www.czasopisma.pan.pl PAN www.journals.pan.pl
Opto-Electronics Review 27 (2019) 58–69

Contents lists available at ScienceDirect

Opto-Electronics Review

journal homepage: http://www.journals.elsevier.com/opto-electronics review

A deep learning ball tracking system in soccer videos

P.R. Kamble*, A.G. Keskar, K.M. Bhurchandi

Department of Electronics and Communication Engineering, Visvesvaraya National Institute of Technology (VNIT), South Ambazari Road, Nagpur, 440010, India

ARTICLE INFO

Article history:
Received 23 November 2018
Accepted 11 February 2019
Available online 13 March 2019

Keywords:
Ball detection
Ball tracking
Convolutional neural networks
Deep learning
Bounding box overlap

ABSTRACT

Increasing interest, enthusiasm of sport lovers, and economics involved offer high importance to sports video recording and analysis. Being crucial for decision making, ball detection and tracking in soccer has become a challenging research area. This paper presents a novel deep learning approach for 2D ball detection and tracking (DLBT) in soccer videos posing various challenges. A new 2-stage buffer median filtering background modelling is used for moving objects blob detection. A deep learning approach for classification of an image patch into three classes, i.e. ball, player, and background is initially proposed. Probabilistic bounding box overlapping technique is proposed further for robust ball track validation. Novel full and boundary grid concepts resume tracking in ball_track_lost and ball_out_of_frame situations. DLBT does not require human intervention to identify ball from the initial frames unlike the most published algorithms. DLBT yields extraordinary accurate and robust tracking results compared to the other contemporary 2D trackers even in presence of various challenges including very small ball size and fast movements.

1. Introduction

Object tracking is a well-explored area of research. Any object, its motion, and appearance have its own difficulty level in tracking in presence of different challenges. High interest and economics in sports offer high importance to accurate ball tracking. Continuous, robust, and accurate ball tracking helps in taking crucial decisions and identifying infringement over game rules during fast sports activities. Thus, tracking of a ball is the most required but challenging machine vision activity in ball-based sports. Few common challenges are occlusion, misdetection, shadowing, and changes in object appearance and environmental condition. In addition to the challenges in arbitrary object tracking, ball tracking is prone to many additional challenges. Few of them include unexpected high velocity of events, frequent occlusion of ball by players, changes in ball size, shape, color, texture, and velocity. Additionally, sports environment is inconsistent, dynamic, and unpredictable.

Soccer is the most loved and appreciated sport in the world [1]. Its popularity attracted many researchers to work on 2D and 3D ball tracking for various applications. They include highlight extraction, tactical and performance analysis, refereeing, shot classification, event detection, content insertion, commentary assistance, etc. All these applications need instantaneous ball position and its track as a reference. Thus, robust ball detection and tracking are important and crucial tasks. A recent survey on ball detection and tracking [2] presents the depth and breadth of this challenging topic.

Research work on tracking of ball in sports began in late 90′s. Researchers at that time were interested in obtaining ball position in 3D [3–5] more than ball detection and tracking [6,7]. Kalman Filter (KF) was mostly used for ball tracking [8,9]. However, unsuitability of KF for tracking non-linearly and dynamically moving objects like ball, with changing appearance, decreased the tracking performance. As a consequence, hybrid approaches started being used which included Nearest Neighbour Data Association (NNDA) [10] and Template Matching (TM) [11]. Small and less challenging datasets resulted in their high quantitative performance. Meanwhile, Particle Filter (PF) was introduced for ball tracking due to its capability to handle non-linearity. A few works [12–15] claim to satisfactorily overcome problems like, varying scales, partial occlusion, non-linear, non-periodic motion, etc. However, they could not handle drastic changes in ball size, multicolored ball, and ball on similar background. The track of ball was lost due to slow updation of number of particles. TM has been extensively used for ball tracking [16–18]. TM was effective while dealing with a merged and missing ball. However, many researchers [19,20] used only long trajectory sequences in which the ball was most unlikely to miss. Presence of ball-like objects in the background also leads to false positives in such approaches. Offline ball tracking approaches like Data Association (DA) was used for tracking only tennis ball in sim-

* Corresponding author.
E-mail addresses: kamble.paresh@gmail.com, paresh.kamble@students.vnit.ac.in (P.R. Kamble).

https://doi.org/10.1016/j.opelre.2019.02.003
1230-3402/

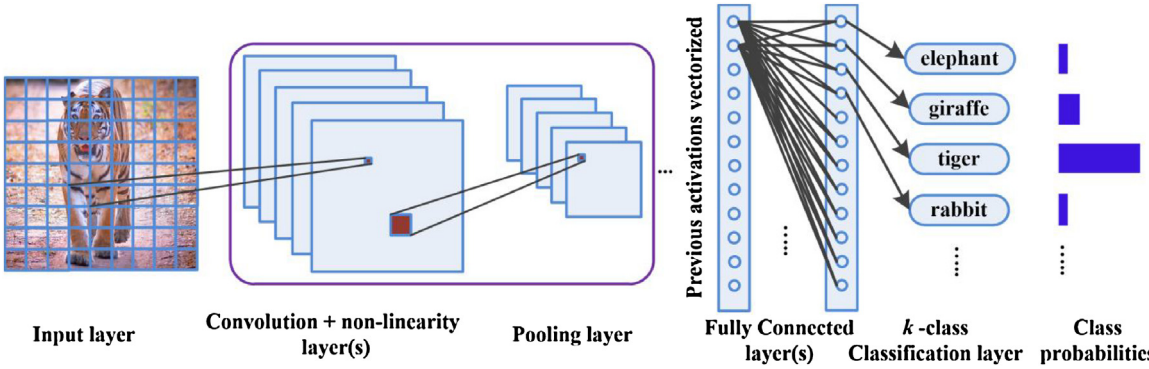

Fig. 1. Basic architecture of CNN.

pler conditions [21–23]. Similar to DA, graph based approach can handle occlusion for a very small duration. Ball missing for longer duration may lead to tracking failure. None of the methods [24–26] can handle size and shape changes of ball due to heavy occlusion and high velocity.

Recently introduced methods [27–29] used energy function optimization approach. They formulated the ball-tracking problem in terms of ball-player interaction [27,29] and introduced physics-based constraints [29]. The approach in Ref. 29 is generic. However, the processing time of this approach increases for bigger fields like soccer. A single approach cannot guarantee the consistency of performance across different sports. Both the approaches [27,29] assigned ball position to the player possessing the ball. Very recently, a supervised machine learning framework has been used for real-time ball-position measurement [30]. The work in Ref. 30 trained their ball detection algorithm using appearance and motion features for ball and no ball patches. Reno *et al.* [31] used Convolutional Neural Networks (CNN) [32] for detecting tennis ball. They designed and trained it, which is time consuming. It offered good performance on the easy task of tennis ball tracking. Ball tracking in team sports is much more complex [2].

The proposed algorithm cascades Intersection Over Union (IOU) with a modified pre-trained CNN [32] for fast ball detection and tracking. The interaction between ball and player is modeled using bounding boxes. Ball position is assigned to a player only if he occupies or occludes the ball. Database image patches are used to train a modified CNN for ball detection using transfer learning. The proposed algorithm handles the challenges like; automatic ball detection, occlusion by players, ball track lost, and ball out of frame.

Further, Section 2 discusses the fundamentals used by the proposed work. Section 3 presents the proposed 2D ball tracking algorithm in detail along with features and limitations of the work. Section 4 presents the experiments and database. Section 5 describes the results, and their discussions followed by conclusion and future scope in Section 6.

2. Learning paradigms

This section offers a brief overview on deep learning, basic architecture of CNN and transfer learning.

2.1. Deep learning

Deep learning algorithms learn data representations rather than task-specific computations [32]. Its relation with representation learning, machine learning, and artificial intelligence is described as Deep learning is element of Representation learning next Machine learning and finally Artificial Intelligence [33]. Deep learning helps computational models, with many layers, to learn data with multiple levels of abstraction. These methods discover intrinsic structures in huge datasets using backpropagation [34]. They also find out internal adjustable parameters to obtain representation of one layer from its previous layer [32]. Deep learning is divided into three major types of learning methods: supervised, semi-supervised, and unsupervised [35,36]. More recently, reinforcement learning was also introduced [37]. Supervised learning has been very popular so far. It requires a huge amount of labeled dataset for training. The CNN takes an image as input and produces a vectored and labeled output after training.

An objective function measures the loss between the output and its true value. The model adjusts its weights using backpropagation [34] to minimize the objective function during training. A deep learning system may have millions of such weights. Major applications of deep learning include image and pattern matching. A few major deep learning architectures include CNN, recurrent neural network, and recursive neural networks [38]. This work uses CNN.

2.2. Fundamentals of CNN

A basic architecture of CNN for classification, using supervised learning method, is shown in Fig. 1. It consists of an input layer, one or more convolution layers accompanied by non-linearity and pooling layer (optional). This follows one or more fully connected layers and finally a classification (regression) output layer. Incoming images have the same size as input layer, while the output layer neurons (k) are same as the required classes. Each convolution layer is initialized with a set of inherent learnable random Gaussian filters (weights), that extracts features from the previous layer. The image patches of filter size ($F \times F$) for each colour plane are convolved with the randomly initialized Gaussian weight filter matrices of the same size to obtain a convolution value (x).

Final feature detection filters for each layer are obtained by successively updating the Gaussian weight matrices of each layer during the process of training. For high dimensional data like images, it is impractical to connect every neuron from current layer to every other neuron from its previous layer. So, each current layer neuron takes inputs from a small region($F \times F$)from its previous layer. The region is called receptive field, whose spatial dimension is same as filter size and is decided heuristically. Each filter has a small height and width but larger depth. The filters convolve with the previous layer and produce activations. Height and width of the activation are computed using (1) [39]:

$$W2 = (W1 - F + 2P)/S + 1. \quad (1)$$

Where, $W1$ and $W2$ are input and output volume size. F, P, and S represent the filter size, amount of zero padding, and stride, respectively. Each convolution layer has a non-linearity function for range compressing the output of a convolution layer. Rectified

Fig. 2. Sample images from the ball-player-background dataset (a) balls; (b) players; (c) background.

Linear Unit (ReLU) implements the most popular non-linearity [40]. ReLU trains several times faster than other non-linearity functions [41]. It has a transfer function $f(x) = \max(0, x)$ for input x. It does not require input normalization for preventing from saturating. Still, Local Response Normalization (LRN) is sometimes used after ReLU for generalization [41]. If a kernel i is applied at position (x, y) of an input volume and its activity is denoted by $a^i_{x,y}$, the ReLU output $b^i_{x,y}$ is given by (2) [41]. If n is the number of adjacent kernel maps over which the sum is taken and N is the total number of kernels in the respective layer:

$$b^i_{x,y} = a^i_{x,y} \Big/ \left(k + \alpha \sum_{j=\max(0,i-n/2)}^{\min(N-1,i+n/2)} \left(a^j_{x,y}\right)^2 \right)^{\beta}. \tag{2}$$

Pooling layer may be periodically inserted between successive convolution layers. Inclusion of pooling layers reduces the spatial size of the representation, the number of learnable parameters and controls over-fitting.

Max operation is mostly preferred also for pooling layer. Height and width of the output of pooling layer is computed using (3) [39]:

$$W2 = (W1 - F)/S + 1. \tag{3}$$

Same parameter definitions as Eq. (1) are used in Eq. (3). More the number of convolution layers, more the amount of abstraction learnt. Output of the last convolution layer is vectorized for use by fully connected layers. The size of the last fully connected layer is equal to the number of classes k. A very popular cross entropy loss function is used as in Eq. (4):

$$E(\theta) = -\sum_{i=1}^{n} \sum_{j=1}^{k} t_{ij} \ln y_j(x_i, \theta). \tag{4}$$

Where, θ represents weights and biases parameter vector, target t_{ij} indicates that i^{th} input belongs to j^{th} class and $y_j(x_i, \theta)$ is the probability with which the CNN associates input i to class j, $P(t_j = 1 \mid x_i)$. Normalized exponential [42], i.e. softmax (5), is the output layer transfer function:

$$y_r(x) = \frac{e^{a_r(x)}}{\sum_{j=1}^{k} e^{a_j(x)}}. \tag{5}$$

Where, $0 \le y_r \le 1$ and $\sum_{j=1}^{k} y_j = 1$. Forward pass includes processing the data from input layer to output layer. The loss is computed between the predicted and true values. Backpropagation [34] updates the CNN parameters using Stochastic Gradient Descent with Momentum (SGDM) algorithm [43]. Conventional Stochastic Gradient Descent (SGD) algorithm gets trapped at a local minima. Adding a momentum term as in Eq. (6) escapes it [43]:

$$\theta_{l+1} = \theta_l - \alpha \nabla E(\theta_l) + \gamma(\theta_l - \theta_{l-1}). \tag{6}$$

Where, θ is the parameter vector, l represents the iteration number and $\alpha > 0$ represents the learning rate of the network. $\nabla E(\theta)$ is gradient of the loss function and γ represents the contribution of gradient from previous iteration to the current iteration. A mini-batch, i.e. a subset of the complete dataset is used for training CNN to save time without compromising performance. A single pass of evaluation of gradients using a mini-batch is called iteration. When all the training samples pass through the network once, it is called an epoch. After a suitable large number of epochs, the CNN is considered trained. It is then performance evaluated on the test sets.

2.3. Transfer learning

Transfer learning is the technique where a part of any connectionist network, that has learned one task, can be reused for other similar task [44,45]. Transfer learning replaces the last few layers of a pre-trained network with new untrained layers. A related labeled dataset is used to train the modified network. During training, parameters of only the new layers are trained, whereas parameters of the retained layers are refrained from training. Transfer learning is advantageous than learning a complete network from scratch. Transfer learning involves tuning less hyperparameters and training only newly added layers to save time.

3. Proposed work

This section presents a detailed description of the proposed work including offline training of CNN using transfer learning, automatic ball finder, bounding box overlap for tracking, concept of grids and all other modules of the proposed 2D ball detection and tracking algorithm.

3.1. Offline training of CNN

We propose a CNN architecture designed by modifying the Visual Geometry Group (VGG) at University of Oxford, named VGG-M [46] network for ball detection followed by its training. This sub-section also offers an insight into generation of our ball-player-background dataset.

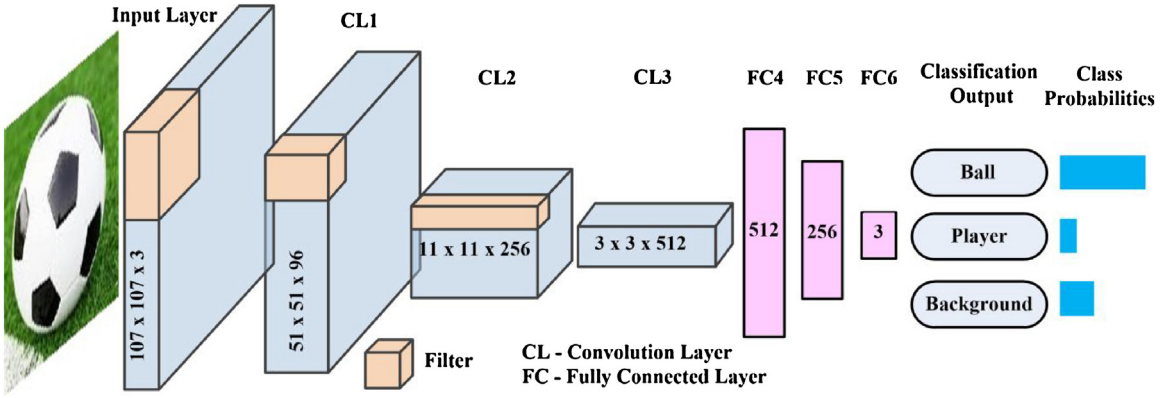

Fig. 3. Modified architecture of VGG-M CNN.

3.1.1. Dataset generation

Due to unavailability of dedicated dataset of ball and players, we designed our own dataset using images cropped from online available and our own soccer videos. We cropped 1500 images for each class, i.e. ball, player, and background. We deliberately used images of only soccer ball. For players, we cropped images of only soccer players playing in the field. For background class, we obtained the images of soccer playfield and stands. For robust training, we selected the class objects of all size, shape, and colour. Figure 2 shows a few samples of all the three classes. We deliberately included images of blurred and occluded objects to train the CNN better, make it a robust classifier, and avoid over-fitting.

3.1.2. Proposed CNN architecture and training

VGG-M is one of the smallest CNN, originally designed for image recognition and object detection [46], whose pre-trained weights are publically available. Since, only three classes are required, small CNN is enough [47]. Smaller size corresponds to less number of layers, less number of learnable parameters, less computation time and eventually higher speed. VGG-M takes an input image of size $107 \times 107 \times 3$. It has five hidden layers with three convolutional and two fully connected layers. All the three convolution layers have ReLU. First two convolution and ReLU layers are followed by LRN and max pooling layers. The LRN and pooling layers are discussed in Section 2.2. VGG-M CNN is originally pre-trained on ImageNet dataset [48] with 1000 object categories. Here, we have three classes viz., ball, player, and background. We replaced the last three layers of VGG-M with two fully connected layers and one classification layer, with the lengths of 512, 256, and 3, respectively as in Fig. 3. First two fully connected layers have ReLU.

We choose the lengths of fully connected layers experimentally. The last output classification layer of length 3, represents the three classes. Softmax layer follows last fully connected layer. In the proposed VGG-M CNN, we used (1), for calculation of all the convolution layers' dimensions. We used hyperparameters of the already trained VGG-M network for object detection, in the proposed ball detection algorithm, using transfer learning. Thus, 96 filters of $7 \times 7 \times 3$ dimension ($F = 7$), filter shift stride ($S = 2$) and no zero padding ($P = 0$) were used for computing dimension of CL1, from input layer as suggested in Ref. 46. For computing output of max pooling layer at CL1 and CL2, we used (3) with$F = 3$,$S = 2$, and$P = 0$. For dimensions of CL2, we used 256 filters, with, $F = 5$ $S = 2$, and $P = 0$.. Finally, for dimension of CL3, we used 512 filters with$F = 3$, $S = 1$,and$P = 0$. Figure 3 shows the modified architecture.

We used SGDM optimization algorithm to minimize loss function in Eq. (4) on the modified VGG-M network. The network parameters update is done using mini-batch of 100 training samples from the training dataset. We experimentally selected a learning rate of $\alpha = 0.01$ for the newly added layers and the other layers from trained at infinitesimal learning rate of$\alpha = 0.00005$. This setting helps to train the newly added layers faster than the earlier ones due to large difference in the learning rates. Further, we train the CNN using our ball-player-background dataset divided in a ratio of 70:20:10 for train, validation, and test set. The process repeats until either the network is converged using loss function threshold or predefined number of epochs (100 in our case) is reached whichever is earlier. We obtained the validation accuracy of 98.44% and test accuracy of 98.89% for the CNN in Fig. 3.

3.2. Proposed 2D ball detection and tracking algorithm

The trained VGG-M is employed for ball detection and classification in the proposed algorithm. This sub-section offers a detailed insight into the proposed preprocessing algorithms for blob detection of moving objects. The proposed novel techniques for removal of all false blobs followed by the robust ball tracking are presented subsequently.

3.2.1. Initialization and blob detection

Initialization and pre-processing is the first module in the proposed algorithm. It includes initialization of constants and CNN. After initialization, the first frame from a video sequence is taken as input. Here, *ROW* and *COL* stands for number of rows and columns of the input frame. We use a novel background modelling approach for ball detection. The proposed fast background modelling algorithm works in two steps: the first step takes incoming *SIZE*1 frames to form a stack. It takes a temporal pixel-position median of the stack to obtain a single median frame, pushes it as the first entry to the second stack of *SIZE*2 frames and clears the first stack. Subsequently, the next *SIZE*1 frames can enter the first stack to repeat the same process and obtain the second entry to the second stack. Thus, *SIZE*1 × *SIZE*2 frames are required to fill the second stack completely. The second step involves taking again the similar median of the available frames of the second stack. The median of the second stack serves as the background for frames until next turn of the second step. The background obtained after this process is the final background for the future frames until the end of sequence. It must be noted here that this background modelling technique is applicable only to fixed camera videos. Unlike successive frame differencing, this technique offers much robust blobs of moving objects. Background pixels are obtained using the background modeling as in algorithm 1.

Algorithm 1: Median filtering background modeling

Input: $SIZE1, SIZE2, ii := 1, jj := 1,\ TEMP1 := zeros(ROW \times COL \times 3, SIZE1),\ TEMP2 := zeros(ROW \times COL \times 3, SIZE2).$

Output: *background* frame

```
if i ≤ SIZE1 × SIZE2
    TEMP1(:, ii) := vectorize(i^th frame)
    if ii == SIZE1
        ii := 0
        TEMP2(:, jj) := median (TEMP1)
        TEMP := median (TEMP2(:, 1 : jj))
        background := reshape (TEMP, [ROW, COL, 3])
        jj := jj + 1
    end if
    ii := ii + 1
end if
```

The current frame is subtracted from the obtained background frame using algorithm 1. A bi-thresholding operation is carried out on the three channels of the frame difference separately using an experimentally derived threshold (0.07). The three thresholded binary channels are logically ORed to yield a single binary image representing moving blobs by logical 1 bits. Blobs of size within the preset range (7×7 and 125 × 125 pixels) are retained whereas others are neglected. Morphological operations, dilation and holes filling are carried out further using a disc structural element of a radius of 2 pixels. This joins the entities like leg blobs and hand blobs with torso blob of the players to yield complete player blobs along with ball blobs. Area, bounding boxes, and centroids of the blobs (ABC_i) in current processed frame (i) are used further. All the bounding boxes are inflated using experimentally decided *SWELLBY* pixels on each side to obtain bounding box overlapping for tracking the closest ball and player(s) together. Further, the CNN is used for classifying all the blobs of i^{th} frame into ball, players, and background classes. Further unwanted blobs, that are outside the circle of $CENT_{i-1}$ centroid and *RAD* radius of the previous ball or player blob are removed. The algorithm checks whether the conditions ball track lost or ball out of frame exist by checking the respective grids till the ball is found to continue tracking. The abnormal sized blobs are neglected as they may not be ball or players.

Algorithm 2: Automatic Ball Finder

Input: ABC_i of the blobs, $BBOX_{i-1}$, ROW, COL

Output: $ABFF$, SF_i, $BBOX_i$, $AREA$, $dBall$, RAD, $BGRID$, $FGRID$

```
ABCBall_i := ABC_i(ball)
if ABCBall_i ≠ {φ}
    SF_i := 1
    if SF_{i-1} == 1
        if (BBOX_{i-1} ⋂ ABCBall_i) ≠ {φ}
            BBOX_i := bounding box of max (BBOX_{i-1} ⋂ ABCBall_i)
            CENT_i := centroid of max (BBOX_{i-1} ⋂ ABCBall_i)
            if Σ_{k=i-M+1}^{i} SF_k == M
                ABFF := 0
                Obtain AREA, dBall & RAD from BBOX_i.
                Obtain BGRID & FGRID from ROW, COL, & BBOX_i
            end if
        end if
    end if
    BBOX_i := bounding box of ABCBall_i
else
    SF_i := 3
end if
```

3.2.2. Automatic ball finder (ABF)

We invoked this module during initialization just after detection of all ball blob candidates. The proposed ball-tracking algorithm uses this module for ball detection. Once the ball is detected and validated,*ABF* flag Automatic Ball Finder (ABF) flag is reset to skip revisiting this module further. The proposed module automatically finds the ball unlike other arbitrary object tracking algorithms [49,50] which take the ball either manually or from Ground Truth (GT). Detection in every frame by other trackers results in increased computational overhead. However, in our approach, we incorporated automatic ball finder. This module searches the ball in initial frames of the input video sequence. After it finds the ball, it validates the ball automatically and starts tracking until it is lost, to save computations. This module, shown in algorithm 2, learns the initial ball to be tracked at a reasonable computational cost. Here, SF_i,$CENT_i$, and $BBOX_i$ denote status flag of the event, centroid, and bounding box of ball in i^{th} frame, respectively. *AREA* and *dBall* denote blob area and blob diameter. If a ball candidate blob is available in$M$ consecutive frames of a video, it may be a true ball. A ball candidate is considered a validated ball if the inflated ball candidate bounding box overlaps maximum with that in the previous frame. This is further validated, if this overlap is non-zero for previous Mconsecutive frames forming a trajectory to indicate the true ball. If a true ball is detected,$BBOX_i$, $CENT_i$, and *AREA* are updated and boundary grid(*BGRID*) and full grid(*FGRID*) are obtained for further use.

3.2.3. Intersection over union for ball tracking

Conventional object tracking by detection in every frame is computationally very heavy. In this work, we propose a novel concept of ball bounding box overlap for tracking a ball using spatial Intersection Over Union (IOU). IOU also known as Jaccard index or Jaccard similarity coefficient metric [51] is a measure for comparing similarity and diversity of sample sets. It is a ratio of intersection of the bounding box in the previous frame and the predicted bounding box in the current frame to their union as in (7):

$$IOU = \frac{\left|BB_A \cap BB_B\right|}{\left|BB_A \cup BB_B\right|} = \frac{\left|BB_A \cap BB_B\right|}{\left|BB_A\right| + \left|BB_B\right| - \left|BB_A \cap BB_B\right|}, \tag{7}$$

where, BB_A and BB_B are areas of bounding box in the previous frame and the current frame. Further, ∩ and ∪ are intersection and union in terms of number of pixels represented by| |. IOU has so far been used on randomly selected image patches [47] for tracking. But mainly it is a performance measure for detection and tracking algorithms [49,50,52]. The proposed work uses it on moving ball blobs which are very small in number compared to the random image patches unlike [47]. This does not require the mathematical or physics model ling of the ball candidate trajectories. *A priori* knowledge of maximum ball speed, camera distance from the ball, and the frame rate of the video sequence are sufficient to decide the factor by which the true bounding box of the moving ball is to be inflated. Ball is the fastest object in soccer that may travel as fast as

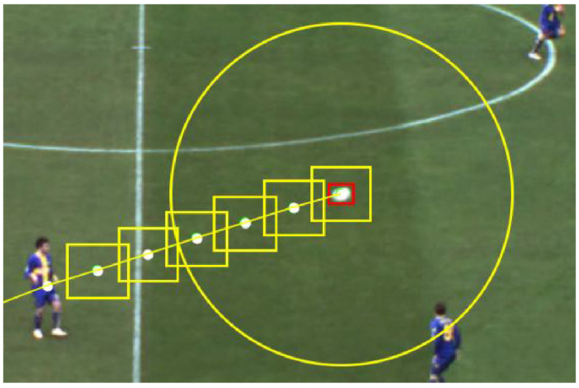

Fig. 4. White dot with red bounding box shows the tracked ball, other white dots denote previous ball positions, yellow circle indicates region of the ball search for next frame, and yellow lines indicate ball trajectories. Yellow bounding box represents inflated bounding box.

129 kmph [53]. The inflated bounding boxes of the true ball from the previous and the ball candidates from the current frame are tested for overlap. In case, more than one ball bounding box candidates overlap, the maximum overlap bounding box indicates the true ball position. If the overlap (intersection) is non-zero ($\neq \phi$), then the ball is considered successfully tracked at the respective position. The same principle is applied for the ball in previous frame and player(s) in the current frame and vice versa. Figure 4 explains the concept of using bounding box overlap for ball tracking. The ball positions from six consecutive frames form a trajectory. The inflated bounding boxes overlap each other even if the true bounding boxes do not if the bounding boxes are inflated enough. This simple and elegant concept of bounding box normalized intersection in the successive frames offers a very robust and accurate tracking algorithm at much less computational cost compared to the conventional tracking algorithm.

3.2.4. Ball tracking module

As discussed in Sections 3.2.2 and 3.2.3, a ball is tracked using the bounding box overlap implemented using IOU. In this module, the filtered ball blobs ($ABCBall_i$) of current frame (i) are checked for spatial overlap of the bounding boxes with that of the previous frame ball ($BBOX_{i-1}$). Parameters like $BBOX_i$, $CENT_i$, and RAD are computed using the maximum IOU position in the current frame. In case of ball_track_lost and ball_out_of_frame the ball blob candidates may be detected using any of the grids; $FGRID$ and $BGRID$. Here, FGF and BGF are full grid and boundary grid flags, which denote the activation (set) and deactivation (reset) of the grids. Once we obtain the true ball bounding box, we set the status flag SF_i as ball track continued and deactivate both the grids. The ball tracking module is presented as algorithm 3. Results are shown in Fig. 4.

Algorithm 3: Ball tracking module

Input: $ABCBall_i$ of the ball blobs, $BBOX_{i-1}$, FGF, BGF

Output: SF_i, $BBOX_{i-1}$, $AREA$, $dBall$, RAD

```
if ABCBall_i ≠ {φ}
    if BGF == 0 AND FGF == 0
        temp := (BBOX_{i-1} ⋂ ABCBall_i)
    else if BGF == 1 AND FGF == 0
        temp := (BGRID ⋂ ABCBall_i)
    else if BGF == 0 AND FGF == 1
        temp := (FGRID ⋂ ABCBall_i)
    end if
    if temp ≠ {φ}
        BBOX_i := bounding box of max(overlap of ABCBall_i)
        CENT_i := centroid of max(overlap of ABCBall_i)
        SF_i := 1
        Inflate BBOX_i by SWELLBY pixels on all sides
        Obtain dBall and RAD
        BGF := 0, FGF := 0
    else
        ABCBall_i := {φ}
    end if
end if
```

3.2.5. Player tracking module

If ball blobs are not detected in the search, then the control is passed to the player-tracking module assuming that the ball is occluded. The control can also be passed to this module if at least one player is present inside the search region (in circle of RAD radius) and both the grids are deactivated. If the ball is occupied or occluded by player(s), the player blob(s) are tracked using same method discussed in Section 3.2.3; the ball blob check also continues. Whenever the ball has an isolated blob, it is tracked further and the player track is discontinued. The new ball position is used to obtain parameters $BBOX_i$, $CENT_i$, andRADas in Fig. 5(a).

The player blobs of the current frame are checked for overlap with previous ball (or player(s)) to detect the probable ball position, as in Fig. 5(b). In soccer, it is difficult to identify which player(s) is(are) possessing the ball. Hence, a bounding box is fit on all the near-by players suspicious of occluding the ball as in Fig. 5(c). Player tracking module is presented in algorithm 4.

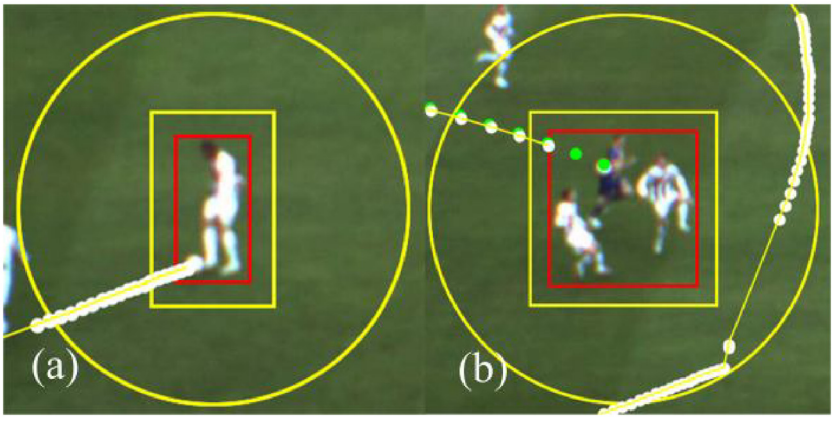

Fig. 5. (a) Red bounding box shows the player position; (b) Red bounding box show most probable players who may occlude ball.

Algorithm 4: Player tracking module

Input: $ABCBall_i$, $ABCPlayers_i$, FGF, BGF, $BBOX_{i-1}$.

Output: SF_i, $BBOX_i$, $CENT_i$, $AREA$, $dBall$, RAD.

```
if ABCPlayers_i ≠ {φ} && ABCBall_i == {φ} && FGF == 0 && BGF == 0
    temp := (BBOX_{i-1} ∩ ABCPlayers_i)
    ABCPlayers_i(~ temp) := {φ}
    if ABCPlayers_i ≠ {φ}
        Form a region with players suspect of ball occlusion
        Obtain BBOX_i, CENT_i, and RAD from it
        SF_i := 2
        if Σ_{k=i-PAT1+1}^{i} (SF_k == 2) == PAT1
            FGF := 1
        end if
    end if
end if
```

During occlusion, status flag (SF_i) is set for players, which denotes the ball is occluded or possessed by the player(s). If only a player(s) is detected for $PAT1$ previous frames and no ball is detected,*FGRID* is activated and the*FGF*is set. The grid remains active until true ball is detected.

3.2.6. Bounding box grid module

Grid is a mesh-like structure made up of connected rectangles of size four times that of the true ball bounding box. The two grids viz., *FGRID* and *BGRID* are shown in Figs. 6(a) and 6(b). . *FGRID* is spread over full frame. It identifies and resumes ball tracking if it is lost due to either presence of false positives or long time occlusion. *BGRID* detects ball_out_of_frame situation and keeps on looking for ball at boundaries in the successive frame(s).

Even after the ball is lost, the moving objects blob spatial position detection continues as in Section 3.2.1. All the blob positions are classified as ball or player blobs. If neither ball nor player positions are found in the current frame, the control passes to the grid module where the grid activation is checked. If*FGRID* is activated, the SF_i is set to ball_out_of_frame and the ball is searched in the entire successive frames. Detected player and background blobs are neglected using the CNN classification of the corresponding resized patches, while all the ball blob candidates are detected as possible ball positions. If it is stationary, it is not a true ball position and will be neglected in the subsequent frame. The next position among the newly detected ball positions in the next frame will be considered for checking a true ball location further. This continues until true ball position is found. *FGRID* remains active until true ball is found in *PAT* 2 frames, after which *FGRID* is deactivated and *BGRID* is activated assuming that the ball has gone out of frame.*BGRID* continues to remain active until the true ball is found, after which it is deactivated. Another case in which the *BGRID* is activated is the bounding box of the ball was overlaps the *BGRID* locations and subsequently the ball is lost. It is assumed that the ball is out of frame. The algorithm keeps a close check on player(s) near to the ball in the*BGRID* , by tracking their blobs and trying to detect the ball around the player blobs. The player may occlude the ball and take it back inside the frame while the algorithm may activate the *BGRID* and set SF_i as ball out of frame. If a player occupying or occluding the ball also leaves the frame, it is considered that the ball is out of frame. The module is presented in algorithm 5.

Algorithm 5: Bounding Box Grid module

Input: FGF, BGF, $BBOX_{i-1}$, SF_i.

Output: FGF, BGF, SF_i.

```
if SF_i == 0
    if FGF == 1
        SF_i := 3
        if (Σ_{k=i-PAT2+1}^{i} (SF_k == 3)) == PAT2
            FGF := 0, BGF := 1.
        end if
    else if BGF == 1
        SF_i := 3
    else if BGF == 0
        temp = (BGRID ∩ BBOX_{i-1})
        if temp ≠ {φ}
            SF_i := 3, BGF := 1
        end if
    else
        SF_i := 3, FGF := 1
    end if
end if
```

3.2.7. Flow diagram of the complete algorithm

All the above-mentioned modules are organized to form the proposed ball detection and tracking algorithm as algorithm 6. The proposed algorithm performs the task of 2D ball tracking from a soccer video sequence obtained from a stationary camera. Here,*ROW*, *COL*, and *numFrames* represent the row, column, and number of image frames of the input video sequence, respectively. The proposed 2D ball detection and tracking flow diagram is shown in Fig. 7.

3.3. Novelties in our approach

The proposed ball-tracking algorithm is able to robustly track soccer ball of any size, appearance, and texture moving with high velocity. We first time proposed a modified VGG-M CNN trained to classify ball, players, and background classes using transfer learning. It not only detects and tracks the ball but also the potential player(s), occupying it, until the ball reappears. Thus, it elegantly models ball player interaction.

We used a novel robust 2-stage median filtering background modelling technique for extracting moving objects by modeling

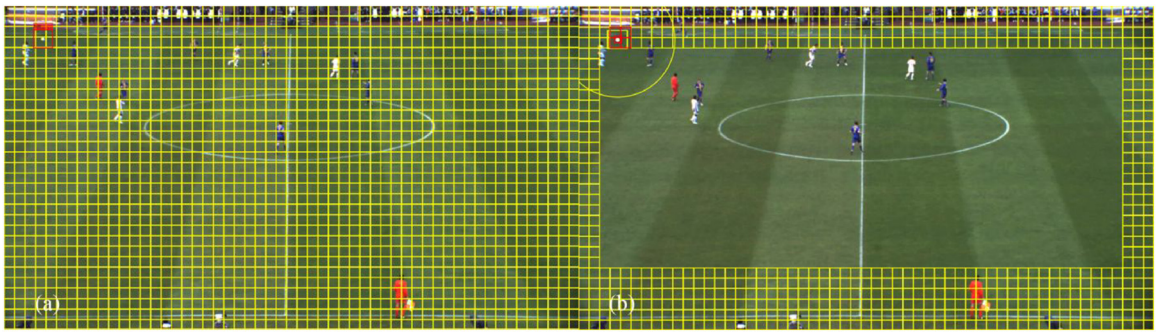

Fig. 6. (a) Full grid; (b) Boundary grid.

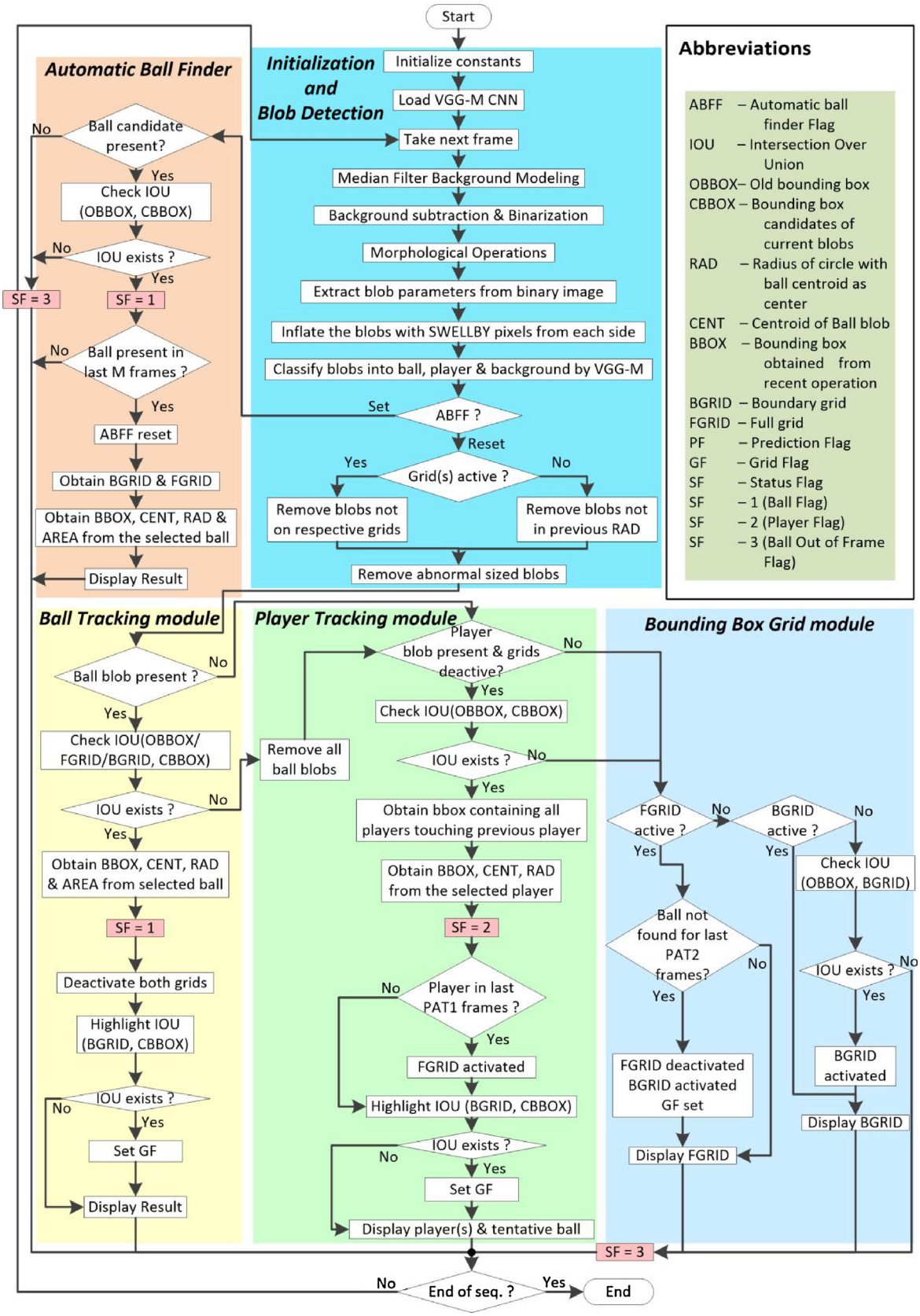

Fig. 7. Detailed flow diagram of the proposed DLBT algorithm.

object-free background from video frames itself. The proposed algorithm exploits the performance measure IOU for ball and player detection and tracking. This simple concept has made the otherwise slow and difficult task of ball tracking under long-term occlusion possible. The proposed algorithm does not need any ball bounding box in the first frame, unlike other trackers. It finds a true

ball on its own and tracks it further. The proposed CNN is pre-trained using earlier frame patches and not a single frame from the current recording is used for training, unlike other contemporary algorithms. The CNN is trained robustly enough to handle ball track lost, ball out of frame, false positives situations, and resume tracking using the novel concept of grids. The problem of partial occlusion is handled by predicting the most tentative ball position by tracking a player occluding the ball. The proposed algorithm achieves an overall speed of 10 frames per second(FPS), which is enough for most of the real-time decisions.

Algorithm 6: Proposed 2D ball detection and tracking algorithm

Input: Video frames, Load GT, VGG-M CNN, Initialize constants $SIZE1$, $SIZE2$, $TEMP1$, $TEMP2$, $SWELLBY$, $BBOX$, $CENT$, ROW, COL, $numFrames$.

Output: SF_i, $BBOX_i$, $CENT_i$.

Recursion:

```
for all i ∈ {1,2,....,numFrames} do
    Read i^th frame
    Obtain background image- Algorithm 1.
    diffimage_i := |background − i^th frame|
    Threshold diffimage_i into binaryimage_i
    Morphologically convert binaryimage_i into blobimage_i
    Get AREA, BBOX_i, & CENT_i of blobs from blobimage_i into ABC_i
    if ABC_i ≠ {φ}
        Inflate all blobs by SWELLBY pixels
        Classify the blobs using VGG-M CNN
        if ABFF = 1
            Find ball- Algorithm 2.
        else if ABFF = 0
            if BGF = 1
                Remove blobs not on BGIRD
            else
                Remove blobs outside RAD radius circle
            end if
        end if
        Obtain ABCBall_i and ABCPlayers_i from rest blobs
        Compute true ball parameters- Algorithm 3.
        Compute player blobs parameters- Algorithm 4.
        Check for grid activations conditions- Algorithm 5.
    end if
end for
```

3.4. Limitations of our approach

Our approach has few limitations. The algorithm offers high robustness when the foreground is of smaller size compared to the background. When the ball blob and a blob of a player wearing similar color uniform, with similar sizes, touch each other, the algorithm fails momentarily. Nevertheless, the tracker resumes tracking when the ball blob assumes its actual shape and size and gets isolated from the player blob. The stands region is the potential source of false positives and noise. In this approach, we masked the stands region, thereby, losing track of the ball when it goes in or flies through the masked region resulting in reduced accuracy. Since the algorithm focusses on ball tracking, it does not track individual players very precisely. It tracks only potential ball snatcher(s).

4. Experiments and database

Two versions of the proposed algorithm are used for performance evaluation and benchmarking. Version 1 includes the initialization cum pre-processing, ball tracking, and player tracking modules for benchmarking. This version requires ball bounding box as an input. Version 1 is benchmarked with Multiple Instance Learning (MIL) [49] and Kernelized Correlation Filter (KCF) [50] in Section 5.1. Version 2 has module, and grid module in addition to version 1 modules. Version 2 also has ability to search the lost ball during tracking and resume tracking after the lost ball is found.

The modified VGG-M network is trained using a mini-batch size of 100 sample images, 100 epochs, global learning rate of 0.0005, and learning rate of 0.01 for new layers [47]. Increasing the epochs even to 300 does not increase its robustness or accuracy.

In the proposed median filtering background modeling algorithm 1, we experimentally selected values 15–25 for $SIZE1$ and $SIZE2$. Smaller value of $SIZE1$ results in more pauses during background modeling. Its larger value results in presence of false blobs. This causes incorrect computations of area, bounding box, and centroid of each blob. The $SIZE2$ parameter prunes out minor differences between frames in $SIZE1$ buffer. We selected size 18 for $SIZE1$ and $SIZE2$ buffers. Thus, a robust background model is obtained after the first $SIZE1 \times SIZE2$ frames of the video sequence.

We thresholded the difference frame by a value 0.07. Small structural element is selected for morphological operations as moving objects in each frame are very small. We choose a disc shaped structural element with radius 2 pixels to avoid too much dilation. We pruned very small (7×7 pixels) and very large (125×125 pixels) blobs that may be neither a ball nor player(s). We inflated the object bounding boxes on all sides by $SWELLBY$ pixels. We chose $SWELLBY = 20$ and $RAD = N \times dBall$, heuristically. $dBall$ is true ball diameter. When a player is occluding the ball, it is found mostly at the boundary of player blobs. Therefore, we limit the search circle radius to $RAD = \max(ballHeight, ballWidth)$ pixels, heuristically.

We evaluated our algorithm on ISSIA dataset [54] that has six HD (1920×1080) videos taken by six stationary cameras set on a single play field, at 25 FPS. Since, this dataset is designed for player tracking, ball is not present in many frames. The videos offer challenges like; partial occlusion, long-term full occlusion, two balls appear in same frame, noisy stands and boundary regions, ball out of frame, and player uniform color. To handle noise and misdetections from the non-field regions and billboards, we masked them and ball is not detected in these regions. This leads to a slight decrease in the performance compared to the other algorithms.

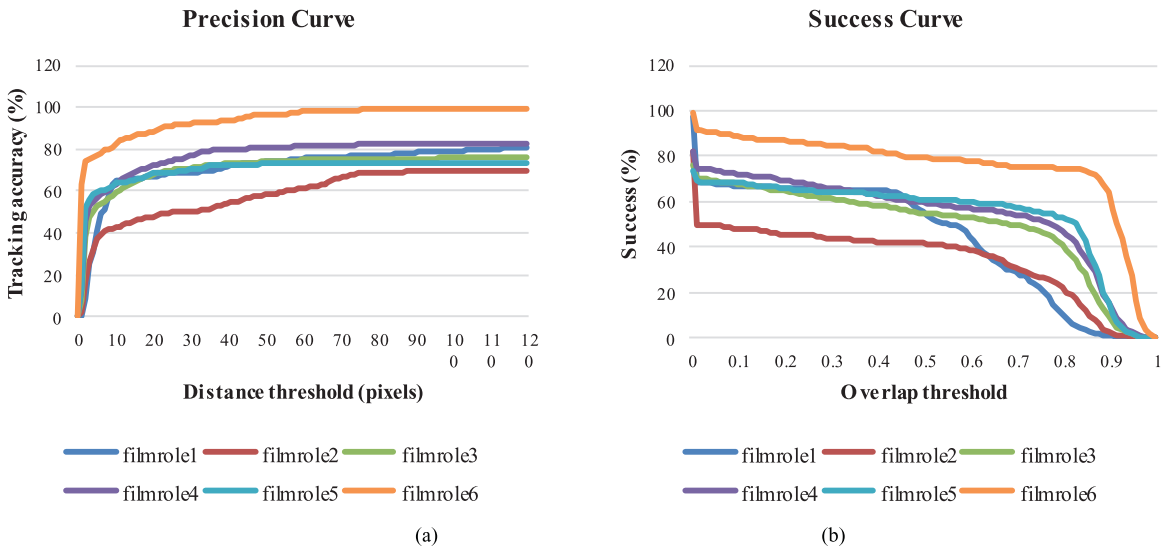

Fig. 8. Results of the proposed DLBT algorithm (Version 2), (a) Precision curves Tracking accuracy), (b) success curve.

Table 1
Comparison of the Proposed Algorithm (Version 1), MIL, and, KCF.

Sequence	# of fr.	MIL [49]		KCF [50]		Proposed (Version 1)	
		PE	IOU	PE	IOU	PE	IOU
filmrole11	224	245.64	0.12	319.16	0.09	**12.52**	**0.52**
filmrole12	156	342.12	0.02	499.29	0.13	**22.68**	**0.33**
filmrole13	95	1068.7	0.01	1169.7	0.01	**12.07**	**0.50**
filmrole14	204	820.11	0.01	945.84	0.01	**15.63**	**0.58**
filmrole21	304	430.83	0.01	414.32	0.00	**18.25**	**0.38**
filmrole22	614	–	–	1095.1	0.00	**269.1**	**0.26**
filmrole31	125	97.92	0.08	16.57	0.70	**2.54**	**0.71**
filmrole32	104	56.79	0.01	100.32	0.01	**57.85**	**0.26**
filmrole33	235	193.23	0.03	366.57	0.10	**59.03**	**0.37**
filmrole34	661	–	–	207.84	0.10	**18.63**	**0.48**
filmrole35	72	326.39	0.02	332.22	0.02	**13.78**	**0.64**
filmrole41	48	698.27	0.03	494.93	0.33	**3.2**	**0.71**
filmrole42	117	549.05	0.01	577.97	0.01	**37.48**	**0.35**
filmrole43	276	–	–	625.93	0.00	**62.79**	**0.50**
filmrole44	377	474.80	0.01	364.38	0.06	**255.1**	**0.27**
filmrole45	133	506.35	0.01	512.48	0.01	**3.95**	**0.75**
filmrole46	210	671.63	0.01	347.82	0.37	**16.75**	**0.51**
filmrole51	191	143.31	0.03	691.39	0.00	**30.35**	**0.43**
filmrole52	81	236.57	0.02	276.64	0.01	**24.02**	**0.38**
filmrole61	220	1191.3	0.01	1287.7	0.00	**14.24**	**0.63**
Weighted Avg.		479.63	0.03	560.43	0.08	**78.24**	**0.44**

* Results of MIL are not available on filmrole22, filmrole34 and filmrole43.
Comparatively better results are highlighted in bold.

5. Results and discussion

This work used MATLAB 2018a on a 64-bit Windows 10 i7 machine, 32GB RAM, and NVIDIA GTX1050Ti GPU.

5.1. Version 1

Since, only ball centroids are available in GT videos, 25 × 25 size of the reference GT bounding box is considered. Twenty sections of the six ISSIA sequences were selected, so that ball is present in all frames. Benchmarking of the proposed algorithm with MIL and KCF is presented in Table 1 in terms of Position Error (PE) and IOU. Version 1 out rightly outperforms both the trackers. KCF and MIL tracked the ball for a few frames and then lost it. Reasons are too small ball size, high-velocity ball movement, and long-term occlusion by players, which lead to trackers learning the player limbs or other objects as a ball. KCF and MIL process at 625 FPS and 0.5 FPS. Both yield poor accuracy and robustness due to frequent loss of track. The proposed algorithm processes at 10 FPS, with all the said features and yields outstanding results.

Table 2
PE and IOU result.

Sequences	PE (in pixels)	IOU
filmrole1	128.10	0.44
filmrole2	70.68	0.34
filmrole3	5.75	0.50
filmrole4	6.1	0.54
filmrole5	3.78	0.54
filmrole6	5.97	0.76
Average	**36.71**	**0.52**

5.2. Version 2

Version 2 could not be compared with other trackers as they do not have the said features. All the other object trackers require manually located object in the first frame. We used PE threshold range of 0 to 200 pixels and success rate (IOU) 0 to 1 for performance benchmarking. Figure 8(a) shows tracking accuracy PE on all the six videos. The algorithm has achieved more than 20 pixel accuracy, which is fairly high on all videos. The accuracy is not close to 100% in many frames because, the ball moved out of play field or was in the stands region. Low accuracy on filmrole2 is due to partial occlusion by a player and ball color matching the player uniform. The proposed tracker tracks the players until the ball completely comes out from the player. During those frames, our tracker gives the centroid of the player bounding box as the ball centroid. When a player handles the ball, he takes a large number of frames to release the ball. During all the above scenarios, GT is present but the proposed tracker cannot detect the ball. Figure 8(b) shows overlap threshold of proposed ball bounding box against GT. Of all the detections, 30% have IOU greater than 0.7. Our algorithm tracks the ball occupying or occluding players and compares them with GT. Thus, IOU is nonzero though the distance between GT and predicted centroid is large. A maximum IOU of 0.99 is achieved. Table 2 shows the average PE and IOU of all the video sequences of ISSIA dataset. The proposed algorithm recognized a few events of the ongoing soccer game. The events are ball detected, player occupying the ball, and ball out of frame. The event accuracy, precision and recall are shown in Table 3. Figure 9(a) shows a sample result

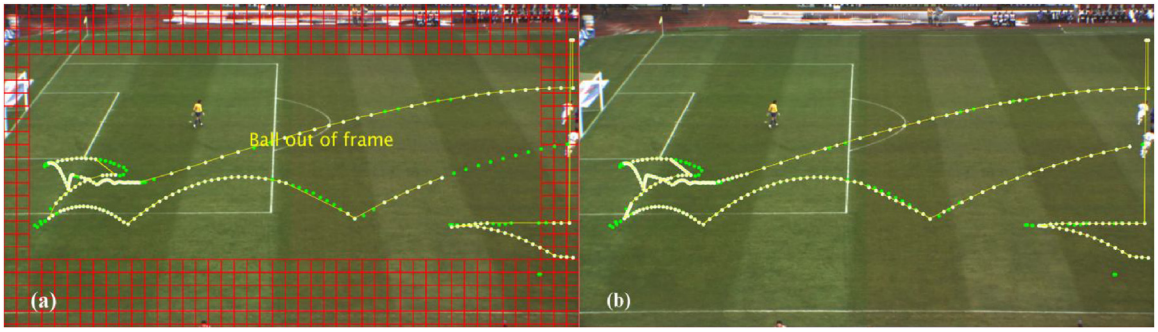

Fig. 9. (a) A sample result of the proposed DLBT algorithm (version 2) on filmrole 6 video. Green and white dots represent GT and tracking results. Yellow lines represent trajectory of the ball. Yellow text represents the current state of the game. Red mesh is the active boundary grid; (b) Result after post processing operation (linear interpolation) is applied for missing ball centroid positions.

Table 3
Event Detection.

Events	Precision (%)	Recall (%)	Overall Accuracy (%)
Ball detected	93.25	73.25	
Player occupying the ball	46.07	56.27	87.45
Ball out of frame	89.08	97.38	

frame of the proposed algorithm along with boundary grid. Figure 9(b) shows the result after prediction of the missing ball position centroids using linear interpolation, which approximates the true ball trajectory.

6. Conclusions

The proposed 2D DLBT algorithm is highly efficient not only for detecting and tracking the ball but also for modelling the necessary ball-player interaction. Excellent results are obtained compared to the other contemporary trackers in terms of PE and IOU. While others fail to track small fast moving objects, our algorithm did that with ease, in reasonable time. The proposed approach is novel in many ways. First, we proposed a new background modeling method using median filtering. Second, it did not require training even on a single frame of the used test dataset. Third, our algorithm finds the true ball without human intervention in the initial frames and also subsequently, unlike all other algorithms. Fourth, our algorithm can keep track of a ball going out of frame and coming back to resume tracking. Finally, our algorithm can predict the ball position for a reasonable time after any player occludes the ball.

The 2D ball detection can be extended to track the ball in 3D space. Constants and hyperparameters can be obtained by training the network beforehand on sport videos. Bounding box can be made adaptive using reinforcement learning. Semantic information can be extracted from video sequences.

Acknowledgements

Authors thank Mr. Dawood Kasim and Mr. Onkar Kaudki for their help in database generation work. The authors acknowledge Visvesvaraya PhD scheme, Ministry of Electronics and Information Technology (MeitY), Government of India for their financial support with grant number PhD-MLA/4(14)/2014 Date 19-12-2014.

References

[1] World Cup 2018, World Cup 2018 Breaks Viewing Records Across Streaming Platforms as Soccer Fans Tune in, 2018 (Accessed 23 November 2018) https://www.cnbc.com/2018/07/14/world-cup-2018-breaks-viewing-records-across-streaming-platforms-as-so.html.
[2] P.R. Kamble, A.G. Keskar, K.M. Bhurchandi, Ball tracking in sports: a survey, Artif. Intell. Rev. (2017), http://dx.doi.org/10.1007/s10462-017-9582-2.
[3] T. Kim, Y. Seo, K.-S. Hong, Physics-based 3D position analysis of a soccer ball from monocular image sequences, computer vision, in: Sixth International Conference on IEEE, 1998, 1998, 721–726.
[4] I.D. Reid, A. North, 3D Trajectories from a Single Viewpoint Using Shadows, BMVC, 1998, 51–52.
[5] Y. Ohno, J. Miura, Y. Shirai, Tracking players and estimation of the 3D position of a ball in soccer games, in: Pattern Recognition, 2000. Proceedings. 15th International Conference on IEEE, 2000, 145–148.
[6] G.S. Pingali, Y. Jean, I. Carlbom, Real time tracking for enhanced tennis broadcasts, computer vision and pattern recognition, in: Proc. 1998 IEEE Computer Society Conference on IEEE, 1998, 1998, 260–265.
[7] Y. Ohno, J. Miurs, Y. Shirai, Tracking players and a ball in soccer games, multisensor fusion and integration for intelligent systems, 1999, in: MFI'99. Proceedings. 1999 IEEE/SICE/RSJ International Conference on IEEE, 1999, 147–152.
[8] W. Chen, Y.-J. Zhang, Tracking ball and players with applications to highlight ranking of broadcasting table tennis video, in: Computational Engineering in Systems Applications, IMACS Multiconference on IEEE, 2006, 1896–1903.
[9] J. Ren, J. Orwell, G.A. Jones, M. Xu, Tracking the soccer ball using multiple fixed cameras, Comput. Vision Image Underst. 113 (2009) 633–642, http://dx.doi.org/10.1016/j.cviu.2008.01.007.
[10] U.B. Desai, S.N. Merchant, M. Zaveri, G. Ajishna, M. Purohit, H. Phanish, Small object detection and tracking: algorithm, analysis and application, in: International Conference on Pattern Recognition and Machine Intelligence, Springer, 2005, 108–117.
[11] D. Liang, Q. Huang, Y. Liu, G. Zhu, W. Gao, Video2 Cartoon: a system for converting broadcast soccer video into 3D cartoon animation, IEEE Trans. Consum. Electr. 53 (2007) 1138–1146, http://dx.doi.org/10.1109/TCE.2007.4341597.
[12] G. Zhu, C. Xu, Y. Zhang, Q. Huang, H. Lu, Event tactic analysis based on player and ball trajectory in broadcast video, in: Proceedings of the 2008 international conference on Content-based image and video retrieval, ACM, 2008, 515–524.
[13] Y. Huang, J. Llach, C. Zhang, A method of small object detection and tracking based on particle filters, in: PAttern Recognition, 2008. ICPR 2008. 19th International Conference on, IEEE, 2008, 1–4.
[14] G. Zhu, C. Xu, Q. Huang, Y. Rui, S. Jiang, W. Gao, H. Yao, Event tactic analysis based on broadcast sports video, IEEE Trans. Multimedia 11 (2009) 49–67, http://dx.doi.org/10.1109/TMM.2008.2008918.
[15] X. Cheng, Y. Shiina, X. Zhuang, T. Ikenaga, Player tracking using prediction after intersection based particle filter for volleyball match video, Signal and Information Processing Association Annual Summit and Conference (APSIPA) (2014) 1–4.
[16] G.S. Pingali, A. Opalach, Y.D. Jean, I.B. Carlbom, Instantly indexed multimedia databases of real world events, IEEE Trans. Multimedia 4 (2002) 269–282, http://dx.doi.org/10.1109/TMM.2002.1017739.
[17] X. Yu, H.W. Leong, C. Xu, Q. Tian, Trajectory-based ball detection and tracking in broadcast soccer video, IEEE Trans. Multimedia 8 (2006) 1164–1178, http://dx.doi.org/10.1109/TMM.2006.884621.
[18] Y. Liu, D. Liang, Q. Huang, W. Gao, Extracting 3D information from broadcast soccer video, Image Vision Comput. 24 (2006) 1146–1162, http://dx.doi.org/10.1016/j.imavis.2006.04.001.
[19] B. Chakraborty, S. Meher, A real-time trajectory-based ball detection-and-tracking framework for basketball video, J. Opt. U. K. 42 (2013) 156–170, http://dx.doi.org/10.1007/s12596-012-0108-7.
[20] C.Ó. Conaire, P. Kelly, D. Connaghan, N.E. O'Connor, Tennissense: a platform for extracting semantic information from multi-camera tennis data, in: 16th International Conference on Digital Signal Processing, IEEE, 2009, 2009, 1–6.
[21] Q. Huang, S. Cox, X. Zhou, L. Xie, Detection of ball hits in a tennis game using audio and visual information, Signal & Information Processing Association Annual Summit and Conference (APSIPA ASC) (2012) 1–10.
[22] F. Yan, W. Christmas, J. Kittler, Ball Tracking for Tennis Video Annotation, Computer Vision in Sports, Springer, 2014, http://dx.doi.org/10.1007/978-3-319-09396-3_2, 25–45.

[23] X. Zhou, Q. Huang, L. Xie, S. Cox, A two layered data association approach for ball tracking, in: IEEE International Conference on Acoustics, Speech and Signal Proc., IEEE, 2013, 2013, 2317–2321.
[24] J. Miura, T. Shimawaki, T. Sakiyama, Y. Shirai, Ball route estimation under heavy occlusion in broadcast soccer video, Comput. Vision Image Underst. 113 (2009) 653–662, http://dx.doi.org/10.1016/j.cviu.2008.10.005.
[25] V. Pallavi, J. Mukherjee, A.K. Majumdar, S. Sural, Ball detection from broadcast soccer videos using static and dynamic features, J. Vision Commun. Image Represent. 19 (2008) 426–436, http://dx.doi.org/10.1016/j.jvcir.2008.06.007.
[26] V. Pallavi, J. Mukherjee, A.K. Majumdar, S. Sural, Graph-based multiplayer detection and tracking in broadcast soccer videos, IEEE Trans. Multimedia 10 (2008) 794–805, http://dx.doi.org/10.1109/TMM.2008.922869.
[27] X. Wang, V. Ablavsky, H.B. Shitrit, P. Fua, Take your eyes off the ball: improving ball-tracking by focusing on team play, Comput. Vision Image Underst. 119 (2014) 102–115, http://dx.doi.org/10.1016/j.cviu.2013.11.010.
[28] X. Wang, E. Türetken, F. Fleuret, P. Fua, Tracking interacting objects optimally using integer programming, in: European Conference on Computer Vision, Springer, 2014, 17–32.
[29] A. Maksai, X. Wang, P. Fua, What Players Do with the Ball: a Physically Constrained Interaction Modeling, arXiv preprint arXiv:1511.06181, 2015.
[30] M. Takahashi, S. Yokozawa, H. Mitsumine, T. Mishina, Real-time ball-position measurement using multi-view cameras for live football broadcast, Multimed. Tools Appl. (2018) 1–22, http://dx.doi.org/10.1007/s11042-018-5694-1.
[31] V. Reno, N. Mosca, R. Marani, M. Nitti, T. D'Orazio, E. Stella, Convolutional neural networks based ball detection in tennis games, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition Workshops, 2018, 1758–1764.
[32] Y. LeCun, Y. Bengio, G. Hinton, Deep learning, Nature 521 (2015) 436, http://dx.doi.org/10.1038/nature14539.
[33] I. Goodfellow, Y. Bengio, A. Courville, Y. Bengio, Deep Learning, MIT Press, Cambridge, 2016.
[34] D.E. Rumelhart, G.E. Hinton, R.J. Williams, Learning representations by back-propagating errors, Nature 323 (1986) 533, http://dx.doi.org/10.1038/323533a0.
[35] Y. Bengio, A. Courville, P. Vincent, Representation learning: a review and new perspectives, IEEE Trans. Pattern Anal. 35 (2013) 1798–1828, http://dx.doi.org/10.1109/TPAMI.2013.50.
[36] J. Schmidhuber, Deep learning in neural networks: an overview, Neural Netw. 61 (2015) 85–117, http://dx.doi.org/10.1016/j.neunet.2014.09.003.
[37] R.S. Sutton, A.G. Barto, Reinforcement Learning: an Introduction, MIT Press Cambridge, 1998.
[38] J. Patterson, A. Gibson, Deep Learning: A Practitioner's Approach, first ed., O'Reilly Media, Inc., 2017.
[39] A. Karpathy, CS231n Convolutional Neural Networks for Visual Recognition, 2019 http://cs231n.github.io/.
[40] V. Nair, G.E. Hinton, Rectified linear units improve restricted boltzmann machines, in: Proc.27th International Conference on Machine Learning (ICML-10), 2010, 807–814.
[41] A. Krizhevsky, I. Sutskever, G.E. Hinton, Imagenet classification with deep convolutional neural networks, Adv. Neural Inf. Process. Syst. (2012) 1097–1105.
[42] C.M. Bishop, Pattern Recognition and Machine Learning (Information Science and Statistics), Springer-Verlag, 2006.
[43] K.P. Murphy, Machine Learning: A Probabilistic Perspective, MIT Press, 2012.
[44] L.Y. Pratt, Discriminability-based transfer between neural networks, Adv. Neural Inf. Process. Syst. (1993) 204–211.
[45] L. Pratt, B. Jennings, A survey of transfer between connectionist networks, Conn. Sci. 8 (1996) 163–184, http://dx.doi.org/10.1080/095400996116866.
[46] K. Chatfield, K. Simonyan, A. Vedaldi, A. Zisserman, Return of the Devil in the Details: Delving Deep into Convolutional Nets, arXiv preprint arXiv:1405.3531, 2014.
[47] H. Nam, B. Han, Learning multi-domain convolutional neural networks for visual tracking, computer vision and pattern recognition (CVPR), in: 2016 IEEE Conference on, IEEE, 2016, 4293–4302.
[48] J. Deng, W. Dong, R. Socher, L.-J. Li, K. Li, L. Fei-Fei, Image net: a large-scale hierarchical image database, computer vision and pattern recognition, CVPR 2009. IEEE Conference on, IEEE, 2009 (2009) 248–255, http://dx.doi.org/10.1109/CVPR.2009.5206848.
[49] B. Babenko, M.-H. Yang, S. Belongie, Robust object tracking with online multiple instance learning, IEEE Trans. Pattern Anal. 33 (2011) 1619–1632, http://dx.doi.org/10.1109/TPAMI.2010.226.
[50] J.F. Henriques, R. Caseiro, P. Martins, J. Batista, High-speed tracking with kernelized correlation filters, IEEE Trans. Pattern Anal. 37 (2015) 583–596, http://dx.doi.org/10.1109/TPAMI.2014.2345390.
[51] P. Jaccard, Étude comparative de la distribution florale dans une portion des Alpes et des Jura, Bull. Soc. Vaudoise Sci. Nat. 37 (1901) 547–579.
[52] J. Redmon, S. Divvala, R. Girshick, A. Farhadi, You only look once: unified, real-time object detection, in: Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition, 2016, 779–788.
[53] G.W.R.L, Fastest Football Kick, Guinness World Records Limited 2018, 2001, 2018 (Accessed 23 November 2018) http://www.guinnessworldrecords.com/world-records/fastest-football-kick/.
[54] T. D'Orazio, M. Leo, N. Mosca, P. Spagnolo, P.L. Mazzeo, A semi-automatic system for ground truth generation of soccer video sequences, advanced video and Signal based surveillance, in: AVSS'09. Sixth IEEE International Conference on, IEEE, 2009, 2009, 559–564.