Generation of Ball Possession Statistics in Soccer using Minimum-Cost Flow Network

Saikat Sarkar
University of Calcutta
Kolkata, India
to.saikatsarkar17@gmail.com

Amlan Chakrabarti
University of Calcutta
Kolkata, India
acakcs@caluniv.ac.in

Dipti Prasad Mukherjee
Indian Statistical Institute
Kolkata, India
dipti@isical.ac.in

Abstract

We present an automatic technique for calculating ball possession statistics from the video of a soccer match. The possession statistics is generated based on the number of valid passes made by an individual team. A valid pass is detected as a split or merge event of the ball with a player. A pass starts when the ball splits from a player. A pass ends when the ball merges with a player. We use a minimum-cost flow network to model number of valid passes in the soccer match. The ball and the players represent the nodes of the network. Each edge of the network is associated with a cost derived from the between-frame correspondences of the ball and the players. The total flow through the network is optimized to track the number of valid passes. Experimental results show that the accuracy of the proposed method is at least 4% better than that of a similar approach.

1. Introduction

Ball possession statistics is a popular performance indicator in a soccer match. Several studies show that possession percentages of the successful teams are higher compared to the unsuccessful teams [2, 3, 4]. As an example, the national team of Germany had the highest 56.71% average possession in the 2014 FIFA World Cup and was the champion of the tournament [10]. All existing methods of ball possession calculation are done manually [14]. One such method, used by Deltatre [9] manually computes the time duration for which a team has the ball.

The time based approach suffers from human error and relies on the expertise of data loggers. Opta Sports, which provides soccer statistics for the Premier League, calculates ball possession based on the pass count of the teams [16]. They manually count the number of successful passes executed by a team during the game. Then the possession of

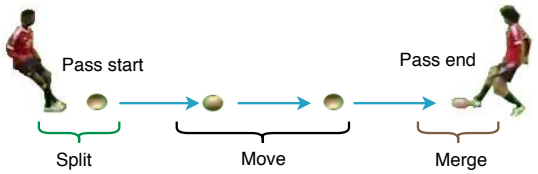


Figure 1. An example of a valid pass.

$team\ i$ is calculated as follows.

$$Possession(team\ i) = \frac{\#Valid\ passes\ by\ team\ i}{\#Valid\ passes\ by\ both\ teams}, \quad (1)$$

where $\#Valid\ passes$ denotes the number of valid passes. A pass is called valid if the pass is made between two different players of the same team. We count the successful passes made by each team and generate the possession statistics using (1). A visualization of a single pass between two players of the same team is shown in Fig. 1.

Different approaches have been proposed for tracking soccer ball and players [21, 15, 12], and for event detection in a soccer match, such as shot on goal, penalty kick, offside [22, 11]. But, very little attention has been paid till date to generate ball possession statistics. In [23], the authors have detected the pass event by analyzing the ball velocity curve. Local minima or maxima in the ball velocity curve represents pass start or end event. Once a pass is detected, the player nearest to the ball is used to get the information of the team in possession of the ball. Based on the pass event, the video is divided into a set of segments called touching segments. The possession statistics of two teams is then calculated by counting the number of touching segments of the teams. A major limitation of the mentioned approach is that the spatial coordinate of the soccer ball and players has to be provided as input to the algorithm to work. Miss-detection of passes arise when there is marginal change in the velocity curve.

The objective of our work is to automatically calculate

ball possession statistics from a soccer video. We map the pass detection process in a graph theoretic framework. We build a minimum-cost network model to detect ball pass event. For a pass start event the connected components representing the ball and the player split from each other. In case of a pass end event, connected components representing the ball and the player overlap each other and become a single connected component. The minimum-cost network detects if objects (ball and players) move, split or merge and appear or disappear between two consecutive frames.

Minimum-cost network has been successfully applied in visual object tracking under data association based tracking framework [13, 19]. Under this framework, first, objects are identified inside a frame. Then corresponding objects are linked across the frames by solving a linearly constrained optimization function.

The rest of the paper is organized as follows. In Section 2, we discuss our proposed methodology including the minimum-cost network model and the pass counting strategy. Experimental results are shown in Section 3 followed by conclusions in Section 4.

2. Proposed method

The block diagram of the proposed method is shown in Fig. 2. The estimation of possession statistics starts from

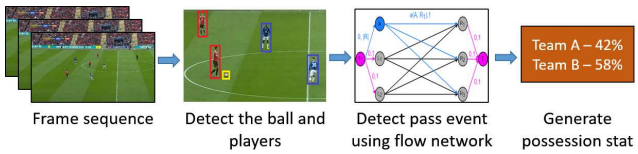


Figure 2. Block diagram of our proposed method.

detecting the soccer ball and players. We have implemented [20] to extract foreground connected components of a soccer video frame.

After segmenting the background representing the soccer field [20], we employ SVM to label the foreground connected components. The classes are, (1) *ball*, (2) *team A*, (3) *team B* and (4) *other*. We use multi-class SVM [5] with RBF kernel as classifier. We use area, eccentricity and RGB histogram as features for classification. A typical detection pipeline is shown in Fig. 3. In the next section we describe the detection of ball pass.

2.1. Detection of ball pass

We use the minimum-cost network model to detect the ball pass event. In a traditional flow network based tracking framework [17], between-frame objects are linked to track individual objects. By enforcing the flow limit through the edges of the network to be one, the paths or the flows going through the network are guaranteed to be mutually exclusive. In our case, pass event detection requires detection

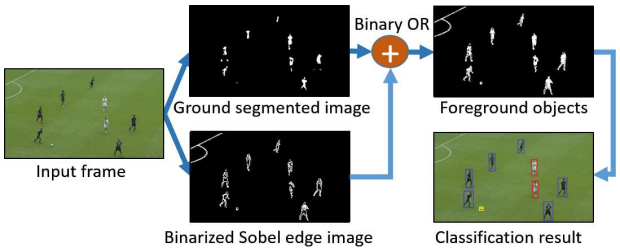


Figure 3. Detection of the ball and players with team labels. The ball is marked with yellow rectangle. Players of two teams are marked with blue and red rectangles.

of connected components representing players and the ball. These connected components split (for pass start) or merge (for pass end) from previous frame to the current frame. However, a single edge of the network can handle only one such split or merge event. In addition, we augment the network with appear and disappear nodes to detect suddenly appearing or disappearing players or ball from the video. Next we describe how to construct the flow network.

2.1.1 Construction of the network

We build a flow network from the objects detected in two consecutive frames. Our objective is to correctly associate the objects in previous frame to those in the current frame. Our formulation does not require any prior assumptions on the number of objects to be associated. Assuming there are total d number of objects in the current frame and the previous frame, we send d amount of flow over the network. The minimum cost solution provides a correct association of all d objects across frames.

A flow network $G = (V, E)$ is a directed acyclic graph with V nodes and E edges. In our case, nodes of the network represent detected foreground objects like players and the ball. The edges of the network connect objects between two consecutive frames. For consecutive frames L and R, Each edge $(u, v) \in E$, $u \in V_L$, $v \in V_R$, has a non-negative capacity $t(u, v) \geq 0$. The nodes of the previous frame L is represented by V_L and the nodes of the current frame R is represented by V_R.

Apart from the capacity $t(u, v)$, each edge has a real-valued cost $a(u, v)$ that denotes cost per unit flow in the (u, v). The cost represents correspondence or similarity between u and v. The cost is minimum for correct association between frames. If we send $x(u, v)$ units of flow over edge (u, v), we incur a cost of $a(u, v)x(u, v)$.

Two vertices that are added in the flow network are the source $T+$ and sink $T-$. The source vertex produces the flow and the terminal vertex consumes the flow. The cost flow problem can be formulated as the minimization problem of the product of cost and flow along the edges of the

network as [1],

$$\text{Minimize} \sum_{(u,v)\in E} a(u,v)x(u,v), \tag{2}$$

subject to,

$$\sum_{u\in V} x(u,w) - \sum_{v\in V} x(w,v) = b(w), \quad \forall w \in V.$$

The difference between all incoming flow from u to w and all outgoing flow to v from w is equal to the flow requirement $b(w)$. The flow requirement is greater than zero for the source vertex, less than zero for the terminal vertex and zero for all intermediate vertices, known as flow conservation constraint.

The decision variables in the minimum-cost flow problem is $x(u,v)$, the flow over an edge (u,v). The problem is to send d amount of flow from source to terminal vertex in such a way that the total cost incurred by the flow is minimized.

We now build the flow network based on the objects detected. Let N_1 objects be detected in the previous frame L and N_2 objects in the current frame R. Therefore, the vanilla flow network has $(N_1 + N_2)$ nodes. Each vertex $u \in V_L$ is connected with every vertex $v \in V_R$, directed from u to v resulting in total N_1N_2 edges. Each of the edges has a cost $a(u,v)$ and capacity $t(u,v) = 1$. In addition, a source vertex $T+$ and a terminal vertex $T-$ are added to the network. Every vertex u in the previous frame has an incoming edge from the source vertex $T+$ with cost $a(T+,u) = 0$ and capacity $t(T+,u) = 1$. Similarly, every vertex v in the current frame has an outgoing edge to the terminal vertex $T-$ with cost $a(v,T-) = 0$ and capacity $t(v,T-) = 1$. A simple flow network with $N_1 = 2$ and $N_2 = 3$ is shown in Fig. 4.

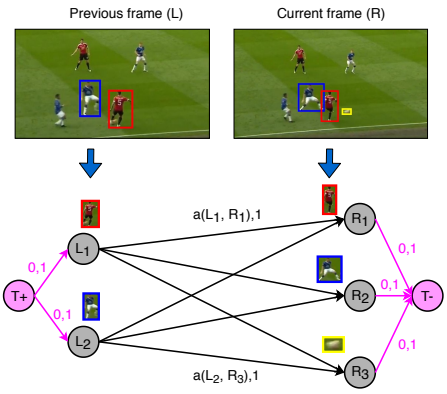


Figure 4. An example of flow network constructed based on the objects detected in two consecutive frames. For each edge, the cost and capacity is written as (cost, capacity). The vertices in L and R are marked in gray while the edges are in black color. The source and the terminal vertices and associated edges are marked with pink color.

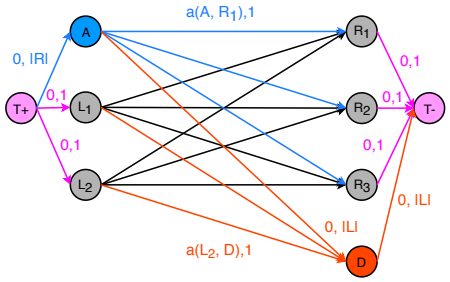


Figure 5. The flow network with the appear vertex A and the disappear vertex D. The appear vertex and associated edges are marked with blue. The disappear vertex and associated edges are marked with orange.

The basic network architecture proposed above needs to be augmented for reasons described next. The first modification is due to appearance (disappearance) of the player and/or ball in (from) a frame. The second modification is due to split or merge of connected components. These are detailed next.

2.1.2 Appearing and disappearing objects

The players and the ball may appear in or disappear from the frame. To incorporate this scenario, appear vertex A and disappear vertex D are introduced in the network. To consider the possibility that an object may suddenly appear in R, all of the N_2 vertices in R are connected to the appear vertex A in L. Each of these edges connecting A and $v \in V_R$ has a cost $a(A, v \in V_R)$ with unit capacity. An edge from the source vertex to the appear vertex with cost $a(T+,A) = 0$ and capacity $t(T+,A) = |R|$, is added for flow conservation, where $|R|$ denotes total number of nodes in frame R. The capacity of the edge $t(T+,A)$ is set as $|R|$ as appear vertex has $|R|$ number of outgoing edges to R.

Analogously, the disappear vertex D has an edge from each of the N_1 vertices of frame L. This means that each object in the previous frame L may disappear in the next frame. Each of these edges connecting $u \in V_L$ with D has a cost $a(u \in V_L, D)$ with unit capacity. An edge from D to the terminal vertex with cost $a(D,T-) = 0$ and capacity $t(D,T-) = |L|$ is added. The capacity of the edge $t(D,T-)$ is set to $|L|$ as the disappear vertex has $|L|$ incoming edges from L. An edge from A to D with zero cost and capacity $t(A,D) = |L|$ is added to preserve the flow conservation constraint of the network. The modified network with appear and disappear vertices is shown in Fig. 5. Next we detail merge and split of connected components.

2.1.3 Splitting and merging of objects

To model splitting and merging of objects, split vertices S and merge vertices M are introduced in the network. Each split vertex has one incoming edge from each of the L vertices and two outgoing edges to two nodes of R. Each of the outgoing edge of split node has zero cost and unit capacity. The incoming edge has cost $a(u, S_{v1,v2})$, $v1, v2 \in V_R$ and unit capacity. There will be total ${}^{N_2}C_2$ split vertices for N_2 number of objects in the current frame.

Similarly, any two connected components of L may merge. Therefore, each merge vertex has two incoming edges from two nodes of the L and one outgoing edge to each node of R. Each of the incoming edges of M has zero cost and unit capacity. The outgoing edge has cost $a(M_{u1,u2}, v)$, $u1, u2 \in V_L$ and unit capacity. For simplicity, we consider that only two objects can be merged at a time. For N_1 number of objects in the previous frame, there will be total ${}^{N_1}C_2$ merge vertices.

Merge vertices violate the flow conservation constraint because each M node has two incoming and one outgoing edges. So, an outgoing edge from each vertex of M to the disappear vertex D is added with zero cost and unit capacity. Similarly an incoming edge from appearance vertex A is added to every split vertices S with zero cost and unit capacity. The extended network with split and merge vertices is shown in Fig. 6.

A constraint is needed to make sure that exactly two units flow to and from split and merge vertices when these vertices are part of the solution. This is called edge coupling constraint [17]. Edge coupling ensures that the flow through a split (or merge) vertex is either 0 or 2. To ensure all of the objects on the previous and current frame are part of the solution, the flow requirement d of the network is set to $d = N_1 + N_2$.

The cost and capacity of different types of edges of the network is summarized in the Table 1. Determination of cost of the edges is described in the next section.

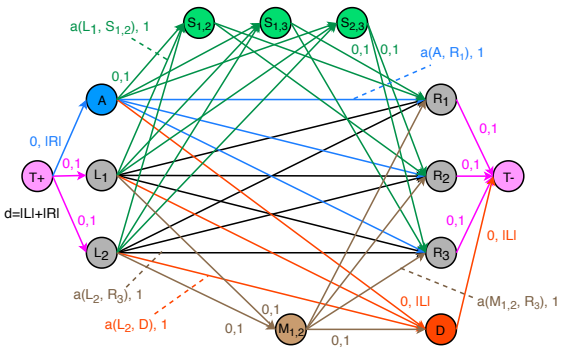


Figure 6. Extended flow network with split and merge vertices. The split vertices and associated edges are marked with green. The merge vertices and associated edges are marked with brown. Flow requirement of the network is d.

Table 1. Summarization of cost and capacity of different edges of the network.

Edge	Cost	Capacity	Remarks
$T+ \rightarrow V_L$	0	1	-
$T+ \rightarrow A$	0	$\|R\|$	Node A is connected with $\|R\|$ objects in the current frame (R)
$V_L \rightarrow V_R$	$a(V_L, V_R)$	1	-
$A \rightarrow V_R$	$a(A, V_R)$	1	-
$A \rightarrow S_{v1,v2}$	0	1	-
$A \rightarrow D$	0	$\|L\|$	To preserve the flow conservation constraint of the network
$V_L \rightarrow S_{v1,v2}$	$a(V_L, S_{v1,v2})$	1	-
$V_L \rightarrow M_{u1,u2}$	0	1	-
$M_{u1,u2} \rightarrow V_R$	$a(M_{u1,u2}, V_R)$	1	-
$M_{u1,u2} \rightarrow D$	0	1	-
$V_R \rightarrow T-$	0	1	-
$D \rightarrow T-$	0	$\|L\|$	Node D is connected with $\|L\|$ objects in the previous frame

2.1.4 Calculation of association costs

We want the edge costs to be low for correct associations between connected components of two consecutive frames. Also, we are trying to minimize the overall cost of the network. The cost of an edge has three components, the color similarity component ρ, the distance component δ and the angular displacement component θ as detailed next.

Assume an object u in frame L is defined using a bounding box. The histogram h_u of u is obtained by concatenating the histograms of equal length red, green, and blue channels of the object. The concatenated histogram h_u has ν number of bins. The histogram is normalized by dividing count in each bin by sum of all bins of the histogram. The color similarity component ρ between objects u in frame L and v in frame R is defined as difference between h_u and h_v as follows,

$$\rho(u, v) = \sum_{\nu} |h_u - h_v|. \tag{3}$$

Assume the center location of the bounding box of u is (u_x, u_y). The distance component between u and v is defined as,

$$\delta(u, v) = \frac{\sqrt{||(u_x, u_y) - (v_x, v_y)||^2}}{\max(c, r)}, \tag{4}$$

where c is the width and r is the height of the frame. Naturally, the correct association between u and v should have minimum displacement between two consecutive frames.

Finally, the angular displacement between u and v is given by,

$$\theta(u, v) = 1 - \frac{(u_x, u_y) \cdot (v_x, v_y)}{||(u_x, u_y)||||(v_x, v_y)||}. \tag{5}$$

We assume that the direction of movement of an object is consistent between frames. The between-frame associations are penalized if the cosine of angle between the position vectors increases.

Association cost We now calculate the cost of each between-frame association. This cost includes cost due to color similarity, between-frame distance and angular displacement. The total cost of association of u with v is calculated as,

$$a(u, v) = \rho(u, v) + \lambda_1 \delta(u, v) + \lambda_2 \theta(u, v), \qquad (6)$$

where λ_1 and λ_1 are two scaling parameters.

Splitting and merging cost To calculate splitting cost, an object in the previous frame is associated with two objects on the current frame. The cost of associating u_1 with v_1 and v_2 is calculated as average of the total association cost between u_1 and v_1 and u_1 and v_2.

$$a(u_1, (v_1, v_2) = \frac{a(u_1, v_1) + a(u_1, v_2)}{2}. \qquad (7)$$

Similarly, the cost of merging u_1 and u_2 with v_1 is calculated as,

$$a((u_1, u_2), v_1) = \frac{a(u_1, v_1) + a(u_2, v_1)}{2}. \qquad (8)$$

Appearance and disappearance cost When an object appears, it is present in the current frame and absent from the previous frame. When an object disappears, it is absent from the present frame but was present in the previous frame. Note that the appear node A and the disappear node D are dummy nodes for which we do not have any defined bounding boxes. So we cannot calculate ρ, δ, θ for the pairs (A, v) and (u, D). We calculate the appearance and disappearance cost based on how far the object is from image border. If the frame has width c and height r, then the cost of appearance of an object v is calculated as,

$$a(A, v) = \frac{\min(v_x, v_y, (c - v_x), (r - v_y))}{\max(c, r)}. \qquad (9)$$

Similarly the cost of disappearance of an object u is calculated as,

$$a(u, D) = \frac{\min(u_x, u_y, (c - u_x), (r - u_y))}{\max(c, r)}. \qquad (10)$$

2.1.5 Solving the cost flow problem using linear programming

The minimum-cost flow problem described in (2) can be solved using a linear programming problem. An incidence matrix $\mathbf{\Gamma}$ is used to represent the flow network. The incidence matrix has the size $|V| \times |E|$ where each column corresponds to an edge and each row corresponds to a vertex of the flow network as shown in Fig. 7. The entry in a column corresponding to the start of the edge is set to +1, and the end of the edge is set to -1. Rest of the entries are zero. First column of $\mathbf{\Gamma}$ in Fig. 7 represents the edge $(T+, L_1)$ of Fig. 6.

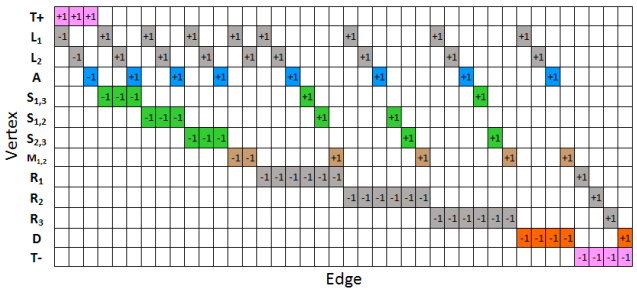


Figure 7. The incidence matrix $\mathbf{\Gamma}$ of Fig. 6. Blank entries represent zero.

The incidence matrix representation of the network has no restriction on selecting exactly zero or two edges of the split (or merge) vertices. In order to enforce the edge coupling constraint, the incidence matrix is modified similar to [17], known as coupled edge incidence matrix. The matrix $\boldsymbol{I}$ in Fig. 8 represents the coupled edge incidence matrix of the network shown in Fig. 6. In the coupled edge incidence matrix, the incoming and outgoing edges of split and merge vertices are combined into a single column.

For example, the columns 4, 6, 18 and 30 of Fig. 7 are coupled into the column 4 of $\boldsymbol{I}$ in Fig. 8. The combined column has four non-zero entries, two +1 (row 2 and 4 of column 4) for starting of incoming edges and two -1 (row 5 and 7 of column 4) represent ending of outgoing edges of a split (or merge) vertex. Split and merge vertices are no longer required and removed from the coupled-edge incidence matrix of Fig. 8.

Using the coupled-edge incidence matrix $\boldsymbol{I}$, finding the optimal matches corresponds to finding a subset of columns in the matrix such that the cost of these columns are minimized. This can be represented as an integer linear programming (ILP) problem as,

$$\underset{\boldsymbol{x}}{\text{Minimize}} \quad \boldsymbol{a}^T \boldsymbol{x}, \text{subject to} : \boldsymbol{I}\boldsymbol{x} = \boldsymbol{b}, 0 \leq \boldsymbol{x} \leq \boldsymbol{t}, \quad (11)$$

where, $\boldsymbol{a}$ is the vector of costs of size $|E| \times 1$. Each $a_i \in \boldsymbol{a}$ represents the cost of an edge i that we have calculated in the previous section. Cost of $\boldsymbol{a}$ will be derived from (6) to (10). The vector $\boldsymbol{b}$ is of size $|V| \times 1$ and represents flow requirement of the vertices. Flow requirement at source vertex is $+d$ and at terminal vertex is $-d$. Flow requirement is zero for all other vertices. The vector $\boldsymbol{x}$ of size $|E| \times 1$ is

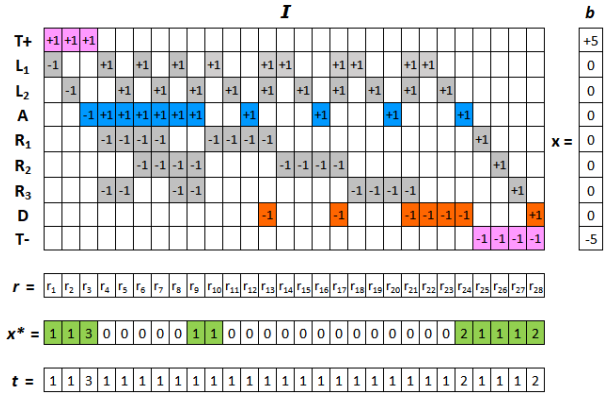


Figure 8. Integer linear programming formulation of Fig. 6. The flow requirement of the network is $d = 5$. Non-zero entries in the optimal solution $\boldsymbol{x}^*$ (marked with green) represent the edges that are part of the solution.

the solution vector whose coefficients represent amount of flow passed through the corresponding edge. The vector $\boldsymbol{t}$ of size $|E| \times 1$ represents capacity of each individual edge of the network.

The integer linear programming based solution of Fig. 6 is shown in Fig. 9. The optimal solution $\boldsymbol{x}^*$ of the problem can be found using linear programming [1]. The vector $\boldsymbol{x}^*$ represents optimal flow over each edge of the network. The solution would never turn to an inconsistent state like unbounded solution or no solution because of the flow requirement d and the flow capacity $\boldsymbol{t}$ of the network [17].

2.2. Counting of passes and generation of possession statistics

A non-zero entry $x_i \in \boldsymbol{x}^*$ represents that the edge i is part of the solution. Now, an edge of the coupled edge incidence matrix $\boldsymbol{I}$ that connects $u \in V_L$ to $v \in V_R$ indicates object u moves to v in the current frame. The split event is

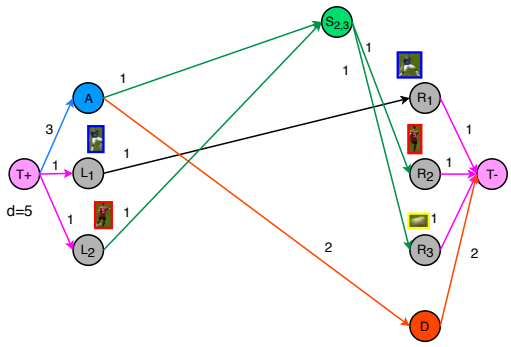


Figure 9. The solution of Fig. 6. L_1 moves to R_1 and L_2 splits into R_2 and R_3. Edge values represent calculated flow on each edge.

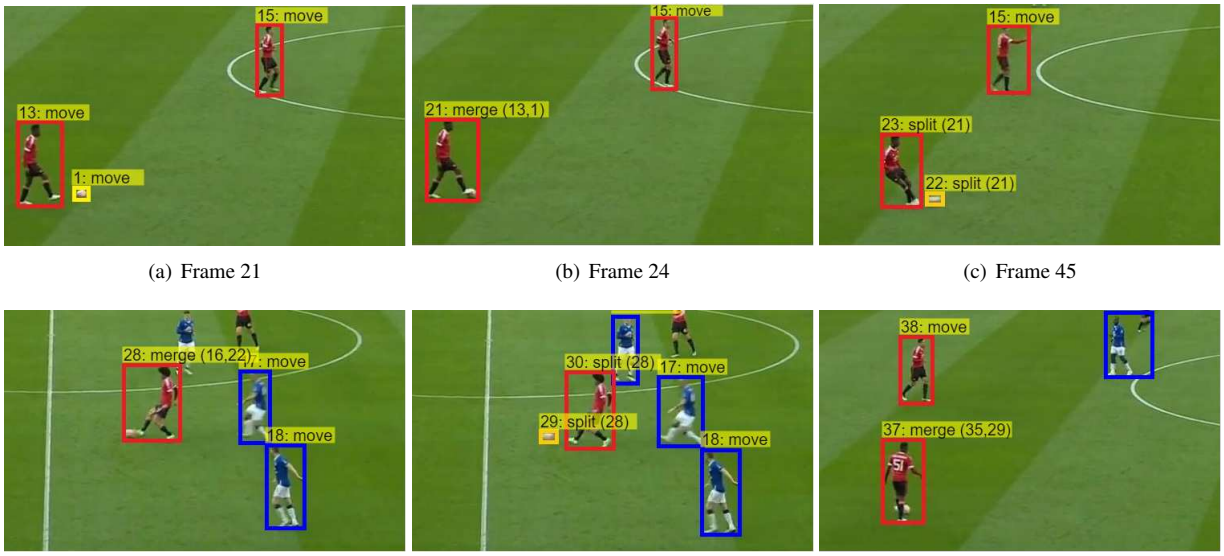
detected when we get a coupled edge that connects a vertex u from previous frame to v_1 and v_2 of the current frame. The coupled edge in column 9 of $\boldsymbol{I}$ connects L_2 to R_2 and R_3, which denotes L_2 splits to R_2 and R_3. Similarly, finding an edge that connects u_1, u_2 from previous frame to v of current frame indicates merge event. Appearance (or disappearance) event is detected on getting an edge that connects object A to v (or u to D).

Now, if we get any solution that involves splitting or merging of a player and the ball, we say a pass event has occurred. We maintain unique id for each individual object moving across the frames. If an object moves from one frame to the next frame, the id get unchanged. For appearance of a new object and splitting or merging of two objects, a new id is assigned. As mentioned earlier, a pass start event followed by a pass end event is considered as a single pass. The pass is valid if both the players are of the same team but their ids are different. We then follow (1) to keep tab on

Algorithm 1: Generation of possession statistics

Input : Frames of soccer video.
Output: Possession statistics of two teams.

Initialize *pass_count_A* = 0
Initialize *pass_count_B* = 0
while *Video read not complete* **do**
  Read the current frame and the previous frame
  Construct the cost flow network as in Section 2.1.1
  Convert the graph to an coupled edge incidence matrix $\boldsymbol{I}$ as in Section 2.1.5
  Get x^* by solving (11) using ILP
  Find an edge $x_i \in x^*$ such that x_i is non-zero and the ball is split from or merged with a player p through x_i
  if *split==TRUE* **then**
    Set *pass_status* = *'pass start'*
    Set *l* = *p*
  end
  if *merge==TRUE* **then**
    if *pass_status is 'pass start'* **then**
      if *Team label of p and l is 'A'* **then**
        pass_count_A += 1
      end
      if *Team label of p and l is 'B'* **then**
        pass_count_B += 1
      end
    end
    Set *pass_status* = *'pass end'*
  end
end
Generate possession statistics by *pass_count_A* and *pass_count_B* using (1);



(a) Frame 21 (b) Frame 24 (c) Frame 45

(d) Frame 73 (e) Frame 75 (f) Frame 111

Figure 10. Text above each object represents the id of the object followed by event status. At frame 24 a merge event of the ball (id=1) with the player (id=13) is initiated. This denotes a pass end event. The merged objects (id=21) split at frame 45. This is starting of a new pass. The pass ends at frame number 73. A new pass starts at frame 75 and ends at frame 111. Frames are cropped for better visualization.

ball possession statistics.

The overall procedure of generation of ball possession statistics is summarized in Algorithm 1. Fig. 10 represents visualization of pass detection process on a frame sequence. A split event of a player and the ball happens at frame 45 followed by merge event at frame 73. This split followed by merge denotes a valid pass for the team marked in red.

3. Experimental results

To evaluate the performance of our model, we experiment with different broadcast soccer video clips available at YouTube [6, 7, 8]. The videos are encoded in mp4 format with 1280x720 resolution and 25 frame rate per second. We have reported the results of 8 long and medium shot video clips in this paper. Each of the clips is about 8-10 minutes duration.

Ground truth \ Our method	Ball	Team A	Team B	Other	Accuracy
Ball	23350	0	0	1650	93.40 %
Team A	0	22650	1683	667	90.60 %
Team B	0	1866	22300	834	89.20 %
Other	1874	389	227	22510	90.04 %

Figure 11. Classification result.

For offline training of the classifier, we have manually marked the ball, players with team label and other objects in video frames. We then train SVM based object classifier with the labeled objects. We use 25000 images of each category for training. The classification results for 3-fold cross validation is shown in Fig. 11. We find the highest classification accuracy for the ball class.

To validate the ball possession stat result of our method, we have prepared ground truth. We mark each frame of the clip with tuple $\{0, A, B\}$, where A denotes that the ball is in possession of team A. Similarly, B denotes that the ball is with team B and 0 indicates the situation when the ball is not in control of any team. For each of the video clips, we

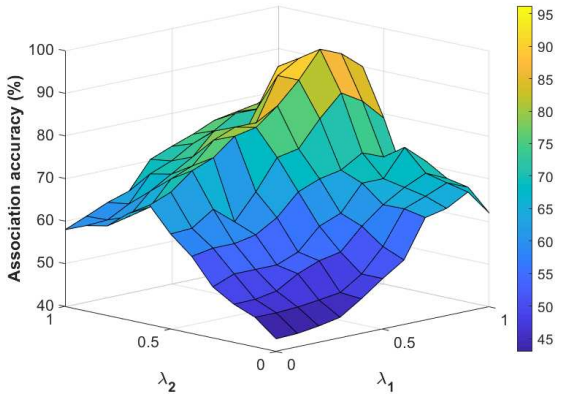


Figure 12. Plot of association accuracy for different values of λ_1 and λ_2.

calculate the possession statistics by manually counting the passes.

The accuracy of association of objects between consecutive frames for different values of λ_1 and λ_2 in (6) is shown in Fig. 12. The plot of Fig. 12 shows that the highest association accuracy is obtained for $\lambda_1 = 0.87$ and $\lambda_2 = 0.63$, which we have used for our experiments. We experimentally set the bin size $\nu = 150$ for the concatenated histograms in (3).

Table 2. Comparison of pass count.

Videos	Ground Truth		Our method		[23]	
	A	B	A	B	A	B
Video 1	72	18	78	29	91	35
Video 2	39	32	21	42	50	44
Video 3	34	46	54	33	23	53
Video 4	37	64	37	79	43	55
Video 5	23	54	29	68	31	76
Video 6	32	36	28	41	44	18
Video 7	61	19	76	24	73	39
Video 8	29	36	29	54	25	58

Table 2 represents comparison of the pass count of our method with ground truth data and the method proposed by [23]. Pass count for [23] is generated by analyzing the ball velocity curve drawn from known ball positions. Columns A and B in Table 2 represent pass count for team A and team B respectively. Table 3 presents comparison of the ball possession statistics of different methods. The possession statistics is calculated based on obtained pass count as listed in Table 2. Table 4 shows the average error of pass count as well as of possession statistics. The error is calculated by comparing the detected result with ground truth information. We have experimentally found that incorrect detection of ball has higher impact than the incorrect detection of a player for calculating possession stat.

Table 3. Comparison of possession stat.

Videos	Ground Truth		Our method		[23]	
	A (%)	B (%)	A (%)	B (%)	A (%)	B (%)
Video 1	80	20	73	27	72	28
Video 2	55	45	33	67	53	47
Video 3	43	57	62	38	30	70
Video 4	37	63	32	68	44	56
Video 5	30	70	30	70	29	71
Video 6	47	53	41	59	71	29
Video 7	76	24	76	24	65	35
Video 8	45	55	35	65	30	70

3.1. Analysis of computational complexity

Given N_1 number of objects in the previous frame and N_2 number of objects in the current frame, we create $N_1 + N_2$ number of vertices and $N_1 N_2$ edges. Adding the source and terminal vertices increase the number of vertices by 2 and number of edges by $N_1 + N_2$. When the appear and disappear vertices are added to the network, the number of vertices are increased by 2 and edges are increased by $N_1 + N_2 + 3$. Adding the split and merge vertices does not increase the rows of coupled edge incidence matrix $\boldsymbol{I}$, but increase the columns by $N_1 \times^{N_2} C_2 + N_2 \times^{N_1} C_2$.

Table 4. Comparison of error of pass count and possession stat.

Method	Pass count error (%)		Possession statistics error (%)	
	Team A	Team B	Team A	Team B
Proposed method	7.6	21.3	17.7	19.3
[23]	16.2	23.9	20.6	24.8

The coupled edge incidence matrix has total $z = (N_1 + N_2 + 2 + 2)$ rows and $\eta = ((N_1 N_2) + (N_1 + N_2) + (N_1 + N_2 + 3) + (N_1 \times^{N_2} C_2 + N_2 \times^{N_1} C_2))$ columns. Columns of the incidence matrix represent the variable x_i and rows represent constraints. Therefore, we have total η variables and z constraints in the ILP. Solving an ILP problem is considered NP-Complete. The ILP in (11) can be solved with runtime complexity $\mathcal{O}((\eta Q)^{z+1})$ [18], where $Q = \eta(z\alpha^{2z+1})$ and x_i takes values between 0 to α. Foreground object detection using binary connected component analysis requires $\beta = \mathcal{O}((rc)^2)$ operations, where c and r are the width and height of the input image respectively. Runtime complexity of applying our method on a video of length ψ is $\mathcal{O}(\psi\beta(\eta Q)^{z+1})$.

The proposed method takes 3.43 seconds on an average to process a pair of frames with an unoptimized MATLAB R2018a code on a PC with Intel i5 2.3 GHz processor, 8 GB of RAM and Windows 10 operating system.

4. Conclusions

We propose a method to automatically generate ball possession statistics in broadcast soccer video. Experimental results show that our method is promising and efficient. We have optimized network flow model using an appropriate cost function. The cost function may be improved incorporating motion vector of the ball and players. Unsupervised team labeling may be explored instead of currently implemented SVM classification. In future we plan to integrate the results of ball possession stat of close shot video frames with the results of long and medium shot soccer video. Effects of other situations on ball possession stat like effect of long pass, dribbling, off-side, different shots and play breaks need to be analyzed using an augmented flow network model. The optimization of objective function and its implementation also need a closer look for real-time result.

References

[1] R. K. Ahuja, T. L. Magnanti, and J. B. Orlin. *Network flows: theory, algorithms and applications*. Prentice Hall, 1993.

[2] J. A. Araya and P. Larkin. Key performance variables between the top 10 and bottom 10 teams in the english premier league 2012/13 season. *Human Movement, Health and Coach Education (HMHCE)*, 2:17–29, 2013.

[3] P. S. Bradley, C. Lago-Peñas, E. Rey, and A. Gomez Diaz. The effect of high and low percentage ball possession on physical and technical profiles in english fa premier league soccer matches. *Journal of Sports Sciences*, 31(12):1261–1270, 2013.

[4] J. Castellano, D. Casamichana, and C. Lago. The use of match statistics that discriminate between successful and unsuccessful soccer teams. *Journal of human kinetics*, 31:137–147, 2012.

[5] C.-C. Chang and C.-J. Lin. LIBSVM: A library for support vector machines. *ACM Transactions on Intelligent Systems and Technology*, 2:27:1–27:27, 2011. Software available at `http://www.csie.ntu.edu.tw/~cjlin/libsvm`.

[6] Dataset1. `https://www.youtube.com/watch?v=7uHGd7yNm6I`. [Online; accessed 13-March-2019].

[7] Dataset2. `https://www.youtube.com/watch?v=cC18Y--L-7w`. [Online; accessed 13-March-2019].

[8] Dataset3. `https://www.youtube.com/watch?v=GGhhbMOp6yY`. [Online; accessed 13-March-2019].

[9] H. Glasser. The Problem With Possession: The Inside Story of Soccers Most Controversial Stat. `http://www.slate.com/blogs/the_spot/2014/06/27/soccer_possession_the_inside_story_of_the_game_s_most_controversial_stat.html`. [Online; accessed 13-March-2019].

[10] K. GÖRAL. Passing success percentages and ball possession rates of successful teams in 2014 fifa world cup. *International Journal of Science Culture and Sport (IntJSCS)*, 3(1):86–95, 2015.

[11] H. Jiang, Y. Lu, and J. Xue. Automatic soccer video event detection based on a deep neural network combined cnn and rnn. In *Tools with Artificial Intelligence (ICTAI), 2016 IEEE 28th International Conference on*, pages 490–494. IEEE, 2016.

[12] P. R. Kamble, A. G. Keskar, and K. M. Bhurchandi. Ball tracking in sports: a survey. *Artificial Intelligence Review*, pages 1–51, 2017.

[13] L. Leal-Taixé, G. Pons-Moll, and B. Rosenhahn. Everybody needs somebody: Modeling social and grouping behavior on a linear programming multiple people tracker. In *Computer Vision Workshops (ICCV Workshops), 2011 IEEE International Conference on*, pages 120–127. IEEE, 2011.

[14] D. Link and M. Hoernig. Individual ball possession in soccer. *PloS one*, 12(7):e0179953, 2017.

[15] M. Manafifard, H. Ebadi, and H. A. Moghaddam. A survey on player tracking in soccer videos. *Computer Vision and Image Understanding*, 159:19–46, 2017.

[16] B. Optasports. A Ball Possessed. `http://www.optasports.com/news-area/blog-a-ball-possessed.aspx`, 2011. [Online; accessed 13-March-2019].

[17] D. Padfield, J. Rittscher, and B. Roysam. Coupled minimum-cost flow cell tracking for high-throughput quantitative analysis. *Medical image analysis*, 15(4):650–668, 2011.

[18] C. H. Papadimitriou. On the complexity of integer programming. *Journal of the ACM (JACM)*, 28(4):765–768, 1981.

[19] H. Pirsiavash, D. Ramanan, and C. C. Fowlkes. Globally-optimal greedy algorithms for tracking a variable number of objects. In *Computer Vision and Pattern Recognition (CVPR), 2011 IEEE Conference on*, pages 1201–1208. IEEE, 2011.

[20] U. Rao and U. C. Pati. A novel algorithm for detection of soccer ball and player. In *In Communications and Signal Processing (ICCSP), 2015 International Conference on*, pages 344–348, 2015.

[21] S. Sanyal, A. Kundu, and D. P. Mukherjee. On the (soccer) ball. In *Proceedings of the Tenth Indian Conference on Computer Vision, Graphics and Image Processing*, page 53. ACM, 2016.

[22] M. Tavassolipour, M. Karimian, and S. Kasaei. Event detection and summarization in soccer videos using bayesian network and copula. *IEEE Transactions on circuits and systems for video technology*, 24(2):291–304, 2014.

[23] X. Yu, H. W. Leong, J.-H. Lim, Q. Tian, and Z. Jiang. Team possession analysis for broadcast soccer video based on ball trajectory. In *Information, Communications and Signal Processing, 2003 and Fourth Pacific Rim Conference on Multimedia. Proceedings of the 2003 Joint Conference of the Fourth International Conference on*, volume 3, pages 1811–1815. IEEE, 2003.