Parallelization of a Multipartite Graph Matching Algorithm for Tracking Multiple Football Players

M. Villalta and F. Siles
Pattern Recognition and Intelligent Systems Laboratory
Department of Electrical Engineering, School of Engineering
Universidad de Costa Rica, San José, Costa Rica
marco.villalta@ucr.ac.cr, francisco.siles@ucr.ac.cr

***Abstract*—This work describes the parallel methodology for a football tracking algorithm based on multipartite graphs using MPI and OpenMP. The proposed algorithm use a consumer-producer scheme to overlap the computing time of the two main procedures of the tracking algorithm: segmentation and tracking; as well a send and receive communication pattern to propagate the blob identities. We show how an hybrid system of data and task parallelization improves the execution time for 4K videos, achieving a speedup equal to 19.24 and a processing speed of 21.71 FPS with 128 threads.**

***Index Terms*—Parallel algorithms; Association Football; Temporal Segmentation; Tracking of Football Players**

I. Introduction

Football is a sport that has shown great popularity over the years, with an estimated 3.5 billion fans all over the world [1]. In the same way as technology has been introduced into everyday life activities, there is also a desire to incorporate it into the sports' field. Such involvement has led to the fact that in recent years, there has been an increased interest for the development of advanced computer systems that favor the automated analysis of sport's results. This information is increasingly important, not only for the general viewer for entertainment purposes, but also for professional football clubs, and their coaching teams, as well as for the sport sciences community, since it will lead to perform physical assessments, fatigue detection, analysis of opponents, evaluation of tactical performance, and others [2]. In order to build such automated analysis systems, one of the first steps consists of the extraction of relevant positional information of the players, which in fact requires primarily their trajectories on the field at any given time. This process of following each player to reconstruct their 3D trajectories is called player tracking. Since Football is a team sport, then the player tracking stage requires detecting multiple players, finding their positions at regular intervals, and associating spatial and temporal information to extract their trajectories [3]. This task is complex because of the patterns of unpredictable movements of the players during the game, in addition to the fact that the players look quite similar among those belonging to the same team, and frequently, the players find themselves in the struggle for possession of the ball creating multiple occlusions. Besides the tracking of players is affected also by external factors such as environmental conditions like rain, light changes, and stadium' shadows. The ability to obtain visual quality and the accuracy of tracked targets is highly desired in a tracking system, this requires a high resolution of the input data that provide more details to the tracker. Low quality or small resolution of videos, system noise, small objects and other factors generate less precise tracking results. In case of such conditions an object would be more difficult to identify and track because the object has less information [4], [5]. The present work is part of ACE, a larger system still under development, to meet the needs mentioned above. ACE is a computational platform for analyzing digital videos of football [6], which generates abstract models of the game for interpretation. This platform is implemented in a multilayer architecture, as shown in figure 1. The first four layers are related to the perception stages, while the last two are related to the semantic analysis. Since the platform was initially developed for videos taken from TV Broadcast [7], [8], [9], a *temporal segmentation* process was required, in order to select the candidate scenes (far-view scenes) where information of the players' positions can be extracted [10], [11]. The current work is focused on the lower stages: from the acquisition of the video to the tracking stage, without going through the temporal segmentation, since the input video comes from the concatenation of two digital videos taken using 2 static 4K cameras placed in the stadium (see figure 4).

Given the configuration of the two 4K cameras that produce the input data to the system, and since each image from each camera has an UHD (ultra high definition) of 3840×2160, and the frame rate of 60 Hz, then the pixels per second (pps) to process in order to cope with the incoming information is of 995 328 000 pps. Therefore it is necessary to consider strategies that accelerate the processing of this information. In this paper we present a proof of concept for a parallel hybrid approach to an off-line football tracker based on multipartite graphs for the ACE platform.

II. Related Work

A study of the different types of trackers for football players is beyond the scope of this work, for that see [13]. Our scope is limited to those that present computational costs' results.

Certainly the main objective of a football player tracker is the precise identification and track of every object on the field but the computational cost and latency has already been considered in the professional literature. These considerations

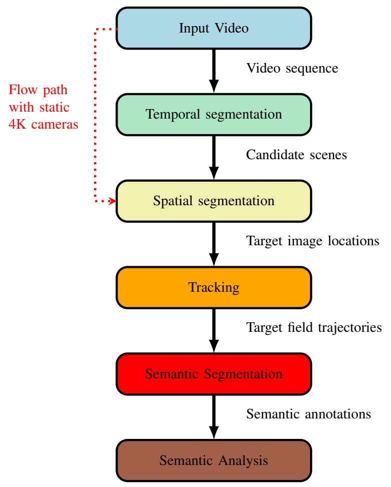

Figure 1: Layered architecture of ACE-Football. Edited from [12].

are found since 2000 where Lefèvre et al. [14] created a method based on a fast snake obtaining a processing time of 2 seconds per frame, with a implementation on Matlab for a sequence of 100 images of 24 bits, with width and height of 384 and 288 pixels respectively. Other works show similar speed performance using Matlab as a development plataform, [15] measures 4 FPS of the motion detection stage for a particle filter tracker, [16] obtain 3 FPS for their visual object tracker, [17] 3.3 FPS in a semi supervised system and [18] for a based adaptive Kalman Filter tracker. Matlab is good for developing and prototyping algorithms but shows lower values of FPS on the implementations of the tracking algorithms, this is because it is a scripting language that uses a JIT compiler to translate a script to machine code.

One approach proposed for a player tracking system using a model field particle filter is Sentioscope [3]. Their algorithm is implemented in C++ on GPU using a task parallel technique running the following tasks on different threads: image acquisition, automatic exposure, light adjustment, foreground extraction and HOG calculation; while the player classification and tracking tasks execute in parallel with a gathering pattern. They mention that the system runs on a laptop with a 4 core Intel i7 CPU achieving a ratio of 14 FPS.

There are similar data association algorithms to the algorithms based on graph theory, for example [19] use a network flow formulation as a linear programming optimization problem, their algorithm is implemented in C++ using STL. They use a 3Ghz PC and utilized a single core, obtaining a speed of 3.95 FPS for the MCNF version of the algorithm on the ISSIA database.

The image processing stages have a great impact on the execution speed of trackers, therefore the parallelization of the spatial segmentation stage of our algorithm has been discussed previously on [20], where a parallel method is designed relying on multi-threading with OpenMP. They perform several tests using standard definition videos obtaining a speedup factor of 4 and efficiency of 0.1 with 8 threads. This algorithm of the parallel method was implemented in C++ with the OpenCV library.

Despite the effort made in [20], it is considered incomplete for the purposes of a tracking algorithm for being decoupled from this idea, also is an algorithm that does not scale well beyond the presented results. Is interesting to observe how is mentioned in the scientific articles, that the execution times can be improved using parallelization techniques, however they do not consider or use the distributed systems or GPGPU [18], [21], [22]. All of the related work are executed on personal computers and the FPS values presented by these works show that their implementation and resources are not sufficient to achieve a desired processing time (equal or greater than 30 FPS).

III. Implementation

A graph is a mathematical abstract representation consisting of a set of nodes, which represent objects and a set of edges, which represent associations between object pairs ([23]) as shown on figure 2. A graph is k-partite if they can be partitioned into *k* disjoint independent sets. A k-partite graph is called multipartite, typically only when $k \geq 3$ and when $k = 2$ is called bipartite [24]. In our propose method, the nodes are visualized as blobs that might or not correspond to football players, and the edges as the weighted value corresponding to the similarity between blobs. The nodes encapsulate a vector of characteristics that describe the blob using their shape, chromatic information, and position from a kinematic model of constant speed, while the edges are the relation between those blobs from frame to frame.

Under these circumstances we propose a parallel hybrid method to solve the tracking of multiple football players for the data association technique of multipartite graphs, relying on the data parallelization and a fine task parallelization.

Let *V* be a video of *v* frames and *G* a multipartite graph of *k* number of graphs, each one with *n* number of nodes. Where f_n corresponds to the processed frame number on the video *V*, G_f is the first graph and G_l the last graph of a multipartite graph.

As can be seen in the algorithm 1, *V* is processed in groups of frames (batch mode) equals to k frames for each *step* (which is the first processed frame on a group or batch) while $step < v$. The frame fn is read from *V* and is applied the segmentation function σ to generate the contours C, then each contour $c_i \in C$ is used to generate the N_j node model for the graph G_i. Once the multipartite graph has been populated with the k-graphs, a tracking function τ generates the T tracking points, which are 2D locations on the image frame as *(x,y)* coordinates. To propagate identities between consecutive multipartite graphs a bipartite matching function ς is performed between the adjacent graphs.

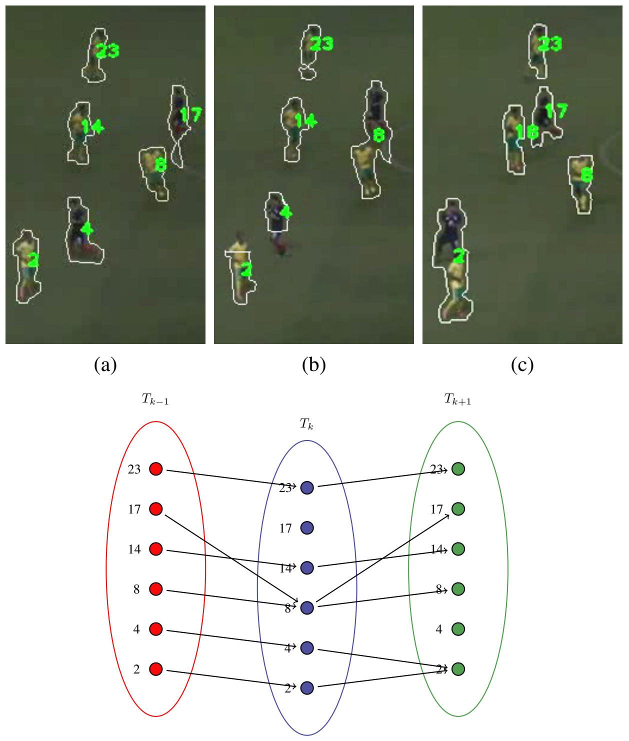

Figure 2: (a) First sequence frame, (b) Middle sequence frame, (c) Last sequence frame and a multipartite graph describing the merge/split events

Algorithm 1 Sequential Multipartite Graph Tracking algorithm

```
    Input V Video
    Output T Tracking points
1:  procedure BATCH PROCESSING
2:      step ← $k$
3:      for $step < v$ do
4:          procedure SEGMENTATION
5:              Initialize G
6:              for $i < k$ do
7:                  $f_n \leftarrow i + step$    ▷ Update first frame number of the multipartite graph
8:                  $C \leftarrow \sigma(f_n)$    ▷ Apply segmentation function to a frame to generate a set of contours
9:                  for $j < n$ do
10:                     Generate $N_j$ model    ▷ Where N is a node
11:                     $G_i[j] \leftarrow N_j$    ▷ Save node model on its corresponding graph
12:         procedure TRACKING
13:             if $step \neq f_f$ then
14:                 $\varsigma$    ▷ Perform bipartite matching function between previous multipartite graph and actual multipartite graph
15:             $\tau(G)$    ▷ Apply tracking function
16:             $G_f \leftarrow G_l$
```

Despite the sequential behavior exhibited by the algorithm 1, it is possible to overlap its two main procedures with a *consumer-producer* scheme where the even processes performs the segmentation function σ and generate the multiparite graphs that the odd processes use on the tracking function τ. Following the same notation as the algorithm 1, it is assumed that there are p processes with a corresponding identification number p_{id}, μ is the maximum number of iterations for those processes, f_a is the start frame for each process, f_o is the stop frame for each process, f_0 the first frame of V, where G_f and G_l are the first and last graph for the batch of process p_{id}.

Algorithm 2 Parallel Multipartite Graph Tracking algorithm

```
    Input V Video
    Output T Tracking points
1:  $\mu \leftarrow v/((p/2) * k)$
2:  procedure BATCH PROCESSING
3:      for $i <= \mu$ do    ▷ Loop until reach max iterations
                             ▷ Compute first and last frame of the multipartite graph for the even and odd processes
4:          if $p_{id}$ % 2 then
5:              $f_a \leftarrow k * p_{id} + i + f_a$
6:              $f_o \leftarrow f_a + k - 1$
7:          else
8:              $f_a \leftarrow k * p_{id} + k * p * i + f_a$
9:              $f_o \leftarrow f_a + k - 1$
10:         procedure PARALLEL SEGMENTATION
11:             if $p_{id}$ % 2 then
12:                 for $j < k$ do
13:                     $f_n \leftarrow i + f_a$    ▷ Current frame number
14:                     $C \leftarrow \sigma(f_n)$    ▷ Apply segmentation function to current frame to find contours
15:                     for $j < n$ do    ▷ Sweep the contours to generate the node models
16:                         Generate $N_j$ model
17:                         $g_i[j] \leftarrow N_j$
18:                 Send multipartite graph G from process
19:                 $p_i$ to process $p_{id+1}$
```

The spatial segmentation algorithm used is an improved version of [20] for tracking purposes, whose results are shown in the figure 3, which includes the following functions:

1) HSV color conversion
2) Normalized Local Spatial Variance for the Hue and Value components
3) Identification of green regions
4) Edge detection
5) Removal of field lines with Hough transformation.
6) Morphological transformations of erosion and dilation.
7) Discard spurious regions and blob filtering using a circularity criterion.
8) Blob model creation within the multipartite graph.

This set of segmentation and filtering functions are observed in the figure 3, where it starts with a raw frame

```
20: procedure PARALLEL TRACKER
21:     if $p_{id}$ % 2 = 1 then
22:         Receive multipartite graph G from process
23:         $p_{id-1}$
24:         if $p \neq 2$ then            ▷ General case of more than 2 processes
25:             if $f_a \neq f_f \vee i\mu$ then
26:                 if $p_{id}1$ then
27:                     Receive graph $G_f$
28:                     from process $p_{id-1}$
29:                 else
30:                     Receive graph $G_f$
31:                     from process $p_{id-2}$
32:                 $\varsigma(G_f)$        ▷ Bipartite matching function to propagate identities
33:             $\tau(G)$        ▷ Apply tracking function
34:             if $i1$ then        ▷ First iteration
35:                 if $p_{id} < p - 1$ then
36:                     Send graph $G\ of\ frame\ f_o$
37:                     from process $p_{id}$ to $p_{id+2}$
38:                 if $i \neq 1 \vee \mu$ then
39:                     if $pid < p - 1$ then
40:                         Send graph $G\,of\,frame\,f_o$
41:                         from process $p_{id}$ to $p_{id+2}$
42:                     else
43:                         Send graph $G\,of\,frame\,f_o$
44:                         from process $p_{id}$ to $p_1$
45:                 else        ▷ Last iteration
46:                     if $pid < p - 1$ then
47:                         Send graph $G\,of\,frame\,f_o$
48:                         from process $p_{id}$ to $p_{id+2}$
49:         else        ▷ Special case of 2 processes
50:             if $fg \neq f_a \cup i \neq \mu$ then
51:                 $\varsigma(fg)$
52:                 $\tau(G)$
53:                 $G_l \leftarrow G(f_o)$
54:             else
55:                 if $i = \mu - 1$ then
56:                     $\varsigma(G_f)$
57:                     $\tau(G)$
58:                 $G_l \leftarrow G(f_o)$
59:
60:
61:
```

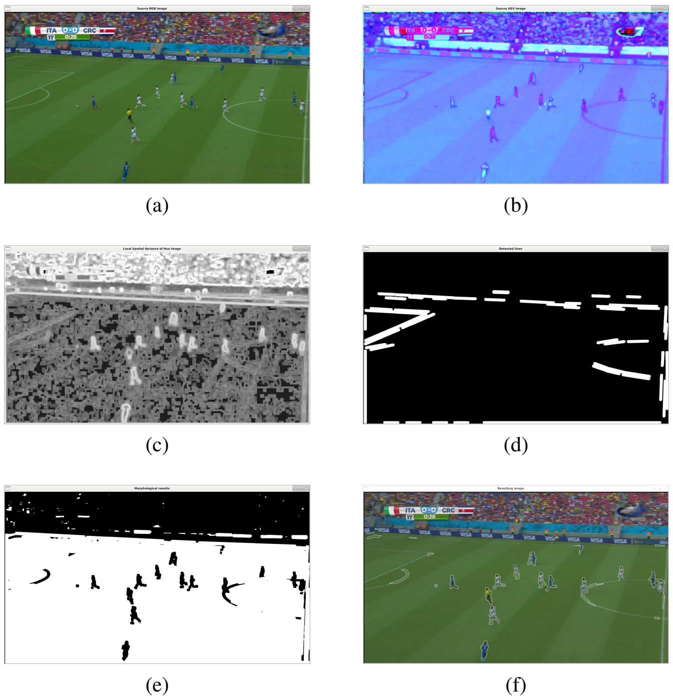

Figure 3: Samples from segmentation procedure: (a) Source frame, (b) HSV frame conversion, (c) Local spatial variance for the Hue component, (d) detected lines, (e) Homogeneous regions with line filtering applied, (f) Resulting segmentated contours

that is then processed to obtain the contours of the players, we use OpenMP as the method of parallelization on each of these functions.

Once, each blob model is created and assigned as a node on the corresponding graph, the tracking algorithm performs the following functions:

1) Pruning the multipartite graph based on the relation between nodes that includes the comparison of static and dynamic descriptors such as: histogram, position, shape, area and speed.
2) Building a coincidence matrix.
3) Forward analysis of borders for merge/occlusion events.
4) Backward analysis of borders for split events.
5) Propagation of blob identities.

The algorithm 2 describes the segmentation and tracking procedures on a batch mode equals to the multipartite graph size. It can be seen from lines 4 to 9 how the distribution of frames is determined for each thread and their corresponding start and ending positions within the video. From lines 18 and 22 the consumer-producer schema is executed, where the parallel segmentation sends multipartite graphs, received by the parallel tracker allowing the overlap of those computational tasks. Another communication pattern between processes is from lines 27 to 31 and from lines 35 to 48 where the send and receive actions are implemented between odd

Table I: Execution times for sequential algorithm

Resolution	Run time (s)	Segmentation time (s)	Track time (s)
4K	2267	2229	38
FHD	542	530	12
HD	286	283	3

Table II: HW specifications of the cluster TARÁ

Node	Quantity	Processor	Ram	HD
Master	1	Intel Xeon E5-2650	62 GB	500 GB
Compute	4	Intel Xeon E5-2650	251 GB	500 GB
Storage	4	Intel Xeon E5-2650	62 GB	40 TB

adjacent processes to transfer a graph, and propagate the blob identities. Between lines 49 and 58 we deal with the special case of two processes: one producer and one consumer.

The generation of the concatenated 4K input video, was performed using an algorithm based on geometric transformations that finds a function which transforms the points in a trapezoid to points of a rectangle in a continuous way. In order to have more information and details of the players, as well as to avoid losing vital information for the tracking process this configuration of cameras was used. An example of the resulting panorama image is shown in figure 4.

The implementation of the segmentation and tracking algorithm was programmed in C++, and have used software tools/libraries consisting of: GCC 4.9.2, openmpi 2.1.3, opencv 2.4.13.5, and the boost library 1.67.0.

IV. Results and Discussion

The experiments were ran on the cluster TARÁ from the PRIS-Lab, the hardware specifications are shown on table II. It has three types of nodes, the master for overall control, the processing nodes, and the storage nodes. All nodes are connected to a private internal network by means of four network interfaces operating in a bonding mode to maximize the available bandwidth. This network topology allowed Lustre, the file system available in the 4 storage nodes, to be in advantage, as this file system has high performance capabilities, and allows to have better reading performance in parallel tasks [25], [26]. Each processing node has 2 Intel Xeon E5-2650 running at 2.0 GHz for a total of 64 cores or 128 threads. For each MPI process 2 OpenMP threads were enabled following a processor architecture affinity criterion, from a previous work [27] it was determined that this was the optimal configuration for the cluster.

Table III: Results for a 4K video

Threads	Run time (s)	Speedup	Efficiency	FPS	Track time (s)
128	117.85	19.24	0.15	21.71	2.07
64	176.69	12.83	0.20	14.48	2.96
32	218.51	14.45	0.45	12.44	3.89
16	426.27	7.41	0.46	6.38	5.20
8	519.44	6.23	0.78	5.39	8.05
4	1215.93	2.69	0.67	2.33	16.91
2	2190.06	1.49	0.75	1.30	37.69
1	2267.00	1.00	1.00	1.27	38.00

Table IV: Multipartite graph size effect

MPG size	Threads	Run time (s)	Speedup	Efficiency	FPS	Track time (s)
20	128	117.85	19.24	0.15	21.71	2.07
10	128	164.27	13.80	0.11	15.58	14.19
5	128	217.96	10.40	0.08	11.74	15.19

Table V: Results for different video resolutions

Resolution	Threads	Run time (s)	Speedup	Efficiency	FPS	Track time (s)
4K	128	117.85	19.24	0.15	21.71	2.07
FHD	128	31.10	24.15	0.19	82.28	2.00
HD	128	17.71	23.99	0.19	144.47	1.95

The speedup was obtained as $S = T_s/T_p$, and the efficiency as $E = S/p$, where T_s is the sequential running time, T_p the parallel running and p as the number of threads. The results of both metrics are shown on figures 5 and 6 for a 4K video sequence, with a resolution of 3648×512, and 2877 frames of length. The results are tabulated in the table III. We inspect the effect of different sizes of multipartite graphs and dimensions of videos on the speedup and efficiency in tables IV and V for 128 fixed number of threads,

From tables I, III, V and IV is observed that most of the execution time is spent on the segmentation part of the algorithm. The growth of acceleration based on the size of the multipartite graph is probably due to a smaller amount of communications between adjacent nodes under the cost of a greater use of memory. The information of table III, figure 5 and figure 6 shows how consistently there is a speedup, but with a penalty in the efficiency.

V. Conclusions and Future Work

In this paper, we deal with the natural sequential process of tracking football players, exploring the hybrid model formed by the combination of MPI and OpenMP parallelism.

Although the complexity of the algorithm 2 is greater than the algorithm 1, the benefits associated with the execution times lead us to the conclusion that a parallel hybrid system must be considered for the conditions of this application, this is the main contribution of our work.

As shown in figure 5, our algorithm has good prospects to scale to a greater number of processes, however caution should be taken with the size of the graph in terms of memory consumption, despite of showing better results by increasing its size according to table IV. The speedup appears to be linear inside a node, but when more nodes are used there is a penalty for the network communication. Also the efficiency values show better results than others methods for a more complex version of the segmentation algorithm.

As we demonstrated from the results of tables I, III, V and IV, the segmentation section is the bottle neck of the algorithm requiring heavy computational power, for future work we expect better results with a GPU version of the segmentation and tracking algorithm in conjunction with the distributed model. Also it is important to explore the use of OpenMP in accelerators such as XeonPhi cards for this type of algorithms.

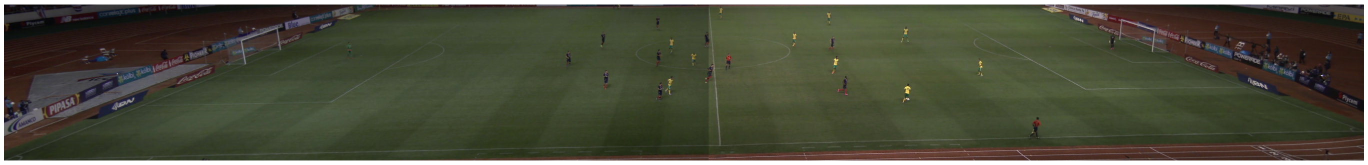

Figure 4: Input video frame.

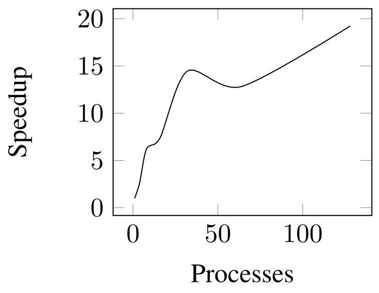

Figure 5: Speedup for a 4K video.

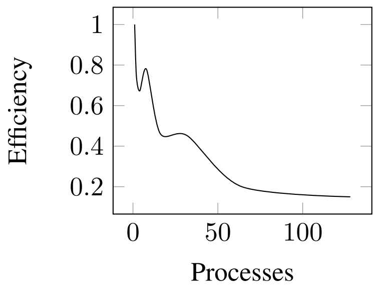

Figure 6: Efficiency for a 4K video.

References

[1] R. Wood, "Top 10 list of the world's most popular sports," http://www.topendsports.com/world/lists/popular-sport/fans.htm, May. 2017. [Online]. Available: http://www.topendsports.com/world/lists/popular-sport/fans.htm

[2] R. Radakovic, M. Dopsaj, and R. Vulovic, "The reliability of motion analysis of elite soccer players during match measured by the tracking motion software system," *IEEE 15th International Conference on Bioinformatics and Bioengineering*, Nov. 2015.

[3] S. Baysal and P. Duygulu, "Sentioscope: A soccer player tracking system using model field particles," *IEEE Transactions on Circuits and Aystems for Video Technology*, vol. Vol. 26, no. No. 7, 2016.

[4] Y. Habibi, D. R. Sulistyaningrum, and B. Setiyono, "A new algorithm for small object tracking based on super-resolution technique," *AIP Conference Proceedings*, vol. 1867, no. 1, p. 020024, 2017. [Online]. Available: https://aip.scitation.org/doi/abs/10.1063/1.4994427

[5] P. Figueroa, N. Leite, and R. M. L. Barros, "Tracking soccer players using the graph representation," in *Proceedings of the 17th International Conference on Pattern Recognition*, 2004, pp. 787–790.

[6] F. Siles, "Ace-football analysing football from tv broadcasting." *24th German Soccer Conference DVS FuSSball in Lehre und Forshung, Weiler*, 2013.

[7] R. C. Quesada and F. S. Canales, "Evaluation of different histogram distances for temporal segmentation in digital videos of football matches from tv broadcast," *2017 International Conference and Workshop on Bioinspired Intelligence*, 2017.

[8] S. C. Ramírez and F. S. Canales, "Deceived bilateral filter for improving the classification of football players from tv broadcast," *Bio-inspired Intelligence (IWOBI), 2014 International Work Conference on*, pp. 98 – 105, 2014.

[9] F. Siles, "Shot classification for association football from tv broadcasting," *Applied Machine Intelligence and Informatics (SAMI), IEEE 12th International Symposium on*, 2014.

[10] R. C. Quesada, S. C. Ramírez, and F. Siles, "Improving the temporal segmentation in digital videos using the deceived bilateral filter," *IEEE 36th Central American and Panama Convention(CONCAPAN)*, 2016.

[11] F. Siles, "Temporal segmentation of association football from tv broadcast." *INES 2013*, 2013.

[12] F. Siles Canales, "Automated semantic annotation of football games from tv broadcast," Ph.D. dissertation, Uni München, 2014.

[13] M. Manafifard, H. Ebadi, and H. A. Moghaddam, "A survey on player tracking in soccer videos," *Computer Vision and Image Understanding*, 2017.

[14] S. Lefèvre, C. Fluck, B. Maillard, and N. Vincent, "A fast snake-based method to track football players," *IAPR International Workshop on Machine Vision Applications*, 2000.

[15] R. C. H. de Vos and W. Brink, "Combining motion detection and hierarchical particle filter tracking in a multi-player sports environment," *Proceedings of the 20th Symposium of the Pattern Recognition Association of South Africa*, 2009.

[16] Y. Yuan, S. Emmanuel, and Y. Fang, "Visual object tracking based on backward model validation," in *IEEE Transactions on Circuits and Systems for Video Technology*, vol. 24, 2014.

[17] R. Martín and J. M. Martínez, "A semi-supervised system for players detection and tracking in multi-camera soccer videos," *Multimedia Tools Appl.*, vol. 73, no. 3, pp. 1617–1642, Dec. 2014. [Online]. Available: http://dx.doi.org/10.1007/s11042-013-1659-6

[18] N. Najafzadeh, M. Fotouhi, and S. Kasaci, "Multiple soccer players tracking," *2015 International Symposium on Artificial Intelligence and Signal Processing*, 2015.

[19] H. B. Shitrit, J. Berclaz, and F. Fleuret, "Multi-commodity network flow for tracking multiple people," *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. Vol. 36, no. Issue 8, pp. 1614 – 1627, Aug. 2014.

[20] F. Siles and J. C. Saborío, "Parallel spatial segmentation for the automated analysis of football," *5th IEEE International Workshop and Conference on Bioinspired Intelligence IWOBI 2015*, 2015.

[21] R. Hoseinnezhad, B.-N. Vo, B.-T. Vo, and D. Suter, "Visual tracking of numerous targets via multi-bernoulli filtering of image data," *Pattern Recogn.*, vol. 45, no. 10, pp. 3625–3635, Oct. 2012. [Online]. Available: http://dx.doi.org/10.1016/j.patcog.2012.04.004

[22] H. Morimitsu, R. M. Cesar, and I. Block, "Attributed graphs for tracking multiple objects in structured sports videos," *IEEE Iternational Conference on Computer Vision*, 2015.

[23] D. Sitton, "Maximum matching in complete multipartite graph," *Electronic Journal of Undergraduate Mathematics*, 1996.

[24] C. A. Phillips, "Multipartite graph algorithms for the analysis of heterogeneous data," Ph.D. dissertation, University of Tennesse, 2015.

[25] T. Zhao, V. March, and S. Dong, "Evaluation of a performance model of lustre file system," *2010 Fifth Annual ChinaGrid Conference*, 2010.

[26] S. Jian, L. Zhan-huai, and Z. Xiao, "The performance optimization of lustre file system," *2012 7th International Conference on Computer Science and Education*, 2012.

[27] M. Villalta, "Diseño, implementación y validación de una biblioteca en paralelo híbrida cpu/gpu de algoritmos de reconocimiento de patrones para el clúster tará del pris-lab," U.C.R., Noviembre 2016.