mathbf{ARM}$$

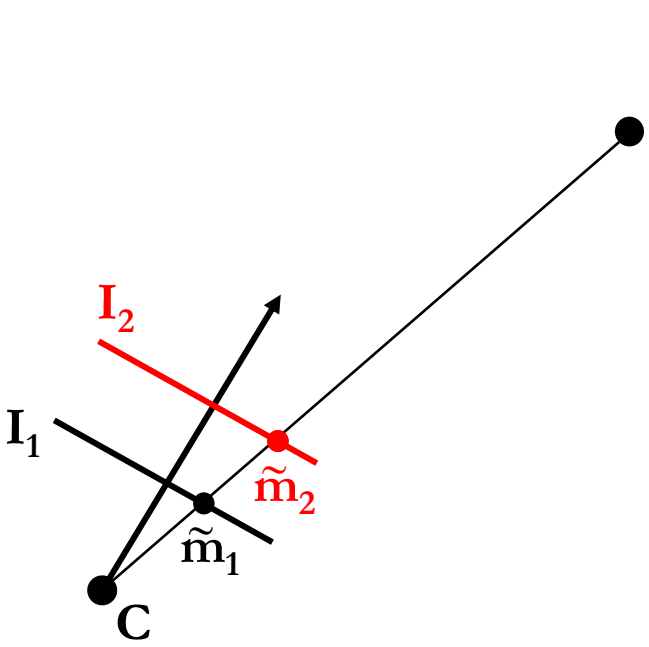
$$\tilde{m_1} = \mathbf{AR}^{-1}\mathbf{A}^{-1}\tilde{m_2} \qquad \tilde{m_2} = \mathbf{ARA}^{-1}\tilde{m_1}$$

$$\mathbf{H}_{21} \qquad \mathbf{H}_{12}$$

Homography

Homographies often occur (3)

1) Any two images taken by different cameras (i.e. different **A**) in a fixed pose (i.e. same CRF) are related by a homography:



$$\widetilde{\mathbf{m_1}} = \mathbf{A_1}[\mathbf{I}\ \mathbf{0}]\tilde{\mathbf{M}} = \mathbf{A_1}\mathbf{M} \qquad \widetilde{\mathbf{m_2}} = \mathbf{A_2}[\mathbf{I}\ \mathbf{0}]\tilde{\mathbf{M}} = \mathbf{A_2}\ \mathbf{M}$$

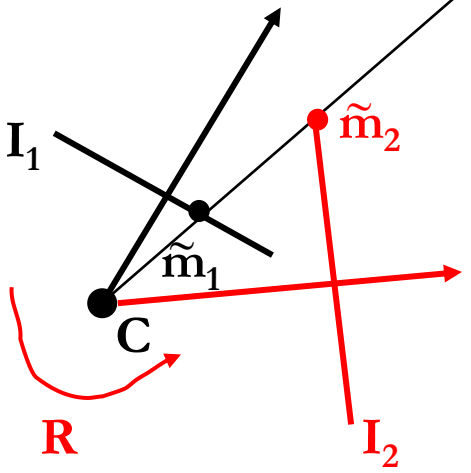
$$\widetilde{\mathbf{m_1}} = \mathbf{A_1}\mathbf{A_2^{-1}}\widetilde{\mathbf{m_2}} \qquad \widetilde{\mathbf{m_2}} = \mathbf{A_2}\mathbf{A_1^{-1}}\widetilde{\mathbf{m_1}}$$

$$\mathbf{H}_{21} \qquad \mathbf{H}_{12}$$

Homography

Homographies often occur (4)

1) Should we both rotate the camera about the optical center and change the intrinsic we still end up with images related by a homography:



$$\tilde{\mathbf{M}} = \begin{bmatrix} x \\ y \\ z \\ 1 \end{bmatrix} = \begin{bmatrix} \mathbf{M} \\ 1 \end{bmatrix}$$

$$\tilde{m_1} = \mathbf{A_1}[\mathbf{I}\ \mathbf{0}]\tilde{\mathbf{M}} = \mathbf{A_1}\mathbf{M} \qquad \tilde{m_2} = \mathbf{A_2}[\mathbf{R}\ \mathbf{0}]\tilde{\mathbf{M}} = \mathbf{A_2}\mathbf{R}\mathbf{M}$$

$$\tilde{m_1} = \mathbf{A_1}\mathbf{R}^{-1}\mathbf{A_2}^{-1}\tilde{m_2} \qquad \tilde{m_2} = \mathbf{A_2}\mathbf{R}\mathbf{A_1}^{-1}\tilde{m_1}$$

$$\mathbf{H}_{21} \qquad \mathbf{H}_{12}$$

Homography

Calibration of a stereo camera (1)

- Given a stereo system (aka *rig* or *head*), we are able to calibrate separately the two cameras. This would provide us with:
  - Intrinsic parameters and lens distortion coefficients
  - Extrinsic parameters, i.e. the rigid motion between the CRF and a given WRF (e.g. associated with the calibration target).
- However, calibrating a stereo system requires determining also the rigid motion, **R** and **T**, between the two cameras (i.e. between CRF_L and CRF_R)
- Knowing the extrinsic parameters wrt the same WRF would allow to compute the rigid motion between the two cameras by simple chaining of the transformations: $G_i(G_j)^{-1}$, with either (i=L, j=R) or (i=R, j=L). With Zhang's method this can be achieved, e.g., by taking the same static pattern position as the first image in both calibration sequences.
- However, this is not a robust solution, especially wrt noise and can easily yield to quite inaccurate results.

Calibration of a stereo camera (2)

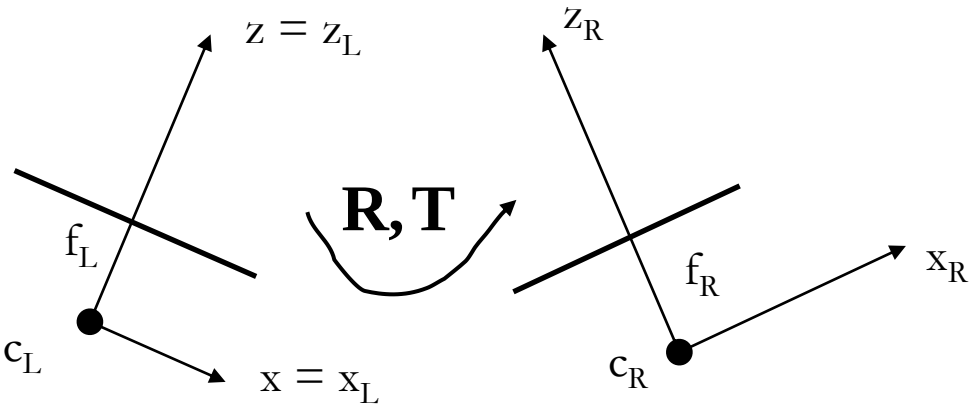
- As already pointed out, a more accurate solution is achieved by starting from an initial guess for **R** and **T** and then refining the estimation by non-linear minimization of the reprojection error.
- To obtain a robust initial guess, the same position of the planar calibration target is seen in each image of both sequences. This allows to estimate the rigid motion wrt to each such pattern positions as $G_i(G_j)^{-1}$.
- The initial guess is then taken as the median over the values $\mathbf{R}_i$ and $\mathbf{T}_i$ computed as $G_i(G_j)^{-1}$ from each pair of corresponding calibration images. Purposely, each $\mathbf{R}_i$ is converted into a 3 elements vector (rotation angles).
- Finally, the estimation is refined by non-linear minimization of the reprojection error in **both images of each pair** by Levenberg-Marquardt's:

$$\sum_{k=L,R}\sum_{i=1}^{n}\sum_{j=1}^{m}\left\|\mathbf{m}^{k}_{ij}-\hat{\mathbf{m}}\left(\mathbf{A},\mathbf{k},\mathbf{R},\mathbf{T},\mathbf{w}_{j}\right)\right\|^{2}$$

Stereo Reference Frame (SRF)

- Once the stereo rig has been calibrated, it is often useful to define a RF attached to the system, which can be thought of as the CRF of the Stereo Camera and will be denoted hereinafter as **SRF** (Stereo Reference Frame).
- The standard choice is just to pick-up the CRF of one of the two cameras, e.g. the left camera. Accordingly, the extrinsic parameters of the right camera in the SRF are given by the previously computed **R** and **T**, with the corresponding PPMs given by:

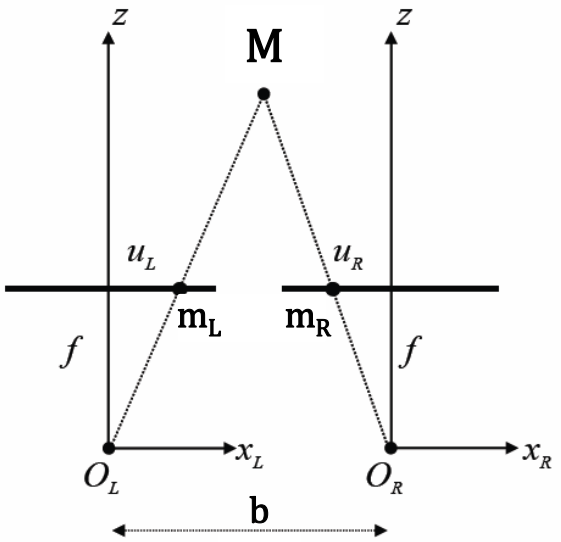
$$\tilde{\mathbf{P}}_L = \mathbf{A}_L \left[\mathbf{I} \middle| \mathbf{0} \right] \quad \tilde{\mathbf{P}}_R = \mathbf{A}_R \left[\mathbf{R} \middle| \mathbf{T} \right]$$



Rectification

- It is well known that a standard (i.e. lateral) stereo geometry is the most convenient choice to search for corresponding pixels:

$$(1)\colon M_L = M_R + \begin{bmatrix} b \\ 0 \\ 0 \end{bmatrix}, (2)\colon f_L = f_R = f \rightarrow A_L = A_R = A$$



- Unfortunately, it is impossible to achieve such geometry by mechanical alignment.
- However, once calibration has been performed, one can define two new cameras (i.e. two new PPMs) in standard geometry by virtually rotating the original ones about their optical centres (1) and constraining the intrinsic parameters of the new cameras to be the same (2).
- Then, the images acquired by the original cameras can be **rectified**, i.e. transformed into those that would have been acquired by the new ones by a warping operation.
- Rectification needs to happen after compensation of lens distortion (i.e. on undistorted images). An initial warp to remove lens distortion is thus always applied before warping again the images according to the rectification transformations.

The new PPMs

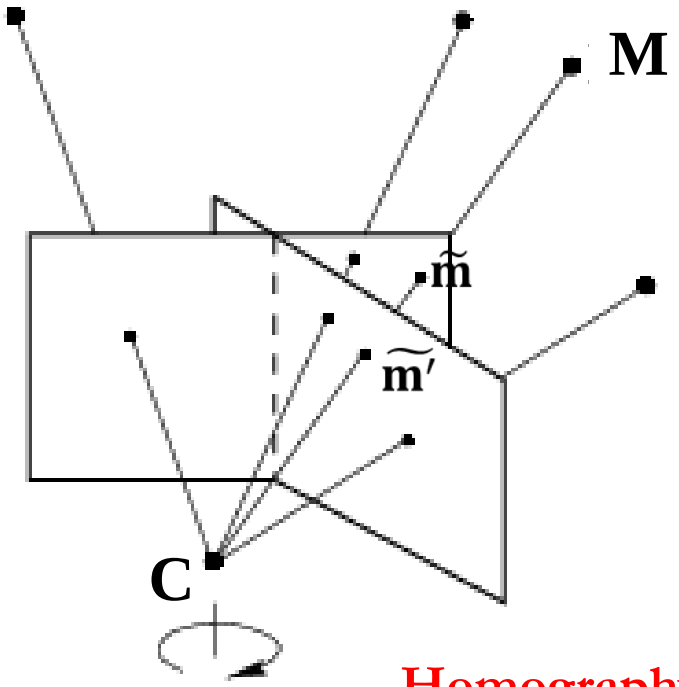
- Given $\mathbf{A}_L$, $\mathbf{A}_R$, $\mathbf{R}$, $\mathbf{T}$ the new PPMs can be achieved as follows:
  - A new intrinsic parameter matrix, $\mathbf{A}_{\mathbf{new}}$, is arbitrarily chosen (e.g. the mean between $\mathbf{A}_L$, $\mathbf{A}_R$).
  - The new rotation matrix for both cameras, $\mathbf{R}_{\mathbf{new}}$, is then determined. As the row vectors in $\mathbf{R}_{\mathbf{new}}$ represent the X, Y, e Z axis of the CRF into the WRF, taking as WRF the previously introduced SRF, a lateral stereo system is achieved as follows.
    - The new X axis is taken parallel to the baseline $\mathbf{B}=\mathbf{C}_R-\mathbf{C}_L$. As we wish to express $\mathbf{B}$ into the SRF: $\mathbf{B} = -\mathbf{R}^T\mathbf{T} - 0 = -\mathbf{R}^T\mathbf{T} = \left[B_x\ B_y\ B_z\right]^T$. Then, the first row of $\mathbf{R}_{\mathbf{new}}$ is given by the unit vector parallel to the baseline: $\mathbf{r}_1 = \frac{\mathbf{B}}{\|\mathbf{B}\|}$
    - The new Y axis is orthogonal to X and to an arbitrary unit vector $\mathbf{k}$, e.g. chosen so to be parallel to the old Z axis of the left camera $(\mathbf{k} = [0\ 0\ 1]^T)$: $\mathbf{r}_2 = \mathbf{k} \wedge \mathbf{r}_1 = \frac{\left[-B_y\ B_x\ 0\right]^T}{\sqrt{{B_x}^2+{B_y}^2}}$
    - The new Z axis is, of course, orthogonal to X and Y: $\mathbf{r}_3 = \mathbf{r}_1 \wedge \mathbf{r}_2$
- The new PPMs are thus: $\widetilde{\mathbf{P}}'_L = \mathbf{A}_{\mathbf{new}}[\mathbf{R}_{\mathbf{new}}|\ 0]$, $\widetilde{\mathbf{P}}'_R = \mathbf{A}_{\mathbf{new}}\left[\mathbf{R}_{\mathbf{new}}| - \mathbf{R}_{\mathbf{new}}\mathbf{C}_R\right]$

Rectification Homographies

- We must compute the transformations that map the original images, i.e. those acquired by the cameras with PPMs given by $\widetilde{\mathbf{P}_L}$, $\widetilde{\mathbf{P}_R}$, into a rectified image pair, i.e. the two the images that would have been acquired by the cameras with PPMs given by the previously defined $\widetilde{\mathbf{P}'_L}, \widetilde{\mathbf{P}'_R}$.
- Given a 3D point in the WRF (i.e. the SRF), for each of the two cameras we can write the equation of the associated optical ray based on both $\widetilde{\mathbf{P}}$ and $\widetilde{\mathbf{P}'}$ (the subscripts L,R are omitted for the sake of simplicity):

$$\widetilde{\mathbf{P}} = [\boldsymbol{P}\ \boldsymbol{p_4}] \longrightarrow \mathbf{M} = \mathbf{C} + \lambda \mathbf{P}^{-1}\widetilde{\mathbf{m}}$$

$$\widetilde{\mathbf{P}'} = [\mathbf{P}'\boldsymbol{p}'_4] \longrightarrow \mathbf{M} = \mathbf{C} + \lambda \mathbf{P}'^{-1}\widetilde{\mathbf{m}'}$$



Homography

$$\mathbf{P}^{-1}\widetilde{\mathbf{m}} = \lambda \mathbf{P}'^{-1}\widetilde{\mathbf{m}'}$$

$$\widetilde{\mathbf{m}} = \lambda \mathbf{P}\mathbf{P}'^{-1}\widetilde{\mathbf{m}'}$$

$$\boldsymbol{H}_L = \mathbf{P}_L\mathbf{P}_L'^{-1}, \quad \boldsymbol{H}_R = \mathbf{P}_R\mathbf{P}_R'^{-1}$$

An example

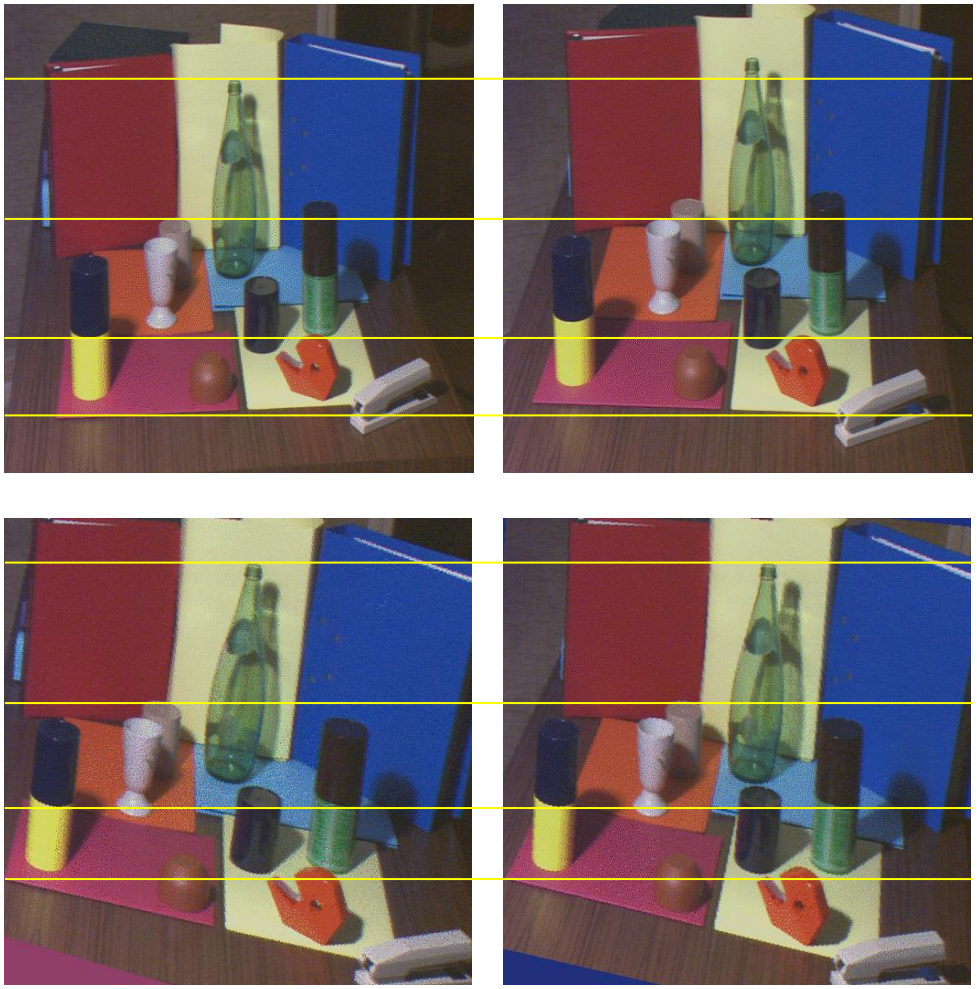
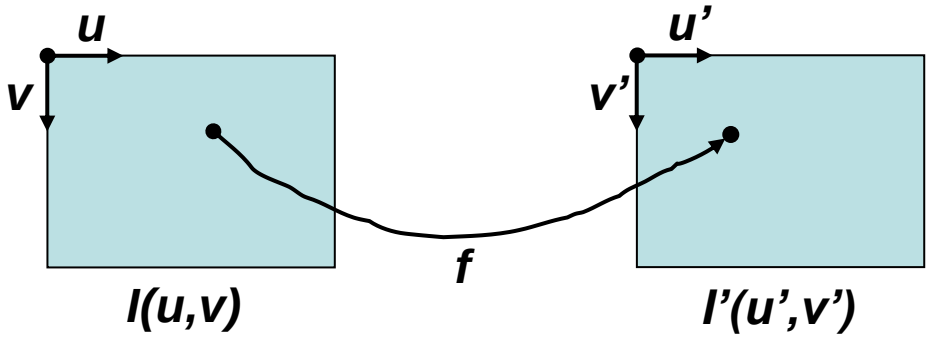


Image Warping



$$\begin{cases} u' = f_u(u,v) \\ v' = f_v(u,v) \end{cases}$$

$$I'\left(f_u(u,v), f_v(u,v)\right) = I(u,v)$$

Examples of image warping:

Rotation → $\begin{bmatrix} u' \\ v' \end{bmatrix} = \begin{pmatrix} \cos\theta & -\sin\theta \\ \sin\theta & \cos\theta \end{pmatrix} \begin{bmatrix} u \\ v \end{bmatrix}$

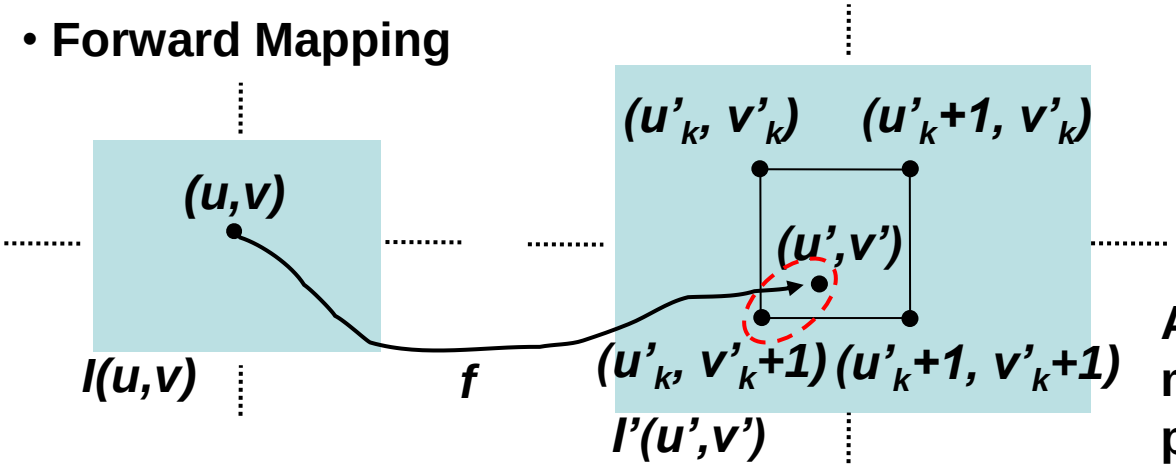
Removal of perspective deformation → $s\begin{bmatrix} x' \\ y' \\ w' \end{bmatrix} = \begin{bmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{32} \end{bmatrix} \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$

Other examples:

- *Remove lens distortion*
- *Stereo rectification*
- *…..*

Forward/Backword Mapping

- **Forward Mapping**



$$(u'_k, v'_k) = (\lfloor u' \rfloor, \lfloor v' \rfloor)$$

A better choice consists in mapping to the closest point into the destination image

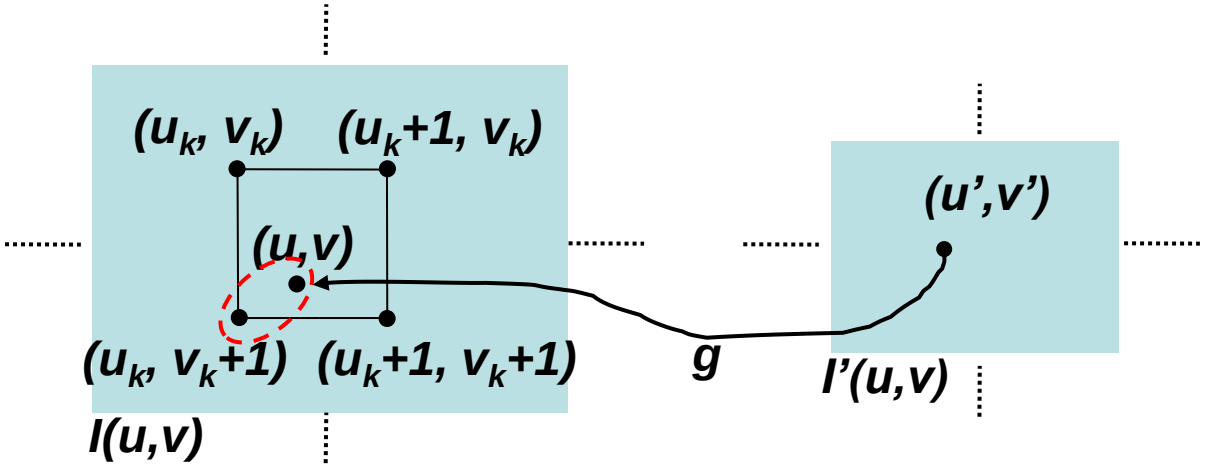
***Holes* and *folds* in I'(u',v') !**

- **Backword Mapping**

$$\begin{cases} u = g_u(u', v') \\ v = g_v(u', v') \end{cases} \Rightarrow \forall (u', v'): \; I'(u', v') = I\left(g_u(u', v'), g_v(u', v')\right)$$

No *holes* and *folds* in I'(u',v'), which mapping strategy ?

Mapping Strategies

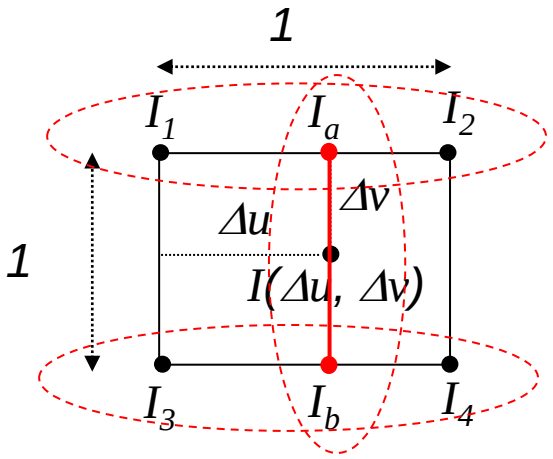


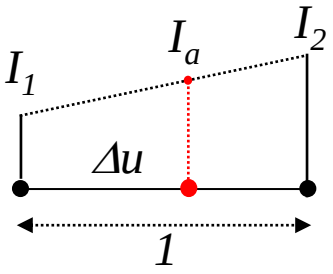
$$(u_k, v_k) = \left(\lfloor u \rfloor, \lfloor v \rfloor\right)$$

- **Mapping from the closest point (Nearest Neighbour Mapping)**
- **Interpolation between the 4 closest points (*bilinear*, bicubic,…)**

$$\Delta u = u - u_k \qquad I_1 = I(u_k, v_k) \qquad I_2 = I(u_k + 1, v_k)$$
$$\Delta v = v - v_k \qquad I_3 = I(u_k, v_k + 1) \qquad I_4 = I(u_k + 1, v_k + 1)$$

Bilinear Interpolation (1)





$$\frac{I_a - I_1}{\Delta u} = I_2 - I_1$$

$$I_a = (I_2 - I_1)\Delta u + I_1$$

$$I_b = (I_4 - I_3)\Delta u + I_3$$

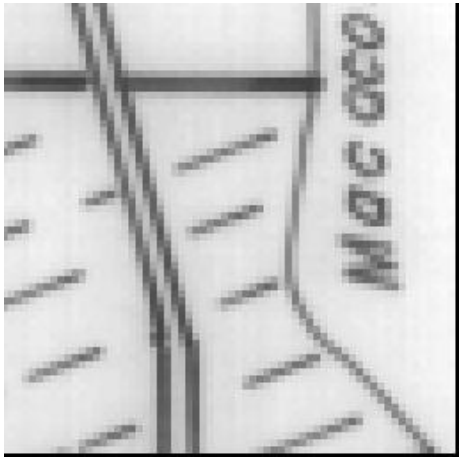
$$I(\Delta u, \Delta v) = (I_b - I_a)\Delta v + I_a$$

$$I(\Delta u, \Delta v) = ((I_4 - I_3)\Delta u + I_3 - ((I_2 - I_1)\Delta u + I_1))\Delta v + (I_2 - I_1)\Delta u + I_1$$

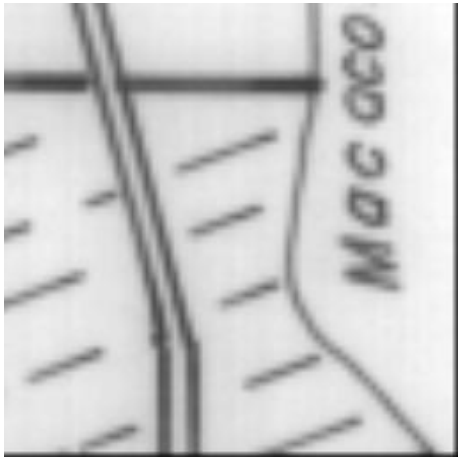
$$I'(u', v') = (1 - \Delta u)(1 - \Delta v)I_1 + \Delta u(1 - \Delta v)I_2 + (1 - \Delta u)\Delta v I_3 + \Delta u \Delta v I_4$$

Bilinear Interpolation (2)

Input

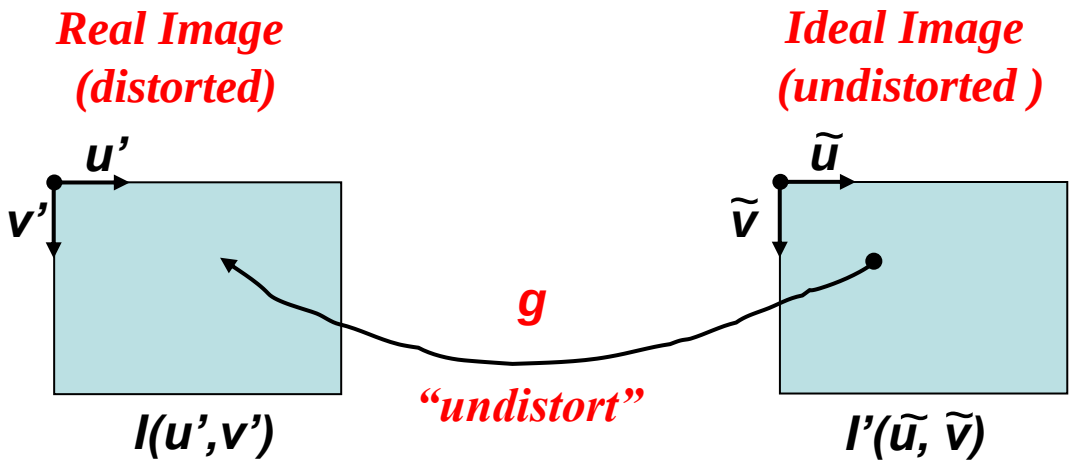


Warp (Zoom) by NNM

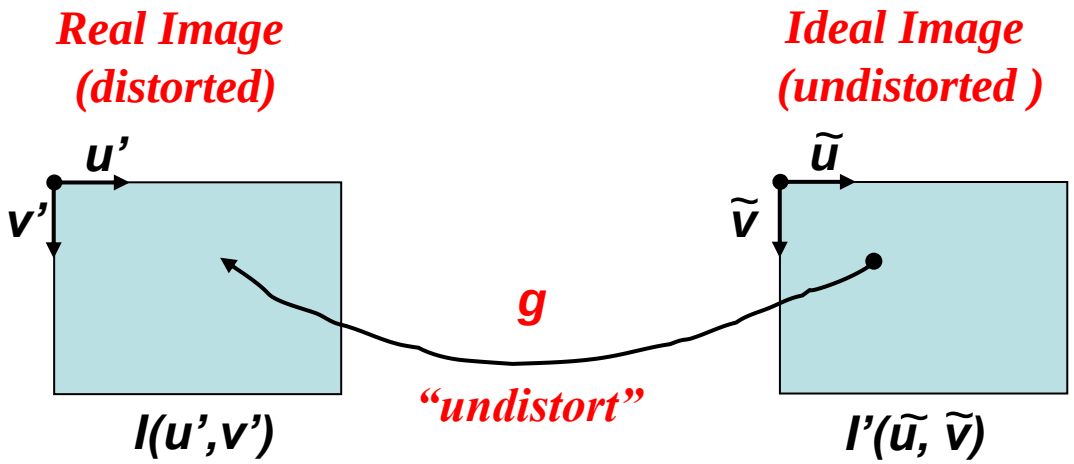


Warp (Zoom) by Bilinear Interpolation

Warping to compensate lens distortion



Warping to compensate lens distortion



Once the lens distortion parameters have been computed by camera calibration, the image can be corrected by a backward warp from the undistorted to the distorted image based on the adopted lens distortion model:

$$\forall (\tilde{u},\tilde{v}):\quad I'(\tilde{u},\tilde{v}) = I\left(g_u(\tilde{u},\tilde{v}), g_v(\tilde{u},\tilde{v})\right)$$

For example, using Zhang's calibration method

$$\begin{cases} u' = \tilde{u} + \left(k_1 r^2 + k_2 r^4\right)\left(\tilde{u} - u_0\right) \\ v' = \tilde{v} + \left(k_1 r^2 + k_2 r^4\right)\left(\tilde{v} - v_0\right) \end{cases}$$

Main References

1) R. Hartley and A. Zisserman, "Multiple View Geometry in Computer Vision", 2° Edition, Cambridge University Press, 2003.

2) Z. Zhang, "A flexible new technique for camera calibration", IEEE Trans. On Pattern Analysis and Machine Intelligence, 22(11):1330-1334, November 2000.

3) A. Fusiello, "Visione Computazionale – Tecniche di ricostruzione tridimensionale", Collana Informatica Franco Angeli, 2013.

4) E. Trucco, A. Verri, "Introductory Techniques for 3-D Computer Vision", Prentice-Hall, 1998.

5) Gary Bradski, Adrian Kaehler, "Learning OpenCV - Computer Vision with the OpenCV Library", O'Reilly Media, 2008.