Calibrating an Overhead Video Camera

Raul Rojas

Freie Universität Berlin, Takustraße 9, 14195 Berlin, Germany
`http://www.fu-fighters.de`

Abstract. In this section we discuss how to calibrate an overhead video camera used for tracking robots moving on a rectangular field. First, the radial distortion of the camera must be eliminated. The resulting corrected image represents a projective transformation of the field's surface to the camera's imaging plane. We show how to find the projective transformation, and how to recover from this information the position of the camera and the rotation of its coordinate system in relation to world's coordinates. We discuss the numerical errors which arise in the computations and how to find an optimal solution. We also show how to track robots of different heights.

1 Overhead Camera calibration

In the RoboCup small-size league, mobile robots are tracked using one or more video cameras overlooking the field from a height of three to four meters. The image registered in each camera provides pixel information from which the positions of robots and other objects in the field can be computed. The robots are marked with blobs of different colors. The geometry of the scene provides all necessary information for tracking the robots, sometimes at 60 frames per second.

Tracking the robots can be done best when the image from the video camera is as undistorted as possible. However, once the video camera has been placed above the field, there is no guarantee that the symmetry axis of the optics will be pointing downwards perfectly. The camera's coordinate system axis could be misaligned with the coordinate system of the field (which we call "world coordinates"). Also, wide angle lenses, necessary for capturing the field from only 3 or 4 meters height, introduce a significant amount of radial distortion in the image. The distance of objects to the center of the coordinate system is underestimated sometimes by as much as 30% to 40%. It is necessary to correct all these imaging artifacts before the coordinates of objects on the field can be recovered.

This chapter provides all the necessary information for computing the best camera-to-field coordinates transformation (so that we can infer from pixel data where our robots are located), as well as the inverse field-to-camera transformation. We proceed as follows:

– We eliminate the radial distortion, calibrating the camera before it is used, and applying a corrective transformation to the pixel's coordinates,

- We find the projective transformation involved in the field-to-camera mapping and show how to recover the field coordinates of the robots using the camera-to-field map.
- We show how to recover the coordinates of robots of different heights.

We also discuss some numerical problems and how to improve the results.

2 Radial distortion

Radial distortion is introduced by the camera optics and can be of pin-cushion type, barrel type, or even a combination of both. In the pin-cushion distortion, the distance of objects to the center of the image overestimates the real distance in the world coordinate system,; in the second, the real distance is underestimated. Both types of distortions are nonlinear, that is, the image of a rectangle in world's coordinates does not look like a rectangle in the image. Straight lines are transformed by the lens into curved lines when radial distortion is present.

Radial distortion is a property of the camera optics. Most of the simple lenses in use today (and many cheap video cameras use only those simple lenses) are spherical. Spherical lenses are easy to grind and have been used for centuries. In spherical lenses, an image is properly transformed (i.e. without distortion) only when its maximum distance from the optical center is small (this means that the angle of incidence of light rays is small). However, when the distance from the center is large, the mapping object-to-image is not perfect anymore. Fig. **??** shows examples of radial distortion of a checkerboard pattern.

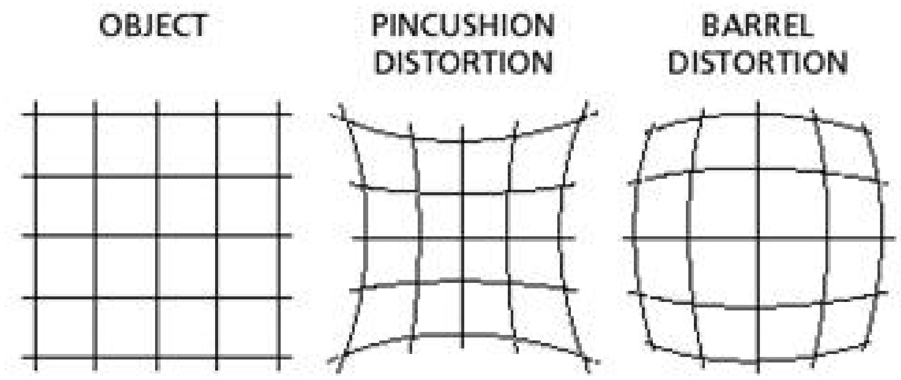

Fig. 1. Examples of barrel and pin-cushion distortion.

Barrel distortion arises when an iris is placed in front of the lens. If the iris is behind the lens, pin-cushion distortion is produced. A pin-hole camera, with an ideal point-like iris, does not need a lens and does not produce any distortion. Many of the computations which follow have been made for a pinhole camera and are not exact for cameras with different iris apertures. This should be kept in mind, before we discuss the numerical errors in the computations. Fig. reffeld shows the distorted image of the 2004 field in our lab, using a 4.2 mm lens and with the camera at about 2.50 m above the field.

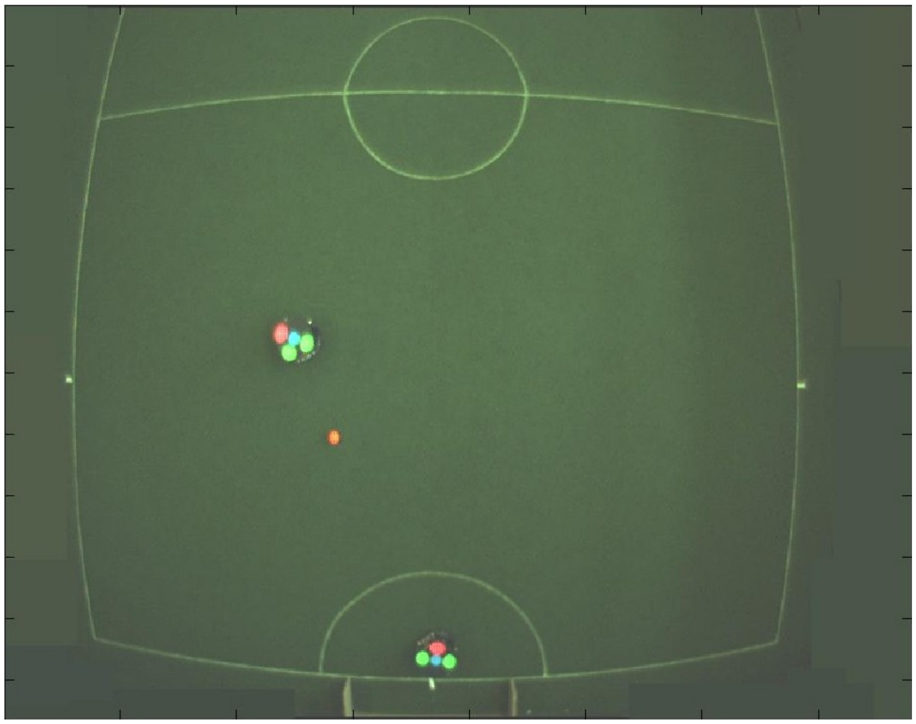

Fig. 2. Barrel distortion of the field with a 4.2 mm lens

The radial distortion of a lens can be assessed by taking a picture of concentric circles centered at the origin. Fig. 2 shows concentric circles, each one of them with a diameter larger by 1 cm than the diameter of the preceding circle (i.e., each successive ring is 1 cm wide). As Fig.2 shows, the image is compressed towards the periphery – the circles seem to shrink more towards the origin the farther away its boundary is.

From this picture it is relatively straightforward to compute the radial distortion and to correct the image pixels. Since the lenses in the camera are radially symmetric, the distortion is the same for any pixel at a distance r from the image center. In barrel distortion the measured distance r is smaller than the real distance r', and the distortion is $d_r = r'/r$. It follows that the coordinates of a pixel (p_x, p_y) in the image can be corrected by just stretching the image by

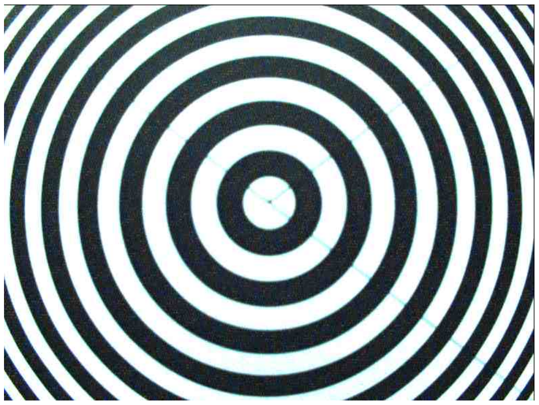

Fig. 3. Barrel distortion produced by a 4.2 mm lens. Each ring is of the same width but the camera distorts the pattern.

the factor d_r, where $r = \sqrt{(p_x^2 + p_y^2)}$, that is

$$(p_x, p_y) \mapsto (d_r p_x, d_r p_y)$$

Of course a table of factors d_r must be precomputed, or better yet, a function relating r' to r can be fitted using an image of known geometry.

Fig. 3 shows the radial distances of the circles borders extracted by a computer program from Fig.2. plotted against the distances we should have obtained without radial distortion. The undistorted pixels distances are obtained by just taking multiples of the distance between the center of the image and the first inner circle's border. As the plot shows, a measured distance of 450 pixels from the origin corresponds to a distance of around 700 pixels without radial distortion. The measured distance is therefore just 64% of the real distance.

Correcting the distortion can be done by computing the inverse of the distortion function. A polynomial can be used for this purpose. Let us assume that the correction factor d_r is a polynomial of the variable r, the measured distance of a pixel to the origin. We can use polynomials up to order 3. Then

$$d_r = a_1 r + a_2 r^2 + a_3 r^3.$$

Notice that there is no constant factor because there is no radial distortion at the origin.

Fig. 4 shows the measured distorted points, plotted now in the x axis, and the undistorted points in the y axis. A polynomial has been fitted to this data (dots in the image). The coefficients, computed to double precision with Matlab, are the following:

$$a_1 = 1.07761103142178$$
$$a_2 = -0.00096240306505$$
$$a_3 = 0.00000401304361$$

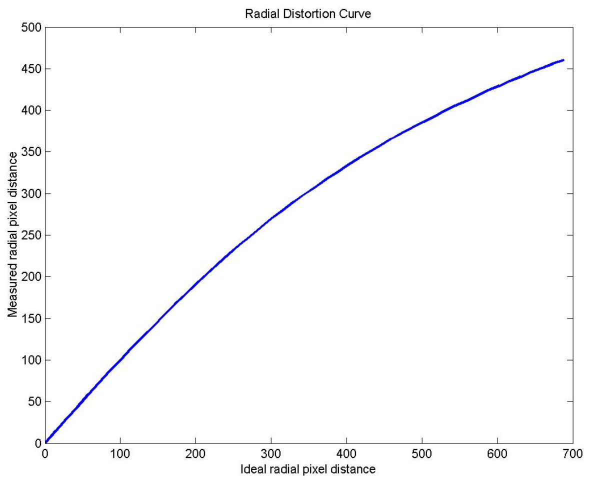

Fig. 4. Radial pixel distortion produced by a 4.2 mm lens

Note that the large number of decimal places is needed because squaring pixel coordinates produces very large values for r. Small numerical variations in the coefficients a_2 and a_3 lead to very different results.

It is important to point out that since barrel distortion depends on the aperture of the camera's iris, the correction polynomial must be computed for exactly the iris aperture later used for tracking the robots. If the iris aperture is changed, the coefficients of the correction polynomial must change. If the iris leaves only a very small opening, barrel distortion will be reduced (because the camera approaches the ideal situation of a pin-hole camera). If the iris is more open, barrel distortion can increase. Fig. 5 shows the corrected image of the field, after computing and eliminating the radial distortion with the camera settings used in our lab.

3 Projective transformations

After radial distortion has been eliminated, the resulting image is still far from perfect. The rectangular field appears as a trapezoid, due to the imperfect centering and alignment of the camera with respect to the field. In Fig. **??** we simply connected with four lines the four corners of the field. The figure shows that there is a certain residual error from the radial correction, and that the perspective of the camera maps the rectangular field to a four sided polygon with sides of different sizes. This is just a projective transformation of the field to the camera imaging chip.

A projective transformation can be best explained with a pinhole camera model. If a pinhole is located at the origin of the coordinate system, any line

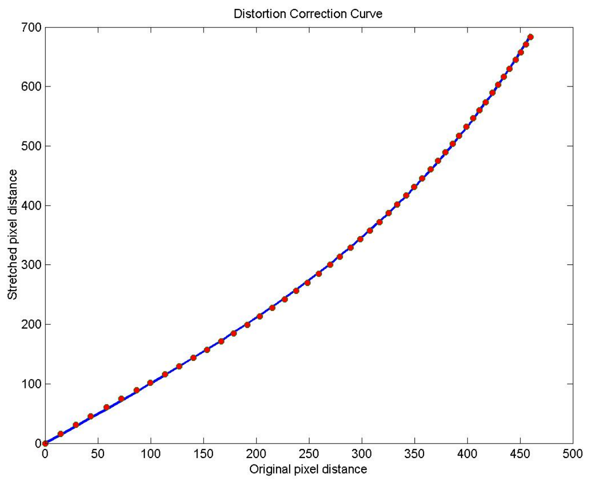

Fig. 5. Radial distortion correction for a 4.2 mm lens

through the origin intersects the projection plane of the camera (which we take as the plane $z = 1$). A point with coordinates (x, y, z) and $z \neq 0$ is projected to the point $(x/z, y/z, 1)$. See Fig. 7.

The main problem for performing the projective transformation is to express all coordinates of points to be projected in the coordinate system of the camera. Assume that the camera is positioned at $\mathbf{t} = (c_x, c_y, c_z)$. Assume that the camera has its own coordinate system (we use primed variables to denote coordinates x', y', z' relative to this camera system), which can be rotated with respect to the world coordinate system (Fig.8. Let R be the 3D rotation matrix which describes the rotation of the camera's coordinate system relative to the directions of the axis in the world coordinates .

We can find the relationship between the world coordinates of a point and its mapping in the camera imaging plane at $z' = 1$ as described next.

First, let us call the projective transformation of a point with coordinates (x, y, z) to the point $(x/z, y/z, 1)$, in any coordinate system, the function f_1, that is

$$f_1((x, y, z)^{\mathrm{t}}) = (x/z, y/z, 1)^{\mathrm{t}}$$

A point (x, y, z) in world coordinates has other coordinates with respect to the camera's system. The change of coordinates under the rotation $\mathbf{R}$ and the translation $\mathbf{t}$ can be written as

$$(x', y', z')^{\mathrm{t}} = \mathbf{R}((x, y, z)^{\mathrm{t}} - \mathbf{t}) = \mathbf{R}(x, y, z)^{\mathrm{t}} - \mathbf{R}\mathbf{t}$$

If all points we are mapping are on the floor, that is, if they have coordinate $z = 0$, then transformation which describes the change of basis reduces to

$$(x', y', z')^{\mathrm{t}} = (\mathbf{r}_1 \quad \mathbf{r}_2 \quad \mathbf{t}')(x, y, 1)^{\mathrm{t}}$$

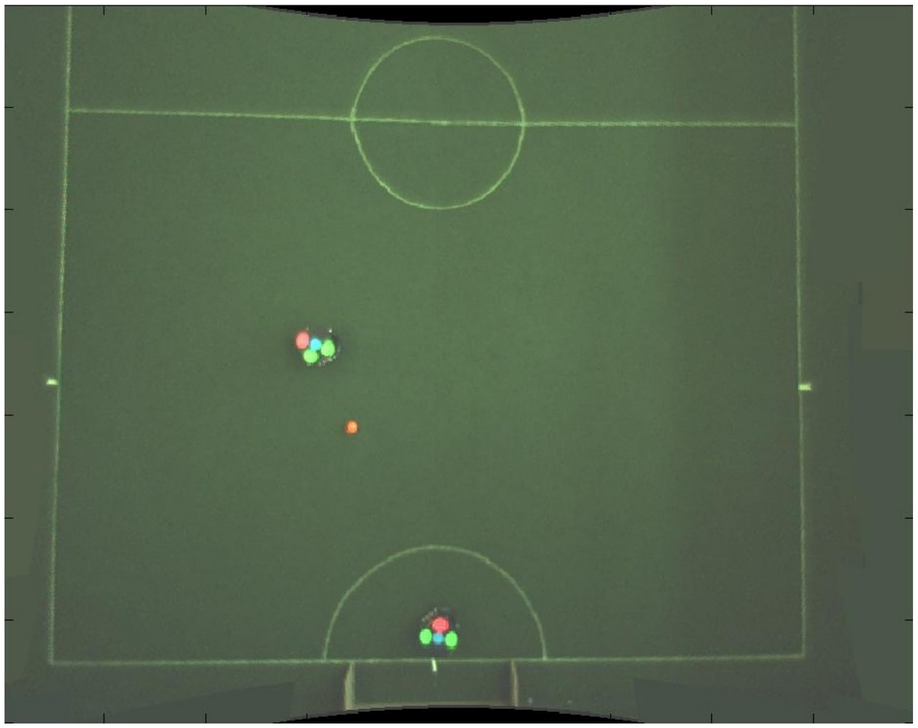

Fig. 6. Corrected field image

where $\mathbf{t}' = -\mathbf{R}\mathbf{t}$ and $\mathbf{r}_1$, $\mathbf{r}_2$ are the first and second columns of the rotation matrix $\mathbf{R}$.

Composing the projective transformation with the change of coordinates we have

$$(x'', y'', 1)^{\mathrm{t}} = f_1((x', y', z')^{\mathrm{t}}) = f_1((\mathbf{r}_1 \quad \mathbf{r}_2 \quad \mathbf{t}')(x, y, 1)^{\mathrm{t}})$$

We denote the matrix $(\mathbf{r}_1 \quad \mathbf{r}_2 \quad \mathbf{t}')$ by $\mathbf{H}$ and we can write simply

$$(x'', y'', 1)^{\mathrm{t}} = f_1(\mathbf{H}(x, y, 1)^{\mathrm{t}})$$

3.1 Mapping from image to field

Until now, we have explored how to transform points on the field to points on the camera projection plane. Actually, what we really need when we are tracking robots, is a transformation from the camera plane to field coordinates. We see the robot, we want to know its coordinates on the field.

Given the camera coordinates of a point (x'', y'') we want to compute the field coordinates (x, y). The mapping we need is just the inverse of the mapping H (up to an scaling factor). Let us call the mapping from camera coordinates to

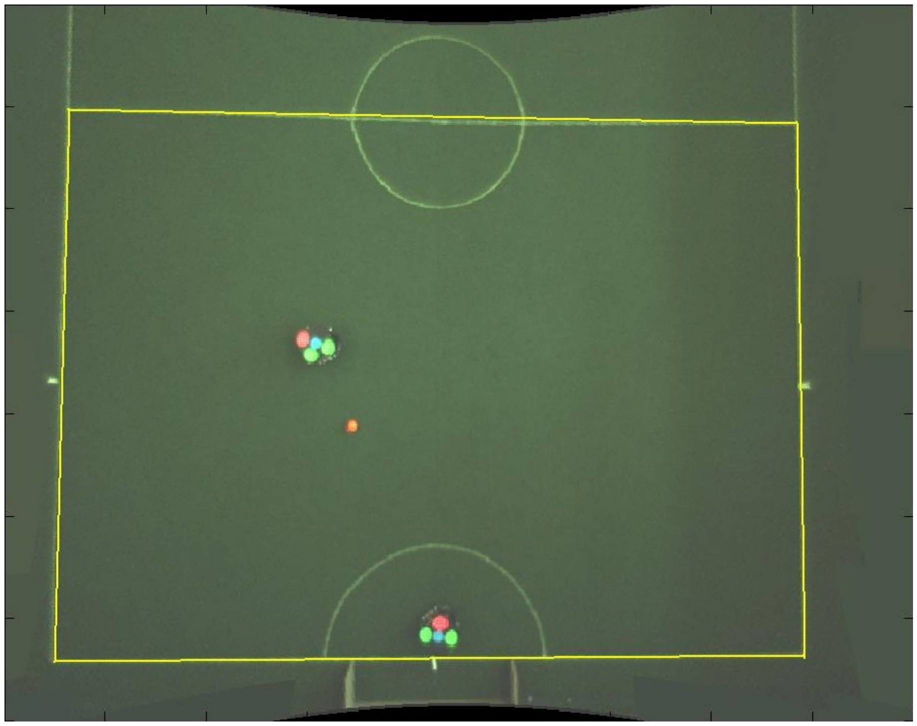

Fig. 7. Corrected field image

field coordinates the matrix $\mathbf{G}$. We want this matrix to satisfy the mapping

$$(x, y, 1)^{\mathrm{t}} = \mathbf{G}(x'', y'', 1)^{\mathrm{t}}$$

where $(x'', y'', 1)$ are the homogeneous coordinates of the image point. Since we know that

$$\mathbf{H}(x, y, 1)^{\mathrm{t}} = (x', y', z')^{\mathrm{t}}$$

and since $x'' = x'/z'$ and $y'' = y'/z'$, it follows that

$$\mathbf{H}(x, y, 1)^{\mathrm{t}} = z'(x', y', 1)^{\mathrm{t}}$$

for a certain z'. It is then obvious that $\mathbf{G} = c\mathbf{H}^{-1}$, where c is a proportionality constant.

Therefore, it is not necessary to compute the matrix $\mathbf{G}$ once the matrix $\mathbf{H}$ has been found. Any multiple of the inverse of $\mathbf{H}$ can be used. Since the matrix $\mathbf{G}$ maps points in the camera to points on the field, projectively, any point of the form $(\rho x, \rho y, \rho)^{\mathrm{t}}$ is mapped to the same point $(x, y, 1)^{\mathrm{t}}$.

3.2 Recovering the transformation matrix

Given four points in the projection plane of a pin-hole camera which correspond to four points in the projected plane of the field, it is possible to find the cor-

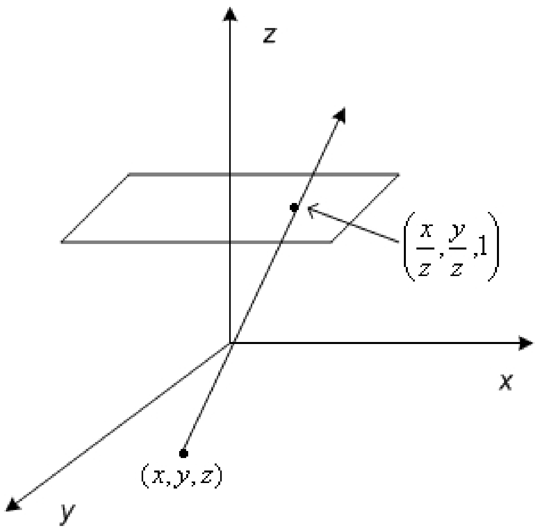

Fig. 8. Projective transformation of three dimensional space to the plane $z = 1$

responding transformation matrix $\mathbf{H}$, and from it we can recover the rotation matrix $\mathbf{R}$ and the translation vector $\mathbf{t}$.

Assume that four points on the field, with coordinates $(x_0, y_0), (x_1, y_1), (x_2, y_2), (x_3, y_3)$ are projectively mapped to four other points $(x_0'', y_0'', 1), (x_1'', y_1'', 1), (x_2'', y_2'', 1), (x_3'', y_3'', 1)$ on the plane $z'' = 1$, with respect to the pin-hole camera coordinate system. We know from the equation above that

$$\begin{pmatrix} x'' \\ y'' \\ 1 \end{pmatrix} = f_1(\mathbf{H}(x, y, 1)^{\mathrm{t}}) = f_1 \left[\begin{pmatrix} h_{11} & h_{12} & h_{13} \\ h_{21} & h_{22} & h_{23} \\ h_{31} & h_{32} & h_{33} \end{pmatrix} \begin{pmatrix} x \\ y \\ 1 \end{pmatrix} \right]$$

Each one of the transformed points yields two linear equations involving the unknown elements of $\mathbf{H}$. The three equations are

$$x' = \begin{pmatrix} x \; y \; 1 \; 0 \; 0 \; 0 \; 0 \; 0 \; 0 \end{pmatrix} \mathbf{h} \tag{1}$$
$$y' = \begin{pmatrix} 0 \; 0 \; 0 \; x \; y \; 1 \; 0 \; 0 \; 0 \end{pmatrix} \mathbf{h} \tag{2}$$
$$z' = \begin{pmatrix} 0 \; 0 \; 0 \; 0 \; 0 \; 0 \; x \; y \; 1 \end{pmatrix} \mathbf{h} \tag{3}$$

where $\mathbf{h}$ denotes the vector $(h_{11}, h_{12}, h_{13}, h_{21}, h_{22}, h_{23}, h_{31}, h_{32}, h_{33})^{\mathrm{t}}$. Since we want to perform the projective transformation f_1, we know that

$$x'' = x'/z' \quad \text{and} \quad y'' = y'/z'$$

therefore

$$x''z' = x' \quad \text{and} \quad y''z' = y'$$

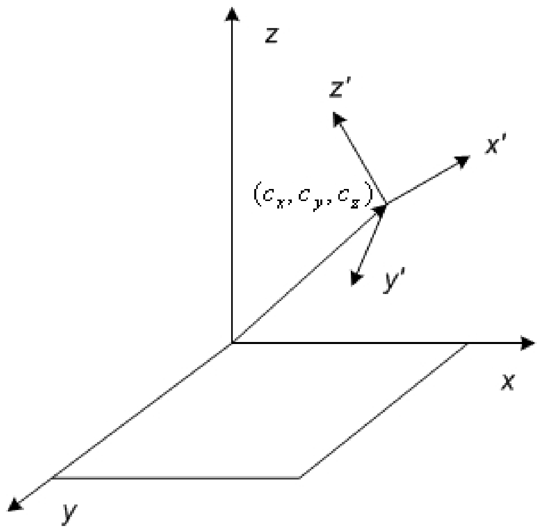

Fig. 9. The world's coordinate system and the camera's coordinate system. The camera is positioned at the point (c_x, c_y, c_z) in world coordinates.

From this we deduce that

$$\begin{aligned} x'' \begin{pmatrix} 0\,0\,0\,0\,0\,0\,x\;y\;1 \end{pmatrix} \mathbf{h} &= \begin{pmatrix} x\;y\;1\,0\,0\,0\,0\,0\,0 \end{pmatrix} \mathbf{h} \\ y'' \begin{pmatrix} 0\,0\,0\,0\,0\,0\,x\;y\;1 \end{pmatrix} \mathbf{h} &= \begin{pmatrix} 0\,0\,0\,x\;y\;1\,0\,0\,0 \end{pmatrix} \mathbf{h} \end{aligned}$$

which we can rewrite as

$$\begin{aligned} \begin{pmatrix} x\;y\;1\,0\,0\,0\;-x''x\;-x''y\;-x'' \end{pmatrix} \mathbf{h} &= 0 \\ \begin{pmatrix} 0\,0\,0\,x\;y\;1\;-y''x\;-y''y\;-y'' \end{pmatrix} \mathbf{h} &= 0 \end{aligned}$$

For the four points given above, we obtain the following eight equations:

$$\begin{pmatrix} x_0 & y_0 & 1 & 0 & 0 & 0 & -x_0''x_0 & -x_0''y_0 & -x_0'' \\ 0 & 0 & 0 & x_0 & y_0 & 1 & -y_0''x_0 & -y_0''y_0 & -y_0'' \\ x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1''x_1 & -x_1''y_1 & -x_1'' \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -y_1''x_1 & -y_1''y_1 & -y_1'' \\ x_2 & y_2 & 1 & 0 & 0 & 0 & -x_2''x_2 & -x_2''y_2 & -x_2'' \\ 0 & 0 & 0 & x_2 & y_2 & 1 & -y_2''x_2 & -y_2''y_2 & -y_2'' \\ x_3 & y_3 & 1 & 0 & 0 & 0 & -x_3''x_3 & -x_3''y_3 & -x_3'' \\ 0 & 0 & 0 & x_3 & y_3 & 1 & -y_3''x_3 & -y_3''y_3 & -y_3'' \end{pmatrix} \mathbf{h} = 0$$

The system of equations can be solved setting $h_{33} = 1$, then the last column of the above matrix can be moved as a vector to the right of the equal sign, and the first eight linear equations can be solved to find the value of the rest of the elements. Note that any multiple of the vector $\mathbf{h}$ is a solution to the original system of equations. The solution we compute is correct up to a scaling factor λ.

3.3 Recovering the rotation matrix

Given four points in the image and their known coordinates in the world, the matrix $\mathbf{H}$ can be recovered, up to a scaling factor λ. We know that the first two columns of the rotation matrix $\mathbf{R}$ must be the first two columns of the transformation matrix. Let us denote by $\mathbf{h}_1$, $\mathbf{h}_2$, and $\mathbf{h}_3$ the three columns of the matrix $\mathbf{H}$.Due to the scaling factor λ we then have that

$$\lambda \mathbf{r}_1 = \mathbf{h}_1$$

and

$$\lambda \mathbf{r}_2 = \mathbf{h}_2$$

Since $|\mathbf{r}_1| = 1$, then $\lambda = |\mathbf{h}_1|/|\mathbf{r}_1| = |\mathbf{h}_1|$ and $\lambda = |\mathbf{h}_2|/|\mathbf{r}_2| = |\mathbf{h}_2|$. We can thus compute the factor λ and eliminate it from the recovered matrix $\mathbf{H}$. We just set

$$\mathbf{H}' = \mathbf{H}/\lambda$$

In this way we recover the first two columns of the rotation matrix $\mathbf{R}$.

The third column of $\mathbf{R}$ can be found remembering that any column in a rotation matrix is the cross product of the other two columns (times the appropriate plus or minus sign). In particular

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

Therefore, we can recover from $\mathbf{H}$ the rotation matrix $\mathbf{R}$. We can also recover the translation vector (the position of the camera in field coordinates). Just remember that

$$\mathbf{h}'_3 = -\mathbf{R}\mathbf{t}$$

Therefore the position vector of the camera pin-hole $\mathbf{t}$ is given by

$$\mathbf{t} = -\mathbf{R}^{-1}\mathbf{h}'_3$$

3.4 Mapping into the camera chip

The analysis above is correct when the camera captures the image at the plane $z = 1$ and when the camera uses the same image units as those used in the world coordinate system. However: a) the camera projects the image to a chip located at the position $z = f$ from the pin-hole, and b) the image is addressed using pixels and not meters .

The correction needed is small. A point $(x'', y'', 1)$ in the camera coordinate system is projected to the plane $z'' = f$, just multiplying the point coordinates with f, that is,

$$(x'', y'', 1)^{\mathrm{t}} \mapsto (fx'', fy'', f)^{\mathrm{t}}$$

If the correspondence between meters and pixels is given by the factor δ, and if the pixels are quadratic, then the pixel coordinates of the transformed point are

$$(p''_x, p''_y)^{\mathrm{t}} = (\delta f x'', \delta f y'')^{\mathrm{t}}$$

We call the factor δf just ϕ. Then

$$(x'', y'')^{\mathrm{t}} \mapsto (\phi x'', \phi y'')^{\mathrm{t}}$$

Define the camera matrix $\mathbf{K}$ as

$$K = \begin{pmatrix} \phi & 0 & 0 \\ 0 & \phi & 0 \\ 0 & 0 & 1 \end{pmatrix}$$

then the complete world coordinates to image projective projection is given by

$$\begin{pmatrix} p''_x \\ p''_y \\ 1 \end{pmatrix} = \mathbf{K} f_1(\mathbf{H}(x, y, 1)^{\mathrm{t}})$$

We can apply the method deduced in the previous section just by pre-transforming pixel coordinates into meter coordinates:

$$\begin{pmatrix} x'' \\ y'' \\ 1 \end{pmatrix} = \mathbf{K}^{-1} \begin{pmatrix} p''_x \\ p''_y \\ 1 \end{pmatrix} = f_1(\mathbf{H}(x, y, 1)^{\mathrm{t}})$$

When tracking the robots we want to go from pixel coordinates to field coordinates. We can just operate on the equation above, inverting the function f_1. This function has not really an inverse, because a projective transformation maps a whole line to a point. But it is true that if $(u, v, 1)^{\mathrm{t}} = f_1((x, y, z)^{\mathrm{t}})$ then $k(u, v, 1)^{\mathrm{t}} = (x, y, z)^{\mathrm{t}}$ for some constant k. In our case we can write

$$k\mathbf{K}^{-1} \begin{pmatrix} p''_x \\ p''_y \\ 1 \end{pmatrix} = \mathbf{H}(x, y, 1)^{\mathrm{t}}$$

and therefore

$$k\mathbf{H}^{-1}\mathbf{K}^{-1} \begin{pmatrix} p''_x \\ p''_y \\ 1 \end{pmatrix} = (x, y, 1)^{\mathrm{t}}$$

Applying f_1 again we obtain

$$f_1 \left(\mathbf{H}^{-1}\mathbf{K}^{-1} \begin{pmatrix} p''_x \\ p''_y \\ 1 \end{pmatrix} \right) = (x, y, 1)^{\mathrm{t}}$$

The mapping from pixel coordinates to field coordinates is complete.

3.5 Numerical errors

In the previous sections we have learned that having the correspondence of four points on the field and four points on the image allows us to recompute the transformation matrix, and from it, the rotation matrix **R** and the position of the camera's pinhole. The four points must be independent (that is, they should be in general position). No one of them should be on a line connecting other two. Otherwise the linear equations are not independent and we cannot solve the system in order to find **H**.

A straightforward application of these ideas to real images produces some puzzling numerical errors. The norm of the two first columns of **H** is not equal, as it should be. Remember that the norm of each column should be one, times a scaling factor λ. Therefore, two different scaling factors are possible, one computed from each column. The difference is very small for real images, but picking any one of the two alternatives leads to columns of the rotation matrix **R** which are inconsistent. The computation of the rotation angles, for example, can lead to imaginary solutions when one asks the computer to find arccos(1.01), for example.

There is an easy fix to this dilemma: just pick as λ the maximum of the norm of $\mathbf{h_1}$ and $\mathbf{h_2}$. In this way at least the angles computed from the reconstructed rotation matrix are real and not imaginary. The next section explores a better solution.

A bigger problem is the correct value of the constant ϕ. This value must be measured directly. Bars of known length can be observed with the camera at fixed and known distances from it. From this measurements it is possible to compute the correspondence meters to pixels, from the plane at $z = 1$. However, the measurement is error prone and this introduces a further source of numerical anomalies. Remember that ϕ is used at the beginning to transform from pixels to meters. Any numerical error in ϕ affects the whole computation.

3.6 Correcting the numerical errors

One improvement for the computation of λ and the rotation matrix is to find the best set of rotation angles and the best λ which can explain the computed transformation matrix.

We did the following. A 3D rotation matrix **R** has the following form

$$\mathbf{R} = \begin{pmatrix} \cos(\beta)\cos(\gamma) & \cos(\beta)\sin(\gamma) & -\sin(\beta) \\ \sin(\alpha)\sin(\beta)\cos(\gamma) - \sin(\gamma)\cos(\alpha) & \sin(\alpha)\sin(\beta)\sin(\gamma) + \cos(\alpha)\cos(\gamma) & \sin(\alpha)\cos(\beta) \\ \cos(\alpha)\sin(\beta)\cos(\gamma) + \sin(\alpha)\sin(\gamma) & \cos(\alpha)\sin(\beta)\sin(\gamma) - \sin(\alpha)\cos(\gamma) & \cos(\alpha)\cos(\beta) \end{pmatrix}$$

where α, β, γ are the rotation angles. We know that the first two columns of **H** are equal to the first two columns of **R**, times an unknown factor λ. We can proceed iteratively by assigning a value to the angles α, β and γ, and also to λ (for example 1.0 in this last case). From the values of the angles we obtain a rotation matrix **R**. We then compute the sum of the quadratic norm of differences

$$E = |\frac{\mathbf{h_1}}{\lambda} - \mathbf{r_1}|^2 + |\frac{\mathbf{h_1}}{\lambda} - \mathbf{r_1}|^2$$

If $E > 0$ then we compute the partial derivatives of E with respect to all four parameters α, β, γ, and λ. We correct the parameters using gradient descent and iterate again. We repeat this procedure until E is sufficiently small.

We performed this iterative computation for a real matrix captured by a camera in our lab. Fig. **??** shows the gradual reduction in the error. The calculation was not optimized for speed. A good initialization of the angles (using for example as λ the maximum of the norm of the two first columns of $\mathbf{H}$) greatly reduces the numbers of iterations.

Perhaps more relevant is the fact that we computed the height of our camera using the direct approach and then we did the same computation using the iterative procedure just described. With the first uncorrected method, the height is computed to be x meters, with the second it is y meters, a difference of x cm. Our own measurement gave a number very near to , within the limits of our measuring precision.

4 Mapping for a certain robot height

The above analysis was done based in the premise that we are mapping points on the floor to the camera chip. If we are mapping the covers of the robots to the field, and since the robots have a certain height, we must necessarily introduce a correction in the matrix $\mathbf{H}'$.

The needed correction is easy to compute noting that if the robot has height ℓ, the camera is then located at a height $c_z - \ell$ from the horizontal plane at the robots' cover. The rotation of the camera's coordinate system has not changed, and therefore the new matrix $\mathbf{H}'$ must have almost the same structure. The first two columns are still the first two columns of the rotation matrix $\mathbf{R}$, which has not changed. The last column $\mathbf{h_3'}$ is, according to what we showed above

$$\mathbf{h_3'} = \mathbf{R}(\mathbf{t} - (0,0,\ell)^{\mathrm{t}})$$

but then

$$\mathbf{h_3'} = \mathbf{h_3} - \mathbf{R}(0,0,\ell)^{\mathrm{t}})$$

and this is equal to

$$\mathbf{h_3'} = \mathbf{h_3} - \mathbf{r_3}\ell$$

The new matrix $\mathbf{H}'$ is just

$$\mathbf{H}' = \mathbf{H} - \mathbf{r_3}\ell$$

5 Experimental results

References

1.