Improvement of Camera calibration in Soccer field Using Simulated Annealing

Homa Fakour
Department of Computer Eng.
Mashhad Azad University
Mashhad, Iran
Homa_fakoor@yahoo.com

Abedin Vahedian
Department of Computer Eng.
Ferdowsi University
of Mashhad, Iran
vahedian@um.ac.ir

Hamidreza Pourreza
Department of Computer Eng.
Ferdowsi University
of Mashhad, Iran
hpourreza@um.ac.ir

***Abstract*— In this paper, we investigate camera calibration in soccer field using a field model defined according to FIFA standards. The proposed algorithm searches for homographic transform matrix for best correspondence between world coordinate and image coordinate systems. To find this matrix, some static markers are required in image and field. A good proposal for these markers is the intersections of lines in field and image. These points are transferred to Simulated Annealing algorithm then it searches for the best correspondence between image and field lines. In this work, we have tested several frames of soccer games and have obtained fairly good results.**

Keywords- camera calibration simulated annealing, Hough transform, ground detection.

I. Introduction

According to a lot of viewers around the world for soccer games, some researchers work on image processing for this purpose.
Camera calibration is a case that so many works is done on it. In this paper we focus on camera calibration for soccer field. In camera calibration, the geometry of 3D points in real world is obvious usually. If the information of image coordinates and it's corresponding world coordinates are available, the calibration is simple but because we don't have any accurate information firstly we would try to find these corresponding points to search for the best homography matrix.

II. Calibration system overview

A 2D point is denoted $m = [u, v]^T$. A 3D point is denoted by $M = [X, Y, Z]^T$. We use $\tilde{x}$ to denote the augmented vector by adding 1 as the last element: $\tilde{m} = [u, v, 1]^T$ and $\tilde{M} = [X, Y, Z, 1]^T$. A camera is modeled by the usual pinhole: the relationship between a 3D point M and its image projection m is given by

$$s\tilde{m} = A[R \quad t]\,\tilde{M} \qquad (1)$$

where s is an arbitrary scale factor, (R, t), called the extrinsic parameters, is the rotation and translation which relates the world coordinate system to the camera coordinate system, and A, called the camera intrinsic matrix, is given by:

$$A = \begin{bmatrix} \alpha & \gamma & u0 \\ 0 & \beta & v0 \\ 0 & 0 & 1 \end{bmatrix}$$

with ($u0$, $v0$) the coordinates of the principal point, α and β the scale factors in image u and v axes, and γ the parameter describing the skewness of the two image axes. We assume the model plane is on $Z = 0$ of the world coordinate system. Let's denote the ith column of the rotation matrix R by ri. From (1), we have

$$s\begin{bmatrix} u \\ v \\ 1 \end{bmatrix} = A[r_1 \quad r_2 \quad r_3 \quad t]\begin{bmatrix} X \\ Y \\ 0 \\ 1 \end{bmatrix} = A[r_1 \quad r_2 \quad t]\begin{bmatrix} X \\ Y \\ 1 \end{bmatrix}$$

By abuse of notation, we still use M to denote a point on the model plane, but $M = [X, Y]^T$ since Z is always equal to 0. In turn, $\tilde{M} = [X, Y, 1]^T$. Therefore, a model point M and its image m is related by a homography H:

$$s\tilde{m} = H\tilde{M} \quad ; H = A[r_1 \quad r_2 \quad t] \qquad (2)$$

As is clear, the 3 * 3 matrix H is defined up to a scale factor [2,4].

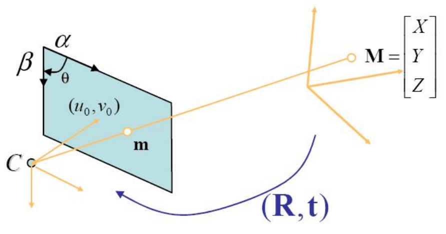


Figure.1. Pinhole camera model

In this paper we try to determine homography matrix according to image processing techniques and SA algorithm. Since the determined points in image and field should be in static points set, we choose these points from the intersection of lines in image and field. Choosing two lines from horizontal lines set and two from verticals, we achieve feature points in image and field, too. Therefore, we have:

$$\begin{bmatrix} x_1 & y_1 & 1 & 0 & 0 & 0 & -x_1u_1 & -y_1u_1 \\ 0 & 0 & 0 & x_1 & y_1 & 1 & -x_1v_1 & -y_1v_1 \\ x_2 & y_2 & 1 & 0 & 0 & 0 & -x_2u_2 & -y_2u_2 \\ 0 & 0 & 0 & x_2 & y_2 & 1 & -x_2v_2 & -y_2v_2 \\ x_3 & y_3 & 1 & 0 & 0 & 0 & -x_3u_3 & -y_3u_3 \\ 0 & 0 & 0 & x_3 & y_3 & 1 & -x_3v_3 & -y_3v_3 \\ x_4 & y_4 & 1 & 0 & 0 & 0 & -x_4u_4 & -y_4u_4 \\ 0 & 0 & 0 & x_4 & y_4 & 1 & -x_4v_4 & -y_4v_4 \end{bmatrix} \begin{bmatrix} h_1 \\ h_2 \\ h_3 \\ h_4 \\ h_5 \\ h_6 \\ h_7 \\ h_8 \end{bmatrix} = \begin{bmatrix} u_1 \\ v_1 \\ u_2 \\ v_2 \\ u_3 \\ v_3 \\ u_4 \\ v_4 \end{bmatrix} \qquad (3)$$

Where $(x_i, y_i, 1)$ are feature points in the field and (u_i, v_i) are in image respectively.

III. Implementations

According to previous section, the starting points of SA are four points in image and four points in field. Four points of field are selected by selecting two lines from six horizontal lines and two lines from seven vertical lines and computing the intersection of them. The method is explained in the following sub sections.

A. Ground detection and line extraction

Since the ground color may vary over different soccer videos, and may vary over shots from different camera angle even in a same soccer video, detecting the ground needs to be performed every frame.

It has been observed that the HSV space is better suited for computing the color changes and separating hue from saturation and brightness adds robustness under most lighting variations.

We assign N_H bins for Hue channel, N_S for Saturation channel, and N_V for Value channel. In this work, we set N_H= 64, N_S= 64 and N_V= 256. Therefore, each histogram for the *i-th* frame is defined as in (4)

$$Hist(\hat{H})_k: 0 \le k < N_H, Hist(\hat{S})_k: 0 \le k < N_S, Hist(\hat{V})_k: 0 \le k < N_V, \qquad (4)$$

By utilizing the definitions in (4), some variables are defined as follows.

PeakIndex(Hist(V))=i, here Hist$(V)_i \ge \mathrm{Hist}(V)_p$ $\quad$ *for all* $0 \le p < N_V$

$\mathrm{LeftBoundary}(\mathrm{Hist}(V)) = j$, where $\mathrm{Hist}(V)_j = \mathrm{Peak}(\mathrm{Hist}(V))$,

for $\mathrm{PeakIndex}(\mathrm{Hist}(V)) > j \ge 0$

$\mathrm{RightBoundary}(\mathrm{Hist}(V)) = k$, where $\mathrm{Hist}(V)_k = \mathrm{Peak}(\mathrm{Hist}(V))/20$,

for PeakIndex(Hist(V))$\le$k<N_V (5)

$\mathrm{LeftBoundary}(\mathrm{Hist}(S)) = k$, where $\mathrm{Hist}(S)_l = \mathrm{Peak}(\mathrm{Hist}(S))/20$,

for PeakIndex(Hist(S))>l$\ge$0

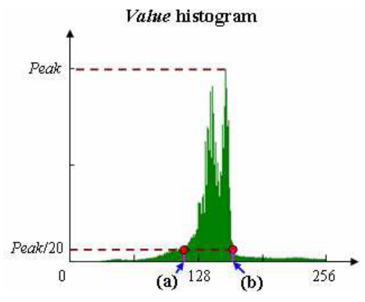


Fig.2. Finding of (a) *LeftBoundary(Hist(V))* (b) *RightBoundary(Hist(V))*.

Peak(Hist(V)) denotes the peak of the *Value* histogram and *PeakIndex(Hist(V))* denotes the bin index of the *Peak(Hist(V))*. *LeftBoundary(Hist(V))* is the index of the bin ranging from *PeakIndex(Hist(V))* to 0, the value of which corresponds to *Peak(Hist(V))*/20 in the *Value* histogram. Finding *RightBoundary(Hist(V))* is same as finding *LeftBoundary(Hist(V))* except the search direction is from *PeakIndex(Hist(V))* to *NV*. *LeftBoundary(Hist(S))* is obtained from the *Saturation* histogram with the same procedure addressed for *LeftBoundary(Hist(S))*. Finding *LeftBoundary(Hist(V))*, and *RightBoundary(Hist(V))* is illustrated in Figure. 2.

With above equations and using RGB space, extraction of ground pixels is as below [1]:

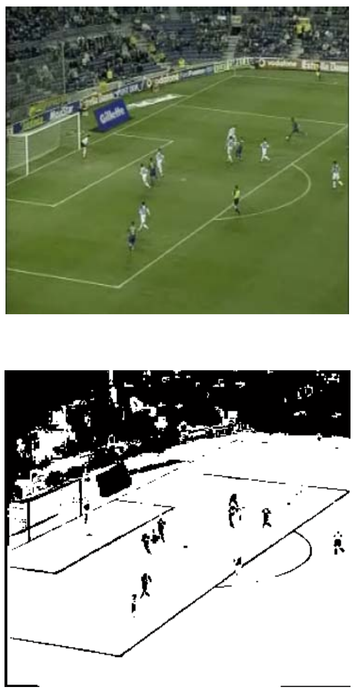


Fig.3. a) Original image, b) the result of ground detection.

$$Ground(x,y) = \begin{cases} 1 & if \begin{cases} g > 0.95r\ ,\ r > 0.95b & and \\ LeftBoundary(Hist(V)) - \theta_1 < g & and \\ g < RightBoundary(Hist(V)) + \theta_2 & and \\ \frac{g-b}{g}.N_S > LeftBoundary(Hist(S)) - \theta_3 \end{cases} \\ 0 & otherwise \end{cases} \qquad (6)$$

In this work, we set θ_1, θ_2, and θ_3 to minimize false detection, to be ten, five, and eight respectively.

After detecting the ground, with applying median filter and filling empty spaces from 3.b, figure 4 is resulted.

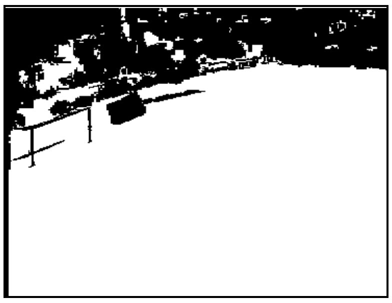


Figure.4. Empty ground

With subtracting two above images, figure 5 is gotten.

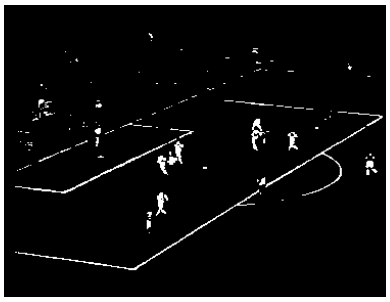


Figure.5. Extracted lines of ground

B. Selecting the entrance points of simulated annealing algorithm

After detecting lines in a frame, with using hough transform we can detect straight lines. In this mode, we can extract angles of these lines and according to them, dividing them into two category of horizontal and vertical lines are possible. For example if the angle of a line is between -30 and 30, that line belongs to horizontal line and other lines with another angle belong to vertical. As mentioned previously for selecting four points firstly two horizontal lines and two vertical lines are selected and their intersections are computed with their line equations. These intersections may be not in the frame. Some examples of these lines are shown in below figure.

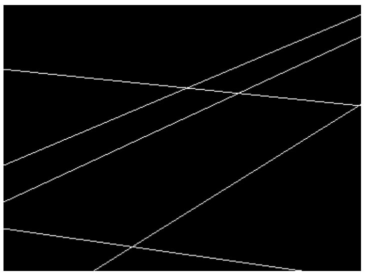


Figure.6. Extracted lines with Hough transform

IV. Simulated Annealing

Homography matrix works well when four points of frame and four points of field would be correspondent completely. So SA algorithm is applied for finding the best Homography transform. The phases of our work are shown in figure 7.
For evaluating the resulted matrix, we start with high first temperature, so we can accept first responses even they have not good result. In this work we started with T=1000000.
At first we transform all the lines in the field using resulted homography matrix and then evaluate these lines with two criterions.

1- Compute angles and spaces using hough transform and compare with angles and spaces of extracted lines of frame. If the correspondence value is upper than a threshold then we go to the next step. (here we consider threshold=0.8)
2- In this section, we compare pixels of lines resulted by H (transline) with pixels of extracted lines (frameline) of frame. Since we evaluate the transformed lines one by one, the output is a line with length of correspondence over 100 pixels [5].

$$f(i,j) = \begin{cases} 1, & if\ frameline(i,j) = 1\ and\ transline(i-1:i+1, j-1:j+1) = 1 \\ 0, & otherwise \end{cases}$$

$$F = \sum_{i=1}^{W} \sum_{j=1}^{H} f(i,j)$$

$$fit = \begin{cases} counter = counter + 1, & F > 100 \\ 0, & otherwise \end{cases}$$

$$fitness = \frac{fit}{length(frameline)} \qquad (7)$$

V. Improvement of proposed algorithm

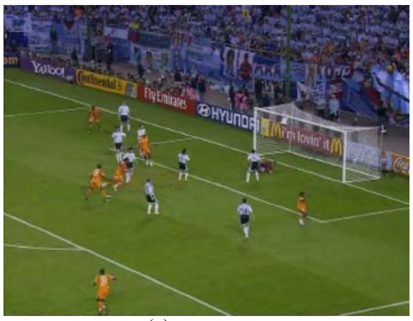
As discussed above, two lines of horizontal set line and two of vertical from field and image enter the SA algorithm. Because SA works accidently and cannot predict next lines, if we limit the space of search by reducing the lines count that can be chosen then it may be receive a good result faster. For this purpose we can determine right or left side of playing field. So some vertical lines can be deleted. This step is described below.

A. The side of playing filed detection

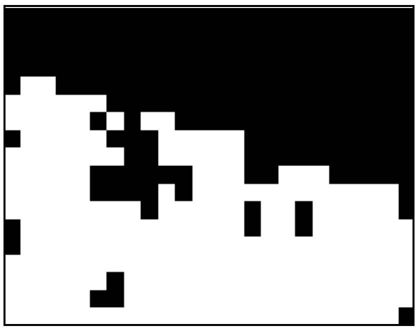
For this purpose, we use some 32*32 blocks on the binary image of the ground like figure 3.b. Then we investigate below of these blocks. If sum of the pixels under a block is greater than a threshold that block comes to 1 otherwise it becomes to 0.

$$CSR(x,y) = \begin{cases} 1, & if \sum_{y=r}^{y=r+31} \sum_{x=c}^{x=c+31} G(x,y) > T*(N^2) \\ 0\ , otherwise & if \begin{cases} c < \frac{C}{2} & leftMouth++ \\ c \geq \frac{C}{2} & rightMout++ \end{cases} \end{cases} \quad (8)$$

At last we consider left and right side of this image, if left side has more black blocks than white blocks, it determines that left goal mouth is in current frame and inverse. The results of this process are shown in figure7.



(a)



(b)

Figure 7. a)original image, b) detection of the side of playing field.

VI. Experimental Results

Our algorithm works on some frames of soccer games. Since this algorithm is performed accidently, so it trys to find the best matching matrix between field feature points and frame feature points. Since most of the field lines are near to goal mouth, we test those frames that contain goal mouth. Figure 8 shows a frame and the extracted lines responded by algorithm. These lines are not the only found lines but they are the best ones that can be corresponded to the real lines in the field.

VII. Conclusion and future works

In figure 9, steps of our algorithm for finding the best lines are shown. Since the algorithm is implemented around the goal mouths, so after finding a goal mouth in current frame the algorithm starts its work for searching the best corresponding lines between field lines and frame lines. For limiting the search area, we use left or right side of the field. With these, some lines of the field are deleted and search areas become limited.

In this work, we focus on finding the position of playing and with that we can find the corresponding point of each pixel in the field. We did this without any knowledge of camera parameters. This algorithm can be used for billboard substitution or inserting objects in the field or other places for future works.

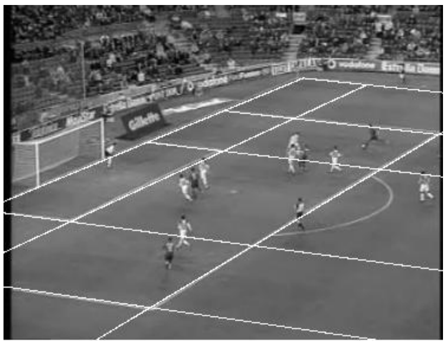


Figure.8. Best matching of field lines

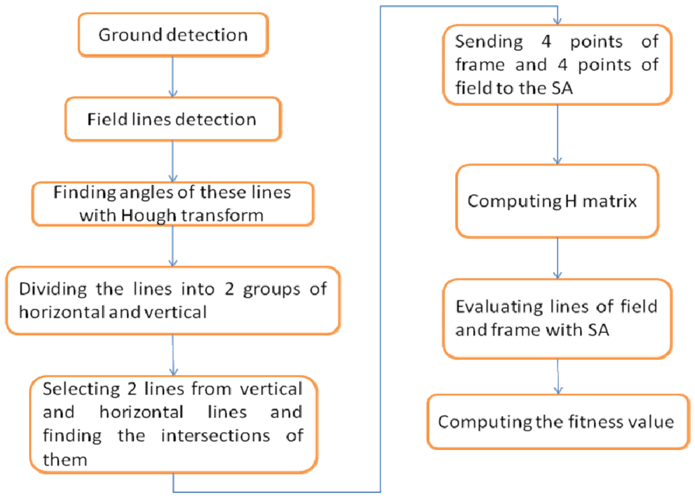

Ground detection
Field lines detection
Finding angles of these lines with Hough transform
Dividing the lines into 2 groups of horizontal and vertical
Selecting 2 lines from vertical and horizontal lines and finding the intersections of them
Sending 4 points of frame and 4 points of field to the SA
Computing H matrix
Evaluating lines of field and frame with SA
Computing the fitness value

Figure.9. Phases of finding corresponding lines

[1] Ilkoo Ahn & Changick Kim, *"Ground color customization of soccer videos"*, IEEE 2007.
[2] Zhengyou Zhang, *"A Flexible New technique for camera calibration"*, Microsoft Research, Technical Report MSR-TR-98-71.

[3] Reza Pourreza & Morteza Khademi & Hamidreza Pourreza & Habib Rajabi, *"Robust camera calibration of soccer video using genetic algorithm"*, IEEE 2008.
[4] Qihe Li & Yupin Luo, *"Automatic camera calibration for images of soccer match"*, world academy of science, engineering and technology 2005.
[5] Homa Fakour, Abedin Vahedian, Hamidreza Pourreza, "Camera calibration in Soccer field Using Simulated Annealing", 15th Tehran computer conference, 2010.