University of Bologna

Image Formation and Acquisition (Part 3)

Luigi Di Stefano (luigi.distefano@unibo.it)

CAMERA CALIBRATION: a very important matter!

- Given a camera, camera calibration is about estimating all its parameters (intrinsic, extrinsic and distorsion ones), doing that by exploting available observable data!

  CALIBRATION of the MODEL: given inputs, outputs get parameters

  USAGE of the MODEL for PREDICTION: Known parameters, given inputs get outputs (or viceversa)

- A lot of camera calibration algorithms do exist: we'll study the one implemented by openCV!

- Common behaviour: a dataset of couples WRFcoords and PIXELcoords are supposed to be Known and a set of equations that includes them must be solved in order to find the said parameters. Indeed, these "couples" (that needs to be Known) are correspondencies between 3D continous WRFcoords points and 2D discrete PIXELcoords points!
  How many corrispondencies we need?
  $\tilde{P}$ has 12 entries (incognitas) and $\tilde{m}=\tilde{P}\tilde{M}$ has 4 scalar equations: needed at least four points (not considering distorsion in $\tilde{m}=\tilde{P}\tilde{M}$ equation); pratically a LOT of points are used to improve robustness.
  ALSO, for calibration they are not used arbitrary scenes BUT we use calibration targets, alias objects for which is quite straightforward to get corrispondences!

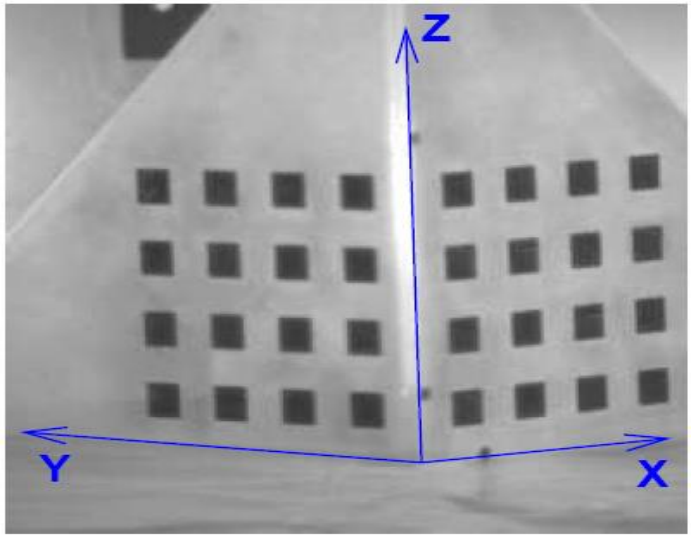
  We may have two dual situations: rely one one image of a 3D calibration target (not so common) or rely on a lot of different images of the same planar calibration target (technically at least 3 of them, pratically in the range $15 \div 20$)(more common).

Camera Calibration (1)

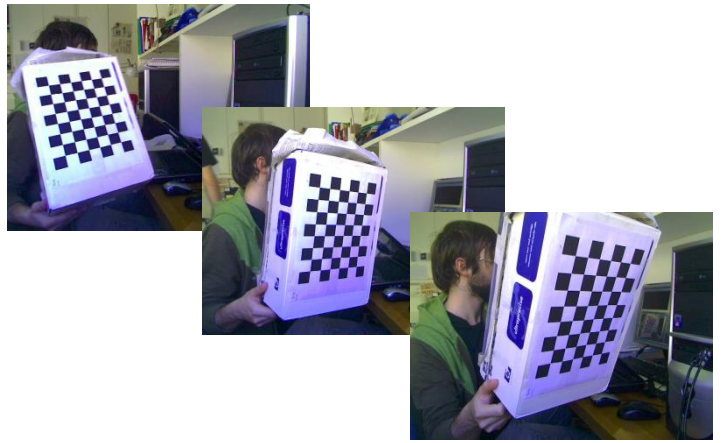
- We have now at disposal a camera model thoroughly explaining the image formation and digitization process.
- Such a model includes the PPM, which, in turn, can be decomposed into 3 independent items: intrinsic parameter matrix ($\mathbf{A}$), rotation matrix ($\mathbf{R}$), translation vector ($\mathbf{T}$), as well as the lens distortion parameters.
- **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 ($\mathbf{A}$, $\mathbf{R}$, $\mathbf{T}$) 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) providing easily detectable image features (such as, e.g., chessboard or dot patterns) are typically deployed.

Camera 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 patterns**.**
  - 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,...)

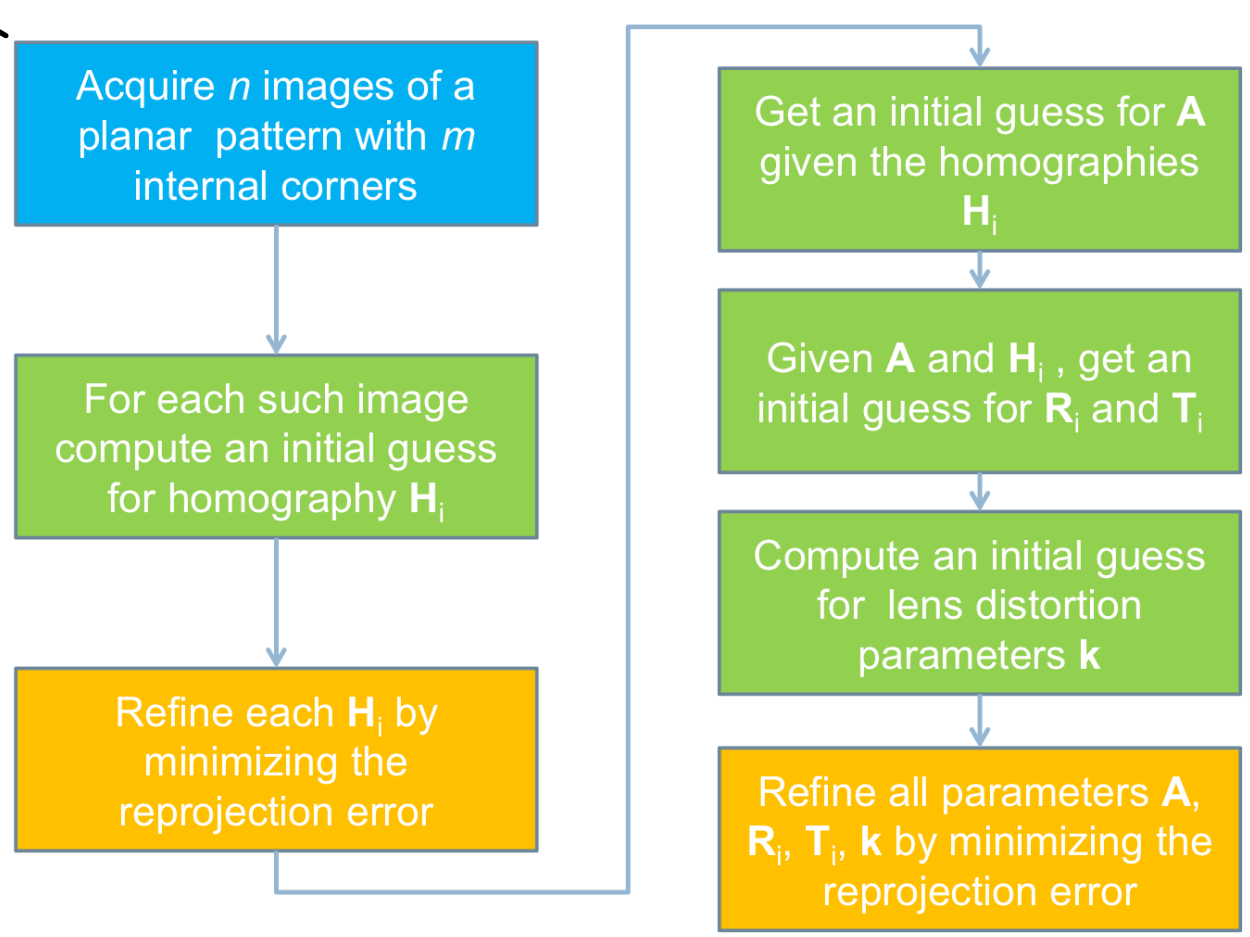


Zhang's Method (1)

The one implemented by OpenCV!
(almost: OpenCV implements some slight variation)

Typically, we acquire $n \geq 15$ images, even if the minimum amount is 3. Also, when dealing with the Zhang's Method the most used calibration target is a planar (2D) chessboard! In that case, m is the number of ALL internal corners of the chessboard (where "internal" means NOT considering the outer border of squares of the chessboard)!

Important: having a planar calibration target IMPLIES that IF choosing a specific and convenient WRF then the PMM collapses to a well-known homography!



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.

INDEED, CV PROCESSES

- Internal corners can be detected easily by standard algorithms (e.g. the Harris corner detector, possibly with sub-pixel refinement for improved accuracy).

CONVENIENT WRF CHOICE

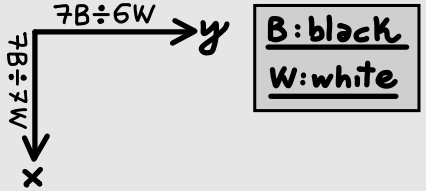
- In each image, the 3D WRF is taken at the top-left corner of the pattern, with plane z=0 given by the pattern itself and the x,y axis aligned to the two orthogonal directions, in particular so as to keep always the same association between axis and directions (e.g. x=rows, y = columns). Thus, as for 3D points:
  - The third coordinate is always 0.
  - x and y are determined by the known size of the squares forming the chessboard.

AGAIN: the calibration target is well-known!

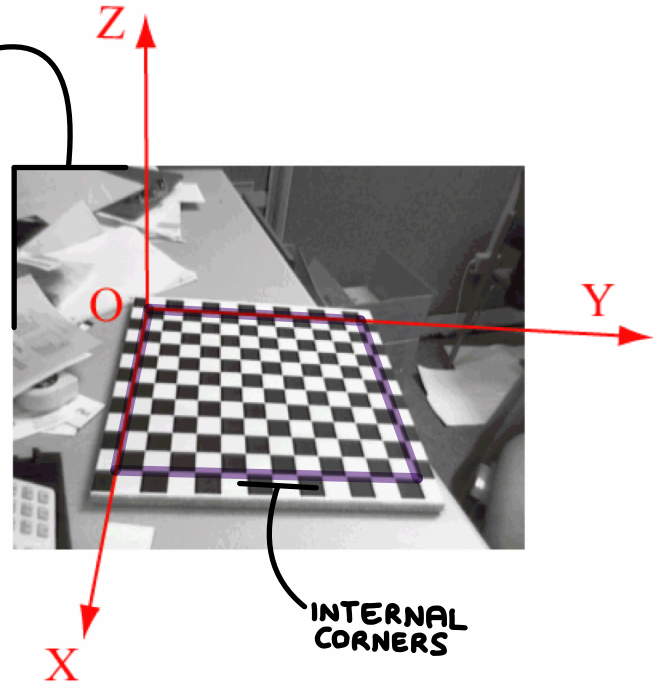
Estrinsic Parameters

- 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 calibration images, e.g. the first.

AGAIN, in order to choose WRF in a non ambigous way, the calibration pattern (of the chessboard) is asymmetric:



WRF coords of internal corners: easy to be computed once we have the said WRF and we known the chessboard dimensions.
pixel coords of internal corners in the image: typically provided by a proper CV process feeded with the image.



Be aware: for each image, that present an orientation of the chessboard which is different w.r.t. other images, we have a different WRF, so different extrinsic parameters and also different pixel coords of internal corners in the image! On the other hand, intrinsic parameters are always the same (cause the camera is always the same), except for noise presence!

Planar Calibration Target → Homographies

- Due to the calibration pattern being planar and the choice of the WRF associated with the calibration images, for each of them we can consider only 3D control points (i.e. corners) with z=0. Therefore, the mapping between the 3D coordinates of the control points and their projections in the calibration images is a **homography:**

$\begin{bmatrix} u \\ v \\ 1 \end{bmatrix}$ Point in the image, pixel homogeneous coords

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

9 unKnowns

Point in the WRF, homogeneous coordinates, called in this context: calibration points!

With (for example) only one calibration point: these are 3 scalar eqs. in 9 unKnown (only 8 independent)

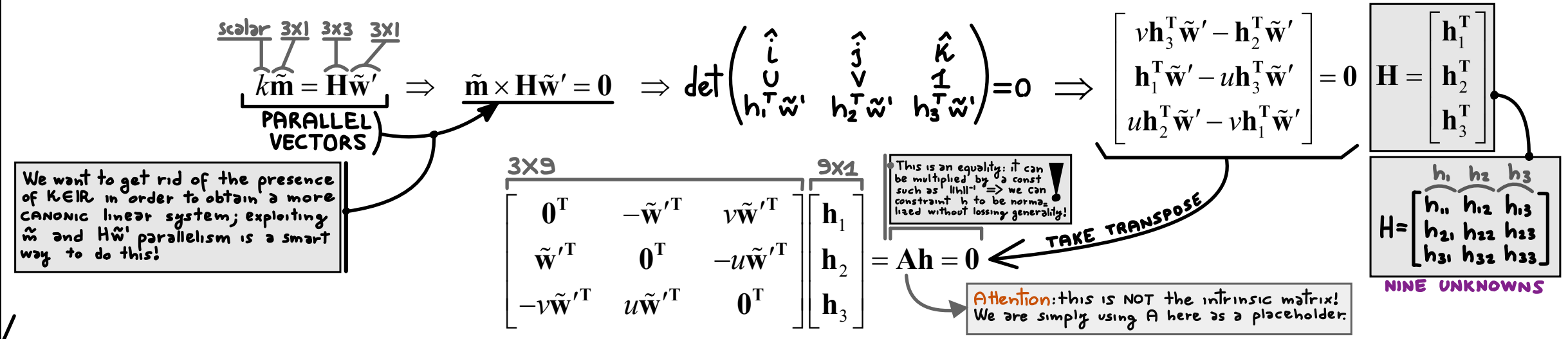
- Given a pattern with *m* corner, we can write *m* systems of 3 linear equations as above, wherein both 3D (with z=0) as well as 2D coordinates are known due to corners having been detected in the calibration image and the unknowns being, therefore, the elements in **H**.

Given all calibration points: overconstrained nonhomogeneous linear system of 2m eqs. in 9 unKnowns (8 independent)

Estimating H (DLT Algorithm)

DLT: Direct Linear Transformation

Starting from the previous homography equation we can write:



Between the previous 3 equations in 9 unknowns, only 2 are linearly independent, so that typically only the first 2 are kept. By deploying all correspondences, for each image we build a linear system of $2m$ equations in 9 unknowns. This allows for estimating **H** by minimizing the "algebraic error" represented by the norm of vector $\mathbf{Ah}$: $\mathbf{h}^* = \text{argmin}_{\mathbf{h} \in R^9} \|\mathbf{Ah}\|$, subject to $\|\mathbf{h}\| = 1$ to avoid the trivial solution ($\mathbf{h}=\mathbf{0}$).

It is known from *linear algebra* that the solution of the above over-constrained system of equations can be obtained in closed form by the Singular Value decomposition (SVD) of matrix **A.**

Ah=0 and how to solve it

If we consider the A matrix of the previous slide ONLY in its two first rows AND also we collect within the same matrix all the contributes for ALL m calibration points, THEN we end up with a overconstrained homogeneous linear system of 2m equations in 9 unKnowns (of which only 8 are independent):

$$Ah = 0$$

$2m \times 9 \quad 9 \times 1 \quad 2m \times 1$

This is solved by searching for the best h, alias the h that makes Ah the most possible near to zero!

$$h^{solution} = argument(min(Ah))$$

Finding $h^{solution}$ in such a way is also called, as a procedure, reducing the algebraic error (which is $\|Ah\|$).

To ACTUALLY solve this overconstrained homogeneous linear system and find $h^* = h^{solution}$ one possibility which is often used is rely on the Singular Value Decomposition (SVD) method, in which accordingly to some proper rules A is decomposed\riexpressed as $A = U\Sigma V^T$ (more details about these matrixes n the next slide) AND h^* can be taKen as the so called smallest right singular vector which is v_9 (where $V = [v_1 \dots v_9]$).

But why 2m>>9?

Considering that the unKnowns can be taKen to be 8, IF we choose to picK exactly m=4 points (within the single image) we get n°equations equal to n°unKnowns (8); in this case an EXACT solution in closed form would be available!
The reason for picKing m>>4 is related to one major advantage that overconstrained systems have, which is producing a solution which is way more robust w.r.t. noises, AKA less prone to them!

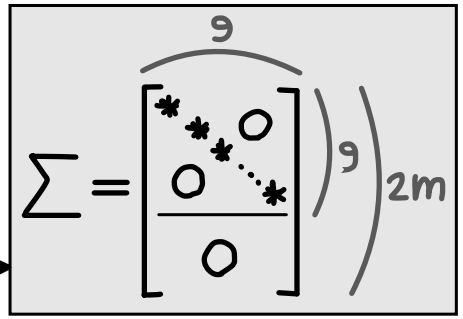
Estimating H by the SVD

- Remarkable properties:
- A orthogonal <=> I identity
- Given $X=A^{+}$ and $Y=\Sigma^{+}$, it holds: $X=UYV^{T}$, where "+" is the pseudoinverse operator!

Orthogonal *Diagonal* *Orthogonal*

$$\mathbf{A} = \mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^{T}$$

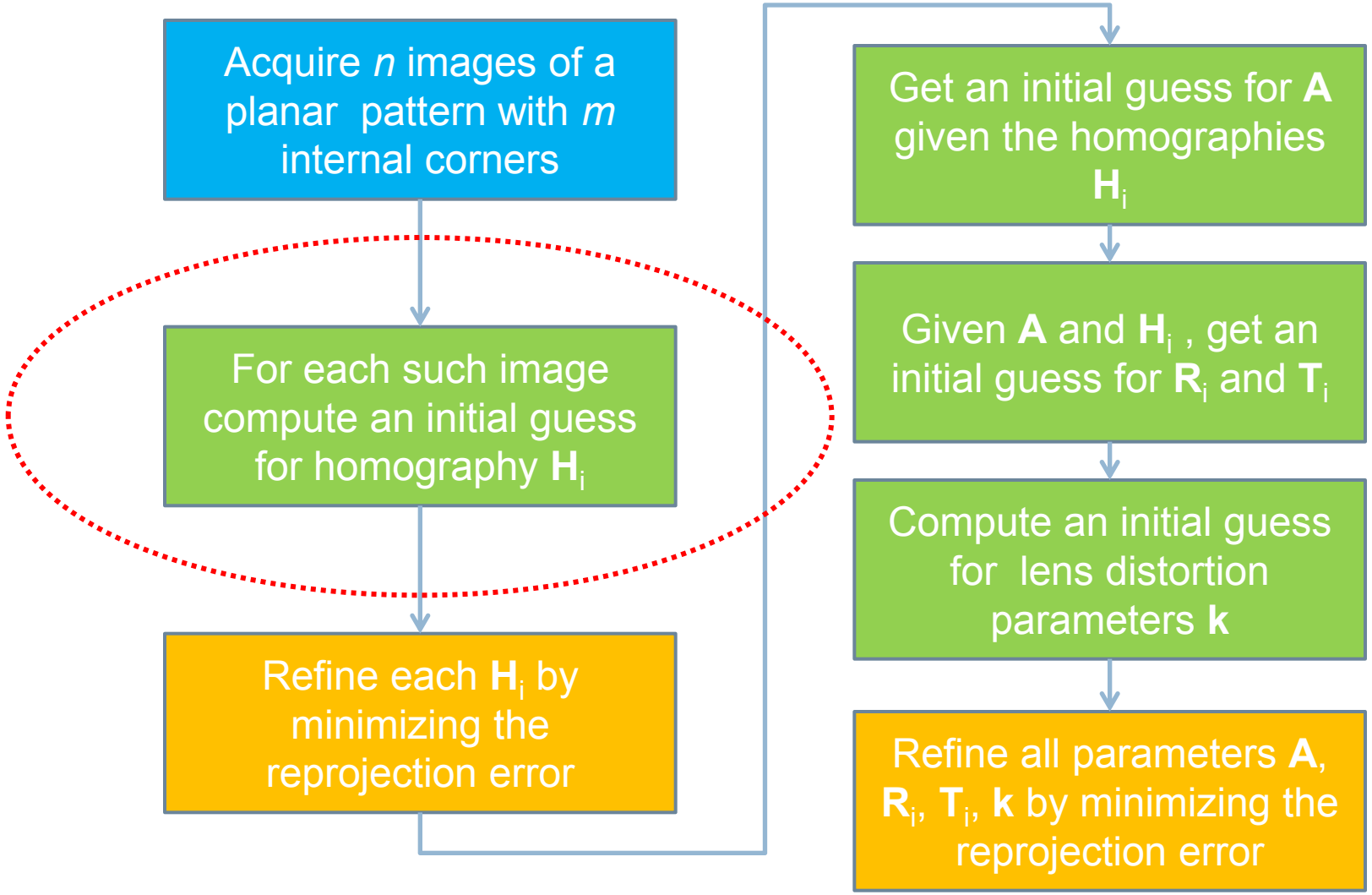
$2m \times 9$ $2m \times 2m$ $2m \times 9$ 9×9



$$\mathbf{V} = [\mathbf{v_1} \ldots . \mathbf{v_9}] \rightarrow \boldsymbol{h}^{*} = \mathbf{v_9}$$

See Appendix 1 for the solution of the homography estimation problems in case only 4 correspondences are available (i.e. the minimum number of required correspondences).

Zhang's Method



Non-linear Refinement

Key-word use to indicate that the scalar cost function to be minimized is indeed obtained as sum over all errors ε_i to be lowered (ideally to zero) in their squares, alias: $\text{cost} = \sum_i (\varepsilon_i)^2$

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

This is a NL optimization problem!

$$\mathbf{H} = \text{argmin}_{\mathbf{H} \in R^9} \sum_{j=1}^{m} \left\| \widetilde{m}_j - \mathbf{H}\widetilde{w}'_j \right\|^2$$

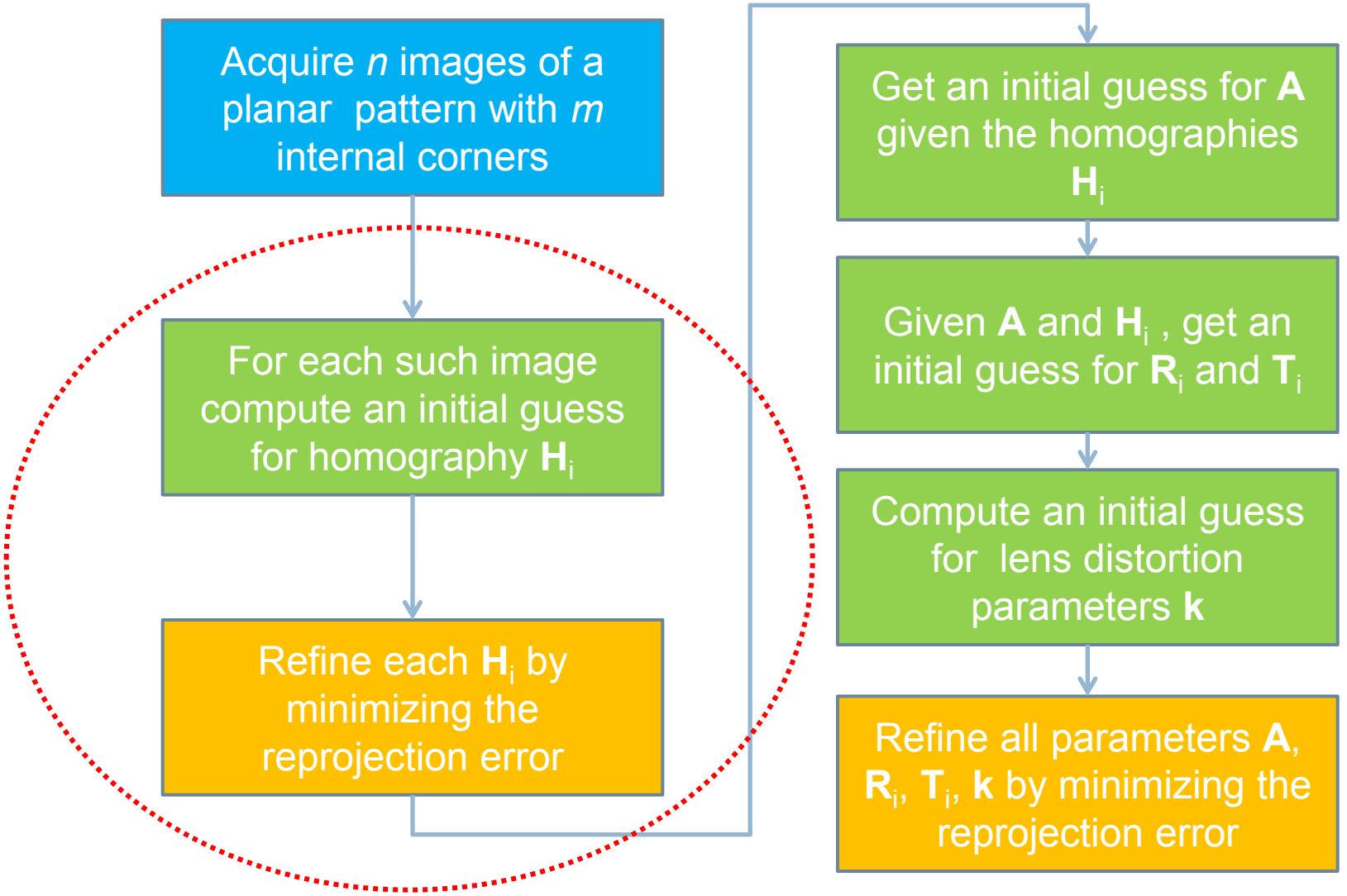
→ To be solved FOR EACH image!

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

→ IMPORTANT: we initialize this problem NOT to a random value BUT to the one given by the previous step!

- 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 referred to as “geometric error”.

Zhang's Method



Estimation of the intrinsic parameters (1)

- As **H** is known up to a scale factor, we can establish the following relation between **H** and the PPM:

UP TO A CONSTANT!

$$P=[p_1\ p_2\ p_3\ p_4]=\lambda A[R|T] \Rightarrow \mathbf{H}=[\mathbf{h}_1\quad \mathbf{h}_2\quad \mathbf{h}_3]=[\mathbf{p}_1\quad \mathbf{p}_2\quad \mathbf{p}_4]=\lambda \overbrace{\mathbf{A}}^{3\times 3}[\mathbf{r}_1\quad \mathbf{r}_2\quad \mathbf{T}]$$

THIS IS THE INTRINSIC PARAMETER MATRIX

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

$$\begin{aligned} h_1&=\lambda A r_1 \\ h_2&=\lambda A r_2 \end{aligned} \Rightarrow \begin{aligned} r_1&=\lambda^{-1}A^{-1}h_1 \\ r_2&=\lambda^{-1}A^{-1}h_2 \end{aligned}$$

$$\mathbf{r}_1^T\mathbf{r}_2=0 \Rightarrow \mathbf{h}_1^T\mathbf{A}^{-T}\mathbf{A}^{-1}\mathbf{h}_2=0$$

$$1=\mathbf{r}_1^T\mathbf{r}_1=\mathbf{r}_2^T\mathbf{r}_2 \Rightarrow \mathbf{h}_1^T\mathbf{A}^{-T}\mathbf{A}^{-1}\mathbf{h}_1=\mathbf{h}_2^T\mathbf{A}^{-T}\mathbf{A}^{-1}\mathbf{h}_2$$

UNKNOWNS (3X3 matrix)

IMPORTANT REMARK: the A matrix is the same accross ALL images, indeed the intrinsic parameters are one unique set for all images!

where the unknowns are the entries of $\mathbf{B}=\mathbf{A}^{-T}\mathbf{A}^{-1}$. As $\mathbf{A}$ is upper triangular, $\mathbf{B}$ turns out to be symmetric, so that the unknowns are just 6. (indeed, it can be proven: M upper triangular $\Rightarrow M^{-T}M^{-1}$ symmetric)

- By stacking 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)

Again, for the single image (AKA the single homography) we obtain the shown 2 scalar equations in 6 unknowns. Given n images, with some proper manipulations (that are SHOWN in the next slide) finally get an overconstrained homogeneous linear system of 2n equations in 6 unknowns! A system like that can be approached with the previously seen techniques! Again, the minimum n is 3 (in order to have a NOT overconstrained system) BUT having n>>3 has the (as already said before) gives us way more robust (w.r.t. noise) solutions!

Estimation of the extrinsic parameters (2)

- By posing:

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

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

A simple way to re-arrange the obtained equations and obtain an homogeneous sys!

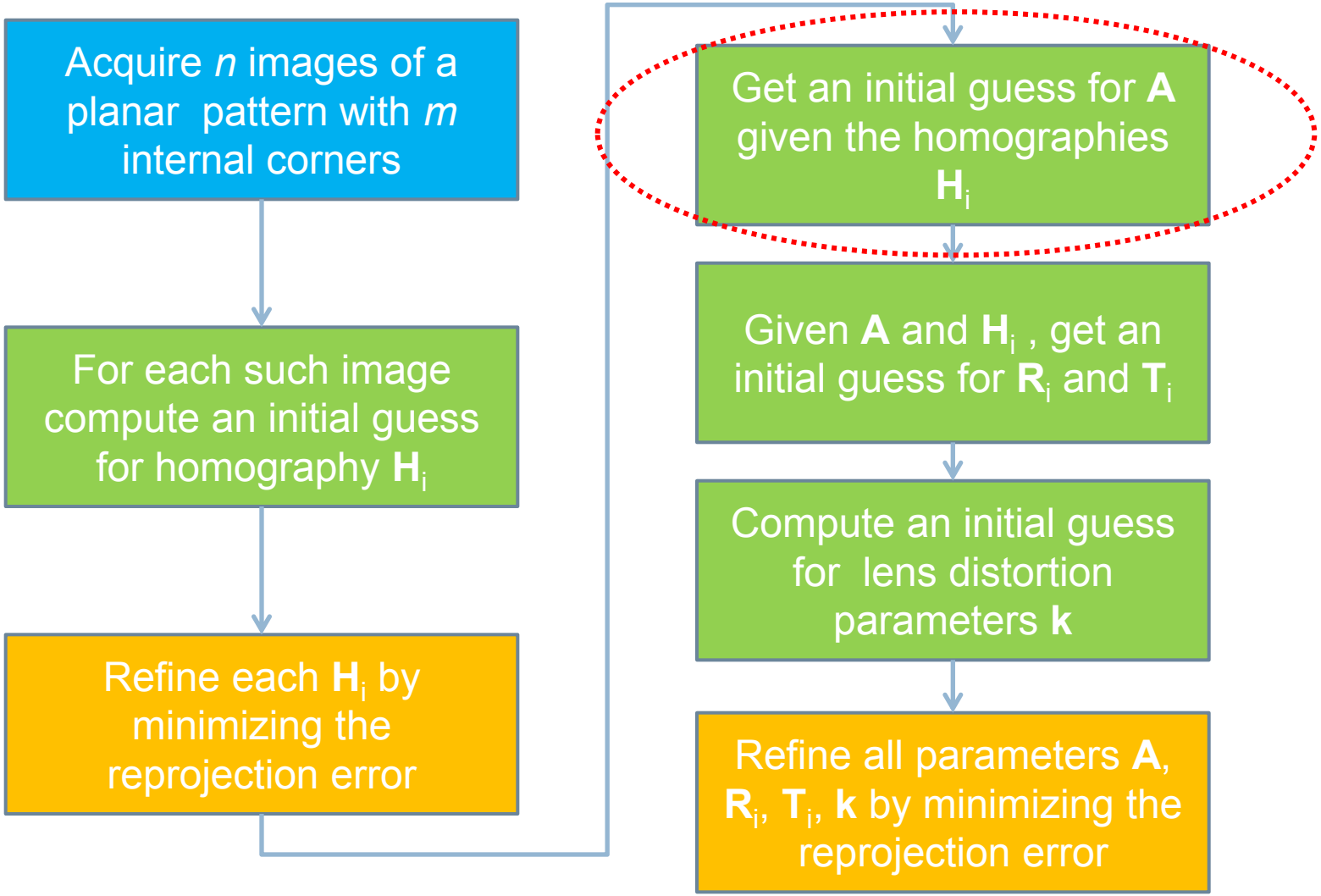
it can be noticed that

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

we get: $\mathbf{Vb} = 0$

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

Zhang's Method



Estimation of the extrinsic parameters

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

recall that we need to pick a value for λ!

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

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

Indeed, for each image we have an H that, exploiting the knoledge of the common A, make us able to compute [R,T]!

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

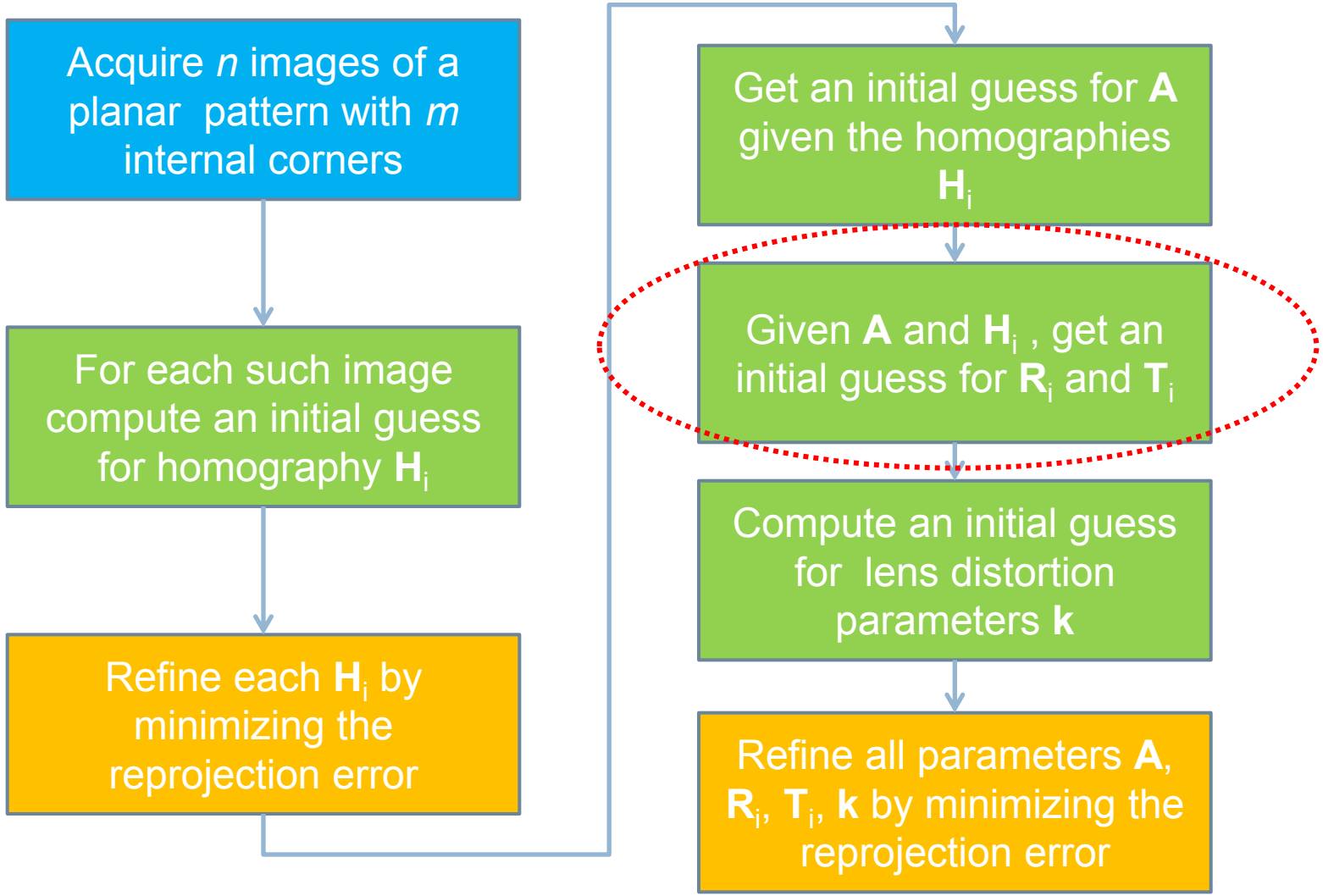
Some important remarks about the obtained R matrix(es)

For each image (alias related homography), the single $R=[r_1\ r_2\ r_3]$ might NOT be orthogonal: surely $\|r_1\|=1$ and $r_1 \perp r_2 \perp r_3$ by construction BUT it might be the case (and typically is) that $\|r_2\| \neq 1$ (regarding r_3, of course if it's defined as $r_3 = r_1 \wedge r_2$ then $\|r_2\|=1 \Rightarrow \|r_3\| \neq 1$)(ALSO, it may be the case that the two obtained r_1 and r_2 are NOT PERFECLTY orthogonal).

To workaround this problem, to get an orthogonal matrix R^{new} starting from a slightly NON orthogonal matrix R^{old} is apply the SVD method obtaining $R^{old}=U\Sigma V^T$ and then compute R^{new} as $R^{new}=UIV^T=UV^T$!

BE AWARE: a final R^{new} matrix obtained such that might have a -1 determinant, representing NOT properly a rotation BUT a reflection! Indeed, in general we simply define: $R^{final}=R^{new}/det(R^{new})$ (typically when the determinant ends up beeing -1 we then simply multiply by -1 the last column of V alias r_3)!

Zhang's Method



Lens distortion parameters (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 parameters k_1, k_2 of the radial distortion function.

- Given the already known intrinsic parameter matrix, $\mathbf{A}$, and the lens distortion model reported below:

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

CONTINOUS DISTORTED — CONTINOUS UN-DISTORTED

REMARK: these are continous coordinates PRE digitalization

BE AWARE: the Zhang's method only include radial distorsion with 2 params, meanwhile OpenCV uses 3 params for radial distorsion and also includes tangential distorsion, for a total set of params: K_1, K_2, K_3, p_1, p_2

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

AS ALREADY SEEN

$$\begin{cases} u = \alpha_u \dfrac{x}{z} + u_0 \\ v = \alpha_v \dfrac{y}{z} + v_0 \end{cases} \rightarrow \begin{cases} \tilde{u} = \alpha_u \tilde{x} + u_0 \\ \tilde{v} = \alpha_v \tilde{y} + v_0 \end{cases}$$

Attention: z is now embedded in x and y: $x^{new} = \frac{x^{old}}{z}$; $y^{new} = \frac{y^{old}}{z}$

UNDISTORTED	DISTORTED
$\begin{cases} \tilde{x} = \dfrac{\tilde{u} - u_0}{\alpha_u} \\ \tilde{y} = \dfrac{\tilde{v} - v_0}{\alpha_v} \end{cases}$	$\begin{cases} x' = \dfrac{u' - u_0}{\alpha_u} \\ y' = \dfrac{v' - v_0}{\alpha_v} \end{cases}$

The core idea is notice that up to now we only miss lens distorsion in our model AND that lens distorsion is the only NL part of the final model we aim to get: that means that up to now our model (linear) provides us with UNdistorted pixel coordinates AND of course image analysis (CV process) provides us with REAL distorted pixel coordinates; we can exploit this to derive the lens distorsion model! BUT our available data are pixel coordinates meanwhile the lens distorsion model deals with continous coords, so we need to match these two!

Lens distortion parameters (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:

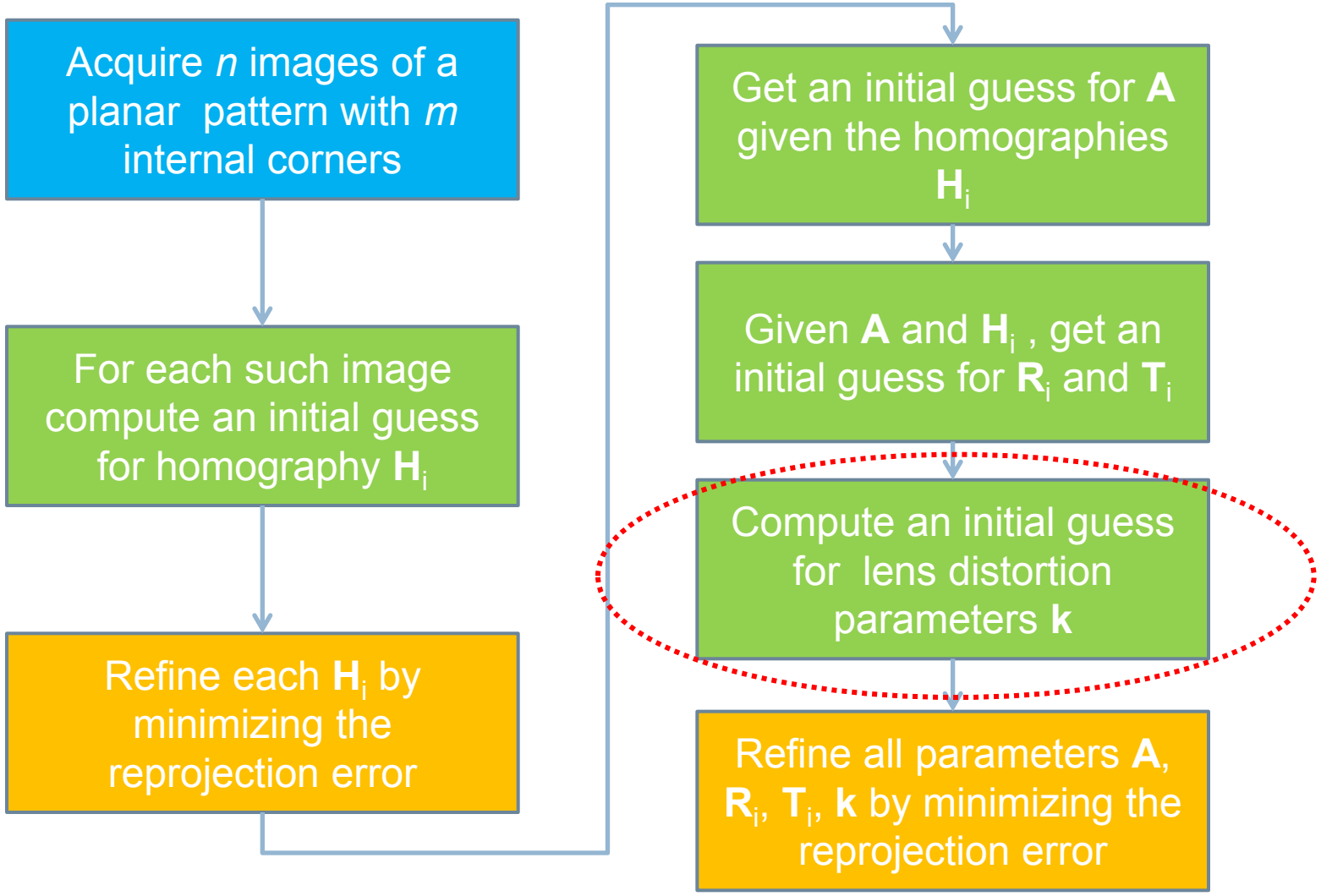
$$\underset{2mn\times 2}{\mathbf{D}}\ \underset{2\times 1}{\mathbf{k}} = \underset{2mn\times 1}{\mathbf{d}} \rightarrow \mathbf{k} = \mathbf{D}^{+}\mathbf{d} = \left(\mathbf{D}^{T}\mathbf{D}\right)^{-1}\mathbf{D}^{T}\mathbf{d}$$

$$\mathbf{D}^{+} = \mathbf{V}\boldsymbol{\Sigma}^{+}\mathbf{U}^{T}$$

Solution in the least-square-sense:
$\mathbf{k}^{sol} = \arg\min_{\mathbf{k}} \|\mathbf{D}\mathbf{k} - \mathbf{d}\|^2$

Indeed, considering all points in all images (recall that the set of lens distorsion parameters is the same accross all of them) we finally get an overconstrained non-homogeneous linear system of 2mn eqs in 2 unknowns $\mathbf{D}\mathbf{k} = \mathbf{d}$ with least-square-sense solution that can be computed as $\mathbf{k} = \tilde{\mathbf{D}}\mathbf{d}$ where $\tilde{\mathbf{D}}$ is computed as $\tilde{\mathbf{D}} = \mathbf{U}\boldsymbol{\Sigma}^{+}\mathbf{V}^{T}$ with $\mathbf{U}\boldsymbol{\Sigma}\mathbf{V}^{T}$ beeing the SVD of $\mathbf{D}$ and $\boldsymbol{\Sigma}^{+}$ beeing the pseudoinverse of $\boldsymbol{\Sigma}$!

Zhang's Method



Refinement by non-linear optimization

- Again, the procedure highlighted so far seeks to minimize an algebraic error rather than the actual reprojection error.
- A more accurate solution can instead be found by a so called Maximum Likelihood Estimate (MLE) aimed at minimization of the reprojection (i.e. geometric) 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 parameters.

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

Some important additional considerations regarding PPMs are reported in Appendix 2.

Again, minimize the geometric error by solving this optimization minimization problem! In particular, if we divide the scalar optimal cost obtained by mn (number of samples) and we take the square root, we obtain the so called RMSE (root mean square error): IF this value is lower w.r.t. a pixel ($\in [0, 0.7]$) THEN we say that globally we have a subpixel reprojection error AND the CALIBRATION is good!

RECALL THAT ALSO SCENES CAPTURED BY MORE CAMERAS IS A RELEVANT CONCEPT! (for example in CV proc. to extract 3D geometry!)

Calibration of a stereo camera (1)

NEW ARGUMENT!

CV LAB Computer Vision Laboratory

SEPARATED CALIBRATIONS

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).

WHAT WE NEED

However, calibrating a stereo system requires determining also the rigid motion, $\mathbf{R}$ and $\mathbf{T}$, between the two cameras (i.e. between CRF_L and CRF_R)

SIMPLE POSSIBLE SOLUTION

Knowing the extrinsic parameters wrt the same WRF would allow to compute the rigid motion between the two cameras by simply chaining 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 by showing the pattern to both cameras in at least one image of their calibration sequences.

$P^L = G_L P^W$; $P^R = G_R P^W$

$\Rightarrow P^R = G_R G_L^{-1} P^L$

$\Rightarrow G_{LR} = G_R G_L^{-1}$

R ≡ right camera ref.sys.
L ≡ left camera r.s.
W ≡ world r.s.

$G = \begin{bmatrix} R & T \\ 0 & 1 \end{bmatrix}$

However, this is not a robust solution, especially wrt noise, and can easily yield to quite inaccurate results.

NOTICE: THIS SIMPLE SOLUTION ONLY RELIES ON THE USAGE OF ZHANG'S METHOD!

A simple solution that relies on showing to cameras the same calibration target: it gives us NOT so good results!

Calibration of stereo camera: getting relative position!

Considering here the case of a stereo system (also called "stereo rig" or "stereo head") with two individual cameras, surely we can run the Zhang's calibration method individually for both of them: for each one of them we get intrinsic params A, lens distorsion params K and a set of extrinsic params Gi ∀image i∈[1,n].

From stereo cameras theory it is known that we ALSO NEED the relative position among them, indeed a rigid motion AKA rototranslation among the two represented as [R|T], typically from left to right AKA from CRF_L to CRF_L (this is foundamental for things as actually computing depth of points from disparities or even at first glance actually conveniently find correspondencies between images).
The whole point of stereo calibration is to find out these R and T!

FOR EACH IMAGE i∈[1,..,n], exploiting out G_{R_i} and G_{L_i} knowledge, we can get an estimate for the required rototranslation, obtaining a set of rototranslations G_{LR_i} none of which is better w.r.t. the others, all of them statistichally associated to the same amount of noise!

The core of the binocular\stereo variation of the Zhang's calibration method is EXPLOITING this set of G_{LR_i} estimates to produce an unique initial guess to be fed to an optimization minimization algorithm of the reprojection error in a least square sense!
Recalling that G is made up of T and R:

- the initial guess for T is the median accross all images (the median of a vectors collection is the one vector WITHIN THAT collection that minimizes the sum of distances from all other vectors).
- the initial guess for R is obtained in the same way BUT biunivocally converting rotation matrixes in triplets of scalar values (for example relying on Euler Angles) before taking the median.

COOL: if considered not good enough, in this opt problem we might improve (and so consider as decision vars of the opt problem) also A,K for R\L cameras!

Calibration of a stereo camera (2)

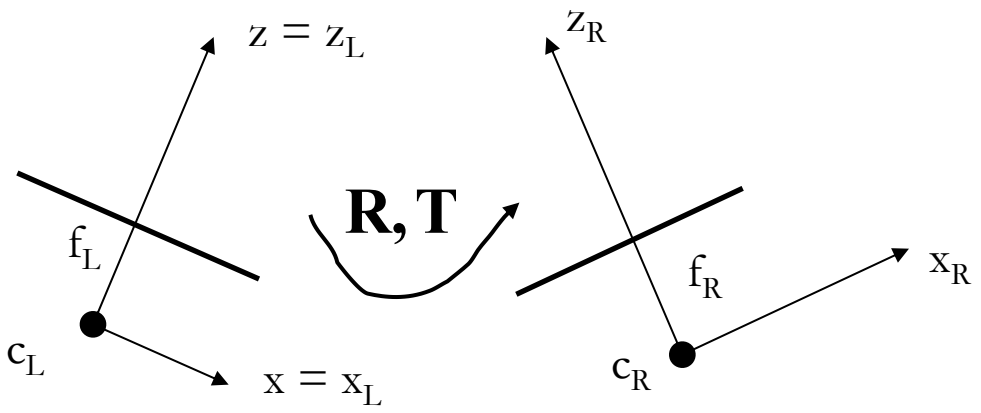
- A more accurate solution is achieved by starting from an initial guess for $\mathbf{R}$ and $\mathbf{T}$ and then refining the estimation by non-linear minimization of the reprojection error.
- To obtain a robust initial guess, the same position of the planar calibration target is seen in each image 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:

$$\sum_{k=L,R}\sum_{i=1}^{n}\sum_{j=1}^{m}\left\|\mathbf{m}_{i,j}^{k}-\tilde{\mathbf{m}}\left(\mathbf{A}_L,\mathbf{A}_R,\mathbf{K}_L,\kappa_R,\mathbf{R},\mathbf{T},\mathbf{w}_j\right)\right\|^2$$

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 "WRF" of the Stereo Camera and will be denoted hereinafter as **SRF** (Stereo Reference Frame)) → the one in which we'll pick our 3D world points!
- 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 $\mathbf{R}$ and $\mathbf{T}$, with the corresponding PPMs given by:

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



Curiosity: if the SRF (indeed the common WRF) is choosen to not be CRF_L nor CRF_R, then it's typically called a cyclopean reference system!

Stereo camera RECTIFICATION

Considering again the case of a stereo system with two cameras, we know from stereo theory that **standard geometry** is highly preferrable in order to have corresponding pixels at same rows in the image, which makes **finding corrispondencies way more efficient and effective**.

In order to have that, we need:
- the intrinsic matrixes of the camera are the same: $A_L = A_R$
- the cameras are perfectly aligned, such that R,T rototraslation from $SRF \equiv CRF_L$ to CRF_R happens to be $R = I$ and $T = [o\ o\ -b]^T$, with b distance between C_L and C_R optical centers of CRF_L and CRF_R.

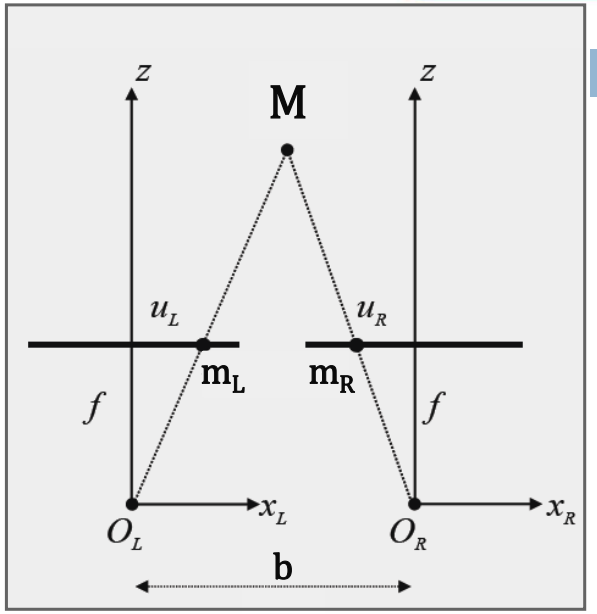
Problem: even if we properly realize the stereo system, these properties will NEVER be accomplished because of non idealities! Similar BUT different intrinsic matrixes, ALMOST aligned CRFs!

Solution: after obtaining the two images from the stereo system (UNdistorted version, undistorsion actuated with the already seen equations that operates on pixel coordinates) just **RECTIFY** them on the fly before usage, which means process them WITH SOME PROPER HOMOGRAPHIES in order to get them as they were obtained with an ideal stereo system that perfectly complies to standard geometry!

How that works? The basic idea is to define new CRF_L' and CRF_R' that BOTH complies to the standard geometry conditions AND aim to be the most possible similar to previous $CRF_L \backslash CRF_R$.

Typical choices: $A = \text{mean}(A_L, A_R)$, CRF_L' picked with $\hat{x}_{CRF_L'} \parallel B$ (where $B \triangleq C_R - C_L$ is the so called **baseline vector**) and $\hat{y}_{CRF_L'} \perp \text{plane}(\hat{x}_{CRF_L'}, \hat{z}_{CRF_L})$, CRF_R' picked by rototraslating from CRF_L' of $R = I$, $T = [o\ o\ -b]^T$.

Rectification



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

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

- 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 each of the two original cameras can be **rectified**, i.e. transformed into those that would have been acquired by the new ones, by a **homography**.
- Rectification needs to happen after compensation of lens distortion (i.e. on undistorted images). An initial transformation to remove lens distortion is thus always applied before transforming again the images according to the **rectification homographies**.

The new PPMs

ALIAS RELATED TO CRF_L' and CRF_R'

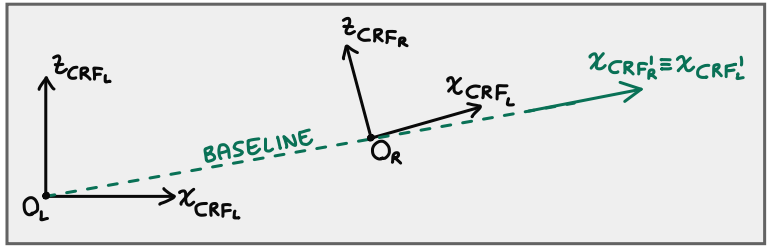
Given $\mathbf{A}_L$, $\mathbf{A}_R$, $\mathbf{R}$ $\mathbf{T}$ the new PPMs can be achieved as follows:

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

These are the PPMs of CRF_L' and CRF_R'!
In particular, from the WRF to CRF_L' we've a pure rotation of Rnew
Instead, from WRF to CRF_R' we've a rotation of Rnew and a translation of $-R_{new}C_R$ (indeed, to find the new optical center – origin – of CRF_R' w.r.t. $WRF\equiv CRF_L$ is sufficient to rotate its coords w.r.t. $WRF\equiv CRF_L$ – AKA C_R – of rotation Rnew; finally the translation vector from CRF_L' to CRF_R' is defined as the coords of CRF_R' origin w.r.t. CRF_L' switched in sign: $-R_{new}C_R$)

The fact that Rnew is the rotation matrix (new one) is related to the fact that $WRF\equiv SRF\equiv CRF_L$ and the orientation of the two new CRF_L' and CRF_R' w.r.t. WRF (given by Rnew) is gonna be the same cause they're gonna be aligned! Notice: the WRF is not modified at all (same for $SRF\equiv CRF_L\equiv WRF$ and CRF_L, CRF_R; we are simply generating CRF_L' and CRF_R')

Important: recall that a rotation matrix from a first system reference frame to a second one, indeed presents in its rows the unit vectors of the axes of the second expressed within the first. In our case of study we've Rnew from WRF to $CRF_L'\backslash CRF_R'$.
Also, recall that given a point P with coords in a first reference frame $P^{(1)}$ and coords in a second reference frame $P^{(2)}$ AND given a rototraslation R,T from the first to the second reference frame, it holds: $P^{(2)}=RP^{(1)}+T$.
In our case of study consider WRF and CRF_R as the first and second reference frames, then $C_R^{(2)}=RC_R^{(1)}+T$ and $C_R^{(2)}=0$, which implies $C_R=-R^{-1}T=-R^TT$ in WRF (recall that for all rotation matrixes it holds $R^{-1}=R^T$).



Rectification Homographies

- Both cameras undergo a rotation about the optical center and a change of intrinsics. We have already studied that, in such a case, the images (i.e. the original and rectified ones) are related by a homography and thus we know how to compute these transformations.
- Considering the left camera first:

$$\widetilde{m}_L = \mathbf{A}_L[\mathbf{I}\ \mathbf{0}]\widetilde{\mathbf{M}} = A_L M$$
$$\widetilde{\mathbf{m}}'_L = \mathbf{A}_{new}[\mathbf{R}_{new}\ \mathbf{0}]\widetilde{\mathbf{M}} = A_n R_n M$$

$$\widetilde{m}_L = \overbrace{\mathbf{A}_L \mathbf{R}_{new}^{-1}\ \mathbf{A}_{new}^{-1}}^{3\times3}\ \widetilde{\mathbf{m}}'_L \rightarrow \mathbf{H}_L$$

Rectification Homography for the left image

- To compute the other rectification homography, we may conveniently think of moving the origin of the WRF into the optical centre of the right camera: (QUICK TRICK)

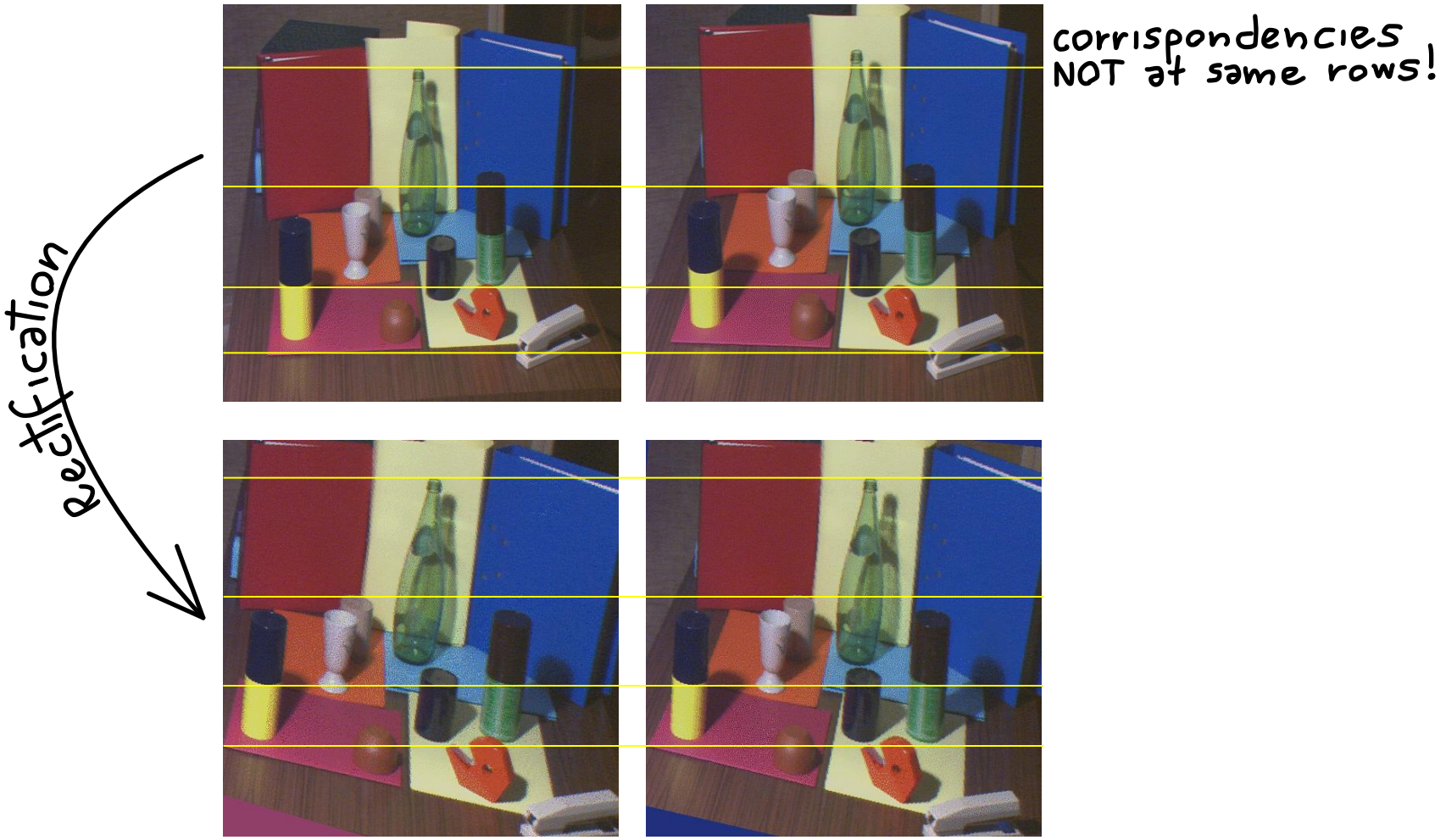
$$\widetilde{m}_R = \mathbf{A}_R[\mathbf{R}\ \mathbf{0}]\widetilde{\mathbf{M}}_R$$
$$\widetilde{\mathbf{m}}'_R = \mathbf{A}_{new}[\mathbf{R}_{new}\ \mathbf{0}]\widetilde{\mathbf{M}}_R$$

$$\widetilde{m}_R = \overbrace{\mathbf{A}_R \mathbf{R}\mathbf{R}_{new}^{-1}\ \mathbf{A}_{new}^{-1}}^{3\times3}\ \widetilde{\mathbf{m}}'_R \rightarrow \mathbf{H}_R$$

Rectification Homography for the right image

A different derivation of the rectification homographies can be found in Appendix 3.

An example



A couple of important remarks:

- ## The direction of the obtained transformations

Notice that, as it also was for the transformation that links distorted pixel coordinates to undistorted one, even these rectification homographies are defined "backward"; indeed they are defined FROM what we typically want to obtain TO what we typically want to transform! Examples: undistorted pixel coords ⇝ distorted pixel coords, rectified pixel coords ⇝ unrectified pixel coords.
There is a very SOLID reason for that which will be illustrated in the following warping theory!

- ## Relation between pixel coords & 3D coords when CRF≡WRF

Given a calibrated camera for which A known where CRF picked coincident with WRF AND for each pixel given to be known its depth (AKA z 3D coord w.r.t. WRF≡CRF of the into-it-projected 3D world point), we can easily write:

$$A[I|0]\begin{bmatrix}x\\y\\z\\1\end{bmatrix}=A\underbrace{\begin{bmatrix}x\\y\\z\end{bmatrix}}_{\triangleq P}=\begin{bmatrix}\alpha_u & 0 & u_0\\0 & \alpha_v & v_0\\0 & 0 & 1\end{bmatrix}\begin{bmatrix}x\\y\\z\end{bmatrix}=\underbrace{\begin{bmatrix}\alpha_u x+u_0 z\\ \alpha_v y+v_0 z\\ z\end{bmatrix}}_{\triangleq p^*}$$

also, we define $p\triangleq\begin{bmatrix}\alpha_u x/z+u_0\\ \alpha_v y/z+v_0\\ 1\end{bmatrix}=\begin{bmatrix}u\\v\\1\end{bmatrix}$ as pixel coords dependent vector

→ Defined p^*, P and p as shown; we have $AP=p^*$ and $p=p^*/z$; we finally obtain:

$$P=zA^{-1}p$$

where again $P\triangleq[x\ y\ z]^T$, $p\triangleq[u\ v\ 1]^T$

In a simpler way, we can recall $\tilde{m}=A[I|0][x\ y\ z\ 1]^T$ and we can develop this equation obtaing exactly $\tilde{m}=AP=p^*$ and then, in order to pass to non-homogeneous (AKA euclidian) pixel coordinates (recall indeed that given the image of course we have non-homogeneous pixel coordinates) divide the obtained result by the value of its third component $\tilde{w}=(p^*)_3=z$, passing from $[\tilde{u}\ \tilde{v}\ \tilde{w}]^T=AP=p^*$ to $[u\ v\ 1]^T=AP/z=p^*/z$ from which again we obtain $P=zA^{-1}[u\ v\ 1]^T=zA^{-1}p$!

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 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**.

Point Cloud: given an image, is defined as point cloud a collection, for all image points, of the related 3D coords within the CRF!

TOF: a time-of-light camera, it only sense depths by sending light rays and sensing them back; for each pixel depth is measured sensing the time interval

Structured Light Sensor: projecting a light pattern (which is well known) before taking the image and then exploit that to compute depths!

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

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

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

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

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

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

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

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

with:

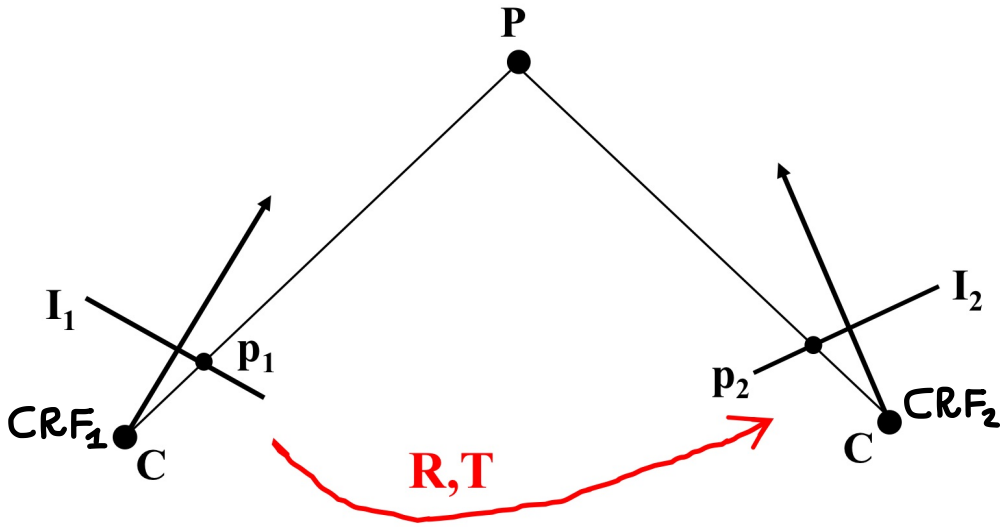
$$A^{-1} = \begin{bmatrix} \frac{1}{\alpha_u} & 0 & -\frac{u_0}{\alpha_u} \\ 0 & \frac{1}{\alpha_v} & -\frac{v_0}{\alpha_v} \\ 0 & 0 & 1 \end{bmatrix}$$

..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).

- Given a pixel $[u_1,v_1]^T$ in image1, Knowing A_1 and z_1 we can easily compute $P_1 = z_1 A_1 [u_1, v_1, 1]^T$ as already shown, with P_1 beeing expressed in CRF1. Then, we can get the SAME point BUT expressed in CRF_2 as $P_2 = RP_1 + T$. Finally, we easily compute $\tilde{m}_2 = [\tilde{u}_2 \ \tilde{v}_2 \ \tilde{w}_2]^T = A_2 P_2$. It's quite relevant to notice that the new pixel coords in image2 are given by $u_2 = \tilde{u}_2 / \tilde{w}_2$ and $v_2 = \tilde{v}_2 / \tilde{w}_2$ AND ALSO that by virtue of equations previously shown $\tilde{w}_2 = z_2$ is the reprojected depth of the same 3D point BUT withing CRF_2!

- Cool remark: this principle is the one onto which RGBD images are produced (they are colored RGB images with also depth information for all pixels): two distinct depth only and RGB only cameras are used and then depths are reprojected from the first to the second!



$$\tilde{p_2} = A_2 T_{1\to 2}(z_1 A_1^{-1} p_1) \qquad \text{with:} \quad T_{1\to 2}(P_1) = RP_1 + T$$

$\tilde{p_2} = \begin{bmatrix} \tilde{u}_2 \\ \tilde{v}_2 \\ \tilde{w}_2 \end{bmatrix}$ $\qquad$ $\begin{bmatrix} u_1 \\ v_1 \\ 1 \end{bmatrix}$

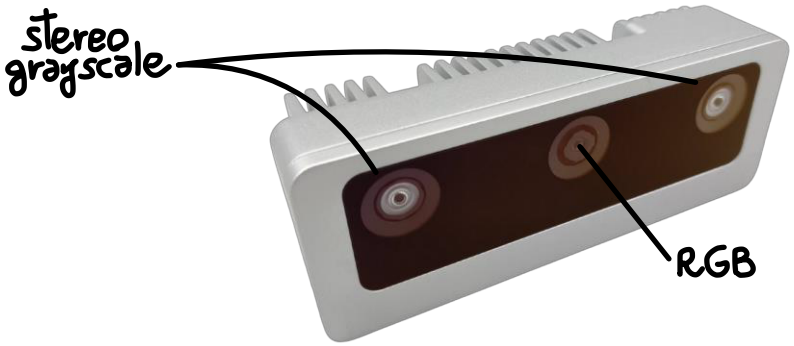
Cameras may be different

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.

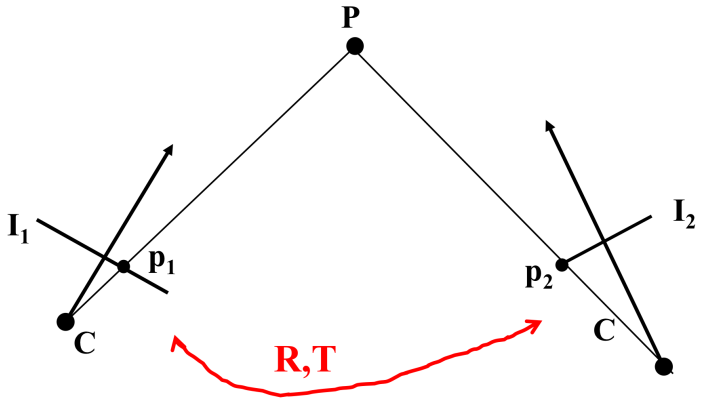
Basler – Blaze ToF + RGB

typical values: RGB up to 2÷20Mpixel
meanwhile TOF camera ≤1Mpx,
typically also 640x420

Luxonis – OAK-D-PoE



Microsoft Azure Kinect
TOF + RGB



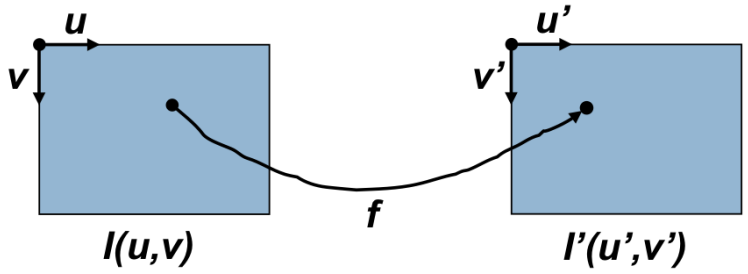
$$p_2 = A_2\, T_{1\to 2}(z A_1^{-1} p_1) \quad \text{with:} \quad T_{1\to 2}(P_1) = RP_1 + T$$

Lucid – Helios2 & Triton

This is how an RGB-D image is obtained in cameras that provide both *depth* and *colour*

Image Warping

THIS IS A CASE OF IMAGE PROCESSING



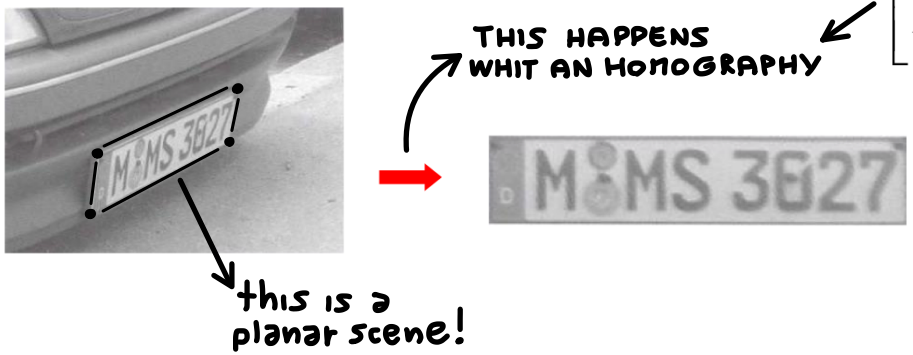
$$\begin{cases} u' = f_u(u,v) \\ v' = f_v(u,v) \end{cases}$$

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

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

THIS HAPPENS WHIT AN HOMOGRAPHY



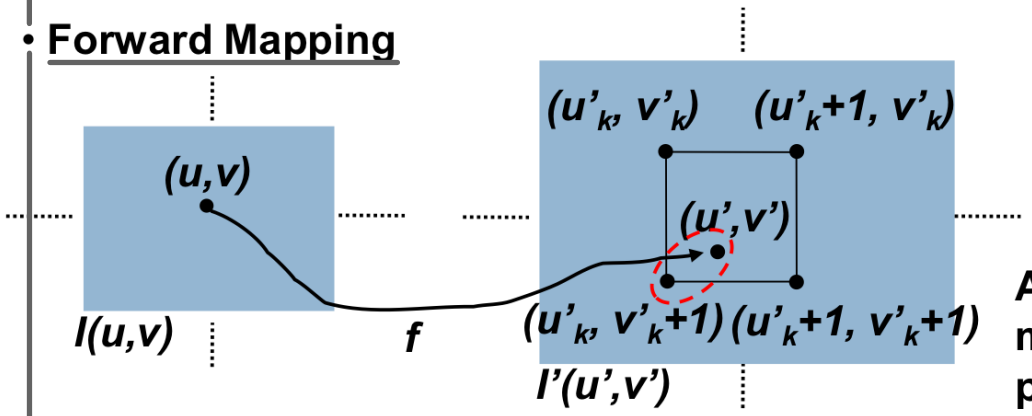
Other examples:
- *Remove lens distortion*
- *Stereo rectification*
- .....

What is warping? It's related to image processing. Indeed, it's an image process in which given an image a new one is synthetized (also said: allucinated) from the first by appling some proper fu/fv functions to all original pixel coordinats: it's a purely geometric remapping of pixels in which each pixel is geometrically moved BUT absolutely not altering its content\information.

IN A NUTSHELL: in general, a warping is defined as an image process that geometrically transforms pixel coordinates accordingly to a certain rule (without altering its informative content).

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**

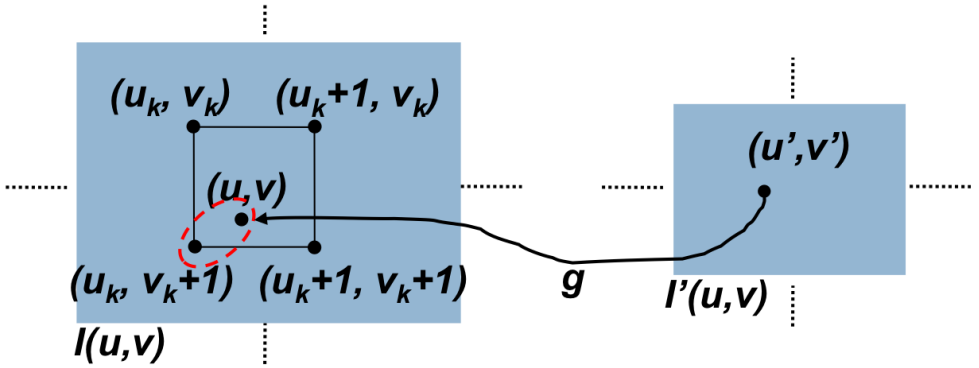
$$\begin{cases} u = g_u(u', v') \\ v = g_v(u', v') \end{cases} \rightarrow \forall (u', v'): \; I'(u', v') = I\left(g_u(u', v'), g_v(u', v')\right)$$

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

A FORWARD warping from the source to the target has the problem that when the original integer pixel coordinates are warped\transformed, what we get are FLOAT pixel coordinates; regardless the methodology used to mutate them to integer values, the target image will always be prone to HOLES (a pixel of the source was never reached) and FOLDS (a pixel of the target was reached more then one time), randomically scattered accross the image.

On the other end, a BACKWORD warping from target to source, in which each pixel of the source is warped to obtain the source pixel (after mutation into integers) that needs to be mapped to it, produces a target image WITHOUT folds and with holes mabye present but isolated in precise peripherical portion of the target image itself!

Mapping Strategies



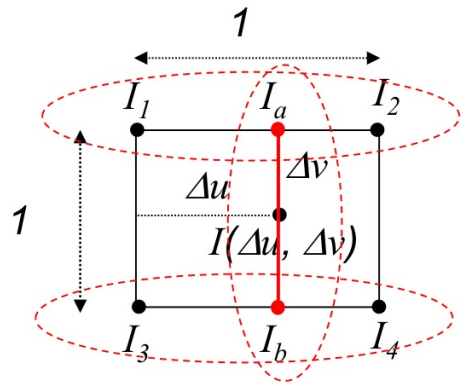
$$(u_k, v_k) = (\lfloor u \rfloor, \lfloor v \rfloor)$$

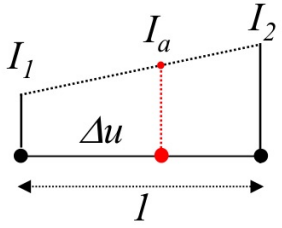
- **Mapping from the closest point** (Nearest Neighbour Mapping)
- **Interpolation between the 4 closest points** (*bilinear*, bicubic,…)

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

Indeed, when we warp integer pixel coords we get in the first place some FLOAT new pixel coords: in order to MAP them into integers values again there are various possibilities.
A trivial one: NNM, nearest neighboor mapping.
A little bit more sophisticated one: bilinear interpolation, a mean of the four neighboors weighted w.r.t. the distance from them (the less the distance, the higher the weight)!

Bilinear Interpolation (1)





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

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

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

two horizontal linear interpolations ($I_1, I_2 \rightarrow I_a$; $I_3, I_4 \rightarrow I_b$)

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

one final vertical interpolation (between I_a and I_b)

bilinear: linear interpolations along the two main directions

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

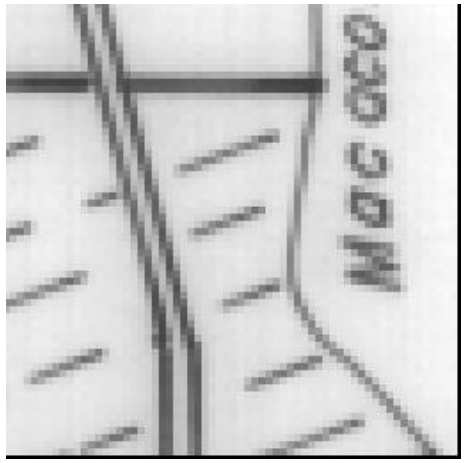
ALGEBRAIC MANIPULATION

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

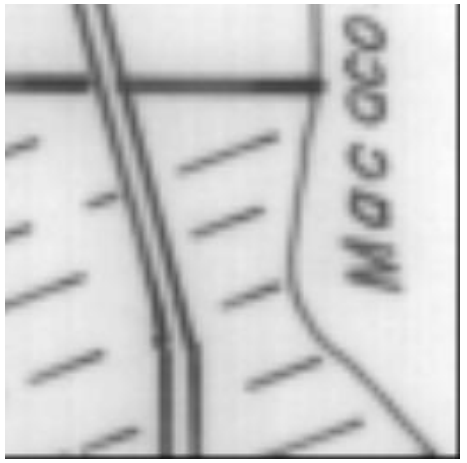
THIS IS INDEED A WEIGHTED SUM (weights sum one and all positive) WHERE THE MORE WE'RE NEAR TO A NEIGHBOOR PIXEL THEN THE HIGHER THE RELATED WEIGHT!

Bilinear Interpolation (2)

Input

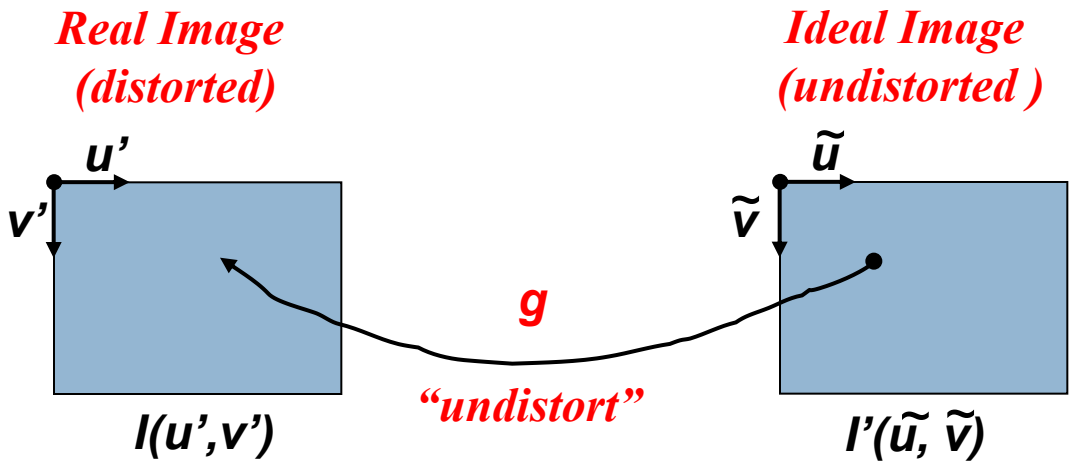


Warp (Zoom) by NNM

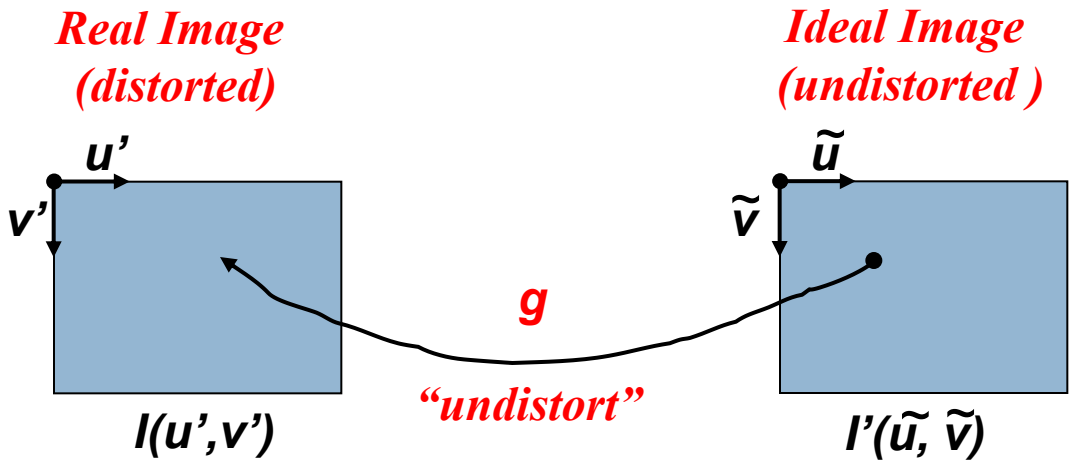


Warp (Zoom) by Bilinear Interpolation

Warping to compensate lens distortion (1)



Warping to compensate lens distortion (2)



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

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

Appendix 1 - DLT Algorithm: 4 points case (1)

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

$$\mathbf{Ah} = 0, \quad \mathbf{A} = \begin{bmatrix} \mathbf{A}_1 & \mathbf{A}_2 \ldots \mathbf{A}_9 \end{bmatrix}, \quad \mathbf{h} = \begin{bmatrix} \mathbf{h}_1 \\ \mathbf{h}_2 \\ \mathbf{h}_3 \end{bmatrix} = \begin{bmatrix} h_1 \\ h_2 \\ \vdots \\ h_9 \end{bmatrix}$$

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, $h_1 \ldots h_9$ 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

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

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

Appendix 1 - 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 to be inverted and then, using the formulas shown above it is possible to compute h_9 first and then **h**

Appendix 2 - Decomposition of the PPM

- Zhang's method provides separately $\mathbf{A}$, $\mathbf{R}$ e $\mathbf{T}$. 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,** so as to factorize a real square matrix into the product between an ***orthogonal*** and ***upper triangular*** matrices.
- Let's then $\mathbf{P}^{-1} = \mathbf{UL}$ be the factorization of the inverse of the square matrix $\mathbf{P}$, with $\mathbf{U}$ orthogonal and $\mathbf{L}$ 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:

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

Appendix 2 - 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 :

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

3D coordinates of the optical center in the WRF

Appendix 2 - Optical ray from the PPM

- The optical ray of an image point, $\widetilde{\mathbf{m}}$, is the 3D line between $\widetilde{\mathbf{m}}$ and the optical centre, , $\tilde{\mathbf{C}} = \begin{bmatrix} \mathbf{C} \\ \mathbf{1} \end{bmatrix}$.

  Indeed, the optical ray is the locus of points, $\widetilde{\mathbf{M}}$, satisfying the equation $k\widetilde{\mathbf{m}} = \widetilde{\mathbf{P}}\widetilde{\mathbf{M}}$ for a given $\widetilde{\mathbf{m}}$.

  int 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)

- 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

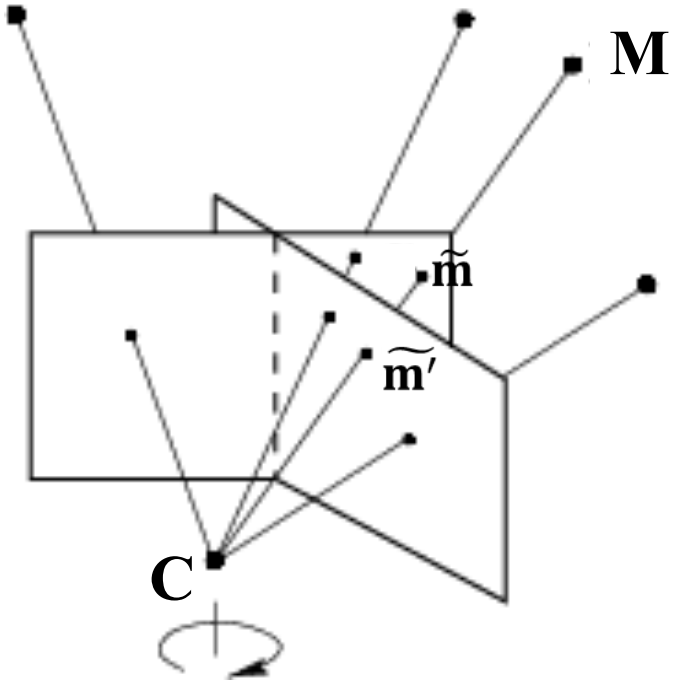
$$\mathbf{V} = \mathbf{P}^{-1}\tilde{\mathbf{m}}$$

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

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

Appendix 3 - Rectification Homographies by optical rays (Appendix 2)

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



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

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

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

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

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

Homography

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) Bradski, Adrian Kaehler, “Learning OpenCV - Computer Vision with the OpenCV Library”, O'Reilly Media, 2008.