Tracking and Identifying in Real Time the Robots of a F-180 Team

Paulo Costa[(1)(2)], Paulo Marques[(2)], António Moreira[(3)],
Armando Sousa[(1)(2)], Pedro Costa[(1)]

paco@fe.up.pt, amoreira@fe.up.pt, asousa@fe.up.pt,
pamarques@riff.fe.up.pt, pedrogc@fe.up.pt

[(1)] Lecturer at the Faculdade de Engenharia da Universidade do Porto (FEUP)
[(2)] PhD. Sutdent at the FEUP, [(3)] Assistant Professor at the FEUP

Abstract - This paper describes the method employed to track and identify each robot during a Robocup match. Also, the playing ball is tracked with almost no extra processing effort. To track the robots it is necessary the use of adequate markers so that not only the position is extracted but also the heading. We discuss the difficulties associated with this problem, various possible approaches and justify our solution. The identification is performed thanks to a minimalist bar code placed in each robot. The bar code solves the problem of resolving some ambiguities that can arise in certain configurations. The procedure described can be executed in real time as it was shown in Paris in RoboCup-98.

1   Introduction

To be able to play consistently a team must, amongst other things, know its kinetic state. More, it must know the other team and the playing ball kinetic states. By kinetic state, we mean all the variables that must be known to uniquely characterize the physical position of a body.

The main source of information, for a F-180 Team, is the global camera that is typically placed above the playing field. Usually, that camera captures an image that covers all the playing field. Having that image transferred to a computer enables us to apply some kind of processing to extract the needed information. That task is very time consuming and the required robustness is sometimes difficult to obtain. Light variations in the illumination can be important disturbances the vision system and the design must be prepared to cope with that kind of problem [2].

2   The Team State

We can see a player as a rigid body constrained to have one of its faces parallel to the xy plane (the floor). With that restriction the full mechanical state of the robot can be described by the vector $(x, y, \quad, v_x, v_y, w)$, where x and y specify the robot's

position,   is the orientation, v_x and v_y are the robot's velocity in the x an y axis and finally w is the robot's rotation speed. If the robot has some kind of extra constraint in the possible trajectories then the state can be further reduced. For example, with the usual differential drive type and if we assume that the robot will not slip sideways then we can define the state only by $(x, y, \ , v, w)$. That happens because assuming the no slippage condition the robot velocity has always the direction of its orientation.
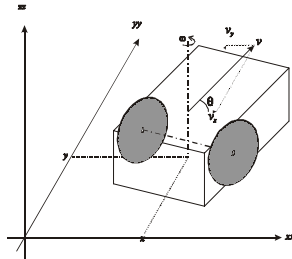

Fig. 1. Robot and state variables

From a still image it is not very easy to extract information about the velocity of an object, especially if the speed is low and we do not have any noticeable motion blur. On the other hand, that information can be obtained using information from past frames so we can concentrate on the data that can be directly extracted from the captured image.

The global kinetic state of the team it is the aggregation of all robot states [3] and [5]. The first step to extract that state from a captured image of the playing field is to get the position of all the available markers.

3   Required Markers

The Robocup Regulation enforce the presence of a Table Tennis ball placed on the top of each robot. Furthermore it is advised that the ball should be placed on the rotation center of the robot. But, using only this ball, we can only know the x, y coordinates. It gives us no information about . That problem can be solved using a second marker placed on top of the robots. We used another tennis table ball of a different color. That extra marker will give the robot's heading with a precision that increases as the distance between the two markers increases. That is why we placed them a few centimeters apart. The distance was not further increased because it is advantageous to have both balls surrounded by the top of the robot. The purpose is to allow the camera to get an image where the border around the ball is always black. This second ball is of the same color for all robots.

4   Image Processing

The main problem, posed in the process of extracting the required information, is

the short time available and the amount of data involved: if we want to extract all the items' positions at 25 *Hz* we have only 40 *ms* to process each image. For an RGB image with PAL resolution of 384 x 288 we can have 216 *KBytes* if we are using 16 bits to hold the RGB values and 340 *KBytes* in the 24 bits case. That is a lot of data and even the simplest filter can take too long to be applied to the image. Clearly there is the need to perform some kind of data reduction in a computationally cheap way.

We choose to process the entire field in an attempt to make the processing of each frame independent from the previous ones. Another advantage is that the time consumed is more predictable as it can be more or less stable from frame to frame. We do not loose the information from previous frames. As mentioned before, the speed of the robots and the ball can only be extracted from previous frames, but the actual image processing can be done without that information.

The overall processing system and its main building blocks are shown in the next figure:
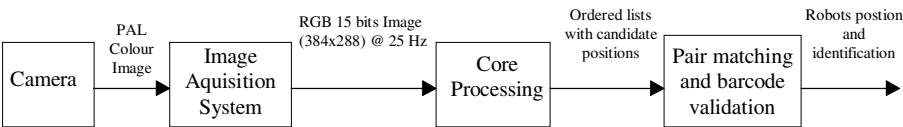

Fig. 2. Overall Image Processing System

We can use the concepts of Fuzzy Logic base our approach [7]. The core of our system relies in a function *f* that catalogues each pixel with its degree of membership to the set of an ideal color *c*. Let us can consider that each color that we want to identify is a fuzzy set *Sc*. Each element of this fuzzy set is a point in the RGB space. In other words it is a three dimensional vector (r, g, b). To define this set we must build a function $m(r, g, b)$ that maps each point to its set membership. To extract *n* colors we must define *n* of these sets and their respective membership functions. The function *f* then applies each of the p_i functions to each pixel and selects the color with the highest degree of membership. Setting a lower bound for the degree of membership we can rule out a lot of pixels that are not interesting. Almost all the pixels related to the green of the table, the white of the lines or the black of the robots cover can be dropped by this filter and we will retain only the interesting pixels. In doing that we can lower the amount of data to be processed by dropping the potentially uninteresting pixels. More, while doing that we can have now an idea of the possible color that each interesting pixel represents.

This function is locally independent and does not use any information from surrounding pixels. That is a big advantage because it can be evaluated only once for each pixel and in any order. The information inherent to the locality of each pixel will be used only in a later processing stage.

We now have a crisp set of interesting pixels and each pixel *p* can be represented by a vector (x, y, c, m) where *x*, *y* are the pixel coordinates, *c* is the color set in which the pixel showed the highest degree of membership and *m* is the respective

degree of membership. Then, we can aggregate the pixels by their positional proximity and build a set of candidate positions for each ball in the image.

5   From Interesting Pixels to Ball Positions

We can label the n colors associated with ball markers as c_1, c_2, …, c_n. After processing the image we can get n ordered lists, cp_1, cp_2, …, cp_n with the candidate positions for each ball. In each list we can have more candidates than there are balls. Each candidate is a vector (x, y, np, tf) where np and tf are figures of merit that are related with this candidate's proximity to the ideal image of a ball. More precisely, np states the number of pixels in this candidate that matched the required color and spatial proximity. The other component, tf, is the sum of each pixel fitness to the ideal color. The coordinates x and y of the candidate position are found by an weighted average of all the x, y coordinate of the points belonging to it.

The algorithm to perform the task is as follows (in pseudo code):

:Order the pixels
   Build a list lp_c of pixels that possibly represent the same color c;
     Order each list by decreasing degree of membership to the S_c
:Cluster
   for each list lp_c
     for each pixel $p(x, y, c, m)$
       if its distance to a existent candidate position of the same color is bellow a certain threshold r_b then add the pixel to that candidate position;
       else create a new candidate position and then add the pixel to that candidate position;
:Order and filter the candidate positions
   for each set of candidate positions for balls of the same color
     np_o := optimal number of pixels associated with a candidate position;
     for each candidate positions in the set
       np := number of pixels associated with the candidate position;
       if the $np < np^{min}$ then discard the candidate position
       tf := sum of each pixel degree of membership m
       if the $tf < tf^{min}$ then discard the candidate position
     order by increasing distance of np to np_o.

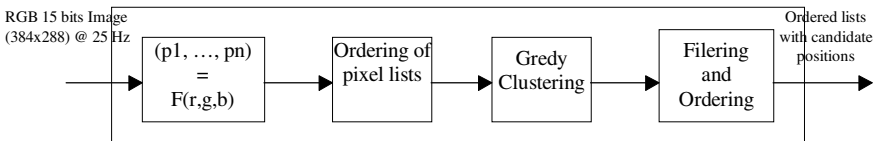The core processing system and its main building blocks are shown in the figure:

Fig. 3. Core processing system details

6   Robots' Positions and Their Identification

With best candidate positions for each ball we can pair then and extract the robot's new observed positions. A simpler procedure can be done to find the position of the playing ball. The best candidate center for the color associated with the ball is elected to give us the new playing ball position. Applying a Kalman filter or similar techniques can produce further refinements. For more on this subject see [6],[4] and [1].

This approach does not supply the identification of each robot. In earlier games we used a locality assumption to identity each robot. By assuming that each robot was the one closest to previous known location we could track each robot almost al the time. The problem with this approach was that it required the initialization by human intervention and in some cases the ambiguities could induce an error that was not easily recoverable and led the mismatched robots to become useless. Also, the pairing of the balls some times struggled with ambiguous cases that added another potential error source.
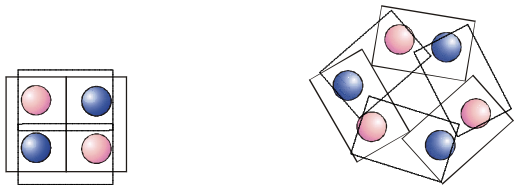

Fig. 4. Some ambiguous robot configurations

To overcome the problem we fitted each robot with a set of black and white marks that can be seen as a bar code label. The size of the markers could not be too small and in the available area on the top of the robot restricted us to use only three markers. With three markers with two states: black and white, we can only have eight different values represented. Restricting the use of all whites and all black we have six possible configurations always with, at least, one black mark and one white mark. That is enough for a five robots team.
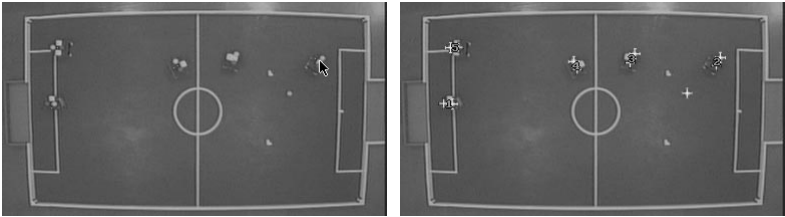

Fig. 5. Bar code in action

Using the bar code to validate a pair of balls we can eliminate the ambiguities shown in figure 4. That can be achieved by setting a luminance threshold to accept a white mark and the same to a black mark. Only if the luminance of the white mark

is above the threshold and the luminance of the black mark is bellow another threshold, we accept the code as a valid. That, coupled with the requirement that the valid bar codes have both kinds of mark, can almost ensure that a false reading is never accepted and gives us a very strong validation of the balls pair. Naturally the bar code identifies each robot without any ambiguity and we do not need anymore to rely in previous information to achieve that goal. Since we started playing with this setup, the reliability of our control increased substantially.

7   Conclusions and Future Work

In this paper we described a vision system for a F-180 team. We used this system in Paris in Robocup-98 with great success. The achieved frame rate of 25 *Hz* and the robustness showed by this system are its greatest strengths. It was implemented in a mix of C and C++ and the computer running it was a PC with a 266 *MHz* PentiumII and 64 *Mbytes* of RAM. For the image acquisition we used a standard camcorder and a PCI frame grabber based on the Bt848 chip.

The bar code technique solved one of the worst problems, the identification of each robot and provided a tool to eliminate ambiguities and the dependence on past information.

Using vision to recognize the overall system state, consisting in the ball position and speed, our team's robots' state and the adversarial robots' state, is still a very difficult task. As the quality of the team behavior is very dependent from the accuracy of that system any improvement in this area shows dramatic results in the overall team performance.

References

1. Arthur Gelb, Joseph Kasper Jr.,Raymond Nash Jr., Charles Price, Arthur Sutherland Jr.: Applied Optimal Estimation, The M.I.T. Press, (1989)
2. J. Borenstein, H. R. Everett, L. Feng, S. W. Lee and R. H. Byrne: Where am I? Sensors and Methods for Mobile Robot Positioning, (1996)
3. J. Carvalho: Dynamic Systems and Automatic Control, Prentice-Hall, (1993)
4. Huibert Kwakernaak, Raphael Sivan: Linear Optimal Control Systems, Willey-Interscience, (1972)
5. Jean-Claude Latombe: Robot Motion Planning, Kluwer Academic Publishers, (1991)
6. Manuela Veloso, Peter Stone, Kwun Han, Sorin Achim: The CMUnited-97 Small Robot Team, RoboCup-97: The First World Cup Soccer Games and Conferences, H. Kitano (ed.), Springer Verlag, Berlin, (1998)
7. Bart Kosko: Neural Networks and Fuzzy Systems, Prentice Hall