6

Tracking Players in Indoor Sports Using a Vision System Inspired in Fuzzy and Parallel Processing

Catarina B. Santiago[1], Lobinho Gomes[1], Armando Sousa[1],
Luis Paulo Reis[2] and Maria Luisa Estriga[3]
[1]*Faculty of Engineering, University of Porto and INESC-Porto, Porto*
[2]*School of Engineering, University of Minho and LIACC, University of Porto, Porto*
[3]*Faculty of Sports, University of Porto and CIFI2D, Porto*
Portugal

1. Introduction

Sports are an important part of nowadays society and there is an increasing interest by the sports' community on having mechanisms that allow them to better understand the dynamics of teams (their own and their opponents). This information is frequently extracted manually by operators that, after the game, visualize game recordings (frequently TV footages) and perform hand annotation, which is a time consuming and error prone task. There is a clear necessity for developing automatic mechanisms and methodologies which allow performing these tasks much faster and systematically. The importance of such systems was first highlighted in the late 80's by Franks et al. ((Franks & Nagelkerke, 1988; Franks et al., 1987)). In this chapter, we present an automatic and intelligent visual system for detecting and tracking handball players based on two cameras that cover the entire playing area. The followed methodology includes the identification of foreground pixels using dynamic background subtraction, the definition of colour subspaces for each team using a Fuzzy inspired model that allows detecting the players based on the colour properties of their clothes. Player tracking is further improved by using one Kalman Filter per player (object to track). The resulting information is aggregated in an undistorted image view of the entire field that is very interesting and meaningful to the target end-user. The generation of the video is a demanding computational task that takes advantage of using parallel computing. The resulting videos are interesting and include several important informations to the human end-user. Tests were conducted on videos collected during the Handball Portuguese SuperCup competition, where the best six Portuguese teams competed for the trophy of the year of 2010/11.
The chapter structure is as follow: the next section presents some relevant information on image segmentation methodologies, parallel processing and implementations of automatic visual systems for detecting and tracking players and describes the main methodologies used. Section 3 discusses the proposed architecture principles, providing an overview of the methodology used. Section 4 presents the results achieved and finally section 5 concludes this chapter with the main conclusions.

2. Related research

This section focus on the three main areas involved on the implementation of the system, and therefore presents some of the most common used methodologies for video/image segmentation and for parallel processing as well as an overview of systems that are able to detect and track players in indoor team sports.

2.1 Video/image segmentation

Video and inherently image segmentation is the first step and probably the most critical step in any vision system. Video segmentation can be subdivided into temporal segmentation and spatial segmentation.

Temporal segmentation corresponds to segmenting the video into meaningful temporal sequences. This kind of segmentation is usually used as the first step of video annotation and tries to segment the video taking into account similarities/dissimilarities between successive frames (Koprinska & Carrato, 2001). On the other hand, spatial segmentation analyses the content of each frame, and divides it into homogeneous regions that correspond to independent objects. The focus of this work is more on spatial segmentation, therefore for a detailed survey on temporal video segmentation please refer to (Koprinska & Carrato, 2001).

Spatial video segmentation may be performed using the methodologies that are used for image segmentation and further enhanced using the temporal characteristics of video. In addition, when performing colour analysis there is also the need to choose a colour space.

Regarding colour image segmentation, a detailed survey is provided by (Cheng et al., 2001). As they state, most of the existing colour image segmentation methodologies have their origins on grey scale image segmentation with the addition of a proper colour space choice. The main categories of image segmentation methodologies (Cheng et al., 2001) are summarized on Table 1.

Nowadays there is the tendency to apply techniques from different categories in order to achieve better results. A good example of this tendency is the JSEG algorithm (Deng & Manjunath, 2001) which initially clusters colours into several representative classes, afterwards replaces each pixel by their corresponding colour class label and at the end employs a region growing process directly to the class map in order to identify homogeneous regions.

On videos, contrary to static images, besides the physical (x and y coordinates) and colour information there is also the time component. Using this property it is possible to segment images based on motion along time. There are two main approaches to perform this task: background subtraction and optical flow.

Background subtraction is usually used in situations where a more or less fixed background exists and subdivides the image into foreground and background regions. Several background subtraction techniques have been proposed in literature. The main issue on these methods is to obtain a good estimate of the background.

The simplest method to model the background is to use a single static image without objects, however its efficiency is low, because it does not take into account changes such as light effects or shadows that may occur in the background. More robust methods include estimating the background model using a moving average (Heikkila & Silvén, 1999), median or even a mixture of Gaussians (Grimson et al., 1998).

Category	Description
Histogram Thresholding	Determine the peaks or modes of the multi-dimensional histogram of a colour image.
Feature Space Clustering	Groups the image feature space into a set of meaningful groups or classes based on intensity, colour or texture characteristics of pixels and not on the spatial relation among them.
Region based	Includes region growing, Watershed transform and region split and merge. These methods try to divide the image domain based on the fact that adjacent pixels in a same region have similar visual features (colour, intensity, texture or motion).
Edge Detection	Segment the image by finding the edges of each region using one of the well known edge detectors (Canny (Canny, 1986), Sobel (Sobel, 1970), Roberts (Roberts, 1963)).
Fuzzy	These methods allow classes and regions to have a certain uncertainty and ambiguity which is generally the case in image processing.
Neural Networks	Allow parallel processing and the incorporation of non-linearities. They can be used either to pattern recognition, classification or clustering.

Table 1. Overview of image segmentation categories

Optical flow (Barron & Thacker, 2005) is based on the fact that when an object moves in front of a camera, there is a corresponding change on the image, however it assumes small displacements during time.
Following the tendency, we propose, for the video segmentation step, a methodology that combines background subtraction for detecting foreground regions, region growing and a Fuzzy inspired categorization for colour calibration.

2.2 Parallel processing

Initially, computing power was pure sequential processing, that is, sequential operations over time in a single dedicated processor. The hunger for usefulness and processing power lead to advanced solutions such as concurrency and distributed computing. Given large "workloads", concurrency is interleaving processing over time for the multiple "workloads" (even in a single processor). Distributed computing is sharing "workloads" (or parts) over different processors linked over a network. Recent advances in computer architecture introduced multi core architectures that enable true parallel processing of several "workloads" in parallel in the same computer. The challenge remains unaltered: to maximize the usefulness of a computer systems and harness as much computing power as possible, possibly circumventing technical limitations of an architecture, whilst maximizing performance but still keeping cost interesting.
When considering a single complex (large) computational job, parallel processing starts by:

- (i) dividing work into several pieces (this can be done by the programmer, at run time or a mixture of both situations) then

- (ii) ”transfers” work parts to collaborators (several processors inside the same computer or not) and later
- (iii) completes the job by assembling all intermediate results into a meaningful final solution.

There are a number of complexities with parallel computing:

- Management of parallelization - the additional computational cost (overhead) to start and maintain parallel execution (sharing program, data, managing requests, extra memory required, etc);
- Communication overhead - additional computational cost to communicate to several other processors (examples: transmission of initial relevant data or fundamental intermediate calculations);
- Synchronization problems - the explicit need for sequential operations caused by interdependence among several work parts or using shared resources;
- Load Imbalancing - the difficulty in sharing workloads as even amounts of work with the likely problem that, if one of the intermediate calculations takes longer to process than the others, optimization is not as perfect as one could wish for.

In parallel computing, the Scalability curve is defined as the speed-up in processing (performance gain), over the number of available processors. Generically, it is not likely at all that a job done in a single processor in a time t_1 could be done in N processors in t_1/N of that time, that is, time to solve a problem in N processors is very frequently $t_N > t_1/N$. By adding too many processors (large N), the scalability curve will drop heavily when too many work parts are generated and computational overhead for parallel processing outweighs the benefits of having many ”workers” (processors). Additionally, adding processors will likely have severe financial impact on the final overall cost of the computational solution.

OpenMP ((OpenMP, 2011)) is a technological framework for cross platform, shared-memory application programming interface (API) for taking advantage of run time distribution of work parts into several processors ((Chapman et al., 2007)). OpenMP (OMP) was introduced formally in 1997 and is maintained by the Architecture Review Board (ARB), a consortium of industry and academia and its licence is essentially free. Advertised benefits include, among others, good scalability, implicit communication and programming at (somewhat) high level. The technological limitations for performance include transferring code and data among several types of memory and assigning work parts (threads) to available resources (processors).

CUDA ((NVIDIA, 2011)) stands for Compute Unified Device Architecture and is a software platform for massively parallel high-performance computing using NVidia’s video graphics hardware. The CUDA software toolkit is proprietary from NVidia, it essentially allows a free licence of use and was formally introduced in 2006. A large portion of the (recent, high-end) hardware from that manufacturer is usable with CUDA. The intent is to turn the resources of the video card into a number of generic processors and memory ((Halfhill, 2008)). The actual number of CPUs and memory available for generic use in the video card is hardware dependent and a highly volatile issue over time. Sometimes 16 or more independent processors are available for custom parallel programming (working frequencies frequently below 1 GHz) and about 512 MB of memory is also frequently usable. These resources are usable if the video card is used for showing 2D images. Communication with the video card

has inherent technological limitations as code and data must be transferred into the video card, an ”external” element when compared to CPUs and Memory Systems. While using the video card(s) as generic processors presents limitations to hardware changes it is most interesting for application sharing lots of data, with the added benefit of freeing up the main processor(s) for other tasks. High-end video cards have risen in complexity, performance and cost and may currently be more expensive than the "main" general purpose CPU of the computer.

The reader should be aware that at, the time of writing this article, parallel programming is still at its infancy. A promising framework exists, however: OpenCL (Open Computing Language) is open, royalty-free, standard for general-purpose parallel programming of heterogeneous systems across different hardware (Group, 2011b). OpenCL provides a uniform programming environment for software developers to write efficient, portable code for high-performance computing using a diverse mix of multi-core CPUs, GPUs, etc. OpenCL is recent and its first introduction is in late 2009 by the Kronos Group (Group, 2011a), a consortium of companies Intel, AMD, ATI, ARM, NVidia, among others. The performance will likely be inferior to pure CUDA as OpenCL calls CUDA when possible. These topics are currently issues of large interest in the Scientific and Technological communities because they promise great performance benefits. The most important topics and dominant toolkits were, however, briefly addressed in this section.

Prospective users of parallel computing should take into consideration that there is a non-negligible effort in learning the concepts at stake and even more in porting large conventional algorithms to take advantage of parallel processing. While OpenMP is not far to common programming techniques, CUDA approach of offering massive parallelism with common data requires different approaches and most likely the new algorithm will demand very large portions of code to be written almost from start. As in other optimization techniques, profiling the application is of great interest to find which part of the code takes more time to execute and what part(s) of the code will benefit from parallel computing.

2.3 Player detection and tracking using vision systems

Although there are devices to detect and track players using other methodologies besides vision (Radio Frequency Identifiers, Local Position Measurement (Stelzer & Fischer, 2004) or Global Positions Systems) these other methodologies are beyond the scope of this paper and will not be addressed.

The sport that has deserved the highest attention by the scientific community has been soccer, nevertheless the focus of this section will be on the work developed in indoor sports since our object of study is handball and also because the challenges posed by indoor/outdoor sports are quite different. Outdoor sports usually receive more attention by the media and therefore it is possible to use images provided by TV broadcast cameras, however the light conditions are much worse and the environment is less controllable. On the other hand, indoor sports usually need a dedicated camera system and despite being a more controlled environment, players tend to be closer to each other (the playing area is smaller) which brings added difficulties since merging and occlusion situations occur more often.

On indoor team sports it is possible to find works on basketball, handball and indoor soccer, using either broadcast video footages or dedicated cameras placed at strategic places and several methodologies for player detection and tracking.

Concerning basketball games we can find (Hu et al., 2011) using broadcast videos. Players are detect by extracting the field (through dominant colour detection) and generating a player mask. Afterwards players are tracked in image coordinates using a CamShift-based tracking algorithm and their positions are converted into real world coordinates using an automatic calibration methodology. Results are presented for 48 consecutive frames and show high precision and recall percentages, 91.38% and 91.34%, respectively. However occlusion and merging between players of the same team affect the detection and are not taken into consideration.

A very promising project is the Autonomous Production of Images based on Distributed and Intelligent Sensing (APIDIS), (APIDIS, 2008), that equipped a basketball court with an acquisition network composed of microphones, conventional and (arrays of) omnidirectional cameras in order to provide a basketball dataset. This dataset was used by (Alahi et al., 2009; Delannay et al., 2009).

Taking advantage of the setup flexibility and amount of cameras (Alahi et al., 2009) were able to minimize occlusion and merging problems and determine the 3D positions of the players. The player detection is performed via a sparsity constrained binary occupancy map based on severely degraded silhouettes. These silhouettes are obtained through a basic background subtraction method. Results show that the usage of more than one camera can tremendously increase both the precision and recall rates for the player detection, from 57% and 62% in a single camera system to 76% and 72% if an omnidirectional camera is added.

(Delannay et al., 2009) detect the players from the foreground activity masks determined on multiple views. They take into consideration players occupy a 3D space and therefore sum the cumulative projections of the multiple views' foreground activity masks on a set of planes that are parallel to the ground plane. Afterwards, regions with larger projection values are considered to be a player and scanned for digits in order to detect the player's number. The tracking propagation is performed frame by frame and is based on the Munkres general assignment algorithm (Munkres, 1957). They compare their methodology with methods that project the activity masks into a single plane and conclude that projecting into multiple planes increases the detection rate and minimizes the shadows effects.

We could find two works (Kristan et al., 2009; Monier et al., 2009) that explore both basketball and handball, use closed world assumptions and two dedicated cameras placed at the ceiling of the sports hall. Placing the cameras at the ceiling has the advantage of minimizing occlusion problems.

On the first work, players are detected by applying a background mask image obtained from thresholding the differences between the background (background is obtained by a simple method, for example a median filter) and the current frame with a dynamic threshold specific for each player. The tracking algorithm is formulated as a closed world problem based on a Boot-Strap Particle Filter and each tracker is initialized manually. The multiple players' tracking is achieved by partitioning the world into several Voronoi cells, one for each player, that are updated at each time step. Results indicate low failure rates per player per minute, less than 1.1 in the worst test case.

The second work follows some of the assumptions of the one from (Kristan et al., 2009), however the foreground pixels are detected by creating a dynamic background model rather than acting on the threshold level or generating a background mask and afterwards apply template matching to track the players. The templates for each player are manually initialized by the user and the tracking is performed by searching in a region of interest determined by

image resolution and players' speed restrictions. The multiple tracking is also achieved by partitioning the world into Voronoi regions where each tracker acts. The fusion between the two images is performed in real world coordinates. Results indicate average correction rates between 0.0019 and 0.00677 correction/frame/player.
Regarding indoor soccer, there are the works of (Needham & Boyle, 2001) and (Kasiri-Bidhendi & Safabakhsh, 2009). (Needham & Boyle, 2001) use a single stationary camera system and apply a multiple object condensation algorithm. Initially a bounding box of each player is detected, then, through a propagation algorithm, the fitness of each bounding box is evaluated and adjusted. The prediction stage of the condensation algorithm takes into consideration position estimates from a Kalman filter. They report on trajectory based results compared with hand annotated values and indicate distance errors less than 1 meter, which is a quite high value given that an indoor soccer minimum field measures 15x25 meters.
On the other hand, (Kasiri-Bidhendi & Safabakhsh, 2009) use TV broadcast images. Initially, the background colour is determined through clustering, afterwards the entire field region is extracted using the Mahalanobis distance between pixels on the frame and on the background colour distribution. Once the image is background free the lines of the field are extracted using a Canny edge operator. The remainder of the information stored in the image corresponds to the players and the ball that are further refined using physical constraints of area and roundness. The player and ball tracking is based on level set contours. By using this tracking technique occluded players are tracked by a single contour instead of two during the occlusion period and are again correctly detected after splitting.
This work follows (Santiago et al., 2011), that present a methodology based on colour in order to detect handball players. In addition, we have included a dynamic background model in order to take into consideration light changes and a Fuzzy inspired methodology to calibrate the colour teams. Moreover, a vector of Kalman filters is used for the tracking process and an HyperVideo with the resulting information is generated.

3. Architecture

3.1 Engineering solution

The challenges of defining a vision system able to identify and track the players in a team game are quite huge due to the dynamic and spatial characteristics of the game itself. Usually indoor team games are quite dynamic with high physical contact among players and rapid movements, for example a player can achieve velocities of more than 5 m/s. These characteristics impose a careful choice on the system's architecture which includes choosing not only the cameras and their disposition but also defining the software design.
In order to minimize crowd interference, merging and occlusion situations and have a good view of the entire field we propose using two cameras placed at the ceiling of the sport's hall to cover each half of the field (this solution was also adopted by (Kristan et al., 2009; Monier et al., 2009)).
The cameras chosen are GigEthernet (DFK 31BG03.H model from Imaging Source) which allows placing the recording unit far away from the cameras and maintaining the signal integrity, have a resolution of 1024x768 pixels and can supply images at 30 frames per second. The images collected from the cameras are shown on Fig. 1 .
We propose a two module software system, one responsible for acquiring the images from the two cameras (Acquisition System) and another responsible for the offline processing of

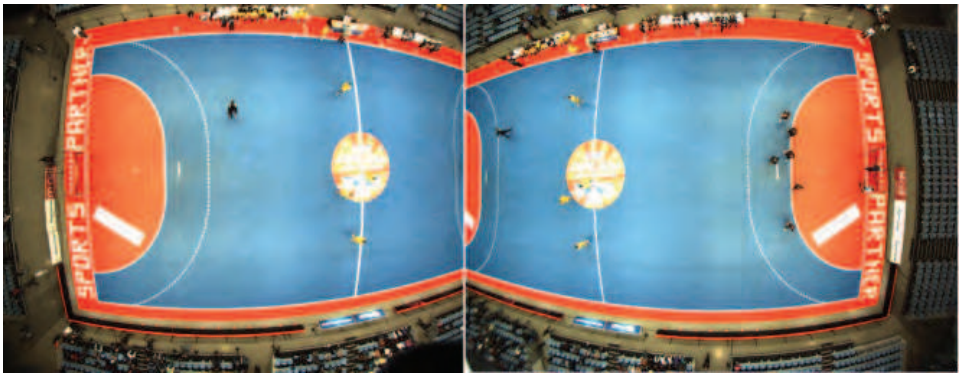


Fig. 1. A single frame from the two video streams.

the two video streams (Processing System). This last module is responsible for detecting and tracking the players as well as for generating an HyperVideo that consists on an unified image of the two streams with the positions of the players. Additionally it generates log files with the players' positions so that they can be used by the sports community to perform game analysis and infer game statistics. Figure 2 presents a scheme of the proposed architecture.

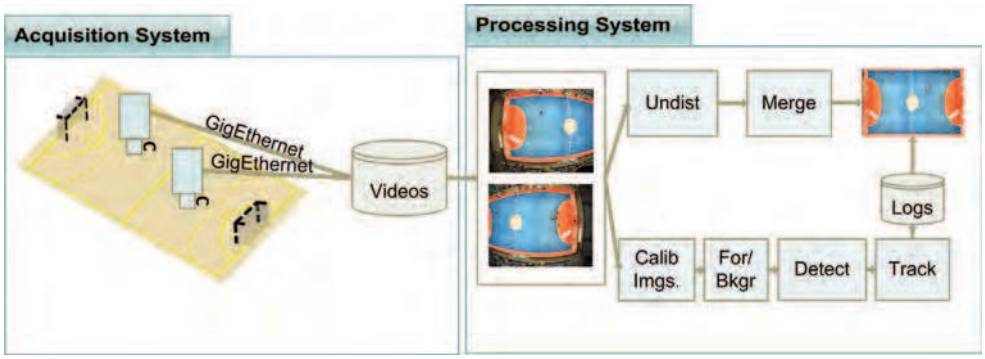


Fig. 2. System's architecture.

3.2 Player detection

Following previous work ((Santiago et al., 2011)) the player detection is achieved through colour identification and is composed of three steps. The first step consists on the colour calibration of each team using a region growing method allied with a Fuzzy inspired categorization methodology. This calibration is responsible for subdividing the colour space into subspaces, which are not necessary disjoint since there may be colours that are common to both teams (for example there are many teams that have uniforms with white stripes).
Afterwards, the user manually indicates the location of each player on the field by clicking on the images. Despite some works make this initialization automatically we choose this approach because both handball and basketball allow unlimited players' substitutions, and this manual initialization enables the possibility to always give the same number to the player and also discard the player when he/she leaves the field.

The second step consists on detecting foreground regions through a dynamic background subtraction method, which uses an empty image of the field and a dynamic threshold that is continuously and locally updated at each new frame (as will be described later on this chapter).
After the foreground pixels are identified, their colour is compared against the colour subspaces and classified into one of the teams. In case there is a belonging tie between teams, the adjacent pixels are searched in order to break the tie. Additionally the teams' colour subspaces are updated with new information.
Finally, pixels are aggregated to form blobs and categorized into player or no player according to size and density restrictions. The centre of mass of the blob is considered the player's position that is afterwards transformed into real world coordinates (court coordinates) using the cameras' homographies.

3.2.1 Colour calibration

The colour calibration is performed under the supervision of the user and is achieved using a region growing method (Santiago et al., 2011).
Let us define colour subspace S_c as the set of RGB colour triplets that are tagged as having the colours of the vests of the team c. The initial colour seeds $C(x_s, y_s)$ for each colour subspace S_c are set manually using the mouse to click on the objects that will be segmented, afterwards the surrounded pixels' colours $C(x_a, y_a)$ are agglomerate around these seeds using colour distance criteria. The colour expansion is performed on the HSL (Hue, Saturation and Luminance) colour space in order to minimize the effects of shadows and light variations.
The regions growth is performed in all directions (using a 8 neighbour mask n_8) in a recursive way until reaching a pixel that in terms of colour is away from the seed more than a global threshold (C_{ThresG}) or from its previous neighbour $C(x_p, y_p)$ more than a local threshold (C_{ThresL}), according to the following definition (both thresholds are user definable):

$$C(x_a, y_a) \in S_c \Leftrightarrow \forall (x_a, y_a) \in n_8(x_p, y_p) :$$
$$\Delta(C(x_a, y_a), C(x_p, y_p)) < C_{ThresL} \wedge \Delta(C(x_a, y_a), C(x_s, y_s)) < C_{ThresG}$$

,where

- $C(x, y)$ is the HSL colour of pixel at location (x,y),
- $n_8(x, y)$ are the eight neighbours of pixel at location (x,y),
- $\Delta(C_1, C_2)$ is a configurable weighed distance function, involving HSL components of colours C_1 and C_2.

During the colour expansion each colour value is attributed a given belonging degree to the subspace being calibrated. This value is stored in a lookup table that contains for each colour triplet the belonging degree to each subspace. Despite the expansion is performed on the HSL colour space, the colour lookup table is built on the RGB (red, green, blue) colour space as well as the remainder image processing operations.
The fuzzy belonging $\mu()$ of the colour $C()$ of a pixel P of coordinates (x_P, y_P) to a given colour subspace Sc is $\mu_{Sc}(C(x_P, y_P))$ and can assume four levels: when not belonging to the colour subspace $\mu() = 0$ and $B() = C_0$, by default and before the calibration takes place, all the colours are categorized with no belong degree to every subspace; for low belong degree to the colour subspace $\mu() = 0.5$ and $B() = C_L$; for full belong degree to the colour subspace $\mu() = 1$

and $B() = C_F$ and additionally a full belong degree with the characteristic of also being a colour seed $\mu() = 1$ and $B() = C_S$. The B_{Sc} function maps the four colour categories (C_0, C_L, C_F and C_S) into the fuzzy belonging according to Table 2.

Colour	B_{Sc}	μ_{Sc}
Not the colour	C_0	0
Resembles the colour	C_L	0.5
Is the colour	C_F	1
Is a seed colour	C_S	1

Table 2. Mapping of B_{Sc} to fuzzy belonging $\mu()$.

In order to determine the belonging degree of the colour triplet the following rules are applied sequentially during the region growing process:

- **Rule1** - if the pixel was assigned to the subspace, is physically quite close to the initial seed pixel and the colour distance to the initial seed pixel is less than one fifth the maximum allowed distance for the growing process (less than $\frac{1}{5}C_{ThresG}$) then it is also assumed to be a seed pixel with a full belonging degree.

$$B_{S_c}(C(x_a, y_a)) = C_S \Leftarrow C(x_a, y_a) \in S_c \wedge \Delta((x_a, y_a), (x_s, y_s)) < \frac{1}{5}C_{ThresG}$$

- **Rule2** - if the colour distance to the initial seed pixel is less than two fifths the maximum allowed distance for the growing process than the pixel is categorized with a full belonging degree but without being a seed.

$$B_{S_c}(C(x_a, y_a)) = C_F \Leftarrow C(x_a, y_a) \in S_c \wedge \Delta(C(x_a, y_a), C(x_s, y_s)) < \frac{2}{5}C_{ThresG}$$

- **Rule3** otherwise and in case the pixel obeys to the region growing conditions it is categorized with a low belong degree (C_L).

By the end of the calibration process the colour space is subdivided into subspaces, which are not necessary disjoint since the same colour can belong to different subspaces. The motivation for allowing non-disjoint subspaces is that teams frequently share colours, for example uniforms with white stripes are common and thus the exact same well known colour belongs to the two opposing teams. Hence the Fuzzy Logic inspiration methodology, however implementation issues make it not interesting to allow for continuous degrees of belonging. The belonging degrees attributed to each colour triplet, as will be seen later, will allow to break ties but also to generate dynamic subspaces that can adapt (either grow or shrink) during the game. Subspaces do not have nor ever create any predefined specific shape as they are created from user-selected seeds on the image, that have been grown in the user selected video frames and in colour space.

3.2.2 Background subtraction

Since the background is more or less static, due to the semi-controlled environment of an indoor game, the subtraction is performed using an empty image of the viewed scene and only the threshold used to distinguish between foreground and background pixels is dynamic and specific for each pixel.

The background subtraction is performed on the RGB colour space, because tests showed that for some pixels a small difference between the RGB colour components of the background and the processed images corresponded to a large difference on the Hue component (HSL colour space). In fact, non-linear colour spaces suffer from the non removable singularity problem as stated by (Cheng et al., 2001).
Also in order to make the processing time shorter, the subtraction is executed locally and not to the entire image. In other words, only predefined regions, which are defined by the Kalman Filter predictive stage, suffer this process.
The threshold applied to each pixel is only updated if the pixel is classified as background, otherwise its value remains unchanged. The update obeys to equation 1 and the value is never allowed to go below 4% or above 23.5% of the entire colour range (0-255) for each colour component. These values were obtained experimentally.

$$\sigma_{t+1}^{c}(x,y) = \begin{cases} \alpha(I_t^c(x,y) - B^c(x,y)) + (1-\alpha)\,\sigma_t^c(x,y) \text{ , if } I_t(x,y) \in B(x,y) \\ \\ \sigma_t^c(x,y) \quad \text{, otherwise} \end{cases} \tag{1}$$

,where

σ is the threshold of the pixel at position *(x,y)*, time *t+1* and colour component *c*,

I is the colour intensity of the pixel at position *(x,y)*, time *t* and colour component *c*,

B is the background colour intensity of the pixel at position *(x,y)* and colour component *c*,

α is a learning constant, that for our specific case was set to 0.02.

Pixels whose colour difference to the background image is less than the respective threshold are labelled as background, the others are labelled as foreground.

3.2.3 Team identification

After the foreground pixels are identified, their colour is compared against the colour lookup table that resulted from the calibration process (3.2.1) and classified into one of the teams.
Since the same colour can belong to different teams (subspaces) it may occur that a pixel is classified into more than one team. To break this tie, information not only from the belonging degree itself but also from adjacent pixels that have already been classified is used.
Spatial information is used by counting the number of adjacent pixels that belong to each team, afterwards if the sum of the team with the highest value is 1.5 times the value of the lowest team then that pixel is assigned a weight of 2 to that team otherwise it is assigned a value of 1 to both teams. Colour calibration information is used by adding to the previous weight the corresponding fuzzy belong values (as shown in Table 2).
The team with the highest final weight is the one assigned to the pixel. This way, it is possible that, although the belonging degree of a pixel to a team based on the colour calibration information is higher than the belonging degree to the other team it may be the winner due to the neighbourhood characteristics.
Additionally, if the winning team has a full belong to that colour triplet and corresponds to a seed colour ($B() = C_S$) then a region growing process is triggered and the colour lookup table that contains the information concerning the colour subspaces is updated. This auto expansion is more restrictive than the one performed during the manual initialization and is performed at time intervals (t_{expans}), an adjustable setting. This setting may change to take

into consideration the speed at which light changes in the pavilion and yields better overall performance to the application.
In order for this update to add not only colour triples to the subspaces but also to remove them (otherwise subspaces would grow too much), each colour triplet has associated a persistence ($p_{S_c}(R,G,B)$) to that subspace. Colours with lower belonging have lower persistence and colours with higher belonging have higher persistence. The initial persistence given to the colour is proportional to the time between auto expansions according to Eq.2.

$$\begin{cases} p_{S_c}(R,G,B) & = \frac{1}{8} t_{expans} & \text{, if } B_{S_c}(R,G,B) = C_L \\ p_{S_c}(R,G,B) & = \frac{1}{4} t_{expans} & \text{, if } B_{S_c}(R,G,B) = C_F \\ p_{S_c}(R,G,B) & = \infty & \text{, if } B_{S_c}(R,G,B) = C_S \end{cases} \tag{2}$$

The persistence is maximum (with the values defined in Eq. 2) when the colour is added to the subspace and diminishes whenever it is not detected in a frame, however seed colours have infinite persistence and will therefore always remain in the subspace. Whenever the persistence value reaches zero the colour triplet is removed from the subspace.
With the introduction of this dynamic it is possible to have mutable subspaces that adapt to light changes either occurring at different regions of the same frame or between frames.
At the same time the foreground pixels are classified, they are also aggregated horizontally to form run length encoding (RLE) structures characterized by the y, x_{min} and x_{max} positions of the RLE. Finally the RLEs are merged vertically to form blobs. A full description of this pixel aggregation can be found in (Santiago et al., 2011) with the particularity that small horizontal RLEs are ignored and do not pass to the vertical merging process in order to minimize noise.
The blobs resulting from this pixel aggregation are further refined according to size and colour density constraints. Therefore, blobs that are too small or too large or blobs that have low colour density are discarded as being players. The colour density is measured as the percentage of pixels inside the bounding box of the blob that belong to the team divided by the total number of pixels of the bounding box. The remainder blobs are considered players that belong to a given team (S_c) and have an (x,y) position on image and world coordinates.
The position on image coordinates is calculated as being the center of mass of the blob according to Eq.3 .

$$(x_{cm_{blob}}, y_{cm_{blob}}) = \left(\frac{\sum_x \sum_y \mu_{Sc}(C(x,y))\, x}{\sum_x \sum_y \mu_{Sc}(C(x,y))}, \frac{\sum_x \sum_y \mu_{Sc}(C(x,y))\, y}{\sum_x \sum_y \mu_{Sc}(C(x,y))}\right) \tag{3}$$

, where c is the team the blob belongs to.

The world coordinates[1] are obtained by first removing the barrel effect produced by the lens (only radial effect was considered, since the tangential component can, in most cases, be neglected) using Eq. 4 . The unknowns in these equations (k_1 , k_2 , k_3 , x_c and y_c) are determined using the information extracted from the field lines.

$$\begin{cases} x_u = x_d + (x_d - x_c)(k_1 r^2 + k_2 r^4 + k_3 r^6) \\ y_u = y_d + (y_d - y_c)(k_1 r^2 + k_2 r^4 + k_3 r^6) \end{cases} \tag{4}$$

[1] We would like to thank professor Paulo Costa and Paulo Malheiros from the University of Porto, Faculty of Engineering for the specific camera calibration software.

, where

- $r^2 = (x_d - x_c)^2 + (y_d - y_c)^2$,
- (x_u, y_u) are the undistorted coordinates,
- (x_d, y_d) are the distorted coordinates,
- (x_c, y_c) are the coordinates of the center of distortion of the lens,
- k_1, k_2 and k_3 are the radial coefficients for barrel distortion.

Fig. 3 illustrates the images before and after removing the barrel effect for the two cameras. Once the barrel effect is removed from the players' positions, it is possible to apply the pinhole camera model in order to obtain the world coordinates of the players. This model uses intrinsic parameters (K) and extrinsic parameters (R and T) to map image coordinates (X) into world coordinates (x) according to Eq.5.

$$x = K[R|T]X \Leftrightarrow x = H_w X \tag{5}$$

The H matrix is defined according to Eq.6.

$$H_w = \begin{bmatrix} f & 0 & c_x \\ 0 & f & c_y \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} \cos\phi\cos\alpha & \sin\omega\sin\phi\cos\alpha - \cos\omega\sin\alpha & \cos\omega\sin\phi\cos\alpha + \sin\omega\sin\omega\sin\alpha & T_x \\ \cos\phi\sin\alpha & \sin\omega\sin\phi\sin\alpha + \cos\omega\cos\alpha & \cos\omega\sin\phi\sin\alpha - \sin\omega\cos\alpha & T_y \\ -\sin\phi & \sin\omega\cos\phi & \cos\omega\cos\phi & T_z \end{bmatrix} \tag{6}$$

, where

- f is the focal length,
- c_x and c_y are the coordinates of the optical center,
- ϕ, ω and α are the rotations around the x, y and x axis, respective,
- T_x, T_y, T_z are the translations on x, y and z directions.

The coordinates are projected at 1.2m from the ground which corresponds to a best effort of the height of the center of mass of an average person, when seen in most frequent positions of the field. This projection allows to have a more correct measure of the players' positions and enables the information fusion between cameras.

3.3 Player tracking

The player tracking is based on a vector of Kalman filters (Kalman & Others, 1960; Welch & Bishop, 2002) (one per player) with state x_k (Eq. 7) and measure z_k (Eq. 8) at instant time k.

$$x_k = [\, x\ y\ v_x\ v_y \,]^T \tag{7}$$

$$z_k = [\, x\ y \,]^T \tag{8}$$

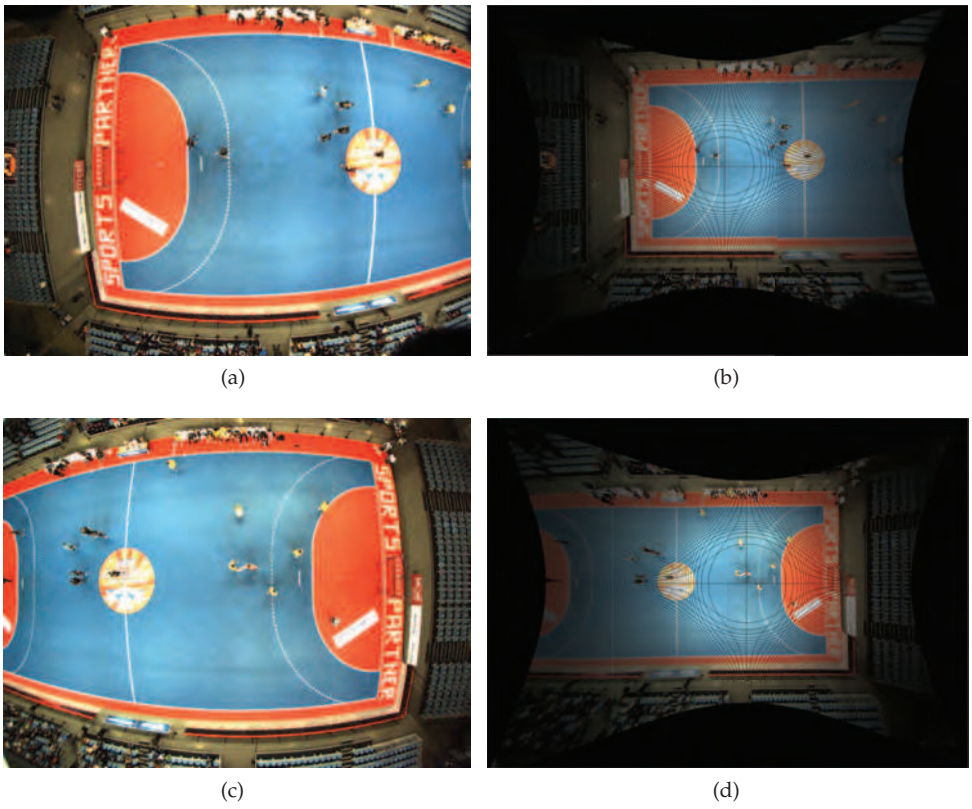
, where

- x and y are the player center of mass position in real world coordinates,
- v_x and v_y are the player velocity in real world coordinates.

And modelled according to the following linear stochastic difference equations (Eq. 9, 10).

$$x_k = Ax_{k-1} + w_{k-1} \tag{9}$$

$$z_k = Hx_k + v_k \tag{10}$$



(a) (b)

(c) (d)

Fig. 3. (a) and (b) left image before and after removing the barrel effect distortion. (c) and (d) right image before and after removing the barrel effect.

Where A is the state model matrix and assumes the form of Eq.11 (where Δt is the time between frames - in this case corresponds to $\frac{1}{30}$ seconds), H is the observation model matrix (Eq.12) and the random variables w_k and v_k represent the process and measurement noise. The usage of real world coordinates allows for a transparent tracking between the two video streams.

$$A = \begin{bmatrix} 1 & 0 & \Delta t & 0 \\ 0 & 1 & 0 & \Delta t \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 1 \end{bmatrix} \tag{11}$$

$$H = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 1 & 0 & 0 \end{bmatrix} \tag{12}$$

Whenever the user indicates a player (using the mouse), a new Kalman filter is added to the vector with the real world position of the player and a default velocity of 0 m/s. Afterwards, the players' locations on the subsequent frames are predicted using Eq.9. The area around

the predicted measure is searched according to the process explained on 3.2.3 to generate a measure (z_k) to updated the estimate.
In addition, and since the players' velocity is not constant through out the game, at each frame each player velocity is updated.
By predicting the position of the players on the subsequent frames it is possible to reduce the computational cost because only a few regions of the entire image are search for players.

3.4 Generation of human meaningful video

This chapter concerns primarily with the generation of a single complete image video stream that is of the utmost importance for the possible human end-users of our system, for example a sports' scientist, educator or coach.
Generating a single, high quality, "undistorted", video stream is a complex task due to the usage of complex optical systems (wide angle zoom lenses) and the need for accuracy of the system that involves dealing with 2 sets of intrinsic and extrinsic camera parameters for the so called pinhole models of the cameras. High accuracy mapping of the pixels of the images onto real world is needed, with the added difficulty of covering large real world areas. This is an even greater task given that high resolution and high frame rate are of interest, thus producing large amounts of data (2 cameras @ 1024 x 768 resolution, RGB colour depth @ 30fps). By using advanced camera calibration techniques, the two different sets of parameters were found, thus allowing image to world accurate mapping (on separate images) - as seen in Fig. 3.
In order to produce the "undistorted" Human Meaningful Video stream, the first task is to optimize the algorithm used. Firstly a pair of "static", off-line created, Look Up Tables (LUTs) are created to map real world pixels into their origin in the original images. Mapping non-overlapped image areas is not complex if all data is available. Additional processing is necessary for the overlapped parts of the image, in order to get a human meaningful image (without neither repeated nor cut objects). LUTs are very useful because they exchange complex mathematical operations with memory storage for the repeated computations which is very interesting in terms of general performance and parallel computing in particular.
One of the issues to be solved is to show an interesting view of the overlapped portions of the images shown in Fig. 1. This is an issue because near the centre line of the handball field, where images overlap, the same player is seen tilted in opposite directions by both cameras (as shown in Fig. 4). The strategy is to have complete objects always from one of the cameras - this is always possible because the largest object is inferior in image size to the size of the overlap in the images (see Fig. 4). It is not computationally immediate, though, because objects and background have to be identified and reconstructed.

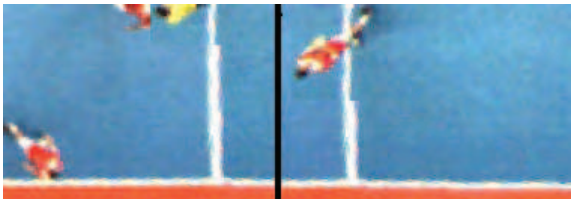


Fig. 4. Same player over time, transitioning from camera to the other.

The implementation was built in C++, under windows and Microsoft Visual C++, using previously mentioned frameworks as OpenMP (OpenMP, 2011) and Compute Unified Device Architecture (CUDA) (NVIDIA, 2011). An example of the processed image is shown in Fig. 5.

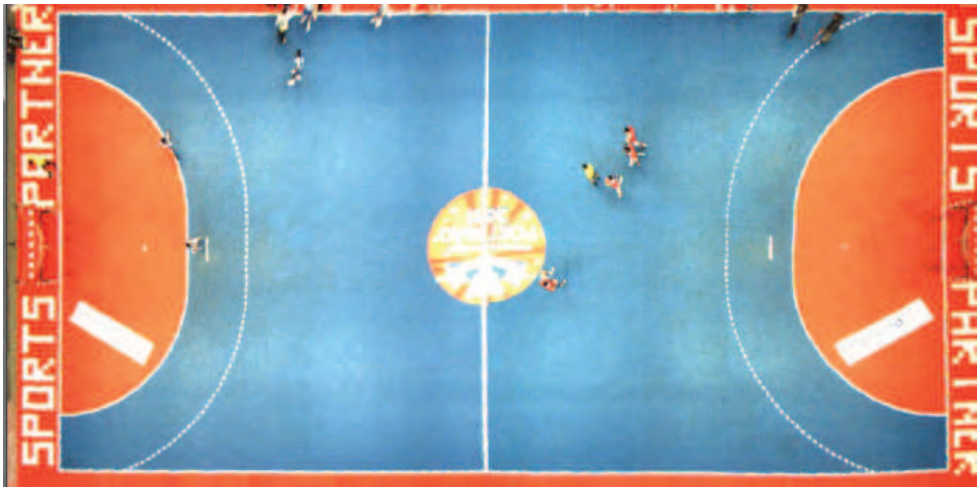


Fig. 5. Single frame from the ”undistorted” human meaningful video.

4. Results

4.1 Player detection and tracking

In order to validate the approach, the system was mounted at the public sports hall of Portimão to film the games of the portuguese SuperCup. The video footages collected validated the engineering solution since we were able to cover the entire field with good resolution and a good overlapped zone as shown on Fig. 1.

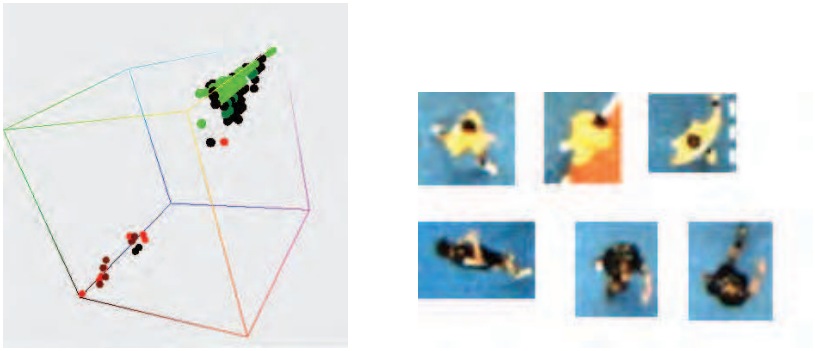
Initially two distinct teams (team A and team B - examples of both teams can be seen on Fig. 6(b)) were calibrated as explained on section 3.2.1. The original seeds were selected by clicking on the players of both teams which resulted on the initial colour subspaces illustrated on Fig. 6(a).

After 1200 frames and a time between expansions (t_{expans}) of 30 frames (which corresponds to 34 expansions), it is possible to verify that the teams' colour subspaces have updated and grown around the initial seeds resulting on the new colour subspaces of Fig. 7.

Table 3 provides an overview of the colour spaces dimensions during the auto calibration process. The @F represents the frame number (remember that the auto expansion occurs at every 30^{th} frame).

The table clearly illustrates that from the initial colour subspaces (Fig. 6(a)) to the final (Fig. 7) the number of colour triplets that are seeds (C_S) increases so that the colour subspaces can adapt to the colour conditions of that team through the entire field and through time.

In addition, if the colour subspace is more condensed, triplets that resemble the colour (C_L) and are colour (C_F) tend to exist in higher number (Team B) than when the colour seeds are more spread in the entire colour space (Team A).



(a) (b)

Fig. 6. (a) Initial colour subspaces. Green dots correspond to team A and red dots to team B. Lighter dots are seed colours (C_S), intermediate are team colour (C_F) and the darkest resemble the team colour (C_L). (b) Examples of players from team A (up) and team B (down).

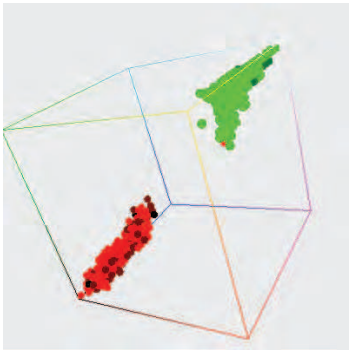


Fig. 7. Final colour subspaces after 34 auto expansions (team A in green and team B in red).

Team	B	@F1	@F5	@F31	@F35	@F61	@F301	@F421	@F691	@F811	@F1021
A	C_L	34	28	12	10	10	10	3	1	4	0
	C_F	6	6	6	6	8	4	9	11	3	7
	C_S	39	39	44	44	51	81	92	105	109	112
B	C_L	2	2	1	0	1	2	4	7	7	3
	C_F	6	6	6	6	6	11	18	42	44	43
	C_S	8	8	12	12	12	25	39	103	116	149

Table 3. Evolution of the number of colour triples that belong to each team and the respective belonging degree. The expansion is performed at every 30^{th} frame ($t_{expans} = 30$)

It is also possible to verify the effect of colour persistence, namely the number of colour triplets that are C_L decreases on Team A from frame 1 to frame 5 (the persistence of these colours is of 3.75 ($p_{S_c} = \frac{1}{8} t_{expans}$)) also they tend to have a more erratic behaviour because they belong to the periphery of the colour subspace and therefore do not appear so often, and triplets with C_F decrease on Team A from frame 61 to frame 301 and on Team B from frame 811 to frame 1021.

However, it is possible to verify that the initial seed choice will influence how well the colour subspace adapts to the environment conditions. In fact, the initial seeds for team A resulted in a faster adaptation: the colour subspace growth is higher at the initial expansions and tends to stabilize, while for team B there is still a reasonable growth even at frame 1021. The initial seed choice is a well known problem of region growing methods.
Comparing the results with and without the Fuzzy inspired model of colour expansion it is possible to verify that the players' detection achieves better results with the mutable colour subspaces as depicted on Fig. 8.
These images show the players' detection at frame 762, where the green crosses correspond to players detected from team A and the purple crosses correspond to players detected from team B. The green and red highlighted pixels correspond to pixels that have been labelled as belonging to one of the teams (green correspond to team A and red to team B).
Analysing the two images it is possible to verify that using the Fuzzy inspired auto expansion model all fielders from both teams were detected (Fig. 8(b)), while using the initial colour subspaces (Fig. 6(a)) the system is unable to detect four players of team B (Fig. 8(a)). In addition, the detected area of the players is higher with the Fuzzy model which allows to have a better measure of the player center of mass.
Moreover, the detection rate increases along with the colour subspaces update as illustrated on Table 4. This increase rate is more visible on players from team B, since its initial colour subspace did not reflect so well the colour properties of the team.

Team	player	1-100	101-200	201-300	301-400	401-500	501-600	601-700	701-800
A	1	1.00	1.00	1.00	1.00	1.00	1.00	1.00	1.00
	2	1.00	1.00	1.00	0.95	1.00	1.00	1.00	1.00
	3	1.00	1.00	1.00	0.96	1.00	1.00	1.00	1.00
B	1	0.00	0.27	0.00	0.70	0.63	1.00	0.93	1.00
	2	0.03	0.35	0.33	0.49	0.92	0.90	0.90	1.00
	3	0.13	0.43	0.24	0.56	1	1	0.98	0.99

Table 4. Player detection of three players from teams A and B from frame 1 until frame 800.

The usage of Kalman filters to perform the tracking allows not only to make the processing time shorter, but also to minimize the miss-detections because the predictive stage (Eq. 10) determines the next position of the player (taking into account the model of the player movement). This position is used on the next frame to search for the blob and in case of not finding a measure the value is used as the position of the player. The following table (Table 5) shows how the tracking of players from team B is improved using the Kalman filter.

Player	1-100	101-200	201-300	301-400	401-500	501-600	601-700	701-800
1	0.00	0.79	0.15	1.00	1.00	1.00	0.95	1.00
2	0.16	0.83	0.57	0.90	1.00	0.98	0.93	1.00
3	0.74	0.86	0.75	1.00	1.00	1.00	0.99	1.00

Table 5. Detection improvement using the Kalman filter of the three players of team B from Table 4.

Additionally, by performing the tracking in real world coordinates it is possible to track the players between cameras in a transparent way. Figure 9 illustrates players from team B crossing the middle line and being identified initially by the left camera and afterwards by the right camera, without the need for user intervention.

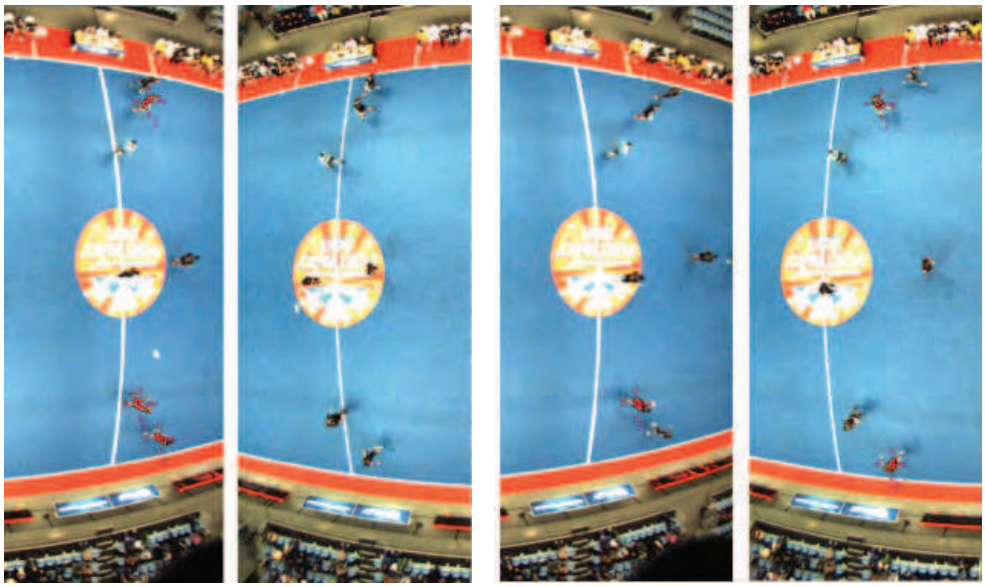
(a)

(b)

Fig. 8. Results of the player detection at frame 762: (a) without colour auto expansion and (b) with colour auto expansion.

4.2 Generation of human meaningful video

In order to ascertain the importance of parallel processing in this application, the same algorithm was ported to the OpenMP and CUDA frameworks, yielding the results shown in Table 6. Naturally, parallel processing is heavily dependent on hardware and results are shown for two laptop solutions, only one of each is able to run the proprietary CUDA toolkit for GPU processing.



(a) (b)

Fig. 9. Tracking players between cameras. (a) Players are being tracked by the left camera. (b) Some players start to be tracked by the right camera.

PC (Year) / Processor / Video Processor / O. S. / CPUs (GPUs)	OMP× CUDA×	OMP✓ CUDA×	OMP× CUDA✓	OMP✓ CUDA✓
Asus V6J (2006) Intel Core Duo no CUDA capabilities Windows 7 2@1.87GHz (none)	145.1+42.1 = 187.2 ms (5.3 fps)	137.9+34.5 = 172.4 ms (5.8 fps)	-	-
Toshiba Tecra S11 (2011) Intel core i7-640M NVidia NVS2100M 512MB Windows 7 4@2.8+GHz (16@0.5+GHz)	91.1+69.7 = 160.8 ms (6.2 fps)	40.7+28.6 = 69.3 ms (14.42 fps)	91.1+22.2 = 113.3 ms (8.83 fps)	40.7+21.5 = 62.5 ms (16.02 fps)

Table 6. Parallel computing execution times: the parcels (a+b) are: a - other algorithms; b - undistorting and correctly joining incoming images. The × indicates NO and the ✓ indicates WITH.

By studying Table 6 and Figure10, it can be found that a single better CPU improves performance marginally - a much better CPU running with a clock at least 1.49 times faster yields about 17% performance gain. This is probably due to the large amount of data being manipulated in this application - the limitations are in the memory area, not pure single CPU power. The same results also demonstrate the advantage of parallel computing with joint usage of OpenMP (OMP) and CUDA, with a performance gain little over 2.6 times (in the same computer). Even larger gains were initially expected but the complexity of the algorithm on the selection of the objects to show on the overlapped portion of the initial images, limited

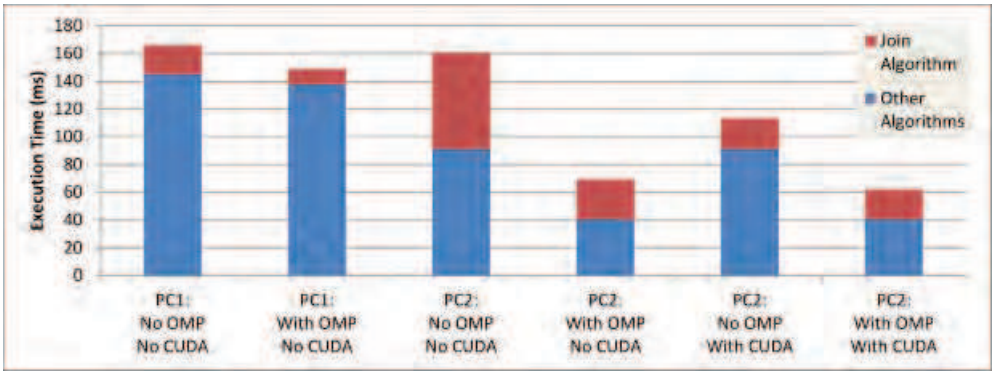


Fig. 10. Execution times for several parallel computing techniques (see Table 6).

considerably the overall performance. As recent processors offer high frequency dual/quad (or more) cores, the interest of using 16 much slower processors is also to be considered - but using the GPU processors of the video card will free up the main CPU for other tasks.
The previous study is related to producing a single frame of the human meaningful video stream with parallel computing. Future work includes using distributed computing to producing the stream, for example, using 2 computers to produce alternating frames is expected to produce excellent performance gains (almost halving processing time). This expectation is justified as little dependencies exist among consecutive frames and, as such, parallel processing is most effective and the "slow" transmission over network would not introduce significant performance loss, when compared to the expected benefits of the double computational power available for processing. A similar approach (several frames at once) would also be possible in a single computer but would likely not be as interesting as most available resources are used fully most of the time. As mentioned earlier, actual performance gains are very much hardware dependent.
In order to really have a meaningful video for humans, the found objects (handball players, etc) have to be marked onto the image, a task that was not parallelized.

5. Conclusions

This chapter presented a visual automatic system for detecting and tracking players in indoor games. The main objectives were to implement a system that could be adaptative and take into consideration the light changes (and subsequent colour change) that frequently occurs in sports pavilions (for example due to windows, clouds, sun orientation, ...). The ultimate goal is to further improve information for sports agents and sportive quality.
Players are identified by color segmentation. The used techniques include foreground detection and classification of team colours using a Fuzzy inspired categorization model that allows a single colour to belong to both teams simultaneously. The colour subspaces have no particular shape and grow or shrink over time in order to take into account spatial and temporal changes of the recognized colours.
Player tracking is improved by making use of a Kalman Filter per player and the resulting information is organized and shown in a single undistorted image view of the entire field,

adequate for human studies. Although the system is multi-camera, the usage of parallel processing allows efficient generation of this final image.
The proposed methodology was validated with a real game footage filmed at an important Portuguese handball championship. Results show that, using simple features such as colour combined with a powerful tool for tracking (Kalman Filter), it is possible to detect and track players throughout the game area with very limited user intervention - initial colour calibration by user and little more. The usage of adaptative colour subspaces generated by a Fuzzy inspired methodology allows to better define the teams' colour properties during the game and increasing detection rates.

5.1 Future work

Future work includes exploring methodologies to automatically define the time between expansions along the game according to the colour subspaces dynamics and also the detection rate and incorporate more information on the Kalman Filter model in order to make it more robust to long merging and occlusion situations and also enable both the detection and the tracking with the benefits of parallel processing.
Additionally, interesting game sequences are also intended to be automatically detected (taking into consideration the players positions) and the generated video tagged accordingly. This video is to be made "active" (searchable, "jumpable", ...) to be integrated into a human interface navigation tool to allow for an easy usage of the extracted information by the final end users.

6. Acknowledgments

We would like to thank Fundação Calouste Gulbenkian by the support given through a PhD scholarship with ref. 104410.

7. References

Alahi, A., Boursier, Y., Jacques, L. & Vandergheynst, P. (2009). Sport players detection and tracking with a mixed network of planar and omnidirectional cameras, *Distributed Smart Cameras, 2009. ICDSC 2009. Third ACM/IEEE International Conference on*, IEEE, pp. 1–8.

APIDIS (2008). Autonomous Production of Images based on Distributed and Intelligent Sensing.
URL: *http://www.apidis.org/index.htm*

Barron, J. & Thacker, N. (2005). Tutorial: Computing 2D and 3D optical flow, *Tina Memo Internal* (2004-12).

Canny, J. (1986). A computational approach to edge detection, *IEEE Trans. Pattern Anal. Mach. Intell.* 8: 679–698.

Chapman, B., Jost, G. & Van Der Pas, R. (2007). *Using OpenMP: portable shared memory parallel programming*, Vol. 10, The MIT Press.
URL: *http://books.google.com/books?hl=en&lr=&id=MeFLQSKmaJYC&oi=fnd&pg=PR7&dq=Using+OpenMP,+Portable+Shared+Memory+Parallel+Programming&ots=5zOPjR26VC&sig=R3WOZRMwGX1tAw-pcR46NuId3xw*

Cheng, H., Jiang, X., Sun, Y. & Wang, J. (2001). Color image segmentation: advances and prospects, *Pattern recognition* 34(12): 2259–2281.

Delannay, D., Danhier, N. & De Vleeschouwer, C. (2009). Detection and recognition of sports (wo)men from multiple views, *Distributed Smart Cameras, 2009. ICDSC 2009. Third ACM/IEEE International Conference on*, IEEE, pp. 1–7.

Deng, Y. & Manjunath, B. (2001). Unsupervised segmentation of color-texture regions in images and video, *IEEE Transactions on Pattern Analysis and Machine Intelligence* 23(8): 800–810.

Franks, I. M. & Nagelkerke, P. (1988). The Use of Computer Interactive Video in Sport Analysis, *Ergonomics* 31(11): 1593–1603.

Franks, I., Willison, G. E. & Goodman, D. (1987). Analysing a team sport with the aid of computers, *Canadian Journal of Sport Sciences* 12(2): 120–125.

Grimson, W., Stauffer, C., Romano, R. & Lee, L. (1998). Using adaptive tracking to classify and monitor activities in a site, *Computer Vision and Pattern Recognition, 1998. Proceedings. 1998 IEEE Computer Society Conference on*, IEEE, pp. 22–29.

Group, K. (2011a). The khronos group inc.
URL: *http://www.khronos.org/*

Group, K. (2011b). Opencl - the open standard for parallel programming of heterogeneous systems.
URL: *http://www.khronos.org/opencl/*

Halfhill, T. (2008). Parallel Processing with CUDA Nvidia's High-Performance Computing Platform Uses Massive Multithreading, *Microprocessor Report* 22: 1–8.
URL: *http://scholar.google.com/scholar?hl=en&btnG=Search&q=intitle:Parallel+Processing+With+CUDA:+Nvidia's+High-Performance+Computing+Platform+Uses+Massive+Multithreading#0*

Heikkila, J. & Silvén, O. (1999). A real-time system for monitoring of cyclists and pedestrians, *Visual Surveillance, 1999. Second IEEE Workshop on,(VS'99)*, IEEE, pp. 74–81.

Hu, M., Chang, M., Wu, J. & Chi, L. (2011). Robust Camera Calibration and Player Tracking in Broadcast Basketball Video, *Multimedia, IEEE Transactions on* 13(2): 266–279.

Kalman, R. & Others (1960). A new approach to linear filtering and prediction problems, *Journal of basic Engineering* 82(1): 35–45.

Kasiri-Bidhendi, S. & Safabakhsh, R. (2009). Effective tracking of the players and ball in indoor soccer games in the presence of occlusion, *Computer Conference, 2009. CSICC 2009. 14th International CSI*, IEEE, pp. 524–529.

Koprinska, I. & Carrato, S. (2001). Temporal video segmentation: A survey, *Signal processing: Image communication* 16(5): 477–500.

Kristan, M., Perš, J., Perše, M. & Kovačič, S. (2009). Closed-world tracking of multiple interacting targets for indoor-sports applications, *Computer Vision and Image Understanding* 113(5): 598–611.

Monier, E., Wilhelm, P. & Ruckert, U. (2009). Template matching based tracking of players in indoor team sports, *2009 Third ACM/IEEE International Conference on Distributed Smart Cameras (ICDSC)* pp. 1–6.

Munkres, J. (1957). Algorithms for the Assignment and Transportation Problems, *Journal of the Society for Industrial and Applied Mathematics* 5(1): 32–38.

Needham, C. & Boyle, R. (2001). Tracking multiple sports players through occlusion, congestion and scale, *British Machine Vision Conference*, BMVA, pp. 93–1022.

NVIDIA (2011). Cuda zone.
URL: *http://www.nvidia.com/object/cuda _home _new.html*

OpenMP (2011). OpenMP.org.
URL: *http://openmp.org/wp/*

Roberts, L. G. (1963). *Machine Perception of Three-Dimensional Solids*, Outstanding Dissertations in the Computer Sciences, Garland Publishing, New York.

Santiago, C., Sousa, A. & Reis, L. (2011). Real time colour based player tracking in indoor sports, *Computational Vision and Medical Image Processing* 19: 17–35.

Sobel, I. E. (1970). *Camera models and machine perception*, PhD thesis, Stanford, CA, USA. AAI7102831.

Stelzer, A., P. K. & Fischer, A. (2004). Concept and Application of LPM - A Novel 3-D Local Position Measurement System, *IEEE Transactions on Microwave Theory Techniques* 52: 2664–69.

Welch, G. & Bishop, G. (2002). An introduction to the Kalman filter, *Technical report*, Department of Computer Science University of North Carolina at Chapel Hill, EUA, North Carolina.

Cutting Edge Research in New Technologies

Edited by Prof. Constantin Volosencu

ISBN 978-953-51-0463-6
Hard cover, 346 pages
Publisher InTech
Published online 05, April, 2012
Published in print edition April, 2012

The book "Cutting Edge Research in New Technologies" presents the contributions of some researchers in modern fields of technology, serving as a valuable tool for scientists, researchers, graduate students and professionals. The focus is on several aspects of designing and manufacturing, examining complex technical products and some aspects of the development and use of industrial and service automation. The book covered some topics as it follows: manufacturing, machining, textile industry, CAD/CAM/CAE systems, electronic circuits, control and automation, electric drives, artificial intelligence, fuzzy logic, vision systems, neural networks, intelligent systems, wireless sensor networks, environmental technology, logistic services, transportation, intelligent security, multimedia, modeling, simulation, video techniques, water plant technology, globalization and technology. This collection of articles offers information which responds to the general goal of technology - how to develop manufacturing systems, methods, algorithms, how to use devices, equipments, machines or tools in order to increase the quality of the products, the human comfort or security.

INTECH
open science | open minds

InTech Europe
University Campus STeP Ri
Slavka Krautzeka 83/A
51000 Rijeka, Croatia
Phone: +385 (51) 770 447
Fax: +385 (51) 686 166
www.intechopen.com

InTech China
Unit 405, Office Block, Hotel Equatorial Shanghai
No.65, Yan An Road (West), Shanghai, 200040, China
中国上海市延安西路65号上海国际贵都大饭店办公楼405单元
Phone: +86-21-62489820
Fax: +86-21-62489821