

SoccER: Computer graphics meets sports analytics for soccer event recognition

*Original*

SoccER: Computer graphics meets sports analytics for soccer event recognition / Morra, L., Manigrasso, F., Lamberti, F.. - In: SOFTWAREX. - ISSN 2352-7110. - ELETTRONICO. - 12:(2020). [10.1016/j.softx.2020.100612]

*Availability:*

This version is available at: 11583/2847886 since: 2020-11-04T09:25:20Z

*Publisher:*

Elsevier

*Published*

DOI:10.1016/j.softx.2020.100612

*Terms of use:*

This article is made available under terms and conditions as specified in the corresponding bibliographic description in the repository

*Publisher copyright*

(Article begins on next page)

# SoccER: Computer graphics meets sports analytics for soccer event recognition

Lia Morra, Francesco Manigrasso, Fabrizio Lamberti

*Politecnico di Torino, Dipartimento di Automatica e Informatica, Torino, Italy*

---

## Abstract

Automatic event detection from images or wearable sensors is a fundamental step towards the development of advanced sport analytics and broadcasting software. However, the collection and annotation of large scale sport datasets is hindered by technical obstacles, cost of data acquisition and annotation, and commercial interests. In this paper, we present the Soccer Event Recognition (SoccER) data generator, which builds upon an existing, high quality open source game engine to enable synthetic data generation. The software generates detailed spatio-temporal data from simulated soccer games, along with fine-grained, automatically generated event ground truth. The SoccER software suite includes also a complete event detection system entirely developed and tested on a synthetic dataset including 500 minutes of game, and more than 1 million events. We close the paper by discussing avenues for future research in sports event recognition enabled by the use of synthetic data.

*Keywords:* sports analytics, computer graphics, event detection, soccer event recognition, game simulator

---

## Required Metadata

### Current code version

Ancillary data table required for subversion of the codebase. Kindly replace examples in right column with the correct information about your current code, and leave the left column as it is.

<b>Nr.</b>	<b>Code metadata description</b>	<b>Please fill in this column</b>
C1	Current code version	v1
C2	Permanent link to code/repository used for this code version	<a href="https://gitlab.com/grains2/slicing-and-dicing-soccer">https://gitlab.com/grains2/slicing-and-dicing-soccer</a>
C3	Permanent link to Reproducible Capsule	
C4	Legal Code License	MIT
C5	Code versioning system used	git
C6	Software code languages, tools, and services used	python, C++, Etalis
C7	Compilation requirements, operating environments & dependencies	python 3.6
C8	If available Link to developer documentation/manual	<a href="https://gitlab.com/grains2/slicing-and-dicing-soccer/-/blob/master/README.md">https://gitlab.com/grains2/slicing-and-dicing-soccer/-/blob/master/README.md</a>
C9	Support email for questions	lia.morra@polito.it

Table 1: Code metadata (mandatory)

## 1. Motivation and significance

Modern professional sports, and especially popular games such as soccer, basketball or football, are increasingly supported by the collection and analysis of massive data quantities about prospective and current players, team performance, fans and their interactions [1, 2, 3]. Major leagues are retrofitting their stadiums with extensive data acquisition capabilities, including wearable technologies and full-HD, multi-view, fixed-cameras arrays [1, 4]. Wearable sensors and trackers (from vests to GPS trackers embedded in the shoes) enable the acquisition of detailed data during training sessions and, sometimes, matches. A variety of affordable sensors have been proposed, which are increasingly used also by lower-level and amateur leagues. It is estimated that the newest and most technologically advanced stadiums can collect up to 50 terabytes of data per month [1]. Emerging technologies such as Virtual and Augmented Reality are able to leverage detailed information about players' positions and movements to recreate game interactions in order to enhance players' training [5, 6], as well as to improve the fruition of games by providing immersive fan experiences [1, 7].

Detecting relevant events in soccer is a fundamental component of many broadcasting, performance analysis, and immersive replay applications [2, 3, 8, 9, 10, 5]. It has been approached using a variety of techniques including machine learning methods, fuzzy logic and Hidden Markov Models [2]. However, scientific progress in this area is hindered by the lack of suitable public annotated datasets, since most of the data generated in sports is privately owned.

Presently, few soccer datasets are available and none are completely suited to the problem of fine-grained event detection. The SoccerNet dataset comprises a large number of broadcast soccer videos [11], but annotations are limited to events that could be parsed from match reports provided by leagues websites (Goal, Yellow/Red Card, and Substitution). Multi-view, fixed camera setups have the advantage of covering the whole field and thus provide a complete coverage of the game, but public datasets are small [4]. A few large scale data repositories provide detailed spatio-temporal information about the events that have occurred in soccer games [12], but do not provide the video sources or players coordinates for the entire games; regrettably, event detection algorithms need to be able to distinguish interesting events from the background and hence must be trained on the original data source.

Synthetic data generation is an affordable solution when the cost of acquisition or manual labeling is prohibitively high, and have already been used in a variety of computer vision tasks [13]. In this work, we propose the Soccer Event Recognition (SocceER) generator, a modified game engine that

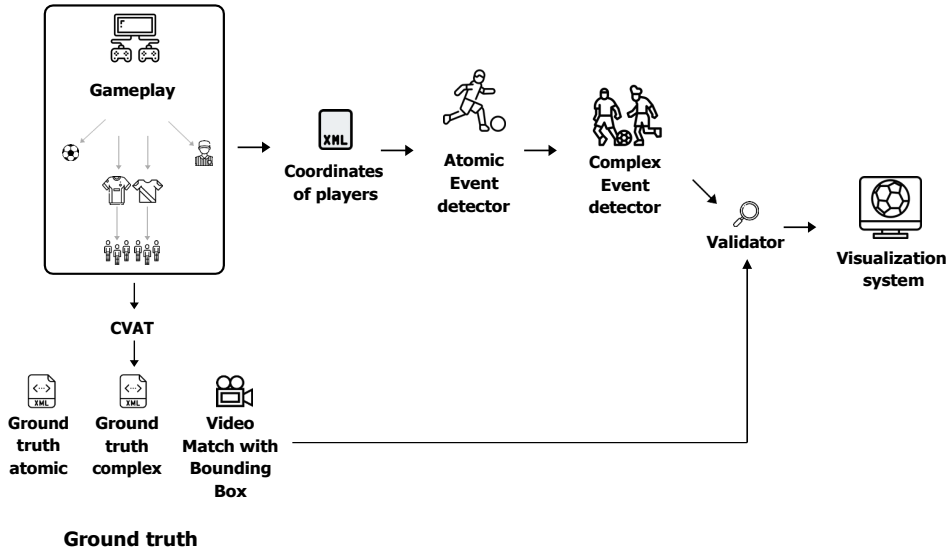


Figure 1: Overall organization of the SoccER repository. The SoccER data generator, based on the Gameplay engine, outputs spatio-temporal positions of all the players along with the ground truth annotation in CVAT format. From the analysis of the spatio-temporal positions a two-stage event detector identifies atomic (e.g., Kicking the Ball, Ball Possession) and complex events (e.g., Pass). Finally, performances is evaluated using a validator script and a visualization system.

41 can produce complete annotated spatio-temporal data from a soccer game,  
 42 and we show how it can be used to develop and evaluate an event detection  
 43 system [3].

## 44 2. Software description

45 The SoccER software repository comprises several distinct modules. The  
 46 core of the system is the *SoccER generator*, a modified version of the Game-  
 47 play Football engine [14]. The generator outputs complete annotated games  
 48 including the videos, spatio-temporal position of all the players, and ground  
 49 truth events. We also provide the software for the *SoccER event detection*  
 50 tool, which was described in detail in a previous publication [3], as an exam-  
 51 ple of event detection system developed using the SoccER data generator.  
 52 The SoccER detection algorithm is based on temporal logics and allows to  
 53 identify a wide range of events occurring during the game. Finally, the repos-  
 54 itory provides auxiliary Python scripts, including a validation script and a  
 55 visualization tool to assess the results obtained. An overview of the entire  
 56 process is reported in Fig. 1.

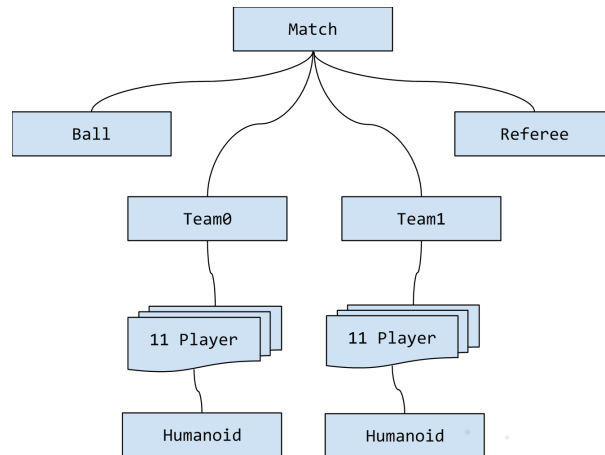


Figure 2: Main classes that implement the game engine in the Gameplay architecture: `Ball`, `Referee`, `Team` and `Player` classes. Each `Player` object is associated to a `Humanoid` object which handles its associated graphical assets and animations.

57 *2.1. SoccER data generator*

58 SoccER builds upon the Gameplay Football simulator, an open source  
 59 engine implementing a complete football game under standard rules, with 11  
 60 players on each team, including all the most common events such as goals,  
 61 fouls, corners, penalty kicks, etc. Among available open source simulators,  
 62 Gameplay Football was deemed the best option with respect to both graphi-  
 63 cal quality and game physics accuracy; being the engine open source, it can  
 64 also be inspected, improved and modified as needed. The environment con-  
 65 trols the opponent team by means of a rule-based bot, which was included  
 66 in the original Gameplay Football simulator [14, 15].

67 The C++ game engine was modified in order to define, extract and  
 68 save the ground truth needed to train and validate event detection sys-  
 69 tems. The main classes that implement the Gameplay engine are depicted  
 70 in Fig. 2. Each class exports a `Process()` method, which is called at each  
 71 frame to update the game state; the `Match::Process()` method calls the  
 72 `Process()` method to update the status of each player, team, ball or ref-  
 73 erree. In order to log the event, we modified when needed the `Process()`  
 74 method of each class, and implemented a new `Logger` class which collects  
 75 and summarizes information from graphics objects emulated in the video, as  
 76 well as commands issued by the player and/or bot. The `Logger` also exports  
 77 data in xml or textual format. The event types included in the ground truth,  
 78 divided in *atomic* and *complex* events, along with the information available  
 79 for each event, are reported in Table 2.

<b>Atomic event</b>	<b>Associated data</b>
<i>KickingTheBall</i>	FrameID, PlayerID, Position, TeamID
<i>BallPossession</i>	FrameID, PlayerID, Position, TeamID, PlayerID closest to the door
<i>Tackle</i>	FrameID, PlayerID, Position, OffensiveTeamID, VictimTeamID
<i>BallDeflection</i>	FrameID, PlayerID, Position, TeamID
<i>BallOut</i>	FrameID
<i>Goal</i>	FrameID, PlayerID, TeamID
<b>Complex event</b>	<b>Associated data</b>
<i>Pass</i>	Start FrameID, End FrameID, PlayerID, receiving PlayerID, TeamID
<i>PassThenGoal</i>	Start FrameID, End FrameID, PlayerID, receiving PlayerID, TeamID
<i>FilteringPass</i>	Start FrameID, End FrameID PlayerID, receiving PlayerID, TeamID
<i>FilterPassThenGoal</i>	Start FrameID, End FrameID, PlayerID, receiving PlayerID, TeamID
<i>Cross</i>	Start FrameID, End FrameID,PlayerID, receiving PlayerID, TeamID
<i>CrossThenGoal</i>	Start FrameID, End FrameID, PlayerID, receiving PlayerID, TeamID
<i>Tackle</i>	Start FrameID, End FrameID,PlayerID, position, Offensive PlayerID,Victim PlayerID
<i>Shot</i>	Start FrameID, End FrameID, PlayerID, TeamID
<i>ShotThenGoal</i>	Start FrameID, End FrameID, PlayerID, TeamID
<i>SavedShot</i>	Start FrameID, End FrameID, PlayerID, TeamID, receiving PlayerID, goalkeeper PlayerID, goalkeeper TeamID, shooting position
<i>Foul</i>	Start FrameID, End FrameID, Offensive PlayerID, Victim PlayerID

Table 2: Associated data extracted for each event in the ground truth.

80 *2.1.1. Atomic event annotation*

81 Atomic events are localized in space and time, hence they involve only  
82 one or two players in a short time window. The ground truth can then  
83 be easily generated by logging the events that trigger the execution of the  
84 specific animation.

85 Specifically, goal events were already detected by the Gameplay engine  
86 and hence they are exported accordingly.

87 Tackles are dueling events that occur when one player seeks to gain  
88 control of the ball from another player. They are recorded in the  
89 `Match::Process()` methods, where the offensive and the victim player  
90 are identified, along with the result of the action, in order to trigger the  
91 correct animations.

92 Ballpossessions are obtained by the game engine through the method  
93 `Player::HasUniquePossession()`, which returns true if the player is  
94 the only one in control or near the ball, which is not true for instance in the  
95 case of a Tackle. At each frame, all the player objects are queried in order  
96 to generate a corresponding `BallPossession` event.

97 Fouls and `BallOut` events are detected through the  
98 `Referee::Process()` method, which continuously monitors the  
99 game for fouls, and the position of the ball with respect to the field. `BallOut`  
100 events is detected through the `Referee::Process()` method, which  
101 continuously monitors the position of the ball with respect to the field.

102 The `BallDeflection` and `KickingTheBall` events are triggered when the  
103 events corresponding animations are activated. Specifically, `KickingTheBall`  
104 are generated for the `ShortPass`, `LongPass`, `HighPass` or `Shot` animations, i.e.  
105 all actions which require the player to kick the ball.

### 106 *2.1.2. Complex event annotation*

107 For complex events, which may involve multiple players over a longer time  
108 span, we designed and implemented a set of finite state machines (FSMs)  
109 which observe the actions of the player and/or the game bot.

110 The FSMs for several families of complex events, namely `Pass`, `Goal`, `Shot`  
111 and `Tackle`, are reported in Figs. 3 and 4, respectively.

112 Let us consider the family of passes and crosses (Fig. 3), which represent  
113 the majority of events in a soccer game. A pass is defined as the action of  
114 transferring possession of the ball between two players of the same team.  
115 The FSM recognizes that a pass is initiated when one the corresponding  
116 animations is activated; let us recall that, in this case, the `KickingTheBall`  
117 atomic event is also generated. When a new player gets in possession of  
118 the ball, the FSM checks whether the action is successful, i.e., the receiving  
119 player belongs to the same team, to generate the ground truth event. A  
120 `Cross` event differs from a `pass` event depending on the position of the player  
121 who initiates the `Pass` (in or out of the field side). A `FilteringPass` is a special  
122 case of `pass` where the receiving player is beyond, as opposed to the previous  
123 case, the defense line of the opposing team.

124 In the case of a fault pass, i.e., the receiving player belongs to the opposite  
 125 team, it is possible to generate the event and flag it as failed, or to return  
 126 to the Idle state without generating any event. The proposed output data  
 127 format supports both options, but we record failed events only in the case  
 128 of crosses, which are less frequent than normal passes. However, with minor  
 129 modifications it would be possible to record also fault pass events.

130 Whenever the receiver of a pass scores a goal, the PassThenGoal, Cross-  
 131 ThenGoal, or FilteringPassThenGoal events are generated, as detailed in  
 132 Fig. 3.

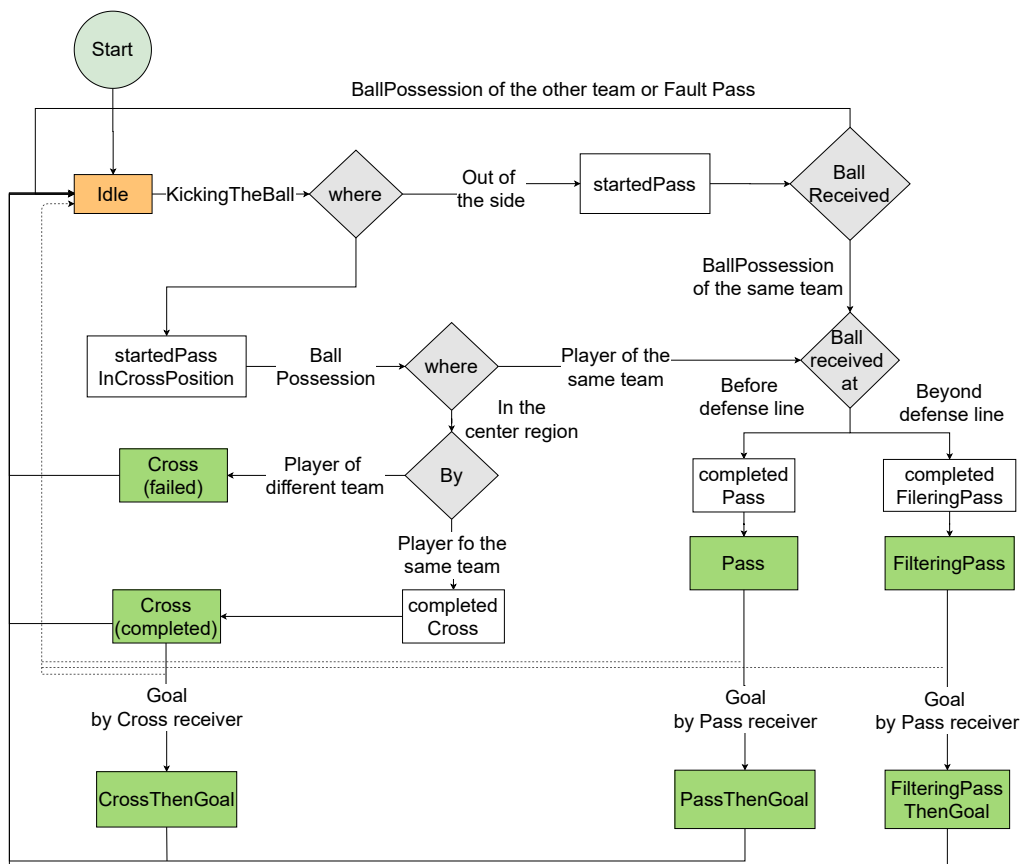


Figure 3: Finite-State Machine to detect the Pass and PassthenGoal event families. A Pass is initiated by a Kicking the Ball command and is successfully concluded if the receiving player is from the same team, otherwise the event is marked as failed. Depending on the position of the starting and receiving players, a pass may be further considered a Cross or a Filtering Pass. If a Goal is scored right after a pass, then a Passthengoal complex event is generated; any other events resets the state to idle .

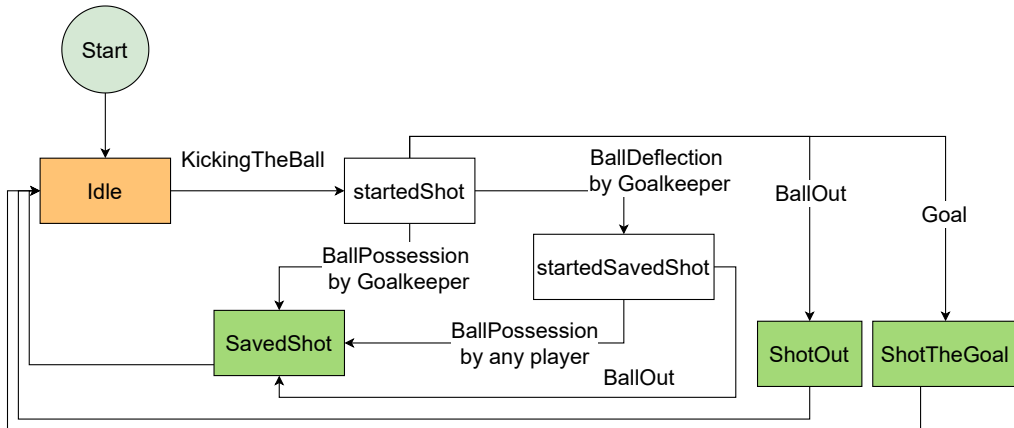


Figure 4: Finite State Machine to detect the Shot event family.

133 *2.1.3. Output format*

134 In addition to the ground truth, the SoccER generator exports the 23  
 135  $(x, y)$  coordinates pairs for each frame: 22 for the players for both the teams  
 136 and one for the ball. The output is in textual format, where each line com-  
 137 prises the following fields:

$$\langle frame \rangle \langle ID \rangle \langle x \rangle \langle y \rangle \tag{1}$$

138 In Eq. 1  $frame$  is the frame number,  $ID$  identifies a single player or a ball,  
 139 encoded in the range [116,127] for the first team, [129,140] for the opposing  
 140 team, and 128 for the ball whereas  $x$  and  $y$  are the coordinates with respect  
 141 to one of the field corners, in the range [0,110] and [0,72], respectively. We  
 142 adopt the same field coordinate system used in the Alheim dataset, which  
 143 includes the position of players obtained from wearable trackers [4].

144 Since the position along the  $z$  axis is not exported, a few events need to  
 145 be manually labeled. The most notable exception is the goal event, since the  
 146 ball height is needed to understand whether the ball crossed the goal line  
 147 below or above the crossbar. To allow the possibility to further expand the  
 148 ground truth through additional manual labeling, we export the annotations  
 149 in the format supported by the Computer Vision Annotation Tool (CVAT),  
 150 an open source tool for video tagging and labeling. An example of annotation  
 151 of atomic and complex events is reported in Figs. 5a and 5b, respectively.

152 The video is recorded with a resolution of 1920 x 1080 px (FullHD) and  
 153 framerate of 25 fps. The bounding boxes of the players with respect to the  
 154 video frame are also included in the ground truth. It is therefore in principle

155 possible to design an event detection system which can take as input spatio-  
 156 temporal data, video recordings, or both. A visualization system allows to  
 157 visualize and export the images with superimposed players' bounding boxes,  
 158 ground truth annotations and detected events; an example is shown in Fig. 6.

<pre> &lt;track id="168741" label="KickingTheBall"&gt;   &lt;box frame="281161" keyframe="1" occluded="0"   outside="0"     xbr="1.2415799999999993e+03"     xtl="1.2005299999999997e+03"     ybr="1.6874000000000009e+02"     ytl="9.1790000000000063e+01"&gt;     &lt;attribute name="playerId"&gt;47&lt;/attribute&gt;     &lt;attribute name="teamId"&gt;0&lt;/attribute&gt;     &lt;attribute name="x"&gt;54.9629&lt;/attribute&gt;     &lt;attribute name="y"&gt;35.7232&lt;/attribute&gt;   &lt;/box&gt; &lt;/track&gt; </pre>	<pre> &lt;track id="545" label="Cross"&gt;   &lt;box frame="212757" keyframe="1" occluded="0" outside="0"     xbr="1.2415799999999993e+03"     xtl="1.2005299999999997e+03"     ybr="1.6874000000000009e+02"     ytl="9.1790000000000063e+01"&gt;     &lt;attribute name="sender"&gt;66&lt;/attribute&gt;     &lt;attribute name="teamId"&gt;1&lt;/attribute&gt;     &lt;attribute name="receiver"&gt;41&lt;/attribute&gt;     &lt;attribute name="outcome"&gt;false&lt;/attribute&gt;   &lt;/box&gt; &lt;box&gt;   ... &lt;/box&gt; &lt;box frame="212835" keyframe="0" occluded="0" outside="0"     xbr="1.2415799999999993e+03"     xtl="1.2005299999999997e+03"     ybr="1.6874000000000009e+02"     ytl="9.1790000000000063e+01"&gt;     &lt;attribute name="sender"&gt;66&lt;/attribute&gt;     &lt;attribute name="teamId"&gt;1&lt;/attribute&gt;     &lt;attribute name="receiver"&gt;41&lt;/attribute&gt;     &lt;attribute name="outcome"&gt;false&lt;/attribute&gt;   &lt;/box&gt; &lt;/track&gt; </pre>
(a)	(b)

Figure 5: Examples of annotated atomic events (a) and complex events (b) in XML format (files AnnotationsAtomicEvents.xml and AnnotationsComplexEvents.xml, respectively). Each event has multiple attributes including ID, label (type of event), frame number, coordinates, player(s) and team ID. An event may include more than one player with different roles (i.e., sender, receiver), and an outcome indicating whether the action was successful or not. A complex event is recorded as a track in CVAT format since it covers multiple frames.

159 *2.2. Soccer event detector*

160 Building on previous works [10], we designed an event detection system  
 161 divided into two different modules: *atomic event detector* and *complex event*  
 162 *detector*. The former, starting from the positional data obtained from the  
 163 data generator, implements a set of rules to identify the atomic events. A  
 164 sliding window allows us to determine if the corresponding rules are satisfied  
 165 by the positional data in the specific interval.

166 The latter expresses complex events as a combination of atomic events  
 167 using *temporal* or *logical* operators. Temporal Interval Logic with Composi-  
 168 tional Operators (TILCO) were chosen as they support both qualitative  
 169 and quantitative temporal ordering, imposing constraints in terms of event



Figure 6: Example of scene generated by the Gameplay Football engine, with superimposed ground truth bounding boxes and IDs of each player and the ball. The ground truth and detected events are also overlaid on the bottom of the scene: in this frame, a shot attempt is correctly detected.

170 duration of events. A well-defined set of rules governs sports, and ITLs provide  
 171 expressive yet compact representations for events that occurred in the  
 172 match. Each event definition was double-checked against the official rules of  
 173 the Union of European Football Association (UEFA).

174 We define a total of 9 atomic events and 12 complex events. We report  
 175 here two examples, one per type, and refer the reader to our previous  
 176 publication [3] for a more complete analysis.

A **KickingTheBall** (Atomic) event consists of a simple kick aimed at executing a cross, pass or shot. The ball should move away from the player throughout the window  $k$ , with sudden acceleration and final increased speed.

$$\begin{aligned}
 &\langle ID, KickingTheBall, t, L = \langle \langle KickingPlayer, p_i \rangle, \langle KickedObject, b \rangle \rangle \\
 &player(p_i), ball(b), Distance(p_i, b, t) < T_{id_1} \\
 &\forall k = 1 \dots n, D(p_i, b, t + k) < D(p_i, b, t + k + 1), \\
 &speed(b, t + n) < T_{s_1} \exists k | acceleration(b, t + k) < T_{a_1}
 \end{aligned}$$

A **Pass** (Complex) event occur when the ball is passed between two players of the same team, and hence can be expressed as a sequence of two

atomic events.

$$\begin{aligned} &\langle ID, Pass, (t, t + k), t, L = \langle ID, KickingTheBall, \\ &\langle KickingPlayer, p_i, t \rangle, \langle KickedObject, b, t \rangle \rangle \\ &THEN \langle ID, BallPossession, \langle PossessingPlayer, p_j, t + k \rangle, \\ &\langle PossessedObject, b, t \rangle \rangle \\ &player(p_i), player(p_j), ball(b), team(p_i) = team(p_j), k < Th3 \end{aligned}$$

### 177 2.2.1. Atomic event detector: implementation details

178 The atomic event detector is implemented by the  
179 EventDetectorAtomic Python module. First, the module calcu-  
180 lates a set of features from the  $x$  and  $y$  positions, such as *velocity*,  
181 *acceleration*, *direction* with respect to the field, *distance from the target*  
182 *line* of both teams, etc. A detailed definition of the features is provided in  
183 previous works [3, 8]. For each time window the rules for atomic events,  
184 described in Section 2.2, are applied. A configuration file defines the  
185 parameters (i.e. thresholds) to be applied for each rule.

186 An evolutionary strategy based on a multi-objective genetic algorithm was  
187 followed to optimize the parameters of the atomic event detection system [16,  
188 3]. To this aim, the Optimizer script leverages the DEAP toolbox for  
189 the genetic algorithm implementation [17] and the Validator script (see  
190 Section 2.2.3) to evaluate each genome, which corresponds to a given detector  
191 configuration.

192 The detected events, both atomic and complex, are saved in the same  
193 XML format as the ground truth.

### 194 2.2.2. Complex event detector: implementation details

195 The complex event detector is implemented by the  
196 EventDetectorComplex Python module. It takes the output, in  
197 XML format, of the atomic event detection, and produces a list of detected  
198 events; performance statistics can be optionally calculated.

199 The proposed implementation is based on the Etalis library [18, 19].  
200 Etalis is a Prolog extension which implements TILCO, and is executed  
201 through the swipl Prolog engine by the Python script, as detailed in Fig. 7.

202 For instance, in Etalis, a complex event is defined by a *pattern*, i.e. a  
203 sequence of events concatenated by logical or temporal operators, and a list  
204 of conditional clauses, with the following syntax:

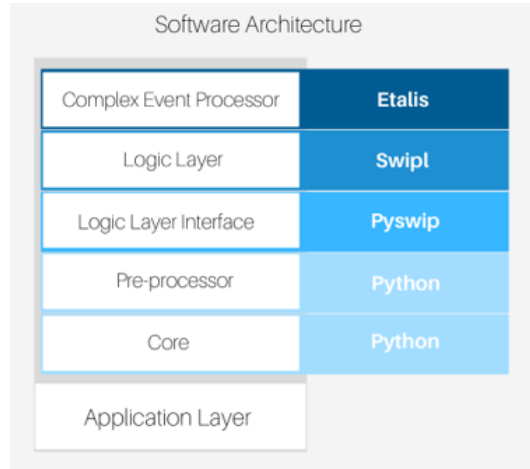


Figure 7: Software architecture of the Complex Event Processor.

```

205 ComplexEvent ←-
206 Event_1 < OP > Event_2 [ < OP > Event_N ]
207 WHERE ( List of conditions )

```

208 In the Etalis language, the Pass event, defined in Section 2.2, is imple-  
 209 mented as follows:

```

210 pass (Kf, Bf, KpId, KtId, BpId) ←-
211 kickingTheBall (Kf, KpId, KtId, Kx, Ky)
212 SEQ
213 ballPossession (Bf, BpId, BtId, Bx, By, BootdpId, BootdpId,
214 Bootdpx, Bootdpy)
215 WHERE
216 (KtId=BtId, Bf>Kf, Bf-Kf <150).

```

217 The `seq` operator is true whenever an instance of `kickingTheBall` is  
 218 followed by an instance of `ballPossession`. The input parameters include  
 219 the  $x$  and  $y$  positions of the event ( $*x, *y$ ), the frame number or time when  
 220 the event occurs ( $*f$ ), the players ID ( $*pId$ ) and team ID ( $*tId$ ).

### 221 2.2.3. Validation script

222 The `Validator` script, implemented in Python, calculates the recall,  
 223 precision and F-score for each event. A ground truth atomic event is detected  
 224 if an event of the same type is found within a pre-defined temporal window  
 225 (e.g., three frames). For complex events, we use the common OV20 criterion  
 226 for temporal action recognition: a temporal window matches a ground truth

227 action if they overlap, according to the Intersection over Union, by at least  
228 a predefined percentage, normally set to 20% [20].

229 The script takes as input a directory which contains the ground truth  
230 and detected events, in the XML-based CVAT format, and outputs an XML  
231 file with the calculated performance. Through command line options and  
232 configuration files, it is possible to specify whether the performance should  
233 be calculated for atomic or complex events, which events should be included  
234 in the evaluation, and the matching parameters.

### 235 **3. Illustrative example: the SoccER dataset**

236 The SoccER dataset comprises 8 complete games synthesized by the  
237 SoccER generator through various modalities (player vs. player, player  
238 vs. AI, AI vs. AI). It comprises a total of 500 minutes of play with  
239 1,678,304 atomic events and 9,130 complex events. The dataset is split  
240 in a training and validation dataset which were used to train and evalu-  
241 ate the SoccER event detector [3]. The dataset is available for download at  
242 <https://gitlab.com/grains2/slicing-and-dicing-soccer>.

243 An important aspect to be considered, when dealing with synthetic data,  
244 is how close the distribution of the data generated by the game engine  
245 matches that of real-life data. For many aspects, this relation could not be  
246 investigated in detail, at least at present, due to the lack of publicly available  
247 data. Nonetheless, an initial analysis yields encouraging results. Based on  
248 comparison with the Alfheim dataset [4], we expect the average player speed  
249 to be higher, although the size of the real dataset is small to draw definitive  
250 conclusions and anyway aspect could be further customized in future SoccER  
251 releases.

252 In terms of the quantity and type of events, data recorded during the  
253 season 2017/2018 of five national soccer competitions reports an average  
254 of 1,682 events per 90-minute game [12]. Passes (including crosses) were  
255 the most frequent event (50%), followed by duels or tackles (28%); shots  
256 (1.5%) and goals ( $< 1\%$ ) were relatively rare events. In our sample dataset  
257 (500 minutes), roughly 1,141 complex events/hour were generated, with the  
258 following distribution: passes and crosses (62%), tackles (29%), shots (7.5%)  
259 and goals ( $< 1\%$ ). The difference in the percentage of shot events partially  
260 stems from the slightly different sub-event types and definitions we adopted.

### 261 **4. Impact**

262 In this section, the potential impact of synthetic data generation is dis-  
263 cussed from the perspective of two research fields: sport analytics and com-  
264 puter vision.

265 *4.1. Impact on sport analytics*

266 The final objective of this work is to use the SoccER synthetic data  
267 generator to improve the recognition of events in soccer games.

268 The SoccER generator outputs both the video, with annotated bounding  
269 boxes, and the spatio-temporal coordinates with respect to the field. So far,  
270 we have validated the detection of events directly from the spatio-temporal  
271 coordinates. In a real-life setting, such these coordinates can be obtained in  
272 a variety of ways, including wearable trackers [8, 21] or wide-camera or multi-  
273 camera setup [4, 12]. To generate spatio-temporal coordinates, aside from  
274 manual annotations, a completely vision-based approach can be implemented  
275 using an object detector and then mapping pixel coordinates to the field  
276 coordinate systems using a properly calibrated setup [7, 3]. Thus, the SoccER  
277 generator can be considered representative of many use cases of interest for  
278 sport analytics.

279 We have demonstrated the possibilities of this approach by developing a  
280 complete event detection system based on ITLs, which is briefly introduced in  
281 Section 2.2 and described in detail in a previous work [3]. At the state of the  
282 art, ITLs have shown promising results in the detection of soccer events [3, 10,  
283 22]. They also allow us to reason about detected events, answering questions  
284 such as "which passes resulted in a goal being scored?".

285 The SoccER event detector successfully detects most complex events,  
286 such as passes, with a F-score greater than 0.8 [3]. Overall, the results  
287 are comparable or better than previously published approaches [10, 22, 8];  
288 the interested reader is referred to our previous publication for an in-depth  
289 analysis [3].

290 Nonetheless, it must be acknowledged that comparing different algorithms  
291 is problematic for two aspects. The first is the lack of a common reference  
292 dataset, which always sparks the question: is the algorithm better or the  
293 dataset easier? The second is that many papers are reporting results on  
294 a limited number of event types, usually passes, ball kicks or shots [10, 8,  
295 22]. Thanks to the SoccER generator, we were able to detect, and more  
296 importantly measure the performance for, a much wider range of events,  
297 highlighting limitations which did not emerge in previous works. The use of  
298 synthetic datasets, though not without limitations, may alleviate both issues  
299 by providing a common and challenging ground for comparison.

300 In particular, our performance highlights the limitations of logic-based  
301 detectors in the case of events, such as tackles, whose recognition strongly  
302 depends on the pose of the players and other visual characteristics. Such  
303 events, which account for 28% of the events occurring in a real game [12],  
304 are not easily detectable with an ITL-based approach, working with syn-

305 thetic data. Future research is needed to evaluate the performance of al-  
306 ternative techniques, e.g. convolutional and recurrent neural networks for  
307 event detection in untrimmed videos streams. Of particular interests will  
308 be hybrid systems that combine low-level pattern detection with high-level  
309 reasoning capabilities. The proposed system allows to generate sufficiently  
310 large datasets to train such systems.

#### 311 4.2. Impact on computer vision research

312 Synthetic data generation is a cost-effective solution when the cost of ac-  
313 quisition or manual labeling is prohibitively high [13]. Hence, it is extensively  
314 used in machine learning and computer vision research to enable the fast, ac-  
315 curate and relatively inexpensive generation of data, along with its ground  
316 truth. For instance, it has been used to train semantic segmentation mod-  
317 els [23, 24, 25], in the field of autonomous driving [23, 26] and for complex  
318 tasks such as in Visual Question Answering [27]. Synthetic data generation  
319 is not only cost-effective, but allows greater control over the generated dis-  
320 tributions, reducing biases, data imbalance and boosting the availability of  
321 rare and infrequent cases [13, 27].

322 From a research point of view, sport event detection offers many chal-  
323 lenges as it combines both low low-level visual pattern detection with high-  
324 level reasoning capabilities. Complex event detection in untrimmed video  
325 sequences is *per se* a challenging problem at the state of the art due to data  
326 imbalance, rare events, and the difficulties in precisely defining temporal  
327 boundaries. To the best of our knowledge, few synthetic datasets exist tack-  
328 ling event detection in untrimmed video sequences. At the same time, event  
329 detection in sports is facilitated by the relatively limited number of classes  
330 and the existence of a known set of rules. It is therefore an excellent "gym"  
331 on which computer vision models can be trained.

332 Games and other types of simulators are also used in reinforcement learn-  
333 ing (RL). In a recent work, the Gameplay Football engine was re-purposed as  
334 a training gym for RL-trained agents, the Google Research Football (GRF)  
335 environment [15]. The GRF and Soccer software suites were conceived with  
336 different, complementary purposes. The GRF does not include or support  
337 the logging system that is needed for training and testing event detection  
338 systems, which goes well beyond recording the input commands and position  
339 of the players. The GRF library includes a more optimized graphical engine,  
340 which is essential for the training of RL algorithms that need to operate  
341 at super-human speed in order to minimize the training time. The genera-  
342 tion of data for event detection benefits from including human players in the  
343 recording session. We found the resulting datasets more varied than those  
344 generated solely by the game AI, and thus does not benefit as much from

345 an extremely optimized graphical engine. Future work will explore a tighter  
346 integration of these two projects.

347 Finally, it should be noticed that the SoccER generator and dataset also  
348 contribute to open research questions related to how intelligent agents can  
349 sense the game state. Examples of such questions are whether it is more  
350 effective to provide machine learning agents with the raw pixel input, or  
351 to extract a compact encoding summarizing aspects such as players coordi-  
352 nates, ball possession, etc., and finally how such a compact encoding can be  
353 effectively engineered [15].

### 354 *4.3. Limitations and future improvements*

355 The main limitations of the SoccER data generator are the relatively  
356 low photo-realism, compared to commercial solutions, and the domain shift  
357 between synthetic and real data. The former issue may be a limitation when  
358 training and evaluating systems that operate directly on the video frame.  
359 Additionally, the type, frequency, and characteristics of events generated by  
360 the game engine have similar, but not equivalent, distributions than those  
361 collected during real competitions [12]. A more in-depth analysis of the  
362 differences and their impact on domain shift should be performed in the  
363 future.

364 The domain shift is a common problem to all synthetic datasets and is  
365 being addressed by extensive research in the field of domain adaptation [23];  
366 however, current research is mostly focused on images (as input), convolu-  
367 tional neural networks (as models/detector) and may not apply directly to  
368 spatio-temporal data. Developing domain adaptation techniques for such  
369 data is an interesting research question.

370 From an application point of view, the proposed SoccER event detector  
371 offers limited performance in the case of tackles, ball deflection and other  
372 events for which visual features are of paramount importance. Exploring the  
373 use of deep learning, in combination or substitution of the current system, will  
374 be particularly interesting. In order to support the training of data hungry  
375 models, a possible extension will consist in modifying the game engine to  
376 generate data on the fly.

## 377 **5. Conclusions**

378 The availability of large scale datasets has become an hallmark of modern  
379 computer vision and data science. Synthetic data generation can bridge this  
380 gap when data acquisition is unfeasible or too expensive. In this paper, we  
381 have presented the SoccER software suite, which comprises both a synthetic  
382 spatio-temporal data generator and an event detection system targeting the

383 soccer domain. While the SoccER detector achieves good performance on  
384 most events, further research is needed to achieve optimal performance on  
385 all classes of events. The SoccER suite aims at enabling further research  
386 in this area by providing a common ground on which algorithms can be  
387 developed and compared.

## 388 **6. Conflict of Interest**

389 We wish to confirm that there are no known conflicts of interest associated  
390 with this publication and there has been no significant financial support for  
391 this work that could have influenced its outcome.

## 392 **Acknowledgements**

393 We thank Claudio Gianfrate, Enrico Guarino and Giuseppe Canto for the  
394 implementation of the SoccER data generator and SoccER event detection  
395 software.

## 396 **References**

- 397 [1] T. Hayduk, The future of sport data analytics, in: *Statistical Modelling*  
398 *and Sports Business Analytics*, Francis & Taylor, 2020.
- 399 [2] H.-C. Shih, A survey of content-aware video analysis for sports, *IEEE*  
400 *Transactions on Circuits and Systems for Video Technology* 28 (5) (2017)  
401 1212–1231.
- 402 [3] L. Morra, F. Manigrasso, G. Canto, C. Gianfrate, E. Guarino, F. Lam-  
403 bertti, Slicing and dicing soccer: Automatic detection of complex events  
404 from spatio-temporal data, in: A. Campilho, F. Karray, Z. Wang (Eds.),  
405 *Image Analysis and Recognition*, Springer International Publishing,  
406 Cham, 2020, pp. 107–121.
- 407 [4] S. A. Pettersen, D. Johansen, H. Johansen, V. Berg-Johansen, V. R.  
408 Gaddam, A. Mortensen, R. Langseth, C. Griwodz, H. K. Stensland,  
409 P. Halvorsen, Soccer video and player position dataset, in: *Proceedings*  
410 *of the 5th ACM Multimedia Systems Conference, MMSys '14*, Associa-  
411 *tion for Computing Machinery*, New York, NY, USA, 2014, p. 18–23.
- 412 [5] A. Cannavó, D. Calandra, G. Basilicó, F. Lamberti, Automatic recog-  
413 *nition of sport events from spatio-temporal data: An application for*  
414 *virtual reality-based training in basketball*, in: *14th International Con-*  
415 *ference on Computer Graphics Theory and Applications (GRAPP 2019)*,  
416 *SCITEPRESS*, 2019, pp. 310–316.

- 417 [6] A. Cannavò, F. G. Praticò, G. Ministeri, F. Lamberti, A movement  
418 analysis system based on immersive virtual reality and wearable tech-  
419 nology for sport training, in: Proceedings of the 4th International Con-  
420 ference on Virtual Reality, 2018, pp. 26–31.
- 421 [7] K. Rematas, I. Kemelmacher-Shlizerman, B. Curless, S. Seitz, Soccer  
422 on your tabletop, in: Proceedings of the IEEE Conference on Computer  
423 Vision and Pattern Recognition, 2018, pp. 4738–4747.
- 424 [8] K. Richly, M. Bothe, T. Rohloff, C. Schwarz, Recognizing compound  
425 events in spatio-temporal football data, in: International Conference on  
426 Internet of Things and Big Data, Vol. 2, SCITEPRESS, 2016, pp. 27–35.
- 427 [9] J. Lee, D. Nam, S. Moon, J. Lee, W. Yoo, Soccer event recognition  
428 technique based on pattern matching, in: 2017 Federated Conference  
429 on Computer Science and Information Systems (FedCSIS), 2017, pp.  
430 643–646.
- 431 [10] A. Khan, B. Lazzerini, G. Calabrese, L. Serafini, Soccer event detec-  
432 tion, in: 4th International Conference on Image Processing and Pattern  
433 Recognition (IPPR 2018), AIRCC Publishing Corporation, 2018, pp.  
434 119–129.
- 435 [11] S. Giancola, M. Amine, T. Dghaily, B. Ghanem, Soccernet: A scalable  
436 dataset for action spotting in soccer videos, in: Proceedings of the IEEE  
437 Conference on Computer Vision and Pattern Recognition Workshops,  
438 2018, pp. 1711–1721.
- 439 [12] L. Pappalardo, P. Cintia, A. Rossi, E. Massucco, P. Ferragina, D. Pe-  
440 dreschi, F. Giannotti, A public data set of spatio-temporal match events  
441 in soccer competitions, *Scientific data* 6 (1) (2019) 1–15.
- 442 [13] S. I. Nikolenko, Synthetic data for deep learning, arXiv preprint  
443 arXiv:1909.11512 (2019).
- 444 [14] B. K. Schuiling, Gameplay football,  
445 <https://github.com/BazkieBumpercar/GameplayFootball>.
- 446 [15] K. Kurach, A. Raichuk, P. Stanczyk, M. Zajac, O. Bachem, L. Espeholt,  
447 C. Riquelme, D. Vincent, M. Michalski, O. Bousquet, S. Gelly, Google  
448 research football: A novel reinforcement learning environment, *CoRR*  
449 (2019). arXiv:1907.11180.

- 450 [16] L. Morra, N. Coccia, T. Cerquitelli, Optimization of computer aided  
451 detection systems: An evolutionary approach, *Expert Systems With*  
452 *Applications* 100 (2018) 145–156.
- 453 [17] F.-A. Fortin, F.-M. De Rainville, M.-A. Gardner, M. Parizeau, C. Gagné,  
454 DEAP: Evolutionary algorithms made easy, *Journal of Machine Learning*  
455 *Research* 13 (2012) 2171–2175.
- 456 [18] D. Anicic, P. Fodor, R. Stühmer, S. Rudolph, Etalis home, [http://](http://code.google.com/p/etalis)  
457 [code.google.com/p/etalis](http://code.google.com/p/etalis).
- 458 [19] G. Canto, Sistema di riconoscimento di eventi sportivi basati su logiche  
459 temporali, Master’s thesis, Politecnico di Torino, Italy (2019).
- 460 [20] A. Gaidon, Z. Harchaoui, C. Schmid, Actom sequence models for effi-  
461 cient action detection, in: *CVPR 2011, IEEE*, 2011, pp. 3201–3208.
- 462 [21] K. Richly, F. Moritz, C. Schwarz, Utilizing artificial neural net-  
463 works to detect compound events in spatio-temporal soccer data, in:  
464 *3rd SIGKDD Workshop on Mining and Learning from Time Series,*  
465 *MiLeTS’17*, 2017.
- 466 [22] M. Stein, D. Seebacher, T. Karge, T. Polk, M. Grossniklaus, D. A.  
467 Keim, From movement to events: Improving soccer match annotations,  
468 in: *International Conference on Multimedia Modeling*, Springer, 2019,  
469 pp. 130–142.
- 470 [23] E. Alberti, A. Tavera, C. Masone, B. Caputo, Idda: a large-  
471 scale multi-domain dataset for autonomous driving, *arXiv preprint*  
472 *arXiv:2004.08298* (2020).
- 473 [24] G. Amato, L. Ciampi, F. Falchi, C. Gennaro, N. Messina, Learning  
474 pedestrian detection from virtual worlds, in: *International Conference*  
475 *on Image Analysis and Processing*, Springer, 2019, pp. 302–312.
- 476 [25] Y.-T. Hu, H.-S. Chen, K. Hui, J.-B. Huang, A. G. Schwing, Sail-vos:  
477 Semantic amodal instance level video object segmentation-a synthetic  
478 dataset and baselines, in: *Proceedings of the IEEE Conference on Com-*  
479 *puter Vision and Pattern Recognition*, 2019, pp. 3105–3115.
- 480 [26] A. Dosovitskiy, G. Ros, F. Codevilla, A. Lopez, V. Koltun, Carla: An  
481 open urban driving simulator, *arXiv preprint arXiv:1711.03938* (2017).

482 [27] Q. Wu, D. Teney, P. Wang, C. Shen, A. Dick, A. van den Hengel, Visual  
483 question answering: A survey of methods and datasets, *Computer Vision*  
484 and *Image Understanding* 163 (2017) 21–40.

485 **Current executable software version**

486 An executable version of the software is not currently available.