DCCAL - Discrete Cameras Calibration using Properties of Natural Scenes

Milestone 5
Multiple Cameras Calibration

Funded by:

FCT Fundação para a Ciência e a Tecnologia
MINISTÉRIO DA CIÊNCIA, TECNOLOGIA E ENSINO SUPERIOR

FCT PTDC/EEA-CRO/105413/2008
January 2010 - December 2012

July 2012

In-Situ Camera Calibration [*]

In-situ camera calibration, i.e. calibration of a camera mounted in the position where it will be used, has the advantage that the camera can be tuned (e.g. focused) in-situ, to fit the local requirements. On the other hand, this camera calibration methodology implies that conventional calibration tools may turn out to be impractical. For instance, considering the well know calibration toolbox by J. Y. Bouguet [1], one has to image a planar chess pattern at various poses relative to the camera, always covering most of the imaging area. For example, if an outdoors camera is mounted at the height of a 3rd floor, it is not simple coming close to it to display an A4 calibration chess pattern, and is not practical constructing and moving a chess pattern with various meters wide. In addition, this calibration is mostly about intrinsic parameters, and thus do not provide distances (rigid pose transformations) among the various cameras of a network of cameras. In this report we discuss alternatives for calibrating cameras in-situ.

1 Map Based Camera Calibration

One way to calibrate a camera is finding a set of correspondences between 3D space points and their 2D projection on the camera image. Obtaining the 2D information can be done easily just by clicking on an image. Obtaining 3D information in a global reference frame for the points observed in the image is much harder. Professionals working with theodolites can provide such information, but the costs are very high. In this report we consider an alternative based on using a blueprint of the location.

In many real cases there is an available a map, or blueprint, of the place where cameras are mounted, as the one shown on Fig.1 (a). This blueprint may contain already the position of the cameras or may allow an easy identification of these locations. Similarly, some landmarks can be visible to the camera, and the calibration procedure can therefore consist of three simple steps:

1. Marking on the blueprint two points, namely the camera location and a point spotted by the camera.
2. Arbitrating the height of the camera, and assuming a typical field of view (lens), which allows to create an initial guess for the projection matrix $P = K\ [R \quad t]$ (nominal calibration).
3. Using the landmarks found on the image and their correspondences on the blueprint, the constructed projection matrix can be fine tuned, minimizing the reprojection error. A P_{opt} projection matrix is obtained by varying the focal distance (intrinsic parameters), the camera rotation angles and height (extrinsic parameters).

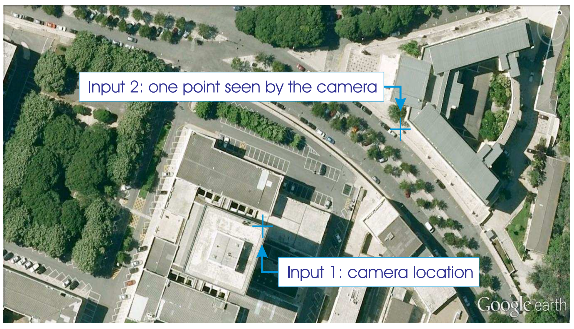
1.1 Nominal Calibration

The camera location and the point can usually be marked by simply inspecting the blueprint and camera image (see Fig.1). Provided some additional information, one obtains an initial guess of the projection matrix.

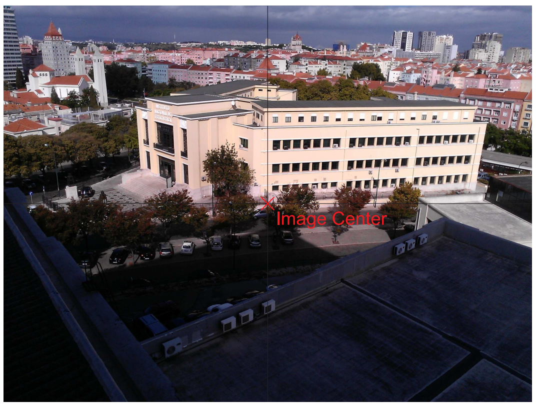
In this work we use the pin-hole projection model, in which a 3D point represented in homogeneous coordinates, $M = [X\ Y\ Z\ 1]^T$, is projected to a 2D point $m = [u\ v\ 1]^T$, also in homogeneous

[*]This report is more detailed in the MSc thesis *NetCam - Network Cameras Calibration*, by Manuel Silva (superv: J. Gaspar) IST/MEEC 2011/2012, and is partially published in *Camera Calibration using a Color-Depth Camera: Points and Lines Based DLT including Radial Distortion*, M. Silva, R. Ferreira, J. Gaspar, In WS in Color-Depth Camera Fusion in Robotics, held with IROS2012. This work was partially supported by the FCT project PTDC / EEACRO / 105413 / 2008 DCCAL, and by the project High Definition Analytics (HDA), QREN - I&D em Co-Promoção 13750.

(a) Local Map



(b) ROI



(c) Camera Image

Fig. 1: Nominal calibration of a camera mounted in its operating place. The local map (a) is zoomed and cropped to a region of interest (ROI), for easier usage, where the camera location and a point seen by the camera (b) can be marked. This scene point is believed to be a point lying on the (camera optical axis), which projects (approximately) on the camera image center (c).

coordinates:

$$m \sim PM, \tag{1}$$

where $\sim$ denotes equality up-to a scale factor, $P = K[R\ t]$ denotes the 3×4 projection matrix, K is a 3×3 matrix containing the intrinsic parameters, t is a 3×1 vector indicating the location of the world origin in camera coordinates, R is a 3×3 matrix representing the orientation of the world frame relative to the camera frame. Note that R and t can be obtained easily from and transformed to world coordinates:

$$\begin{cases} R = ({}^wR_c)^T \\ t = -({}^wR_c)^T\,{}^wt_c \end{cases} \Leftrightarrow \begin{cases} {}^wR_c = R^T \\ {}^wt_c = -R^T t \end{cases} \tag{2}$$

where ${}^wt_c$ denotes the location of the camera in world coordinates, ${}^wR_c$ denotes the camera orientation also in world coordinates, and we are assuming that all rotation matrices are ortho-normal meaning $R^TR = RR^T = I$ and thus $R^{-1} = R^T$.

In [2] an initial guess of the camera projection matrix, P_{ini}, can be obtained following three steps:

1. Pointing the camera location, p_1, and selecting a ground point, p_2, on the blueprint of the building (see p_1 and p_2 marked as $Input1$ and $Input2$ in Fig.1(b));

2. Entering an elevation angle, θ;

3. Entering an horizontal field of view, φ and the aspect ratio of the camera image

Steps 2 and 3 will usually have default values, in order to make as simple as possible the task to the user. For instance, $\varphi = 40^o$ corresponds to a common 8mm lens in a 1/4 in CCD. Default values for θ depend on the location, but in the case of In-Situ camera calibration many cameras are at the level of the second floor (about $6[m]$ high), imaging objects in the ground plane closer than $20[m]$, and thus having a typical value of $\theta = 17^o$.

With the parameters referred in steps 1 to 3, one can compute completely a pin-hole (perspective), projection model: p_1 and p_2 define the azimuth direction, the elevation is given by the user, and the roll of the camera is assumed null (these three parameters suffice to define the rotation matrix, R); p_1, p_2 and θ define the projection center of the camera, t in world coordinates; the field of view combined with the size of an image, and assuming the principal point equal to the image center, give the intrinsics matrix, K. Hence we obtain the perspective projection model: $P(\theta_j) = K\ [R \quad t]$ where θ_j represents a vector containing the listed parameters for camera j.

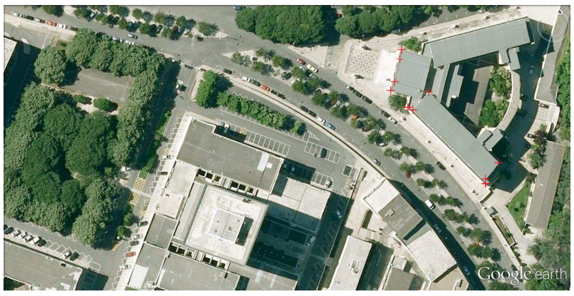
The three steps methodology has been tested on the data shown in Fig.1. Figure 1(b) shows the input data, namely the camera location and a point spotted by the camera. It was assumed that the camera is at a height of approximately $4.5[m]$. Using the defaults of steps 2 and 3, together with a zero roll of the camera, one finally obtains a projection matrix P_{ini} which visual representation is shown in Fig.2(b).

1.2 Optimized Calibration

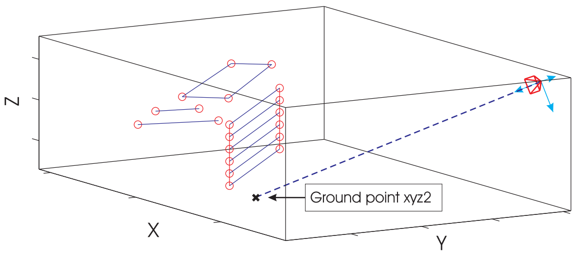
The nominal calibration is just a rough estimation of the projection matrix P. A more precise calibration can be obtained using more information available in the blueprint and the camera image. This methodology is based on information consisting of points of the floor plane, lines on the floor plane and lines orthogonal to the floor plane.

Extracting points of the ground plane is done with a graphical user interface where one clicks points in the blueprint (see Fig.2(a)) and matches these points with the ones on the camera image, as illustrated in Fig.2(c) and (d) (green). The points clicked on the blueprint represents the 3D data on the calibration process, which can be grouped to form lines on the ground plane and vertical lines (e.g. door frames) which are visible and can be marked in the image. The projection of the ground points, using the initial guess P_{ini}, is shown on Fig.2(c).

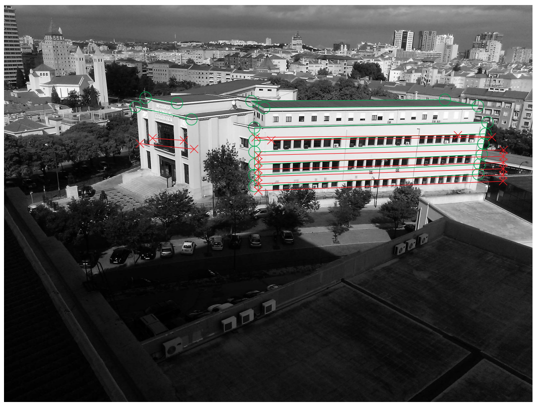
Figure 2 shows some additional information extracted from the blueprint and from the image, namely ground lines and vertical lines. Minimizing the error of the 3D lines projection and 2D lines found in the image, allows fine tuning the projection matrix P_{ini}, and actually obtaining a better estimate for the



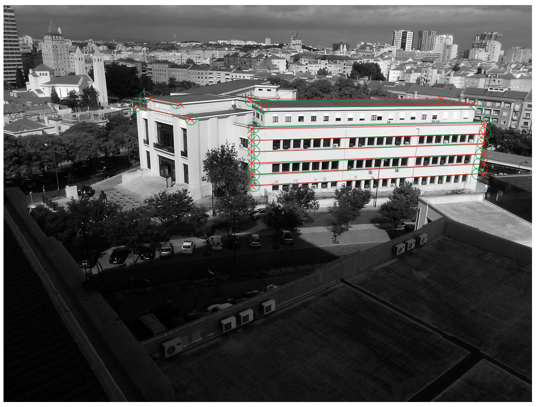
(a) Ground points indicated in the map



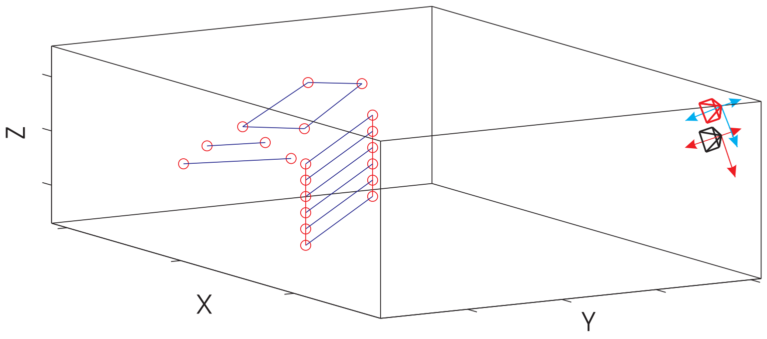
(b) 3D display of points, lines and the camera



(c) Nominal calibration gives a rough matching



(d) Optimized calibration



(e) Ground and vertical lines used for calibration. Camera location before and after optimization

Fig. 2: Additional information, obtained from the local map and the image acquired by the camera, allows the calibration fine tuning. The information contains points on the wall of a building, horizontal and vertical lines (a, b, e). Using the nominal calibration, the projection of 3D vertical lines (red) over the image lines (green) gives just a rough matching (c). After optimizing the calibration (d), there is a good matching of the projected data (red) over the 2D data (green).

camera pose (see Fig.2(e)). In a first approach, the optimization procedure was allowed to change just the focal lengths in K, the rotation R (varying roll, pitch and yaw angles) and the translation vector t.

It was concluded, after some experiments, that is not convenient to allow the the principal point ($K(1:2,3)$) to change freely, as it will compromise the reasonable estimate of K and R. The other parameter that is convenient to keep untouched is the location of the camera at least in its ground coordinates $t(1:2)$. All the 16 ground points, and derived horizontal and vertical lines, have been used in the minimization process. The resulting projection matrix P_{opt} can be decomposed [3] as:

$$K_{opt} = \begin{bmatrix} 827.6 & 0 & 352 \\ 0 & 578.6 & 144 \\ 0 & 0 & 1 \end{bmatrix}, \; R_{opt} = \begin{bmatrix} 0.97 & 0.08 & -0.24 \\ 0.25 & -0.23 & 0.94 \\ 0.03 & -0.97 & -0.24 \end{bmatrix} \; and \quad t_{opt} = \begin{bmatrix} 31.28 \\ 30.44 \\ 4.23 \end{bmatrix}. \tag{3}$$

The reprojection error was found to have the value $Err = \sum (m_{bp} - \hat{m}_{bp})^2/N = 4.9942[pix^2]$, where m_{bp} are the points marked on the camera image, which are assumed to be the projection of the blueprint points (green points in Fig.2(c) and (d)), and $\hat{m}_{bp}$ is the projection of the blueprint points using P_{opt}.

2 Case Study, ISR Camera Network

In this section, the In-Situ calibration methodology proposed in the previous section is used to calibrate a set of cameras mounted at ISR. Figure 3(a) shows the blueprint of the 7th floor of ISR and the location where the cameras are installed, and some calibration data for one of the cameras (isolated green crosses).

In the ISR camera network some cameras have significant radial distortion (see Fig.3(b)). The correction of the radial distortion is made by a nonlinear optimization process which imposes straightness to bended image lines known to be straight, using the radial distortion model as in [1]. The nonlinear optimization problem is solved by a Levenberg-Marquardt algorithm.

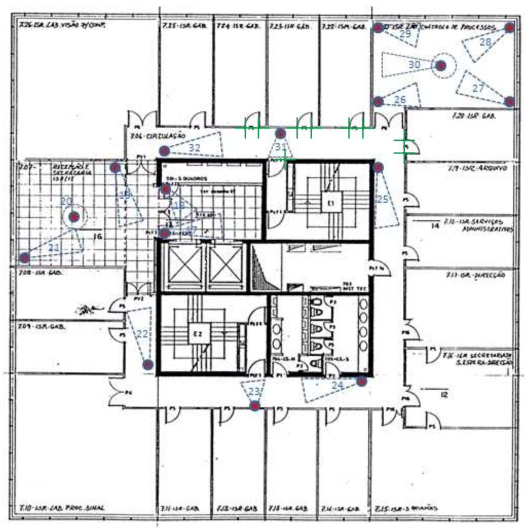
Having corrected the radial distortion, Fig.3(c), the next step consists of marking the camera location in the blueprint, in order to obtain a rough characterization of the nominal projection matrix (See Fig.4). Considering the camera image size and using the same default values and conditions as in Sec.(to estimate the intrinsic parameters), allows to obtain an initial guess to the camera projection matrix, P_{ini}. Figure 4 shows this nominal calibration. In this dataset, the central point of the image is not on the ground, and thus the height was provided to the algorithm ($height = 0.7[m]$).

In order to obtain an optimized estimate of the projection matrix, one needs to input ground points, horizontal lines and vertical lines. This involves a user to click points on the blueprint and on the camera image. Same work can be simplified here by fitting the blueprint over the image. Note that the transformation model is just a 3×3 matrix (*homography*) which can be estimated from four (or more) corresponding points. Figure 5(a) shows the result of superimposing the blueprint over the image.

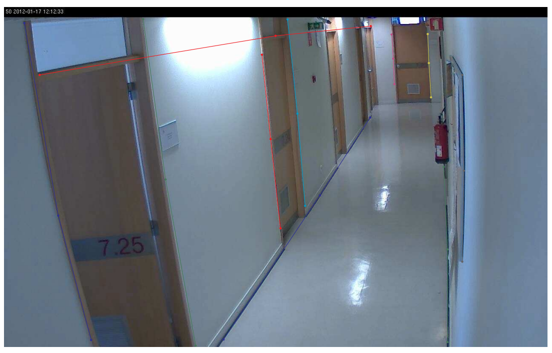
Given that now the blueprint is superimposed over the image, a graphical interface can be designed to help marking points: each point clicked on the image (red circles on Fig.5(a)) has a direct (automatic) correspondence to a ground point clicked in the blueprint (green crosses on Fig.3(a)).

The optimization process, implemented to search for a better estimation of the projection matrix (P_{opt}), was set to vary just the focal length (not the principal point), rotation matrix and camera position (Y component is left fixed). The optimization process searches for the best values, which minimize the reprojection error. The process starts with the nominal estimation P_{ini} and ends with an optimized estimation P_{opt}. Figure 5(c) shows that the reprojection errors are smaller due to the optimization, comparing them to P_{ini} (see Fig.5(b)). Visual evaluation on site allows to state that the optimized pose and orientation of the camera represents better the real mounting, as shown on Fig.5(d).

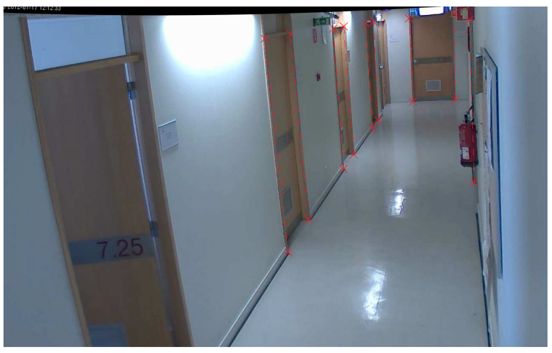
The intrinsic parameters are adapted during the optimization process. In case there are various similar cameras mounted, the intrinsic parameters found for one camera can be used to initialize the optimization process for other cameras. Decomposing the projection matrix P_{opt} [3], leads to the following values to



(a)

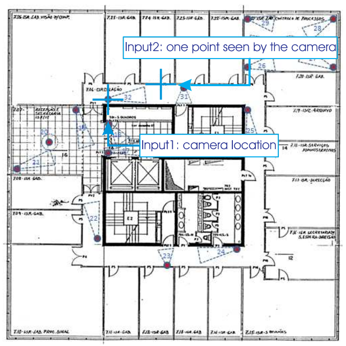


(b)

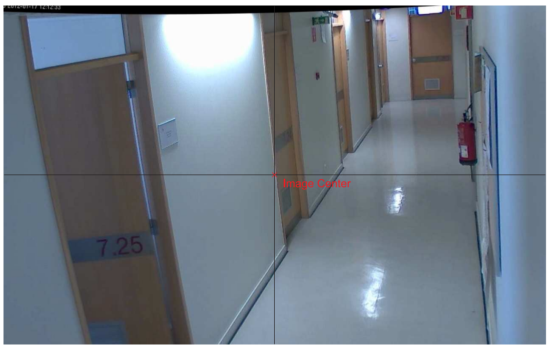


(c)

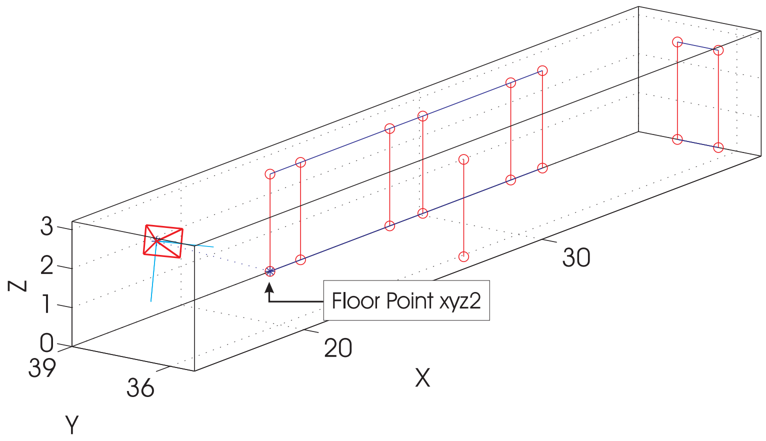
Fig. 3: Blueprint of the 7th floor of ISR where the location of the cameras and field of views (FOVs) are marked as red dots and triangles (a). The green crosses indicate some ground points that are on the FOV of a camera. Calibrating a radial distorted camera (b) involves firstly, correcting its image by imposing the straightness of bended lines. The information, acquired by inspecting the blueprint (a) and the corrected image (c), is used to find an initial guess for the projection matrix, P_{ini}.



(a)

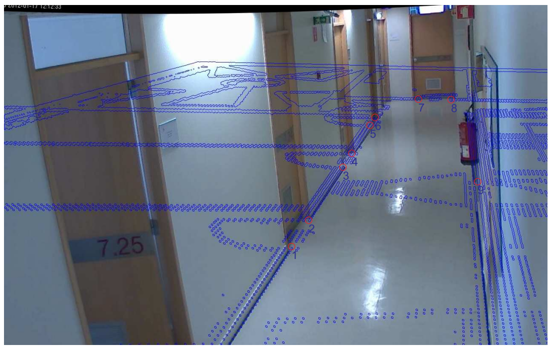


(b)

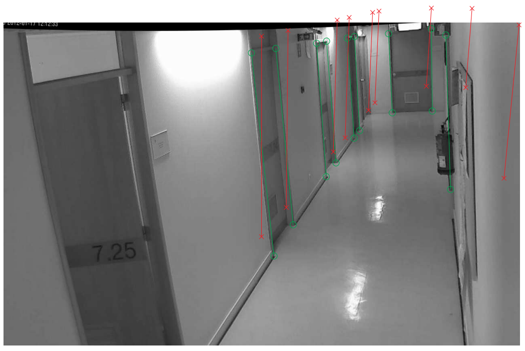


(c)

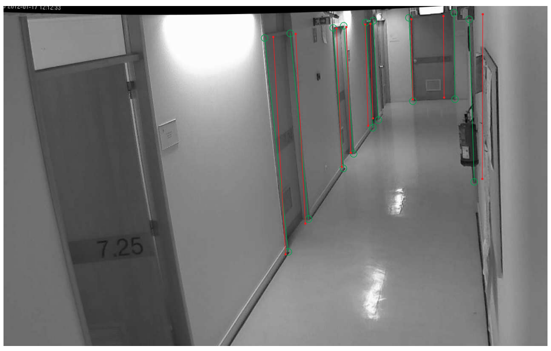
Fig. 4: Nominal calibration. Central point observed by the camera (b) and camera location marked on the blueprint (a). An initial guess of the camera projection matrix, P_{ini}, is estimated. The localization and orientation of the camera can be extracted from the projection matrix [3] (c). The additional points, marked on the blueprint, Fig.3 (green crosses), are used to acquire additional 3D data (red circles).



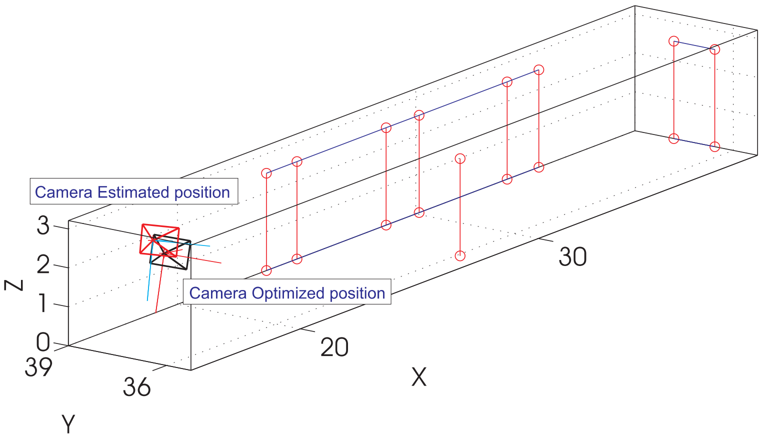
(a) Blueprint superimposed over the camera image



(b) Projection of 3D points using P_{est}



(c) Projection of 3D points using P_{opt}



(d) Estimated camera pose before and after optimization

Fig. 5: Optimized estimation of the projection matrix. The input data consists of ground points, ground lines and vertical lines identified in the blueprint and in the image (b). An alternate way to input ground points both in the blueprint and in the image (superimposed blueprint), can be used to simplify the task of matching 3D and 2D data (a). After optimization (c), the reprojection (red dots) should match closely the observed data (green circles). The estimation of the camera pose is improved by the optimization (d).

the intrinsic and extrinsic camera parameters:

$$K_{opt} = \begin{bmatrix} 1247.9 & 0 & 640 \\ 0 & 1642.9 & 400 \\ 0 & 0 & 1 \end{bmatrix},\ R_{opt} = \begin{bmatrix} 0.29 & \text{-}0.22 & 0.93 \\ \text{-}0.95 & 0.01 & 0.30 \\ \text{-}0.075 & \text{-}0.98 & \text{-}0.20 \end{bmatrix},\ t_{opt} = \begin{bmatrix} 16.27 \\ 36.7 \\ 2.56 \end{bmatrix}. \quad (4)$$

The reprojection error has the value $Err = \sum (m_{bp} - \hat{m}_{bp})^2/N = 13.3091[pix^2]$, where m_{bp} are the points marked on the camera image, which are assumed to be the projection of the blueprint points, while $\hat{m}_{bp}$ is the projection of the blueprint points using P_{opt}.

The proposed calibration process starts with an initial estimate obtained directly by pointing in a blueprint the camera location and a point seen by the camera. Assuming some priors such as the knowledge of the lens and the CCD size, one has a good estimation of the intrinsic parameters matrix, which allows to initialize an iterative optimization process where good approximations to the real parameters of the camera are found.

However, there are some parameters whose estimation makes the process very sensitive to noise. Those parameters are in essence: the principal point and the camera position in the blueprint. This is the reason why in most of the cases we do not optimize these parameters, as they are close to default values (the principal point is close to the image center) or marked with enough quality on the blueprint.

3 *DLT-Lines* Calibration

In this section we introduce the calibration methodology *Direct Linear Transformation based in Image Lines* (*DLT-Lines*) including the estimation of radial distortion. As indicated by the name, we consider lines identified on the image of the camera to obtain its parameters. Contrarily to 3D lines, which are normally represented using Plucker coordinates [3], 2D lines have simple representations as cross products of image points in homogeneous coordinates. In the following we explore this representation to build the calibration methodology. The use of lines, as opposed to using isolated image points, brings an advantage. Image processing can be used for fine tuning the location of the lines in the image and therefore automatically improving the calibration data input.

3.1 *DLT-Lines*

Two cases are considered, namely (i) non-existent radial distortion and (ii) significant radial distortion. In the case where the radial distortion is considered, it is modeled using Fitzgibbon's division model.

Given a 3D line L_i, its projection on the camera image plane, l_i can be represented by the cross product of two image points in projective coordinates, $l_i = m_{1i} \times m_{2i}$. Any point m_{ki} lying in the line l_i implies that $l_i^T\ m_{ki} = 0$. Applying the multiplication by l_i^T on both sides of Eq.1, i.e., $l_i^T\ m_{ki} = l_i^T\ P\ M_{ki}$, leads to:

$$l_i^T\ P\ M_{ki} = 0 \quad (5)$$

where M_{ki} is a 3D point in projective coordinates lying in L_i. As in the case of *DLT-Points*, using the Kronecker product one obtains a form factorizing the vectorized projection matrix:

$$(M_{ki}^T \otimes l_i^T)\ vec(P) = 0. \quad (6)$$

Each pair of 3D point and its corresponding image line, (M_{ki}, l_i), allows writing Eq.6 once, and thus provides one linear constraint in the entries of $vec(P)$. In order to estimate P one needs at least 12 pairs (M_{ki}, l_i). [1] Considering $N \geq 12$ pairs (M_{ki}, l_i), one forms a matrix B, $N \times 12$, by stacking the N matrices $M_{ki}^T \otimes l_i^T$. The least squares solution, more precisely the minimizer of $\|B\ vec(P)\|^2$ s.t. $\|vec(P)\| = 1$, is the right singular vector corresponding to the least singular value of B.

Comparing *DLT-Lines* with *DLT-Points* [3], it is important to note that while in *DLT-Points* one has to provide one 3D-point to one 2D-point correspondences, in *DLT-Lines* one 2D-line, l_i is an image of a

[1] Alternatively, one can state that nondegenerate six 3D lines configuration and their corresponding six image lines are enough to estimate P.

3D-line, L_i and thus indicates, for example, many-3D-points to one-2D-line correspondence. Any point $M_{ki} \in L_i$ forms a linear constraint with l_i (Eq.6). On the other hand, any image line l_i can be paired with any 3D point lying on L_i, i.e., more than one image line can be paired with a 3D point. This property of *DLT-Lines* allows to apply additional image processing tools that add robustness to the extraction of calibration data.

3.2 *DLT-Lines* with Radial Distortion

As noted by Fitzgibbon [4], true lens distortion curves are typically very complex to represent, implying the use of high-order models or lookup tables to model camera radial distortion effect with high precision. On the other hand, considering typical computer vision applications, accuracies of the order of a pixel are all that is required, and an approximation to the cameras' true distortion functions perform as well as the preciser ones.

Fitzgibbon proposed the so called *Division Model* where an undistorted image point, $\hat{m}_u = [u_u \; v_u]^T$ is computed from a radially distorted image point $\hat{m}_d = [u_d \; v_d]^T$ as:

$$\hat{m}_u = \hat{m}_d / (1 + \lambda \left\| \hat{m}_d \right\|^2) \tag{7}$$

where λ represents the radial distortion parameter. The *Division Model* can also be conveniently written in homogeneous coordinates

$$\begin{bmatrix} u_u \\ v_u \\ 1 \end{bmatrix} \doteq \begin{bmatrix} u_d \\ v_d \\ 1 + \lambda \left\| \hat{m}_d \right\|^2 \end{bmatrix}. \tag{8}$$

Note that an undistorted point, $m_u = [u_u \; v_u \; 1]^T$ is a simple function of a distorted point, $m_d = [u_d \; v_d \; 1]^T$,

$$m_u \doteq m_d + \lambda e_d \tag{9}$$

where $e_d = [0 \; 0 \; \|\hat{m}_d\|]^T$.

The coordinates of $\hat{m}_u$ and $\hat{m}_d$ are expressed in a 2D coordinate system having the origin coincident with the image principal point $\hat{c}_o = [c_u \; c_v]^T$. Redefining a distorted image point, $m_d = [u_d \; v_d \; 1]^T$, to have the principal point $\hat{c}_o$ as its reference, one obtains the coordinates to use in Eq.8 simply with a translation, $m_d = [u_d - c_u \; v_d - c_v \; 1]^T = T \; m_d$, where T is a 3×3 matrix.

Using Eq.8 a line l_{12} can be defined as the cross product of two points:

$$l_{12} = \begin{bmatrix} u_{1d} \\ v_{1d} \\ 1 + \lambda s_1^2 \end{bmatrix} \times \begin{bmatrix} u_{2d} \\ v_{2d} \\ 1 + \lambda s_2^2 \end{bmatrix} = \hat{l}_{12} + \lambda e_{12} \tag{10}$$

where s_i is the norm of distorted image point i, $s_i^2 = u_{id}^2 + v_{id}^2$, the distorted line is denoted as $\hat{l}_{12} = [u_{1d} \; v_{1d} \; 1]^T \times [u_{2d} \; v_{2d} \; 1]^T$ and there is a distortion correction term $e_{12} = [v_{1d}s_2^2 - v_{2d}s_1^2, \; u_{2d}s_1^2 - u_{1d}s_2^2, \; 0]^T$. Applying Eq.10 into the point-to-line constraint, Eq.6, one has:

$$\left(M_{k12}^T \otimes (\hat{l}_{12} + \lambda e_{12})^T \right) vec(P) = 0 \tag{11}$$

which can be rewritten as:

$$(B_{ki1} + \lambda B_{ki2}) \; vec(P) = 0 \tag{12}$$

where $B_{ki1} = M_{k12}^T \otimes \hat{l}_{12}^T$, $B_{ki2} = M_{k12}^T \otimes e_{12}^T$ and M_{k12} denotes the k^{th} 3D point projecting to the distorted line l_{12}. Considering $N \geq 12$ pairs $(M_{ki}, \hat{l}_i)$, where $N = k_{max} i_{max}$, one forms two $N \times 12$ matrices, B_1 and B_2, by stacking matrices B_{ki1} and B_{ki2}.

Using once more Fitzgibbon's suggestion [4], left-multiplying the stacked matrices by B_1^T results in a Polynomial Eigenvalue Problem (PEP):

$$(B_1^T B_1 + \lambda B_1^T B_2) \; vec(P) = 0 \tag{13}$$

which can be solved for example in Matlab using the `polyeig` function. Its solution gives simultaneously the projection matrix, $vec(P)$, the radial distortion parameter λ, and $P' = T^{-1}P$, where T is defined in sec II.C. In a similar way as explained before, both DLT methods applied to the radial distorted camera, can be combined to estimate P and λ.

3.3 Solving the Polynomial Eigenvalue Problem

Considering the matrix polynomial (or λ-matrix) of degree m

$$P(A,\lambda) = A_0 + \lambda A_1 + \cdots + \lambda^m A_m = \sum_{k=0}^{m} \lambda^k A_k, \tag{14}$$

where $A_k \in \mathbb{C}^{n\times n}$ for $k = 0 : m$, and $A = (A_0, A_1, \ldots, A_m) \in \mathbb{C}^{n\times n\times(m+1)}$, the Polynomial Eigenvalue Problem consists of finding the nm pairs of eigenvalues λ and the corresponding eigenvectors x satisfying $P(A,\lambda)x = 0$.

The Polynomial Eigenvalue Problem (PEP) extends the well-known Standard Eigenvalue Problem (SEP) and the Generalized Eigenvalue Problem (GEP). In the case of the SEP, $m = 1$ and $A_1 = -I_n$, $P(A,\lambda) = A_0 - \lambda I_n$ where I_n denotes the $n \times n$ identity matrix. In the case of the GEP, $m = 1$ and A_1 is not constrained to be the identity matrix, $P(A,\lambda) = A_0 + \lambda A_1$. Another important case of PEP is the Quadratic Eigenvalue Problem (QEP) which can be obtained with $m = 2$, $P(A,\lambda) = A_0 + \lambda A_1 + \lambda^2 A_2$.

The standard methodology for solving PEPs consists of three steps. In the first step, the PEP is converted into the GEP form through a process called *linearization* [5, 6, 7]. Let the $n \times n$ matrices A_k be combined to form larger, $nm \times nm$, matrices A and B

$$A = \begin{bmatrix} A_0 & 0_n & \cdots & 0_n \\ 0_n & I_n & \ddots & \vdots \\ \vdots & & \ddots & 0_n \\ 0_n & \cdots & \cdots & I_n \end{bmatrix},\ B = \begin{bmatrix} -A_1 & -A_2 & \cdots & -A_m \\ I_n & 0_n & \cdots & 0_n \\ 0_n & \ddots & \ddots & \vdots \\ 0_n & \cdots & I_n & 0_n \end{bmatrix} \tag{15}$$

where I_n and 0_n represent the $n \times n$ identity and zeros matrices. As proposed by [8], there is a decomposition of the matrix polynomial $P(A,\lambda)$ such that

$$\begin{bmatrix} P(A,\lambda) & 0 \\ 0 & I_{n(m-1)} \end{bmatrix} = E(\lambda)(A - \lambda B)F(\lambda), \tag{16}$$

where $E(\lambda)$ and $F(\lambda)$ are unimodular matrices. The left side of Eq.16 defines a block diagonal matrix, where the elements are the matrix polynomial $P(A,\lambda)$ and $m-1$ identity matrices, $n \times n$ each. The eigenvalues and eigenvectors of the block matrix contain the PEP solution.

Equation 16 allows to represent the PEP in terms of the matrices pair (A,B) (Eq.15), $E(\lambda)$ and $F(\lambda)$. Left-multiplying the right half of the equation by $E(\lambda)^{-1}$ leads to

$$[(A - \lambda B)F(\lambda)]x = 0 \Leftrightarrow (A - \lambda B)\hat{x} = 0, \tag{17}$$

where $\hat{x} = F(\lambda)x$. The PEP eigenvectors can be obtained from solving the generalized eigenvalue problem $(A - \lambda B)\hat{x} = 0$, and taking $x = F(\lambda)^{-1}\hat{x}$.

The second step of solving the PEP involves solving the GEP just defined. The GEP can be solved, for example, using the QZ algorithm [9]. For a given matrix pair $(A,B) \in M_n(\mathbb{C})^2$ in Eq.17, this algorithm computes the *generalized Schur decomposition* [10] in which the matrices Q and Z are unitary and S, T are upper triangular, such that $A = QSZ^T$ and $B = QTZ^T$. The upper triangular matrices S and T are known as the *Schur form* of A and B, respectively. Applying these matrices in Eq.17 leads to

$$(QSZ^T - \lambda QTZ^T)\hat{x} = 0 \Leftrightarrow (S - \lambda T)y = 0, \tag{18}$$

where $y = Z^T\hat{x}$ are the eigenvectors of the GEP defined by $(S - \lambda T)$. As noted in [9], the GEP eigenvalues can be obtained as a ratio between the diagonal elements $\lambda = diag(T)/diag(S)$, which

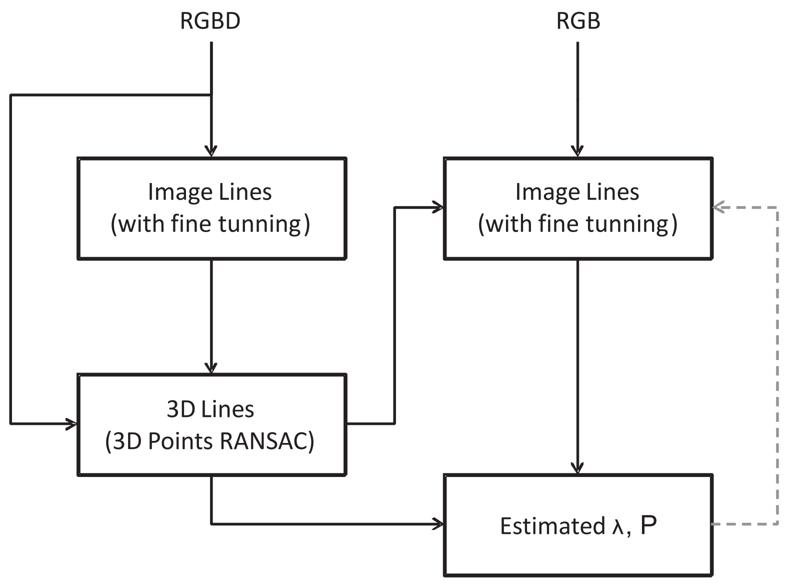

RGBD
RGB
Image Lines (with fine tunning)
Image Lines (with fine tunning)
3D Lines (3D Points RANSAC)
Estimated λ, P

Fig. 6: Camera calibration methodology, using Color-Depth (RGBD) camera and *DLT-Lines*.

contains the PEP eigenvalues as noted earlier. The eigenvalues in Eq.18 allow computing the eigenvectors y as the null space of $(S - \lambda T)$.

The third step of solving the PEP, consists of determining the PEP eigenvectors from the eigenvectors y, considering the variables changes made in the first two steps,

$$x = Z\, F(\lambda)^{-1}\, y. \tag{19}$$

3.4 Summary

In order to organize and summarize the aspects already described, we outline now the complete *DLT-Lines* calibration methodology (see Fig. 6). As input one has 2D lines in a RGBD image acquired by a calibrated camera[2], and 2D lines in a RGB image acquired by the camera to calibrate.

Each 2D line of the RGB image is described by a number of points. Usually one needs more than two points per line in order to identify the radial distortion.

- Step 1 of the methodology consists of estimating λ, P and finally P' (using equations 10 till 12).
- Step 2 consists of a local fine tuning of the lines in the RGB image. The image lines are composed by a number of parts (as described by the original number of 2D points). For each of the parts of a line, an optimization process is run using a Levenberg-Marquardt algorithm.

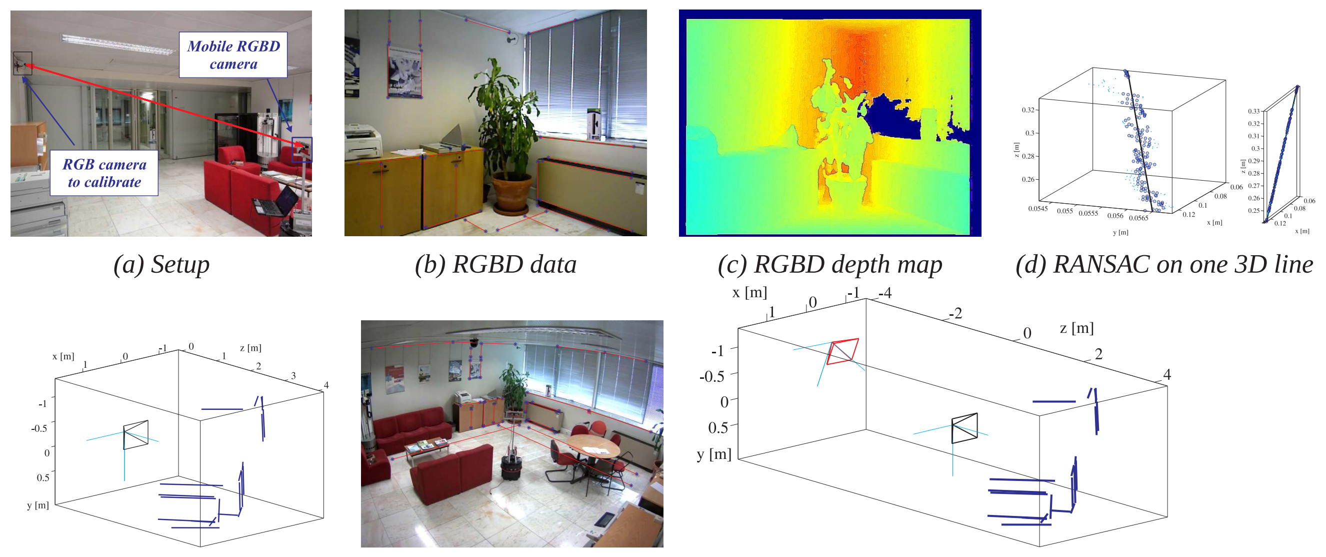
Steps 1 and 2 are repeated until the fine tuning of the lines does not change significantly the lines. The optimization process is bootstrapped with an approximation of the principal point obtained by factorizing the current estimate of the projection matrix.

4 *DLT-Lines* Calibration Experiment

In this experiment, the objective is to calibrate an Axis P1347 high definition surveillance camera (RGB), with radial distortion, installed on a waiting room. A ASUS X-Tion (RGBD) camera, mounted on mobile platform, is used to capture 3D scene information. The RGBD camera is assumed to be calibrated.

Figures 7(b) and (f) show the lines identified on the RGBD and RGB cameras. The noise in the depth map, Fig. 7(c), implies noise in the 3D lines which can be attenuated using RANSAC (see Fig. 7(d)). The data in Figs.7(e) and (f), allow applying *DLT-Lines* and obtaining the results shown in Fig.7(g). As in the

[2] Alternatively, one can have simply 3D points describing 3D lines (two points per line).



(a) Setup *(b) RGBD data* *(c) RGBD depth map* *(d) RANSAC on one 3D line*

(e) RGBD lines and cam. *(f) RGB data* *(g) Result, RGB cam. in red*

Fig. 7: Calibration of a surveillance camera, Axis P1347 (RGB), using a mobile robot equipped with a color-depth camera, Asus X-Tion (RGBD) (a). Lines in the RGBD image (b,c) define 3D lines (d,e). Each line formed directly from the depth map (cyan dots) is filtered using RANSAC (blue and black dots), as shown in (d), where the left/right plot has different/equal scales in the axis. RGBD (e) and RGB lines (f), form the input dataset for *DLT-Lines*. Decomposing the estimated projection matrix as $K[R\,t]$, provides the camera position and orientation on the world coordinate system (g).

simulated setup, a qualitative assessment of the precision of the calibration can be made by transporting edges from the RGBD image to the RGB image. The RGB image, Fig. 7(f), has been cropped to show just the area covered by the RGBD camera.

In order to obtain quantitative assessment, some more tests have been conducted. In particular J. Y. Bouguet's calibration toolbox [1] was used to estimate the intrinsic parameters of the RGB camera. The difference between the (horizontal) focal length obtained using Bouguet's toolbox and the one extracted from the estimated projection matrix using *DLT-Lines*, was found to be $K_{err} = 0.05$. The real distance from RGBD to RGB was $d = 3.55[m]$, measured with a tape, while the estimated using *DLT-Lines* was found to be $d_e = 3.53[m]$. The small relative errors assert that *DLT-Lines* can provide accurate results.

References

[1] J. Bouguet. Camera calibration toolbox for Matlab. `http://www.vision.caltech.edu/bouguetj`.

[2] A. Ortega, B. Dias, E. Teniente, A. Bernardino, J. Gaspar, and J. Andrade-Cetto. Calibrating an outdoor distributed camera network using laser range finder data. In *IROS*, pages 303–308, 2009.

[3] R. Hartley and A. Zisserman. *Multiple View Geometry in Computer Vision*. Cambridge University Press, 2000.

[4] A. Fitzgibbon. Simultaneous linear estimation of multiple view geometry and lens distortion. In *Proc. IEEE Conf. Comp. Vision and Pattern Recognition.*, volume 1, pages 125–132, 2001.

[5] F. Tisseur. Backward error and condition of polynomial eigenvalue problems. *Linear Algebra Appl*, 309:339–361, 1999.

[6] D. Mackey, N. Mackey, C. Mehl, and Volker. Vector spaces of linearizations for matrix polynomials. *SIAM J. Matrix Anal. Appl*, 28:971–1004, 2005.

[7] N. Higham, D. Mackey, and F. Tisseur. The conditioning of linearizations of matrix polynomials. *SIAM J. MATRIX ANAL. APPL*, 28(4):1005–1028, 2005.

[8] M. Berhanu. *The Polynomial Eigenvalue Problem*. Ph.d. thesis, University of Manchester-School of Mathematics, 2005.

[9] C. Moler and G. Stewart. An algorithm for generalized matrix eigenvalue problems. *SIAM J. Numer. Anal.*, 10(2):241–256, 1973.

[10] H. Lütkepohl. *Handbook of Matrices*. Wiley and Sons, 1996.