POLITECNICO DI TORINO

Corso di Laurea Magistrale in Ingegneria Informatica

Tesi di Laurea Magistrale

Generazione di dati sintetici tramite computer graphics per il riconoscimento di eventi sportivi

Relatori

Prof. Fabrizio Lamberti
Dr. Lia Morra

Candidato

Enrico Guarino

Sessione di laurea di Ottobre 2019

Sommario

Capitolo 1

Introduzione

In ambito sportivo, la tecnologia sta svolgendo un ruolo sempre più determinante, grazie alle possibilità che oggi offre.

Sempre più addetti ai lavori, tra cui scout di talenti, allenatori e giocatori, fanno uso dei risultati che sono derivati dall'analisi delle performance sportive per migliorare le proprie prestazioni e/o comprendere meglio il gioco. Questo è tanto vero anche nell'ambito televisivo, dove il pubblico è interessato ad analisi sempre più precise.

Uno degli sport che possiede un seguito molto ampio in questo ambito, è il calcio. I lavori di ricerca che hanno come oggetto questo sport, infatti, sono sempre di più e possono far uso di una grande mole di dati che è prodotta ogni giorno dalle miriadi di telecamere, dai sistemi di tracciamento (GPS o indossati dai giocatori) che sono installati nei campi di gioco.

Molte aziende sono specializzate nel fornire quasi in tempo reale, le statistiche, la lista di eventi e di azioni che si sono verificate durante un incontro, facendo uso di software semi automatici che a partire dai dati riesce a determinare, con poche correzioni manuali, tutte queste informazioni.

Non di rado però i dati non sono reperibili, o semplicemente non sono sufficienti. Molte soluzioni di machine learning, ad esempio, attualmente necessitano di una grande mole di dati di alta qualità. Il più delle volte il lavoro di annotazione che deve essere svolto manualmente richiede un enorme sforzo umano e vi è comunque un rischio elevato di errori.

Alcuni lavori di ricerca hanno allora cercato di colmare l'assenza di informazioni di questa natura producendoli artificialmente, facendo uso della computer graphics. Con questo approccio è virtualmente possibile generare il quantitativo di dati necessari usando il grado di qualità ricercato, minimizzando al contempo lo sforzo umano richiesto nel processo. Ne sono un esempio le immagini sintetiche fotorealistiche di oggetti, ricostruiti in scene arbitrarie usando motori di gioco, o le sequenze video sintetiche utilizzate per l'addestramento di sistemi a guida autonoma.

In questo contesto, la ricerca scientifica ha fatto molti progressi, ma la scarsa disponibilità di dati posizionali pubblici relativi ai giocatori e alla palla, cerca di trovare

una possibile soluzione in questo lavoro di tesi. L'obiettivo complessivo, portato avanti in questo ed altri due lavori di tesi, è quello di sviluppare un sistema di riconoscimento automatico di eventi nel calcio. Pertanto, il lavoro, svolto in un gruppo di tre persone, è stato organizzato nel seguente modo: un lavoro di tesi ha analizzato e realizzato un riconoscitore di eventi di tipo atomico, un secondo lavoro ha realizzato invece il sistema di riconoscimento degli eventi complessi, e infine questo lavoro ha realizzato il sistema per ottenere e visualizzare i dati facendo uso delle tecniche di computer vision.

In particolare, lo scopo di questa tesi è quello di fornire un dataset di dimensione abbastanza estesa di informazioni circa le posizioni, gli eventi e i video di partite di calcio, da poter usare nel campo dello sviluppo di sistemi esperti e/o di machine learning.

Per portare a termine l'obiettivo di questa tesi, si è fatto uso di un gioco *open-source*, debitamente modificato, per produrre durante una partita tutti i dati necessari a un sistema come quello che viene presentato più avanti. I dati prodotti cercano di essere quanto più ampi e quanto più verosimili possibile.

Per comprendere meglio la validità del dataset prodotto, si è fatto uso di una serie di metriche di confronto, da relazionare ai dati di una partita vera. Pertanto, una parte della tesi prende in esame queste metriche e propone una analisi qualitativa del dataset generato.

Capitolo 2

Stato dell'arte

Per affrontare al meglio l'argomento di questa tesi, si è proceduto con un lavoro di ricerca che ha toccato alcuni temi differenti, ma tutti inerenti alle metodologie di analisi, visualizzazione e raccolti dei dati in ambito sportivo, con particolare riferimento al calcio.

2.1 Analisi su sistemi e software commerciali per il riconoscimento di eventi nello sport

Riguardo all'ambito di ricerca sul riconoscimento tramite video di eventi nello sport, e in particolare nel calcio, c'è relativamente poco materiale e la ricerca è molto orientata a un sottoinsieme di casi.

Molte aziende sono invece specializzate nel fornire i dati delle partite alle emittenti TV. Questi possono essere sia i dati grezzi estrapolati (ovvero brutalmente la lista di tutti gli eventi che sono verificati in quella partita), oppure i dati risultanti da delle analisi più articolate che mostrano risultati aggregati e/o statistici.

È pertanto mio interesse stabilire e determinare quali sono le metodologie, e quindi qual è lo stato dell'arte, che sono definite dai processi e dai software che sono in dotazione alle grandi aziende che si occupano da anni di queste attività in questo settore. Si vuole quindi analizzare il grado di sviluppo dei prodotti commerciali nell'ambito del riconoscimento automatico di eventi nel calcio, e fornire una visione generale su ciò che viene offerto oggi al cliente.

2.1.1 Opta

Opta [9] è una delle aziende leader nel settore per quanto riguarda la fornitura di dati associati alle partite di calcio. In particolare, fornisce dati da oltre 5 anni alla "Premiere League", e i loro dati sono mostrati e usati dalle maggiori emittenti televisive inglesi, quali BBC, Sky Sport ecc.

“Siamo l’unica attività di sport data che colleziona e distribuisce dati live contestuali completi, con timestamp, con l’aggiunta di coordinate xy (anche z dove possibile, come ad esempio i tiri nel calcio), e una granularità del tipo di evento unica tra i fornitori di dati”, si legge nel loro sito web.

Per raggiungere tale risultato, Opta si serve di un team di analisti addestrati ad usare i tool e le metodologie che permettono loro di garantire qualità e consistenza dei dati, e di fornirli in modo rapido e sicuro ai clienti mediante molte possibili soluzioni.

Il processo comincia con la collezione stessa dei dati, affidata ai data analyst negli hub in giro per il mondo e supportata da quelli presenti nello stadio. Viene usato per l’occorrenza un software per registrare l’azione al più alto grado di dettaglio, in diretta, per una serie di sport. Queste informazioni sono dunque salvate in un database e poi distribuite ai vari clienti nel mondo in base alle opzioni scelte dagli stessi mediante “feeds”, “widgets” ecc.”[1]

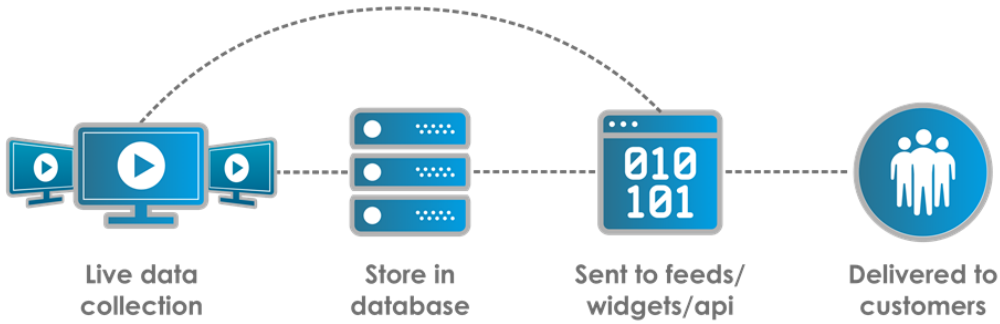


Figura 2.1 - Flusso dei dati, dal momento della raccolta al loro invio.

Vengono forniti un alto numero di analisi avanzate, che includono modelli predittivi, metriche analitiche e dati aumentati da diverse fonti.

In sostanza il processo di annotazione degli eventi si svolge in questo modo, secondo quanto riferito in “stuff.tv”[2]: “per ogni partita ci sono tre uomini che usano un software che sovrappone al live video la grafica di un campo di gioco. Uno si occupa di osservare la squadra in casa, l’altro guarda la squadra in trasferta e infine il terzo essenzialmente fa un controllo sui dati. Utilizzando una combinazione di tasti e click del mouse tracciano chi è sulla palla e cosa succede su di essa. Così per esempio se Wayne Rooney la prende sopra la linea di metà campo e la passa largo ad Antonio Valencia, loro cliccheranno dove Rooney è sullo schermo quando la gioca, e di nuovo dove Valencia la riceve e il software registrerà delle coordinate xy e il timestamp del passaggio. Ogni partita produce tra i 1600

[1] "The Opta difference - Opta Sports." https://www.optasports.com/about/the-opta-difference/. Ultimo accesso: 16 ott. 2018.

[2] "The inner beauty of the beautiful game: How Opta analyses ... - Stuff." 1 feb. 2014, https://www.stuff.tv/features/inner-beauty-beautiful-game-how-opta-analyses-football. Ultimo accesso: 16 ott. 2018.

e i 2000 pezzi di dati individuali che sono analizzati dal team in Opta e impacchettati per l'uso alle tv, alla stampa ecc."

Inoltre, si aggiunge come: "Tutti nei nostri datacenter sono addestrati ad avere la stessa definizione di contrasto e di duello [...]. La coerenza è molto importante".

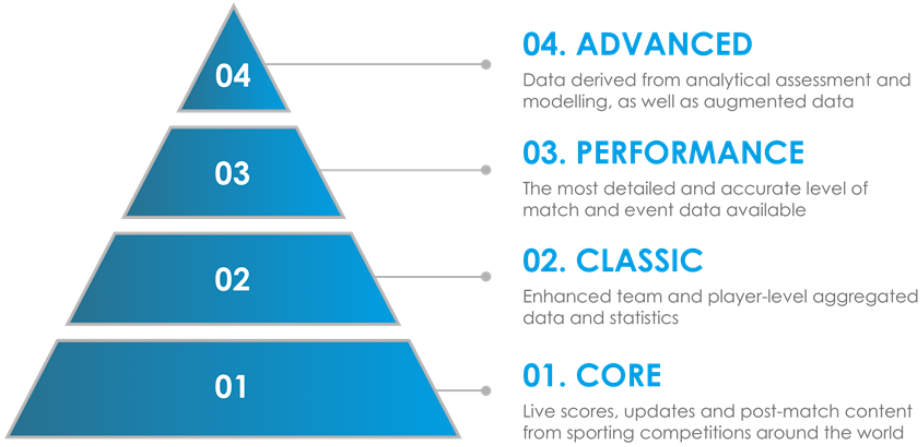


Figura 2.2 - Descrizione delle possibili tipologie di servizio fornito al cliente.

Allo stato dei fatti del 2014 (quando è stata rilasciata l'intervista a "Stuff") si evince che Opta svolge l'annotazione degli eventi manualmente. Gli eventi sono infatti riconosciuti e individuati da tre persone che grazie a un software ad-hoc, evidenziano il momento e la posizione in cui è avvenuto. Pertanto, il software, che sovrappone il video in diretta con le posizioni dei giocatori su un campo fittizio, più che altro serve ad annotare, piuttosto che a riconoscere. Il processo è dunque guidato dalla mano dell'uomo, in particolare di tre addetti per ogni singola partita. Buona parte dei servizi forniti sono di analisi in post-processing, ovvero aggregazione dei dati per trovare statistiche di più alto livello.

Non ci sono invece informazioni al riguardo del metodo di tracking dei giocatori. Si fa riferimento solo alla posizione che viene segnata a partire dalla sovrapposizione dell'immagine in diretta, quando si sta annotando un evento.

Tutte queste informazioni, riferite al 2014, con molta probabilità non sono più valide, sebbene ciò non sia verificabile in quanto al 2018 non sono stati resi pubblici ulteriori dettagli sul loro processo di annotazione. D'altronde, considerato il grado di sviluppo tecnologico in questi ultimi anni e in questo ambito, suggerisce che lo stato delle cose è molto diverso e gran parte del lavoro di annotazione è ormai svolto in modo automatico, con la sola supervisione dell'uomo.

2.1.2 Stats

Stats [10] è una compagnia che si occupa dell'analisi e fornitura di dati inerenti agli sport. Questa azienda è in questo settore da oltre 35 anni, ed è partner di numerose leghe e squadre a cui fornisce dati di tracking sui giocatori e analisi sui video.

In particolare, il sistema di tracking usato da Stats è denominato Stats SportVU. Quest'ultimo è composto da una serie di telecamere installate indipendentemente sul campo, le quali "forniscono statistiche sulle performance mediante l'estrazione e il processamento delle coordinate dei giocatori e della palla con un software sofisticato e con algoritmi statistici"[3].

Figura 2.3 - Esempio di setup delle telecamere in un campo di gioco.

Il sistema di telecamere ha due opzioni: quella standard, che usa un set di tre telecamere in alta definizione in una singola posizione; il setup da 6 telecamere aggiunge un secondo set di telecamere con il vantaggio di fornire dati avanzati sul posizionamento 3D. I dati vengono collezionati a una velocità di 25 volte al secondo, seguendo la palla e ogni giocatore nel campo. Le informazioni fluiscono in tempo reale fornendo il posizionamento dei giocatori in coordinate xy e della palla in xyz[4]. Le statistiche prodotte dal sistema includono, ma non sono limitate a:

- Distanza percorsa
- Velocità media
- Velocità massima
- Velocità istantanea
- Numero di scatti
- Mappa di copertura
- Tempo di possesso

[3] "Football Player Performance Analysis | Football Video Analysis | STATS." https://www.stats.com/football/. Ultimo accesso: 16 ott. 2018.

[4] "SportVU - Wikipedia." https://en.wikipedia.org/wiki/SportVU. Ultimo accesso: 17 ott. 2018.

- Possesso del giocatore
- Velocità della palla
- Zona di copertura
- Formazione della squadra[5]

2.1.3 Interplay Sports

Questa azienda fornisce uno strumento di analisi video che può essere usato in molti sport con soluzioni ad-hoc, in base alle esigenze. Permette di analizzare fino a 4 flussi video diversi, da sorgenti varie (in diretta dalla tv o da internet, video importati, videocamere ecc.). Si possono scegliere delle variabili da analizzare per le azioni e i passaggi dei giocatori di entrambe le squadre. L'analisi viene supportata da una serie di eventi che sono parte di una ontologia che supporta l'utente, ma di fatto è svolta a mano da quest'ultimo.[6]

Il software stesso viene definito come "Professional Front-End Video Analysis System".

2.1.4 Camargus

Il sistema sviluppato da Camargus permette di usare una matrice di 16 telecamere poste nello stesso punto per creare una immagine video non distorta e unica del campo di gioco.

Tutte le zone dello schermo possono essere ingrandite senza perdere qualità o avere distorsione, spostandosi in continuità senza salti. Camargus è un ottimo sistema di ripresa, che tuttavia è manchevole di uno strumento di analisi[7]. È quindi utilizzabile per la generazione dei soli dati video, ma non per ottenere direttamente le informazioni riguardo la posizione dei giocatori, o delle azioni che questi hanno effettuato nel campo.

2.1.5 TRACAB by ChironHego

TRACAB è un sistema di tracking ottico che promette di fornire le posizioni dei giocatori e della palla in 3D in tempo reale. In particolare, usa "telecamere Super-HD e un sistema brevettato di processamento immagini per fornire il tracking di tutti gli oggetti in

[5] "Football Player Tracking | Football Tracking System | STATS." https://www.stats.com/sportvu-football/. Ultimo accesso: 16 ott. 2018.

[6] "Interplay-sports." http://interplay-sports.com/. Ultimo accesso: 17 ott. 2018.

[7] "Camargus Broadcast System - YouTube." 21 mar. 2013, https://www.youtube.com/watch?v=SO32pEgCeDI. Ultimo accesso: 17 ott. 2018.

movimento con un ritardo massimo di tre frame. Un software analizza ogni immagine per estrarre le posizioni in xyz per ogni oggetto"[8].

Il sistema include anche dei moduli per l'estrazione di ulteriori statistiche che vanno di pari passo con i dati sul tracking, permettendo ad esempio la creazione di mappe di calore (heat maps), grafo dei passaggi e così via.

"Caratteristiche chiave:

- Tracciamento in tempo reale in 3D di tutti gli oggetti, inclusi tutti i giocatori, l'arbitro e la palla;
- Sviluppati diversi sport, inclusi calcio, tennis, basket, baseball, cricket e football americano;
- Necessarie solo due unità telecamere Super-HD compatte;
- Relazione tra la grafica del broadcast e i fornitori di dati assodata;
- L'unica soluzione dimostrata per il tracking in diretta"

2.1.6 LongoMatch

LongoMatch è un software che permette di effettuare analisi sui video, sia in diretta, che in una fase successiva. Esso permette di importare i flussi video (fino ad un massimo di 8 insieme) e di taggare a mano gli eventi che si verificano, selezionando da una lista di eventi possibili e associandoli ai giocatori che vi hanno preso parte.

Inoltre, fornisce una serie di statistiche a partire da tutti gli eventi che sono stati annotati, come ad esempio l'area del campo in cui sono avvenuti più eventi, il possesso della palla e così via.[9]

Anche in questo caso si tratta di un sistema che supporta il riconoscimento manuale e non automatico delle azioni. E non consente, almeno in modo diretto, di riconoscere e tracciare i giocatori o la palla per ottenere le loro posizioni.

2.1.7 Intel® True View (FreeD)

Il sistema di Intel consiste essenzialmente nel dotare un campo con una griglia video per coprire tutti i possibili angoli visivi. In particolare, si fa uso di un set di 32 telecamere fisse ad alta risoluzione (5K), ognuna delle quali fornisce una risoluzione di 5120 x 2880 pixel.

[8] "TRACAB Optical Tracking Product Info Sheet - ChyronHego." https://chyronhego.com/wp-content/uploads/2018/02/TRACAB-PI-sheet.pdf. Ultimo accesso: 17 ott. 2018.

[9] "LongoMatch PRO." https://longomatch.com/en/pro/. Ultimo accesso: 18 ott. 2018.

Sei di queste telecamere sono dotate di zoom, mentre le altre hanno lunghezze focali fisse. Questo garantisce la possibilità di coprire tutto lo spazio del campo e di poter effettuare lo zoom sulle azioni e sui punti importanti del campo di gioco. La griglia ha inoltre permesso di assegnare a ogni sezione del campo coordinate XYZ e di generare un'immagine 3D.[10]

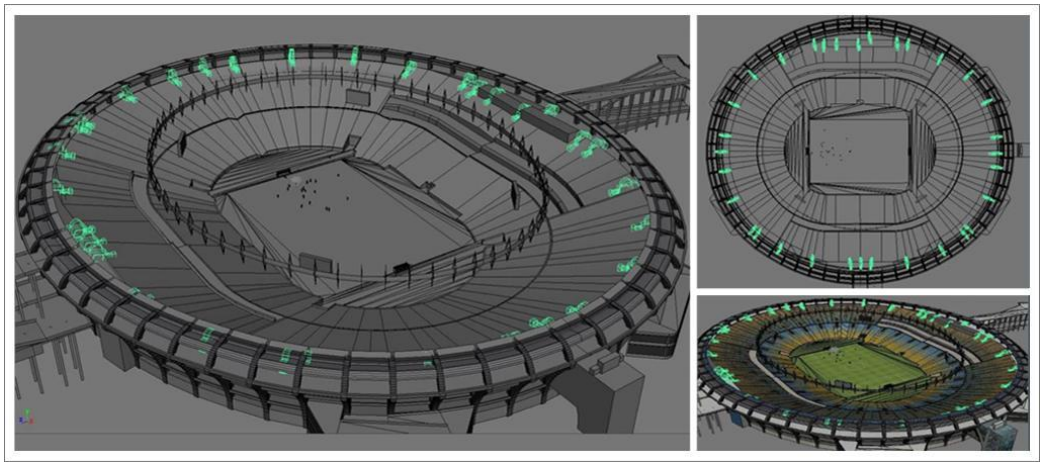


Figura 2.4 - Posizionamento delle 32 telecamere nel campo di gioco.

In un secondo di utilizzo, alla velocità massima, le 32 telecamere raccolgono una elevata quantità di dati, pari a 1 terabyte, che viene elaborata da una serie di server con processori XEON e i7 ad alte prestazioni che effettuano la ricostruzione 3D del video. L'intero processo inizia con la matrice di telecamere che cattura i dati volumetrici (altezza, larghezza e profondità) di tutte le azioni. I dati catturati sono quindi analizzati, ricostruiti, compressi e codificati prima di essere memorizzati. I "Voxel" (pixel con volume) permettono a questa tecnologia di ottenere una multi-prospettiva 3D di ogni zona del campo.[11]

In sostanza questo è un sistema ad elevata precisione e prestazioni che consente di visualizzare ogni zona del campo come si si fosse lì presente. La possibilità di ottenere le posizioni degli elementi sul campo, inoltre, lo renderebbe perfetto per un lavoro che necessita delle posizioni come il nostro.

Tuttavia, anche in questo caso non ci sono moltissime informazioni riguardo le tecniche usate per generare il video 3D finale e come eventualmente riconoscere automaticamente le scene e i momenti salienti delle azioni. Si fa solo riferimento all'enorme potenza di calcolo richiesta per elaborare la grossissima mole di dati che viene generata in ogni secondo dalle oltre 30 telecamere ad altissima risoluzione. L'elaborazione, inoltre, è di tipo video, ovvero è un flusso unico a partire dalle numerose

[10] "FreeD: la rivoluzionaria tecnologia di streaming 5K offre ... - IQ Intel." https://iq.intel.it/freed-la-rivoluzionaria-tecnologia-di-streaming-5k-offre-replay-di-partite-di-calcio-a-360-gradi/. Ultimo accesso: 24 ott. 2018.

[11] "NFL + Intel® True View." https://www.intel.com/content/www/us/en/sports/nfl/overview.html. Ultimo accesso: 24 ott. 2018.

sorgenti. Questo consente, ad esempio, di avere i replay di una azione pronta a pochi secondi dal momento in cui viene ripresa.

2.1.8 Conclusioni

Una parte dei prodotti commerciali realizza un sistema che applica il lavoro umano per riconoscere alcune o tutte le azioni nel campo: “L’integrazione dei sistemi di statistiche dei giocatori e dei sistemi video richiede un grande quantitativo di lavoro manuale. Per esempio, gli eventi annotati dagli allenatori o da altri annotatori umani esperti devono essere estratti manualmente dai video, spesso richiedendo ore di lavoro davanti i computer. Inoltre, collegare le statistiche dei giocatori al video richiede anche del lavoro manuale” come si evince in [11].

Tuttavia, sebbene non siano disponibili particolari riferimenti alle tecniche impiegate per quanto riguarda i maggiori software commerciali, è lecito pensare che lo sviluppo osservato in questi ultimi anni in letteratura, sia stato abbracciato dalle maggiori aziende per rendere sostenibile il proprio lavoro. Lo dimostra il crescente numero di aziende che si occupano di fornire un tale servizio, e il numero di queste che lo fornisce in tempo reale e per un numero elevato di partite. I sistemi installati nei campi da gioco sono sempre in maggior numero e sempre più sofisticati, fornendo un tracciamento dei giocatori a 3 dimensioni con crescente precisione.

2.2 Visualizzazione di eventi in ambito sportivo

Nelle trasmissioni sportive, molto spesso, sono presentate delle statistiche riguardo i giocatori o riguardo le azioni che si svolgono. Sempre più si osserva come l’uso di nuove tecniche di visualizzazione riescano a dare allo spettatore una serie di indicazioni sulle performance, lo stile e l’abilità dei giocatori, o a mostrare con dettaglio le traiettorie prese dalla palla. In sostanza si può dare allo spettatore una nuova esperienza che gli permetta di osservare qualcosa che normalmente non sarebbe possibile vedere, come ad esempio la ricostruzione 3D di una azione, il resoconto degli spostamenti del giocatore, le aree in cui ha corso di più ecc.

L’analisi fornita risulta essere molto più efficace se mostrata con un adeguato mezzo di visualizzazione, soprattutto alla luce del fatto che il quantitativo di dati che viene generato ad ogni singolo evento è sempre più grande. Dunque, il problema di mostrare questi dati in modo efficace è sicuramente una sfida ogni giorno crescente.

“Il numero di visualizzazioni dei dati nello sport è cresciuto rapidamente nelle decadi passate. La stampa e i media online, incluso il The New York Times, presentano i dati sportivi utilizzando visualizzazioni infografiche e interattive.”[12]

[12] "State of the Art of Sports Data Visualization - Perin - 2018 - Computer ...." 10 lug. 2018, https://onlinelibrary.wiley.com/doi/full/10.1111/cgf.13447. Ultimo accesso: 07 nov. 2018.

Si vogliono allora definire e individuare quali sono le tecniche migliori, disponibili in letteratura, per visualizzare i dati di eventi sportivi. Più precisamente si farà riferimento al calcio, ma anche a tutti gli altri sport in cui si possono applicare con successo tali tecniche.

2.2.1 Dati di tracking

I dati di tracking che sono a disposizione sono sempre di più, principalmente grazie alla possibilità di reperire dati spazio-temporali con maggiore precisione anche con alcune delle tecniche illustrate precedentemente. Questi dati possono essere usati per effettuare l'estrazione di informazioni utili, non precedentemente disponibili. Il modo più efficace per estrarre queste informazioni è aggregarle o visualizzarle mediante delle tecniche ad-hoc.

2.2.2 Event density map

Un modo semplice ed efficace per mostrare dei dati di tracking non elaborati è usare delle "dot map", ovvero delle mappe di punti, in cui ogni punto corrisponde a un evento. In questo modo si può caratterizzare l'abilità di un giocatore e mostrare quella che è la traiettoria della palla, segnando il punto di partenza e la destinazione della stessa nello spostarsi mentre si verifica l'evento.

Ne è un esempio il sistema Baseball4D [12], con cui vengono mostrati tutti i tocchi della palla sul campo di gioco per ricostruire così l'azione e le traiettorie della palla e dei giocatori nel campo di baseball. È così possibile esplorare ogni giocata utilizzando una serie di schermate, la cui principale mostra l'intero campo, l'insieme alle posizioni e le traiettorie dei giocatori. Può anche essere cambiata la posizione della telecamera per esaminare una particolare zona del campo, applicando anche uno zoom. In una seconda finestra possono essere mostrate delle "heatmap" (o mappe di calore) che danno un'altra visione dei dati. Tutto questo non è legato a un singolo evento, ma possono essere visualizzati più eventi contemporaneamente.

Un altro strumento della stessa natura, ovvero che mostra i tocchi sul campo e le relative traiettorie a partire dai semplici dati di tracking, è quello proposto da "Bo Moon e Richard Brath" in [13]. Le tecniche di visualizzazione da loro usate includono filtraggio, visioni legate per famiglia, glifi multivariati con collegamento intuitivo tra colori e forme, aggregazione con griglie di calore e mappe di contorni, e un sistema di supporto per l'analisi da e per la visualizzazione. Un chiaro esempio di applicazione delle seguenti tecniche è mostrato nell'immagine seguente: a sinistra sono presenti i colpi subiti da chi riceve, mentre a destra i punti di inizio e fine delle traiettorie della palla e dei giocatori.

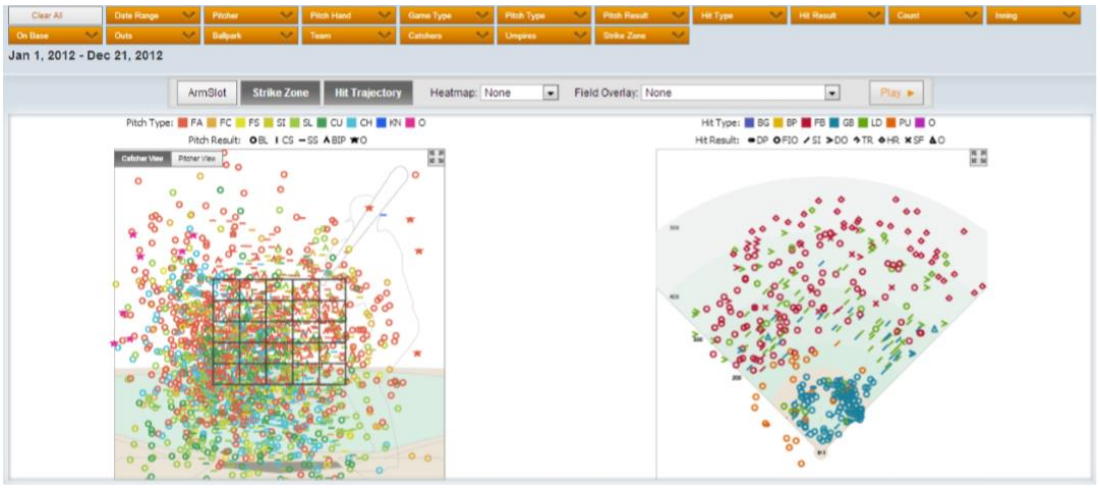


Figura 2.5 - A sinistra dot map con i colpi in ricezione, a destra inizio e fine delle traiettorie.

Un altro sistema che sfrutta un simile sistema di visualizzazione, ma associato al basket, è stato elaborato da Peter Beshai[13]. In questo caso vengono mostrate le statistiche sui giocatori in dettaglio, riferite ai tiri effettuati. Sul campo sono mostrati dei punti per ogni tiro, mentre dei grafici mostrano le performance aggregate per distanza dal canestro (ovvero il numero di tiri per ciascuna distanza).

Un metodo del tutto analogo è stato usato da Kirk Goldsberry, che con "CourtVision[14]" mette a disposizione delle tecniche per visionare, quantificare e comunicare gli aspetti spaziali dei tiri effettuati in NBA dai giocatori, usando delle "dot map" e "spread charts", ovvero una mappa di calore discretizzata, con colori variabili.

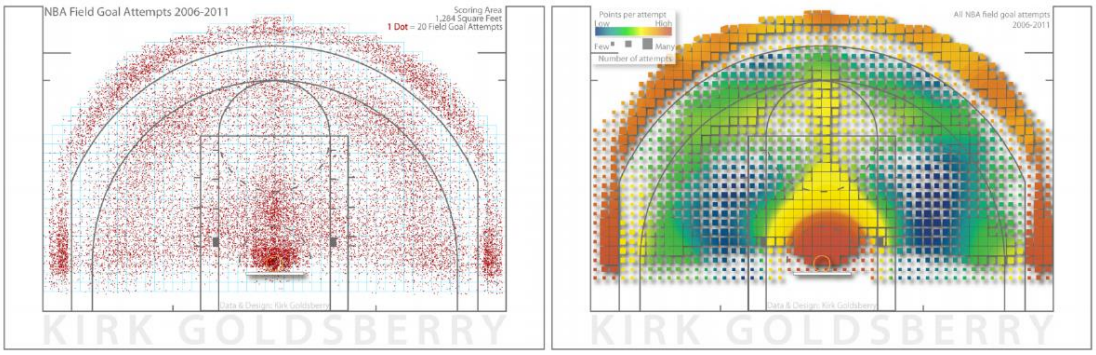


Figura 2.6 - A sinistra density map, a destra il punteggio medio ottenuto in base alla zona di tiro.

L'immagine di destra mostra la densità di tiri, in cui il colore rappresenta il quantitativo di punti ottenuti in media rispetto i tentativi effettuati in base alla posizione.

[13] "Buckets: Basketball Shot Visualization - Semantic Scholar." 5 dic. 2014, https://pdfs.semanticscholar.org/8413/cc6a30ce8813a60bb85a1d039bc5ad70e39a.pdf. Ultimo accesso: 07 nov. 2018.

[14] "New Visual and Spatial Analytics for the NBA - Semantic Scholar." https://pdfs.semanticscholar.org/46e4/a7271de62e9118625dec935c4aef1bc0ea74.pdf. Ultimo accesso: 07 nov. 2018.

Le immagini prese dal lavoro di Goldsberry evidenziano quello che è riportato in [14]: "Le mappe di densità sono spesso decorate con le linee di separazione del campo di gioco, per fornire una scala e contestualizzare i dati con all'interno le regole di gioco. In alcuni casi le linee emergono nella visualizzazione in automatico proprio a causa della distribuzione dei dati. Nel caso del basket ad esempio la linea dei 3 punti appare per la mancanza di tiri effettuati appena al suo interno".

2.2.3 Rappresentazione di eventi simili

Gli eventi che accadono nello sport sono determinati principalmente dalle coordinate (x, y) in cui avvengono. Tuttavia, può essere utile anche rappresentarli raggruppandoli in base alle similitudini che li caratterizzano. Se due eventi possiedono un pattern simile, allora questi sono indicati assieme in modo da identificare i tratti tipici di quelle giocate da parte di un giocatore.

Ad esempio, Damien Demaj in [15], ha raggruppato nell'ambito tennistico i servizi simili effettuati da Roger Federer localmente, utilizzando il K-means[15]. Così facendo è stato possibile visualizzare e quantificare statisticamente la maggior variabilità spazio-temporale dei servizi di questo giocatore rispetto a quella di altri.

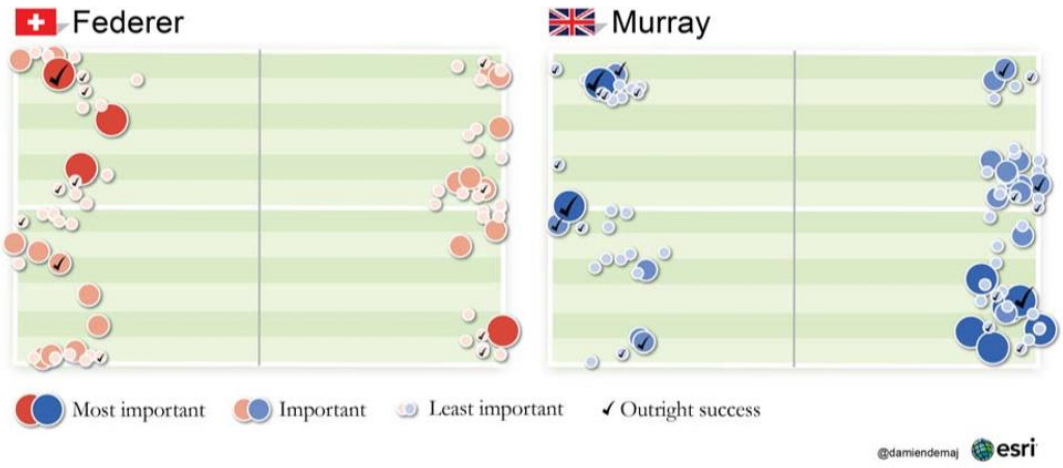


Figura 2.7 - Raggruppamento in cluster dei servizi di due giocatori di tennis.

Questo è un esempio dei cluster ottenuti dopo l'applicazione del K-means sui dati dei servizi.

2.2.4 Traiettorie

Le traiettorie sono il risultato della connessione di più punti discreti. Ciascun punto rappresenta un singolo evento che è avvenuto in uno spazio e ad un certo istante, ognuno dei quali è della stessa tipologia. Possono ad esempio far riferimento alla palla, ai

[15] Algoritmo di classificazione per raggruppare in cluster elementi con caratteristiche simili.

giocatori e così via. Inoltre, possono essere rappresentate traiettorie più corte per ridurre l'ingombro nella visualizzazione, quando vengono mostrate più entità contemporaneamente.

Ad esempio, Gennady Andrienko ed altri[16] usano le traiettorie del movimento dei giocatori per mostrare la pressione che essi esercitano nelle aree di gioco nel calcio. Queste sono mostrate con colorazioni diverse in base alla squadra, mentre la palla, in blu, è affiancata da aree colorate che rappresentano le zone di contrasto tra difensori e attaccanti. Le linee, inoltre, tendono ad essere sempre meno visibili al trascorrere del tempo.

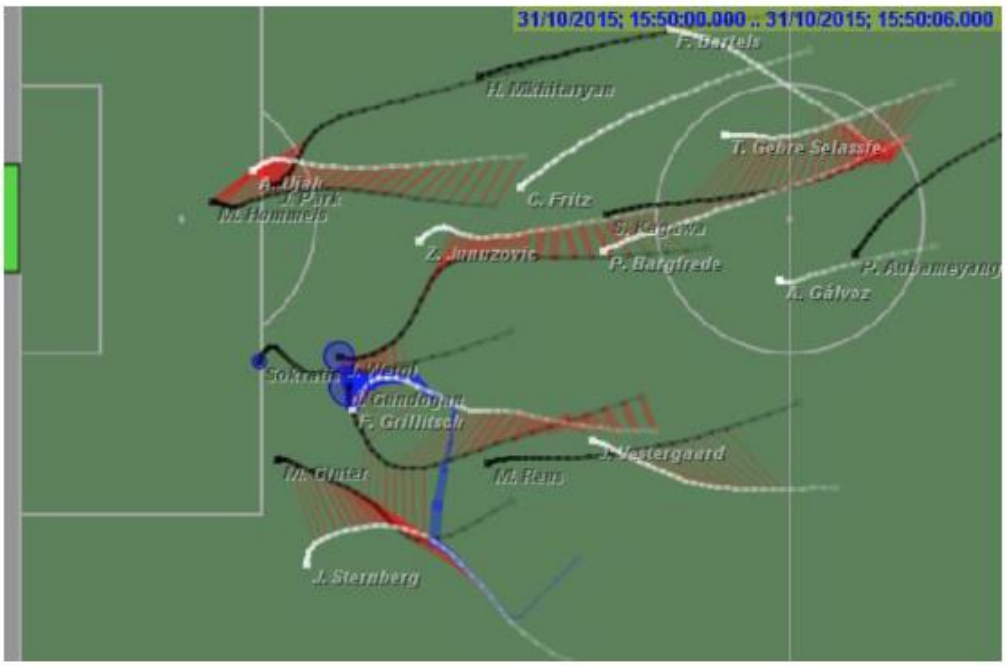


Figura 2.8 - Traiettorie dei giocatori e della palla.

Infine, si può osservare come tra le traiettorie di giocatori diversi siano presenti ulteriori linee che rappresentano la pressione esercitata tra i due.

Uno dei problemi più evidenti del disegnare sulla rappresentazione del campo di gioco numerose traiettorie, è quello di cadere nel problema dell'overplotting e di non riuscire più ad individuare i tratti salienti di una azione o di parte di essa. Lo scopo della ricerca di J. Heer e altri[17] è proprio quello di sintetizzare le azioni individuate, fornendo una rappresentazione più schematica della stessa. La Figura 2.9 sottostante rappresenta il problema dell'overplotting, in cui sono rappresentate le traiettorie di 22 giocatori più quella della palla per intervalli di tempo crescenti.

[16] "Exploring pressure in football - ACM Digital Library - Association for ...." 29 mag. 2018, https://dl.acm.org/citation.cfm?id=3206558. Ultimo accesso: 09 nov. 2018.

[17] "Dynamic Visual Abstraction of Soccer Movement - ACM Digital Library." 1 giu. 2017, https://dl.acm.org/citation.cfm?id=3128425. Ultimo accesso: 09 nov. 2018.

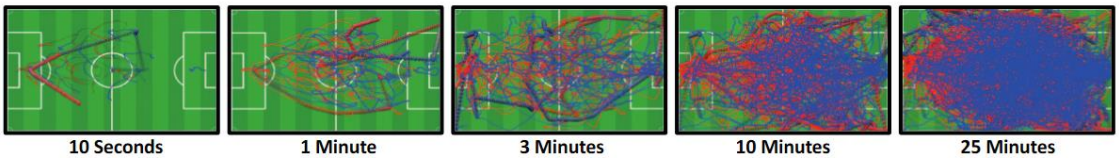


Figura 2.9 - Rappresentazione delle traiettorie al crescere del tempo.

Il primo passo effettuato è quello di selezionare i dettagli interessanti ai fini dell'azione, ovvero il movimento dei giocatori nei pressi della palla, riducendo così considerevolmente il quantitativo di dati da mostrare. Alcuni movimenti, come i passi, i dribbling e gli stop della palla sono mostrati con tratti più spessi. Il secondo passo è quello di semplificare le curve impiegando una tecnica di interpolazione con cui la traiettoria principale è evidenziata e gli eventi rilevanti sono mostrati. Infine, il terzo livello è quello di aggregazione, con cui le varie traiettorie sono unite grazie al K-means in cluster. In questo livello sono presenti anche dei messaggi che appaiono sui punti della traiettoria che specificano i giocatori coinvolti e gli eventi che si sono verificati.

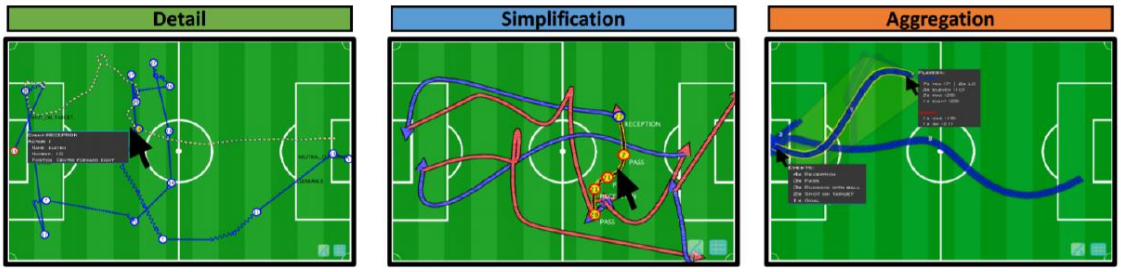


Figura 2.10 - Passi effettuati durante la semplificazione dei dati.

2.2.5 Passaggi e tiri

"Uno degli eventi più comune è il passaggio, e molti di questi passaggi sono a breve distanza. I passaggi possono avvenire in sequenza, con una serie di piccoli passaggi. In termini di effetto, un serie di passaggi con la palla (definita cluster), che parte dal giocatore A e finisce con il giocatore B, è equiparabile a un passaggio diretto da A a B rendendo quest'ultimi i giocatori più importanti nel cluster.

Comunque, è a volte interessante sapere informazioni più specifiche riguardo un cluster, come quale giocatore ne è stato coinvolto e l'ordine dei passaggi. Poiché i cluster possono assumere diversi ruoli in una fase, proponiamo di visualizzarli in modi differenti e ognuno di essi presenta i suoi pro e contro. Per esempio, tali compiti possono dipendere dalla posizione spaziale dell'azione, dall'ordine cronologico degli eventi, e dalla frequenza alla quale i giocatori appaiono all'interno di una serie di passaggi."[18]

[18] "SoccerStories: A Kick-off for Visual Soccer Analysis - IEEE Xplore." https://ieeexplore.ieee.org/iel7/2945/6634084/06634087.pdf. Ultimo accesso: 10 nov. 2018.

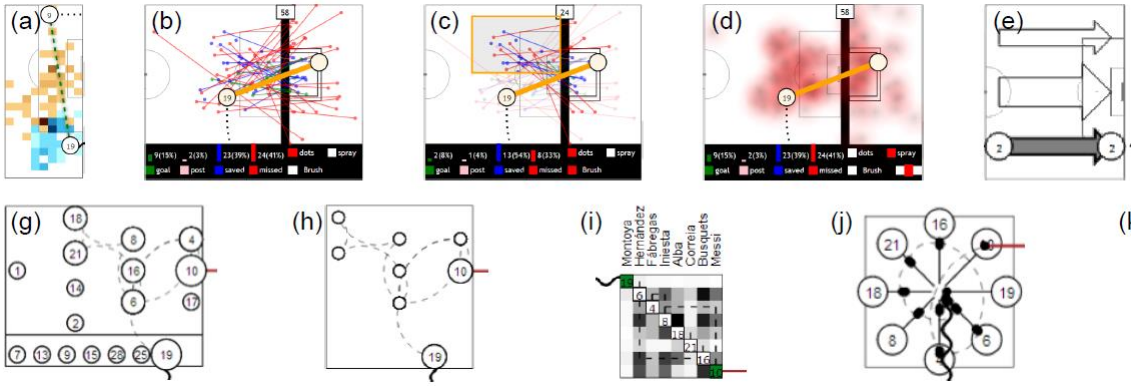


Figura 2.11 - Diversi esempi di visualizzazione di passaggi e tiri.

In (g) viene mostrato un esempio dell'organizzazione in cluster dei passaggi tra i giocatori, mentre le altre figure mostrano vari esempi di visualizzazione di passaggi tra i giocatori, sfruttando raggruppamenti per meglio esemplificare il peso delle varie coppie.

In (b) vi sono invece i tiri effettuati, con il punto di inizio, e quello di fine. In (c) sono mostrati solo quei tiri che possono finire in un goal. In (d) è mostrata una "heatmap" in modalità "spray" per indicare i tiri presenti in (b), ma evitando la possibile confusione dovuta al numero di linee.

2.3 Generazione di dati sintetici mediante computer graphics

I sistemi di machine learning o in generale i sistemi intelligenti hanno mostrato grandi progressi e grandi risultati nelle applicazioni del mondo reale, ma posseggono un forte limite, dovuto ai dati che essi necessitano. Si è visto infatti come i risultati e le performance siano fortemente dipendenti dai dati, dalla loro quantità e qualità soprattutto. È però evidente come il processo manuale di collezionamento e annotazione per produrre questi dati, richieda un lavoro e uno sforzo elevatissimo, che cresce ovviamente di pari passo con la necessità di avere sempre più dati.

L'uso di sistemi di simulazione come metodo di raccolta di dati, allora, diventa una soluzione efficace che consente di produrre molti di più dati di prima, di migliore qualità e con uno sforzo umano decisamente inferiore. È infatti possibile annotare in modo programmatico i dati che si producono, e farlo con una velocità di gran lunga superiore.

Ad esempio, nel paper "Understanding Real World Indoor Scenes With Synthetic Data" [16], si è costruito un sistema di generazione di dati sintetici per addestrare sistemi in grado di riconoscere e comprendere delle scene al chiuso. In questo caso Ankur Handa ed altri, hanno sfruttato un set di modelli 3D preesistenti di alcune stanze, per generare in modo automatico mediante OpenGL, una grossa quantità di scene diverse. Queste sono state costruite usando telecamere virtuali diverse poste in punti randomici della stanza.

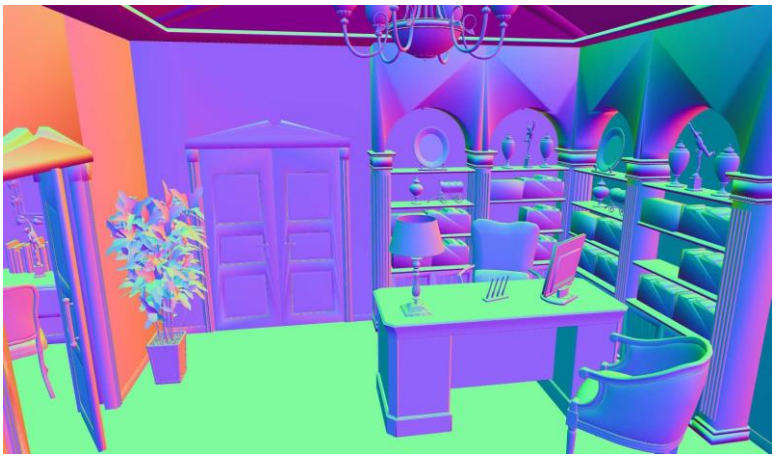


Figura 2.12 - Esempio di scena di partenza.

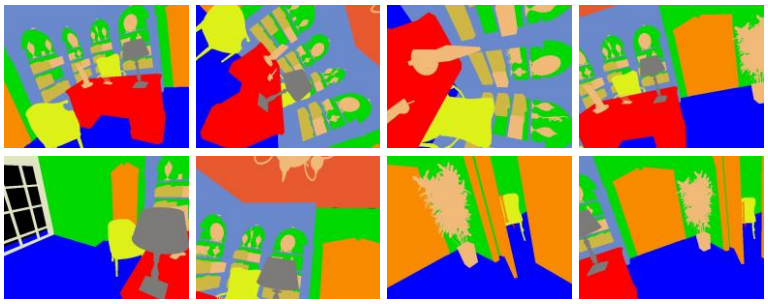


Figura 2.13 - Esempio di scene generate in modo randomico mediante computer grafica.

Le scene generate sono tutte provviste di annotazioni riguardo la distanza degli oggetti. Il processo, dunque, è totalmente automatico sia nella generazione dei dati che delle annotazioni correlate.

Un lavoro affine è quello di Alexey Dosovitskiy ed altri [17], che hanno sviluppato un sistema analogo di generazione sintetica per il riconoscimento mediante reti neurali del campo di flusso ottico di due immagini. Anche in questo caso la ragione di un tale lavoro è da ricercarsi nella limitatezza dei dati a disposizione, soprattutto per addestrare una CNN. È stato generato un dataset di 22'872 coppie di immagini in un dataset denominato "Flaying Chairs". Mediante l'uso di immagini pubbliche ottenute da "Flickr", sono state prodotte delle immagini randomiche utilizzando degli effetti grafici che riproducevano il movimento dell'oggetto (in questo caso delle sedie) e il movimento dello sfondo.

Il risultato della ricerca mostra che il sistema addestrato su un dataset sintetico, come quello brevemente qui descritto, sia in grado di generalizzare a sufficienza da avere delle prestazioni pari a quelle dello stato dell'arte, utilizzando poi come testing set immagini reali. In sostanza non è stato necessario utilizzare dei sample reali per addestrare il sistema, almeno nell'ambito del problema del flusso ottico, mostrando quindi la validità della tecnica.

In [18] Qi Wang ed altri hanno affrontato il problema della generazione di un dataset a partire da un video gioco. In particolare, il problema di effettuare il conteggio delle persone in una scena è stato risolto generando un grosso dataset utilizzando il gioco

Grand Theft Auto 5[19]. È stato sviluppato un collezionatore di dati in accoppiata a un etichettatore. Tramite questi due strumenti è stato possibile produrre un dataset di larga scala che avesse i seguenti quattro vantaggi rispetto agli altri dataset: "1) annotazione e raccolta gratuita; 2) volumi di dati più grande e a risoluzione migliore; 3) scene più diversificate e 4) annotazioni più accurate".

2.4 Valutazione di metriche per un dataset

Uno studio che può essere interessante da svolgere è quello di capire e determinare le caratteristiche di un dataset, anche in relazione ad altri insiemi di dati. Infatti, ogni dataset può avere delle caratteristiche anche abbastanza diverse, in funzione di numerose variabili.

Il documento di ricerca [19] prende in esame due leghe diverse, la Premier League inglese e La Liga spagnola, per verificare quanto il sistema di gioco fosse differente e quanto le prestazioni dei giocatori fosse legata ad una serie di parametri definiti. Per valutare queste differenze, i parametri esaminati da Dellelal e altri, sono:

- distanza totale percorsa;
- distanza percorsa ad alta intensità con palla al piede;
- distanza percorsa ad alta intensità senza palla al piede;
- azioni tecniche;
- gioco aereo;
- gioco a terra;
- numero di passaggi;
- possesso palla;
- numero di tocchi palla;

Tutti i valori riportati dall'articolo sono espressi in termini di valori medi e deviazione standard. Sono state analizzate un totale di 5938 osservazioni, fatte su 600 partite diverse, e per ciascuna di esse sono state calcolati i valori di distanza percorsa da ciascun giocatore a tre velocità differenti. In particolare, sono state studiate le distanze a due possibili velocità, ovvero ad alta intensità (oltre i 24 km/h) e a bassa intensità (tra 21 e 24 km/h). I risultati hanno mostrato che i giocatori de La Liga hanno corso per distanze comprese tra 10496 metri e 11779 metri, che sono in sostanza equiparabili alla distanza percorsa dai giocatori inglesi della Premiere League. Questa comparazione effettuata anche su base del posizionamento del giocatore, ha inoltre rilevato come non ci sia una particolare differenza. Pertanto, conclude il documento, la distanza totale percorsa dai giocatori non può essere considerata un importante fattore discriminante per indicare le differenze di performance tra le diverse leghe sportive.

[19] Gioco open-world di simulazione, ad elevato realismo grafico.

Capitolo 3

Progettazione

Si è considerato, come punto di partenza, un documento che indicava gli eventi di interesse nello sport del calcio. Questi definiscono un lavoro tassonomico abbastanza ricco che ha richiesto una definizione matematico-geometrico rigorosa degli eventi. Questa definizione rigorosa è fondamentale in quanto identifica quali sono i dati finali necessari nel dataset, e che quindi devono essere riconoscibili dal sistema nel suo complesso.

In questa prima fase di analisi dei requisiti, il lavoro delle tre tesi è stato fortemente coordinato. Infatti, in prima battuta è stata concordata e decisa, come meglio vedremo più avanti, la definizione degli eventi dividendoli in due categorie: gli eventi atomici e quelli complessi.

Gli eventi atomici sono caratterizzati dalla peculiarità di essere istantanei, ovvero si verificano in un determinato momento, e non hanno una vera e propria durata. Gli eventi complessi, invece, sono caratterizzati da un istante di inizio e uno di fine che non coincidono. Hanno dunque una durata estesa e sono tendenzialmente ottenuti dalla combinazione di eventi atomici con altri eventi atomici, di eventi atomici con eventi complessi o ancora da eventi complessi con altri eventi complessi.

In questo lavoro ci si è concentrati sulla realizzazione di un sistema di generazione di dati sintetici da poter utilizzare per il riconoscimento di entrambe le categorie di eventi, affiancato ad un software in grado di mostrare il dataset generato e le relative annotazioni. Uno dei lavori di tesi, invece, si è occupato di sviluppare la parte atomica dell'event detector, utilizzando un sistema di regole. Il terzo ed ultimo lavoro si è infine occupato degli eventi complessi, sfruttando un framework a regole logico-temporali denominato ETALIS [20] al fine di riconoscerli.

3.1 Descrizione dei dati

I dati necessari, come già ripetuto, sono le posizioni dei giocatori e della palla rispetto al campo di gioco, e le annotazioni riguardo agli eventi che sono avvenuti durante l'incontro. In una seconda fase si è deciso di arricchire queste informazioni, con le coordinate dei

giocatori relative allo schermo, in modo da poter visualizzare mediante lo strumento di visualizzazione tutte queste informazioni direttamente sul video.

Si è stabilito, allora, come sistema di riferimento per le posizioni dei giocatori e della palla, un sistema cartesiano con origine degli assi posto in uno dei quattro angoli del campo di gioco preso in considerazione. Un esempio di tale sistema è disponibile in Figura 3.1. La variabile x può assumere valori compresi tra $[0, x_{max}]$, mentre la variabile y può assumere valori compresi tra $[0, y_{max}]$, dove $x_{max} \in [100, 110]$ e $y_{max} \in [64, 75]$, secondo la regolamentazione delle dimensioni dei campi di calcio.

In determinate circostanze, le coordinate possono assumere valori più grandi o più piccoli dei limiti indicati. Ciò ad esempio avviene quando la palla finisce in fallo laterale o un giocatore esce dalle linee di gioco del campo.

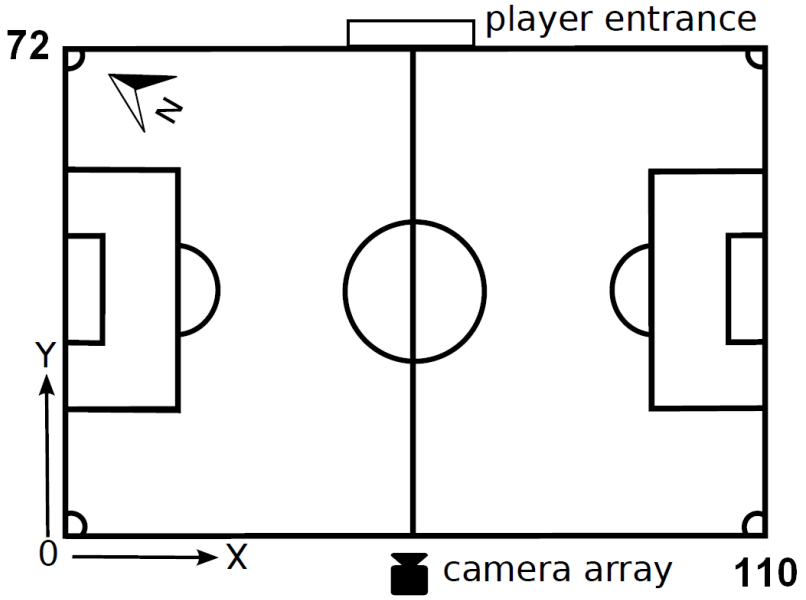


Figura 3.1 - Schema di un campo con relativo sistema di coordinate.

Le informazioni, inoltre, avranno un certo passo di discretizzazione, ovvero avranno una certa frequenza di emissione. Questo vuol dire che ci si aspetta una lista completa delle posizioni di ogni elemento sul campo, per ogni singolo frame della partita, ciascuno di essi distanziato dall'altro di un certo tempo fisso. Le annotazioni sugli eventi avranno anch'esse una informazione temporale (istante di inizio ed eventualmente di fine) sincronizzata con le informazioni video e/o con i dati sulle posizioni.

3.2 Generazione dei dati sintetici

L'obiettivo finale di questo lavoro è la produzione di un dataset. È sorta infatti la necessità di avere un dataset molto grande così da rendere valido ed efficace il sistema di riconoscimento di eventi nel maggior numero possibile di situazioni. Sfruttando quindi le

potenzialità di una generazione sintetica si è pensato di utilizzare un gioco open-source che permettesse, mediante la sua modifica, di ottenere i dati richiesti. Come visto le informazioni richieste per un tale set di dati sono molteplici e sono di diversa natura. Si passa da informazioni posizionali, a informazioni video, alle annotazioni circa gli eventi di interesse.

I dati posizionali, sia rispetto il campo di gioco che rispetto al video, devono avere un riferimento che li inquadri temporalmente nella partita. La cadenza temporale deve essere costante e i sample, quindi, equi-spaziati. Analogamente le annotazioni devono essere caratterizzate dallo stesso identificativo univoco temporale, in modo da poter associare l'azione alle posizioni. Le annotazioni devono rispettare la separazione in eventi atomici ed eventi complessi, e coprire la tassonomia definita al paragrafo 3.3. Le due macro-classi sono corrisposte da altrettanti file, con uno specifico formato di rappresentazione XML. In particolare, ogni classe di evento deve possedere la serie di attributi tipici di quell'evento, così da descriverne la dinamica.

Le informazioni video delle partite devono essere quanto più verosimili possibile, e devono mantenere una qualità elevata in modo da poter sfruttare anche un eventuale riconoscimento di eventi a partire dalle immagini. I dati associati alle posizioni video invece, sono necessari per poter mostrare sul video la posizione dei giocatori sottoforma di *bounding box*, e di avere una sincronia con il video che altrimenti non sarebbe possibile avendo i soli dati posizionali, come si vedrà più avanti.

La generazione, inoltre, deve essere quanto più autonoma possibile, in modo da semplificare e velocizzare il processo di esportazione, e consentire la produzione estesa di dati.

3.2.1 Esportazione delle annotazioni

Per esportare gli eventi sottoforma di annotazioni, si è pensato inizialmente di utilizzare una classe per scrivere su un file testuale i dati prodotti via via dal gioco. Da un'analisi più attenta durante la fase di definizione dell'interfaccia trai i vari componenti software del sistema, si è invece deciso di fare uso di un formato più strutturato, ma comunque leggibile, delle annotazioni, ovvero il formato *XML*. Il formato dei dati specifico utilizzato è il formato del software *Cvat*. Cvat è un software open-source di annotazione che permette di visualizzare e annotare al contempo delle etichette su una sequenza video. In questo modo è stato utilizzato un formato già testato e indicativamente standardizzato per dati associati a un video. Lo scopo di questa scelta è anche avere la possibilità di poter verificare su video gli eventi annotati ed eventualmente sfruttare il software per correggere eventuali annotazioni non precise o scorrette. Per rispettare il formalismo introdotto da Cvat, si è optato per la scrittura di una libreria di classi di esportazione in *XML*, così da disaccoppiare l'implementazione del formato di esportazione, dalla logica del simulatore.

Si è pensato di aggiungere anche una classe che esportasse informazioni non strutturate in plain text, così da fornire dati fruibili mentre si sta simulando una partita così da poterli verificare. I dati *XML,* infatti, non sono disponibili fintanto che l'intero albero non sia

stato generato e questo non avrebbe consentito di verificare la correttezza dei dati in fase di debug.

3.2.2 Esportazione delle posizioni sul campo e a video

I dati posizionali rispetto il campo di gioco, che seguono il sistema di riferimento degli assi in Figura 3.1, non necessitano di una struttura complessa come quella delle annotazioni. Pertanto, si è pensato di usare un file in plain text, che contenesse per ogni riga i dettagli di una entità (un giocatore o la palla) rispetto l'indicazione temporale, la posizione e l'identificativo dell'entità stessa. Poiché si vuole avere la totalità delle posizioni a ogni dato istante temporale, ogni marcatura temporale viene ripetuta un numero di volte pari al numero di giocatori più la palla.

3.3 Tassonomia degli eventi

Gli eventi che sono stati richiesti come requisito di partenza dall'intero sistema, sono esemplificati nella Tabella 3.1. Alcuni di questi sono eventi in sé, ovvero che sono determinati in un preciso instante temporale, e che sono stati svolti da precisi attori. Altri elementi di questa lista, invece, sono il risultato di una aggregazione di eventi (dunque una sorta di statistica) e sono quindi computabili a partire dagli eventi riconosciuti.

L'obiettivo finale dall'intero lavoro sviluppato nelle diverse tesi è quello di consentire il riconoscimento e l'annotazione di una lista di eventi ogni volta che si analizza una partita di calcio. Questa lista di eventi è stata fornita sottoforma di una tassonomia descritta qui di seguito.

ENTITÀ	EVENTO	TIPOLOGIA
GIOCATORI	**Passaggi**	**Effettuati**
		Filtranti effettuati
		Ricevuti
		Filtranti ricevuti
		Ricevuti che sono sfociati in goal
		Filtranti ricevuti che sono sfociati in goal
	Cross	**Effettuati**
		Corretti effettuati
		Ricevuti
		Effettuati che sono sfociati in goal
		Ricevuti che sono sfociati in goal
	Tackle	**Fatti**
		Vinti
	Contrasti	**Fatti**
		Vinti

	Dribbling	**Vincenti fatti**
	Tiri	**Totali (nello specchio della porta + fuori)**
		Nello specchio della porta (Goal + parati)
		Goal fatti
		Parati
PORTIERI	**Parate**	**Su tiri dentro area piccola**
		Su tiri dentro area di rigore
		Su tiri da fuori area
		Su rigori
		Su punizioni
STATISTICHE	**Tocchi palla**	**Effettuati prima di passare la palla**
	Distanza	**Percorsa con palla al piede**
		Percorsa senza palla
	Stop	**Riusciti**

Tabella 3.1 - Tipologia di eventi da riconoscere.

3.4 Eventi atomici

Come precedentemente detto, gli eventi atomici sono un tipo di evento che non hanno una durata prolungata nel tempo, ma che sono considerati, nella loro formalizzazione, come un accadimento "istantaneo". Questo vuol dire che questa classe di eventi non possiede un momento di inizio e uno di fine, ma solo una collocazione temporale istantanea nel tempo.

La descrizione di un evento atomico è determinata da parametri e valori che hanno natura fisica. Un esempio molto semplice, ma esplicativo, è quello calcio alla palla (più avanti definito come "*KickingTheBall*"), il quale è definito come il momento in cui un giocatore calcia la palla, ed è associato a una forte accelerazione di quest'ultima. Per la loro natura, dunque, gli eventi atomici sono riconoscibili mediante una analisi delle *feature* fisiche estratte dalle informazioni posizionali della palla e dei giocatori. Come vedremo più avanti, essi fanno da base per il riconoscimento di un'altra categoria di eventi, ovvero gli eventi complessi.

3.4.1 Calcio alla palla

Consiste in un calcio dato da un giocatore alla palla, assumendo che essa sia stata distante al giocatore stesso entro una certa soglia massima. Questa azione può coincidere con un

tiro, l'inizio di un passaggio, di un cross o di altre azioni generiche in cui il giocatore allontana la palla da sé.

La semplice nozione di distanza non è sufficiente a discriminare con successo questo evento da altri, infatti quando il giocatore corre con la palla, le dà dei piccoli calci per farla avanzare, che inevitabilmente porterebbero all'identificazione errata dell'evento. Pertanto, nella sua definizione, sono considerati altre due variabili fisiche, ovvero velocità e accelerazione. Quando la palla viene calciata, subisce un incremento di velocità. Questo incremento però non è istantaneo, ma graduale, quindi consente di dividere quei casi in cui la palla è lenta (come il caso precedente di possesso palla), dai casi in cui si muove a velocità più elevata.

Aggiungere l'accelerazione nella definizione, ha permesso di valutare l'evento sulla base di un parametro che ha un valore istantaneo molto elevato. Infatti, non appena si sta verificando il tiro, la palla ha una accelerazione che è massima, ed è quindi più facilmente distinguibile da quei casi in cui l'accelerazione è bassa (ovvero quando non si tratta, con molta probabilità dell'evento in questione).

3.4.2 Contrasto

L'evento contrasto[20] si verifica quando un giocatore tenta di rubare la palla al giocatore che ne ha correntemente il possesso. In generale un contrasto potrebbe avvenire tra più di due giocatori, ma per semplicità si è considerato solo il caso con due giocatori. Gli elementi che prendono parte a questo evento sono i due giocatori e la palla, i quali devono essere in un raggio spaziale molto piccolo affinché si possa parlare di contrasto.

In particolare, si è deciso di spezzettare un evento come il contrasto (che generalmente ha una durata più o meno estesa), in una sequenza di eventi "contrasto atomico". Quest'ultimo è valutato in ogni singolo frame, e per farlo, si controlla la condizione di distanza per la palla e i due giocatori, individuando in uno di essi l'attaccante (ovvero colui che cerca di ottenere il possesso palla) e l'attaccato (colui che lo deteneva prima dell'inizio del contrasto).

Si può immaginare come la sola informazione posizionale può non essere sufficiente a determinare con precisione chi dei giocatori ha effettivamente il controllo della palla, a causa della dinamica confusionaria intrinseca dell'evento. Infatti, questo si può manifestare in modi e modalità non sempre chiare neanche osservando per via diretta il video dell'azione. La formalizzazione dell'evento prende in esame anche la velocità della palla che deve avere un valore inferiore ad una certa soglia.

[20] Contrasto: in inglese è definito come "Tackle" e in questa tesi viene usata la parola con questa accezione

3.4.3 Deflessione della palla

Questo evento, inizialmente definito come la generica deviazione della palla, mentre questa si muove a grande velocità, da parte di un giocatore, è stato ridefinito come la deflessione operata dal solo portiere. Le ragioni di questa restrizione saranno trattate più avanti. La deflessione viene usata come evento base e di supporto per determinare le parate effettuate dai portieri delle due squadre di calcio.

3.4.4 Goal

Per individuare il goal è necessario determinare l'evento in cui la palla entra nello specchio della porta e lo attraversa. Essendo in possesso di tutte le coordinate spaziali, il compito risulterebbe evidentemente privo di ambiguità e quindi semplice. Nel caso che prenderemo in esame più in avanti, la posizione della palla è fornita delle sole coordinate in x e y, ed è priva dell'altezza da terra (asse z). In assenza della terza coordinata, si inserisce inevitabilmente l'ambiguità dovuta alla possibilità di un tiro finito sopra la traversa. In questo caso infatti, osservando la proiezione a terra della palla, essa risulterebbe dentro la rete, e non al di fuori di essa.

Per ovviare a questa problematica non indifferente (il goal è l'evento più importante di una partita), si potrebbe osservare il resto della traiettoria della palla e verificare se essa stazioni all'interno del rettangolo della rete, o se finisca oltre. Nel primo caso si potrebbe affermare che vi sia stato a tutti gli effetti un goal, mentre nel secondo caso si potrebbe dire con certezza di non averlo trovato.

3.4.5 Palla fuori

Questo evento rappresenta il caso in cui la palla esce fuori dalle linee di gioco, a seguito di un tiro o di qualsiasi altro evento. La palla fuori (o "*BallOut*", come definito più avanti) è stato inserito tra gli eventi atomici perché si considera solo in momento in cui la palla attraversa la linea di gioco, e non tutto il periodo in cui la palla resta fuori dal campo.

3.4.6 Possesso della palla

Il possesso della palla è stato inserito nella categoria degli eventi atomici. In quanto tale, esso presenta una durata equivalente a un solo frame di gioco. Questo vuol dire che tutta quella fase di tempo in cui un giocatore ha la palla vicina ai propri piedi, o ne è l'unico "possessore", è scandita da una serie di eventi atomici di possesso palla.

In questo tipo di evento il giocatore può trovarsi in diversi stati: ad esempio può correre, oppure stare fermo, aver appena ricevuto la palla, oppure essere in procinto di lanciarla. L'evento in sé, dunque, non tiene conto della natura dell'azione che si sta compiendo, ma solo se il giocatore è colui che ne ha il controllo. L'evento è dunque

caratterizzato dall'avere una distanza giocatore-palla piccola, e di non avere interferenze a breve raggio di altri giocatori (cosa che farebbe scaturire un caso di contrasto).

Per evitare che il caso in cui la palla passa sopra un giocatore (ovvero alle stesse coordinate x e y, ma con una z differente) sia annotato erroneamente come un caso di possesso palla, è stato necessario aggiungere un ulteriore controllo fisico sulla palla: si controlla che la velocità della stessa non sia superiore a una certa soglia. Generalmente la palla portata da un giocatore avrà nel caso limite una velocità orizzontale inferiore a quella di una palla che viene lanciata in aria.

3.5 Eventi complessi

L'altra categoria di eventi che è stata determinata in questo lavoro, consiste negli eventi qui definiti come complessi. Questi eventi hanno la caratteristica di essere conseguenza di uno o più eventi atomici e di avere una durata estesa, che non è quindi limitata a un instante temporale isolato. Un evento complesso può quindi iniziare all'accadere di un particolare evento atomico, e concludersi quando avviene un altro evento atomico. A differenza degli eventi atomici, solo alcuni eventi tengono conto di parametri fisici per essere correttamente individuati, parametri che sono comunque contenuti negli eventi atomici consumati dall'evento complesso. Ne è un esempio il passaggio filtrante, che, come vedremo nei prossimi paragrafi, usa l'informazione spaziale inerente al giocatore più esterno della difesa avversaria.

3.5.1 Contrasto

Due giocatori che entrano in conflitto diretto per ottenere il possesso della palla, sono in contrasto. L'evento si riferisce proprio a questo caso, alla fine del quale solo uno dei due giocatori avrà effettivamente il possesso della palla. Il contrasto risulterà vinto se il possesso palla passa da un giocatore all'altro, mentre risulterà perso se chi ha deteneva il possesso palla continua a mantenerlo anche dopo la fine del contrasto. La formalizzazione di questo evento prevede una successione di eventi contrasto atomico. Dunque, ogni sequenza continua di eventi atomici di tipo *Tackle* sarà considerata come un unico contrasto, e dalla valutazione del possesso palla prima e dopo la sequenza, si determinerà il tipo di contrasto, se vincente o meno.

3.5.2 Cross effettuato

Il cross è un tipo di passaggio che avviene dalla fascia verso l'area del portiere. Un cross infatti taglia parte del campo di gioco in modo da dare la palla agli attaccanti affinché essi possano tirare e segnare un goal. Le condizioni di questo evento sono le seguenti: *KickingTheBall* a cui segue un evento *BallPossession*, con la condizione che il tiro sia avvenuto su uno delle due fasce, mentre il possesso palla nell'area centrale della metà

campo avversaria. Per delimitare con precisione le aree interessate, si sono definite delle costanti. Le zone evidenziate in blu sono le fasce a cui si fa riferimento, mentre la zona rossa rappresenta l'area considerata valida per la ricezione della palla per considerare il passaggio un cross valido effettuato. Ovviamente i discorsi fatti per una squadra si invertono in modo speculare per l'altra squadra (e quindi metà campo).

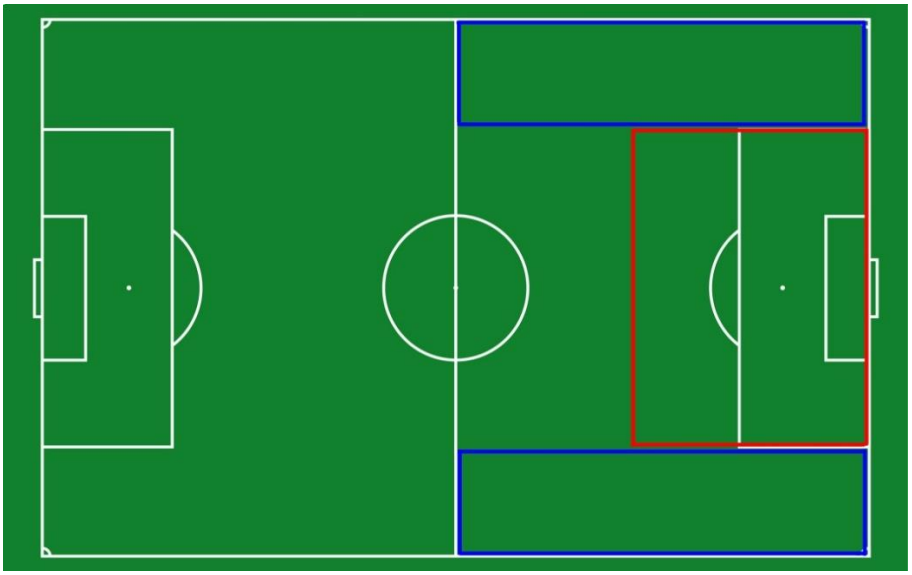


Figura 3.2 - Rappresentazione delle aree di interesse per il cross.

3.5.3 Cross fallito

Il cross fallito è del tutto analogo al cross effettuato, con una sola differenza: il giocatore che riceve la palla deve essere un giocatore avversario. Il resto delle condizioni invece resta invariato.

3.5.4 Cross sfociato in goal

Quando un semplice cross dà vita a un'azione in cui il giocatore che riceve la palla effettua un tiro e segna un goal, si parla di cross sfociato in goal. Questo evento, definito più avanti con il nome di *CrossThenGoal*, è dato da una sequenza di *Cross* che ha avuto successo e uno *ShotThenGoal*.

3.5.5 Passaggio

Il passaggio è l'azione che si verifica quando un giocatore lancia la palla verso un giocatore della propria squadra, e questo la riceve correttamente. È stato allora definito come una sequenza di un calcio alla palla susseguito da un possesso palla, con la condizione di avere entrambi i giocatori coinvolti nei due eventi atomici della stessa

squadra. Un qualsiasi evento che si frappone tra il *KickingTheBall* e il *BallPossession*, deve far cancellare l'evento passaggio, perché non più valido.

3.5.6 Passaggio sfociato in goal

L'evento in questione è quello che nel gergo calcistico viene definito "assist", ovvero un passaggio che ha consentito a un giocatore della propria squadra di segnare. Questo evento, definito più avanti come *PassThenGoal*, è determinato da una sequenza di *Pass* e *ShotThenGoal* (ovvero il tiro sfociato in goal, descritto più avanti).

3.5.7 Passaggio filtrante

Il passaggio filtrante è un particolare caso di passaggio, che si verifica quando il giocatore che riceve la palla ha superato la linea della difesa avversaria. La condizione di emissione è la stessa di un semplice passaggio, ma con l'aggiunta di un ulteriore controllo sulla posizione del giocatore nel momento in cui riceve la palla. Questo significa che un *Pass* sarà anche un *FilteringPass* se l'evento atomico *BallPossession* si verifica quando il giocatore è oltre la linea di difesa avversaria.

3.5.8 Passaggio filtrante sfociato in goal

Analogamente al passaggio sfociato in goal, questo evento è determinato dall'evento finale tiro sfociato in goal. In questo caso, però, l'evento che precede il tiro è un passaggio filtrante.

3.5.9 Tiro fuori

Il tiro fuori è definito come un semplice tiro, ma che termina la sua corsa al di fuori del campo di gioco. La sequenza di eventi atomici che danno vita a un tiro fuori è la successione di un calcio alla palla e un evento palla fuori. Il momento di terminazione dell'evento coincide con l'istante in cui si trova l'evento atomico *BallOut*.

3.5.10 Tiro parato

È il tiro che viene parato da un portiere, che evita quindi un possibile goal. L'evento è definito come la sequenza di eventi atomici tiro e deflessione della palla, dove quest'ultima sia stata però eseguita dal portiere. Un altro caso di tiro parato si ha quando il portiere riesce a bloccare il tiro. L'evento diviene allora una successione di tiro e

possesso palla ad opera del portiere. Come si vede ci sono due possibili strade che portano al medesimo evento.

3.5.11 Tiro sfociato in goal

Equivale al tiro che ha consentito alla squadra di segnare un goal. Viene definito come tiro a cui succede un goal, per cui in termini di eventi atomici la sequenza prevede un *KickingTheBall* seguito da un *Goal*. L'istante in cui si dà per terminato il tiro è il momento in cui la palla supera la linea della porta ed entra in rete.

3.6 Sistema di visualizzazione

Le feature richieste dal sistema sono le seguenti:

1) possibilità di visualizzare direttamente sul video della partita i *bounding box* dei giocatori e della palla con i relativi id, in modo da poterli identificare facilmente;
2) poter osservare nel momento in cui accadono nel video, gli eventi del *ground truth*, con a fianco gli eventi riconosciuti dai detector;
3) spostarsi agilmente di azione in azione per visionare sezioni scelte del dataset.

La realizzazione di un tale sistema richiede l'uso di un framework che consenta di manipolare informazioni video e di generare una interfaccia di interazione per l'utente anche minimale. Tra i vari framework analizzati, il più usato e il più documentato è risultato essere *OpenCV*. OpenCV è una libreria grafica digitale open-source che consente la manipolazione di contenuti multimediali, ma anche di generare UI non troppo complesse. Accanto a questo framework, è stato necessario individuare una libreria per la lettura e la deserializzazione di dati in formato XML, ovvero le informazioni del dataset. La libreria individuata in questo caso è *xml-tree*.

Un ulteriore requisito per realizzare il sistema di visualizzazione è la sincronia tra i dati posizionali dello schermo e delle annotazioni con il video. Poiché il conteggio in *frame* che viene usato internamente dal gioco non è parallelo allo scorrere del tempo, è richiesta l'aggiunta di un parametro temporale ulteriore che indichi proprio lo scorrere del tempo. Il conteggio dei frame è infatti messo in pausa durante determinate fasi del gioco, ad esempio, in seguito ad un fallo laterale o di una qualsiasi altra azione del genere. Anche nel passaggio da un tempo all'altro, il contatore dei frame del gioco viene arrestato, mentre la registrazione del video della partita continua, rendendo impossibile mantenere il sincronismo.

Capitolo 4

Analisi dei dataset disponibili

4.1 Dataset pubblici

La ricerca dei dataset è stata effettuata in primo luogo sul web e tramite i portali di ricerca per articoli accademici. Molti lavori di ricerca, infatti, fanno uso di dataset che però spesso non sono disponibili liberamente. In qualche caso, invece questi erano disponibili con precisi riferimenti scaricabili.

Nei successivi paragrafi verranno presentati quindi i dataset più rilevanti rispetto lo scopo della tesi, in modo da evidenziarne un possibile uso in questo lavoro stesso.

4.1.1 Dataset Alfheim

Il dataset in questione, usato nella ricerca [1] e messo a disposizione in [2], è basato su una serie di tre diverse partite di calcio che si sono svolte nello stadio di Alfheim di Tromso (in Norvegia) nel corso del Novembre 2013. Il dataset è disponibile gratuitamente ed utilizzabile solamente a scopo di ricerca.

I dati sono stati collezionati attraverso dei sensori presenti sulle cinture indossate dai giocatori della sola squadra di casa (11 in totale), il cui segnale è stato raccolto da corrispondenti 11 stazioni radio posizionate intorno allo stadio. Questi sensori sono composti da un accelerometro per ognuno dei tre assi direzionali, un giroscopio, un monitor per la frequenza cardiaca e una bussola. Tramite il flusso di dati è stato possibile tracciare gli spostamenti dei giocatori nell'arco della partita.

Per la rappresentazione dei dati è stato scelto un sistema di coordinate che parte dall'angolo nord-ovest del campo di gioco, le cui dimensioni sono 105 x 68 metri, per cui i valori di x e y sono nell'intervallo [0, 105] per la x e [0, 68] per la y, fatta eccezione dei casi in cui i giocatori sono fuori dai limiti del campo, come ad esempio durante le rimesse laterali. La seguente immagine mostra chiaramente il sistema di coordinate utilizzato:

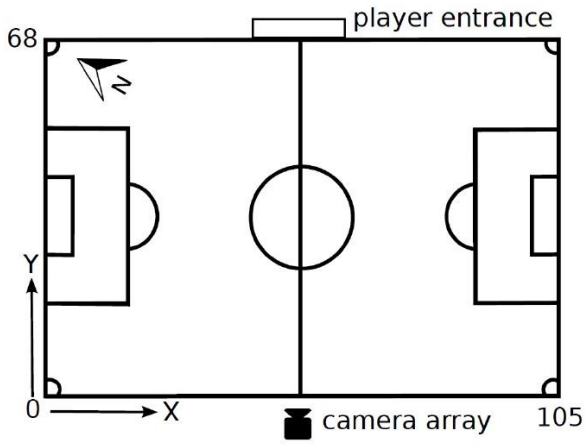


Figura 4.1 - Sistema di coordinate adottato per il dataset.

I dati estratti sono stati memorizzati in *plain* CSV ASCII, disposti uno per riga, con il formato seguente: 'timestamp', 'tag_id', 'x_pos', 'y_pos', 'heading', 'direction', 'energy', 'speed', 'total_distance'

- timestamp (string): espresso in forma 'yy-mm-dd hh:mm:ss'
- tag_id (int): identifica il sensore (l'identità dei giocatori è anonima)
- x_pos (float): posizione relativa all'asse x espressa in metri
- y_pos (float): posizione relativa all'asse y espressa in metri
- heading (float): direzione in cui il giocatore è rivolto espressa in radianti, dove 0 è la direzione dell'asse y
- energy (float): consumo stimato di energia rispetto all'ultimo sample
- speed (float): velocità del giocatore espressa in metri al secondo
- total_distance (float): numero totale di metri percorsi durante la partita fino a quel momento

Oltre ai dati raccolti dai sensori, sono disponibili i dati video delle partite, filmate attraverso due insiemi di telecamere che formano rispettivamente due matrici.

- Il primo insieme è formato da tre telecamere grandangolari poste al centro di uno dei lati del campo. Queste riprendevano la partita contemporaneamente per mostrare le tre diverse parti del campo. I video sono stati processati in seguito alle riprese per rimuovere la distorsione grandangolare.

Figura 4.2 - Immagini dei tre flussi video ripresi dalle rispettive telecamere grandangolari.

- Il secondo insieme è composto da 5 telecamere grandangolari i cui video sono stati processati ed uniti in modo da formare un'immagine che mostri l'intero campo da gioco contemporaneamente, come in Figura 4.3. Quest'immagine panoramica presenta una forte distorsione.

Figura 4.3 - Esempio di immagine processata prodotta dalle 5 telecamere grandangolari.

È possibile ottenere la posizione dei giocatori nel primo tipo di video a partire dai dati estratti dai sensori, calcolando una matrice di trasformazione 3 x 3 da moltiplicare alla posizione XYZ dei giocatori. Questa può essere ottenuta a partire da punti fissi la cui posizione sul campo è conosciuta, come quelli mostrati nella seguente figura:

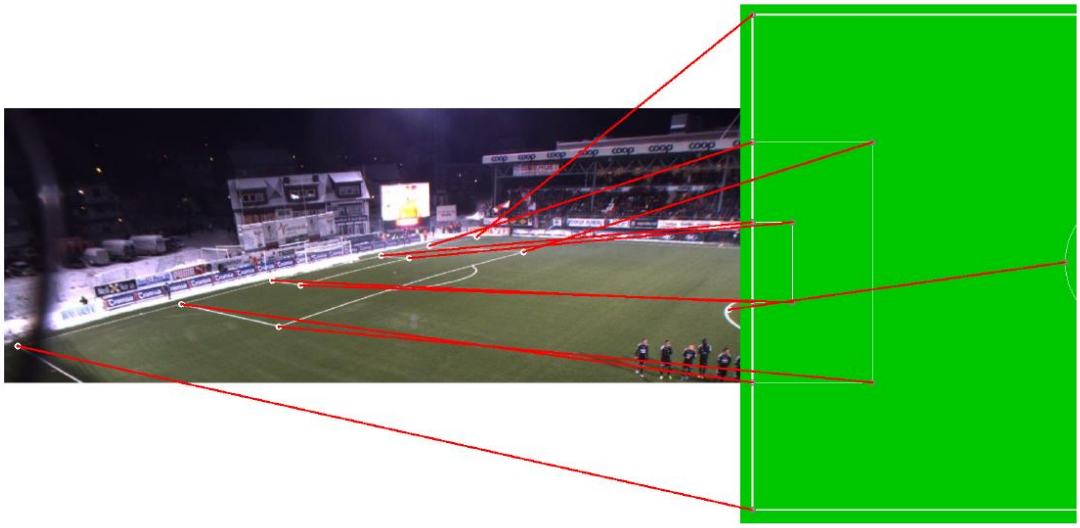


Figura 4.4 - Punti a coordinate note per individuare le posizioni sul video.

Sul sito web sono disponibili 3 dataset: 2 di questi contengono due intere partite per cui sono forniti i tipi di dati sopra descritti, l'altro fa riferimento a 40 minuti all'interno di una sola partita e non contiene il primo tipo di video, le tre viste del campo da cui è stata cancellata la distorsione grandangolare, ma solo il panorama in cui per altro la telecamera è stata avvicinata al campo, aumentandone la distorsione in modo evidente,

come in Figura 4.5. Su questo video panoramico è stata anche manualmente tracciata la posizione della palla per ogni frame, rispetto alle immagini del video.

Figura 4.5 - Immagine prodotta avvicinando la matrice di 5 telecamere al campo.

Questo dataset era potenzialmente ottimo in virtù del suo essere basato su dati raccolti da sensori che quindi rendono disponibili direttamente dei valori (o feature) utili quali la velocità, la direzione di movimento e l'energia. Tuttavia, risultando manchevole di una serie di elementi, ne è stato precluso l'utilizzo nell'ambito del lavoro di questa tesi. Infatti, i dati sono esclusivamente relativi ai giocatori di una sola squadra, impedendo quindi di individuare eventi in cui i giocatori delle due squadre avversarie interagiscono, come ad esempio i contrasti, limitando notevolmente la portata del riconoscimento degli eventi.

Inoltre, la posizione della palla è fornita soltanto in uno dei tre dataset, ed è relativa alla sola posizione sullo schermo in un video che presenta una forte distorsione prospettica. Questo rende necessario operare al fine di correggere la distorsione se si vogliono ottenere dei dati corretti una volta convertiti nel sistema di coordinate xy del campo da gioco. Un'alternativa poteva essere estrarre la posizione della palla dai video che non presentano questa deformazione addestrando ad esempio un object detector, e avendo quindi a disposizione anche quei dati che sono assenti ma richiesti.

4.1.2 Dataset SoccerNet

Il dataset SoccerNet, presentato nell'ambito della ricerca [3], e disponibile sul sito web [2], è stato costruito con lo scopo di individuare le azioni in video di calcio. È composto da 500 partite complete da sei delle principali lege Europee, per una durata totale di 764 ore di gioco. Un totale di 6,637 annotazioni temporali è stato automaticamente derivato da report online di partite di calcio con una risoluzione di un minuto per tre classi principali di eventi (Goal, Cartellino Giallo/Rosso, e Sostituzioni). Queste annotazioni sono state corrette manualmente a una seconda risoluzione associandole a un singolo *timestamp* che segue le ben definite regole del calcio. Con una media di un evento ogni

6.9 minuti, questo dataset si focalizza sul problema della localizzazione di eventi sparsi in video lunghi.

Le peculiarità di questo insieme di dati sono le seguenti:

- Costruzione del dataset in 3 passaggi: (i) raccolta dei video da fonti in rete; (ii) sincronizzazione del tempo della partita con quello del video individuando e leggendo l'orologio mostrato a video; e (iii) lettura dei report disponibili sul web per generare annotazioni temporalmente allineate al video.
- Compilazione di un insieme di 500 partite. Ogni partita è composta da due video non tagliati, ognuno di un tempo di gioco. I video provengono da diverse sorgenti online, usando un insieme di codifiche (MPEG, H264), contenitori (MKV, MP4 e TS), frame rate (25 fino 50 fps), e risoluzioni (SD e Full-HD). Il dataset occupa circa 4 TB, per un totale di 764 ore. Inoltre, le partite sono separate in modo casuale in 300, 100, e 100 partite rispettivamente per il training, validazione e testing assicurando una distribuzione similare degli eventi tra le classi e i gruppi di partite.
- Le annotazioni degli eventi usate in questo dataset sono state ottenute gratuitamente a partire dai report disponibili nei vari siti di leghe di calcio. In questi si riassumono gli eventi principali, a livello di azioni, della partita, con il corrispondente istante di tempo in cui si sono verificati.

A livello di usabilità, questo dataset non è appropriato per il lavoro di questa tesi per le ragioni di seguito esposte. Innanzitutto, i video del dataset derivano da diverse tv broadcast, dunque sono stati catturati a telecamera e inquadratura variabile. Per questa ragione non sono facilmente analizzabili. I dati delle annotazioni, invece sono leggibili seguendo i passi forniti dall'autore, e sono in formato "npy". Il dataset, tuttavia, contiene solamente le annotazioni riguardo 3 eventi, e non contiene i dati posizionali dei giocatori o della palla. Pertanto, anche questo dataset è stato scartato in quanto poco utile ai fini della tesi.

4.1.3 Dataset Issia CNR

Il dataset Issia CNR [5] contiene le posizioni dei giocatori, della palla e dell'arbitro annotate manualmente per due minuti (3000 frame) di partita. Nei 6 file di metadati disponibili, sono dunque presenti i bounding box per tutti gli elementi in movimento nel campo di gioco, ognuno con una etichetta assegnata fissa (uguale nei 6 file). I giocatori di una squadra con etichette che vanno da 1 in poi, mentre per quelli della seconda squadra a partire da 201

I file messi a disposizione sono 6 perché nel raccogliere i video sono state impiegate 6 telecamere disposte in posizioni e angolazioni differenti attorno al campo in modo da riprenderne ogni sezione. Nei primi 300 frame di ogni sequenza non è stato etichettato alcun elemento in modo da fornire una fase di avvio per inizializzare l'algoritmo di sottrazione dallo sfondo.

Le restanti caratteristiche salienti del dataset sono le seguenti:

- Sei visuali sincronizzate acquisite da 6 telecamere Full-HD, tre su ogni lato lungo del campo di gioco, a 25 fps (6 file AVI).
- Sono forniti, insieme ai meri dati posizionali, anche i dati di calibrazione delle telecamere. Precisamente vi si definiscono venti punti forniti in coordinate polari con cui recuperare dalle immagini le posizioni relative agli attori sul campo. In particolare, sono disponibili sei immagini contenenti alcuni punti di riferimento al campo di gioco e alle misure relative per calibrare ciascuna telecamera in un sistema di coordinate sul campo, piuttosto che sullo schermo.

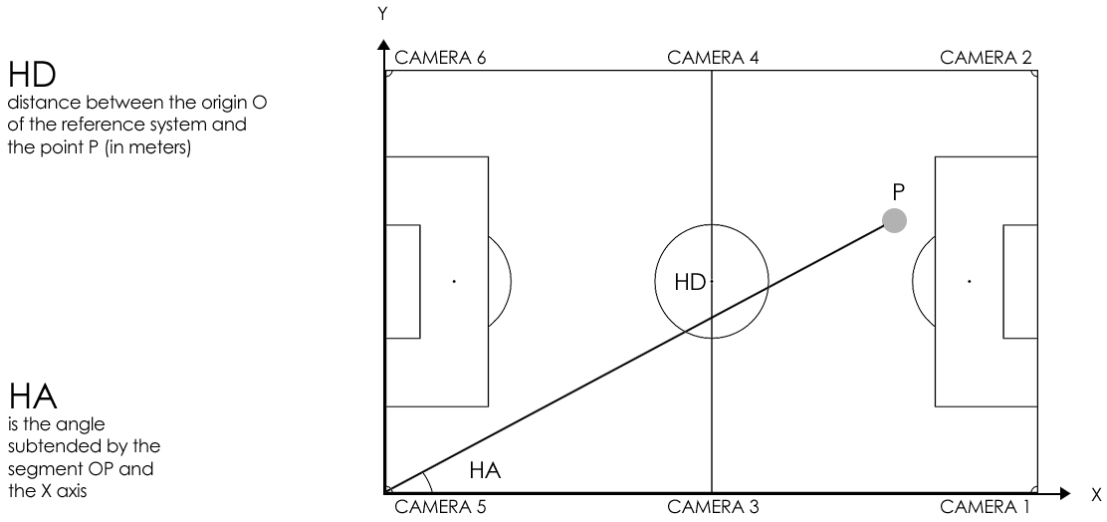


Figura 4.6 - Sistema di riferimento polare adottato con disposizione delle telecamere.

Questo dataset aveva un alto potenziale, in virtù della possibilità di ottenere i dati posizionali, frame per frame, dei giocatori di entrambe le squadre e della palla tramite una conversione da coordinate su schermo in coordinate sul campo. I dati di calibrazione, infatti, sono disponibili assieme ai video e ai bounding box.

Tuttavia, a fronte di un lavoro non indifferente, il dataset finale sarebbe stato di appena 2 minuti, non sufficienti per addestrare un sistema di regole di riconoscimento. Inoltre, questo dataset non presenta le annotazioni sugli eventi che si sono verificati nella sequenza video, che quindi avrebbero dovuto essere annotati successivamente e in modo manuale.

Un potenziale problema per usare questo dataset, infine, poteva derivare dall'estrazione dei metadati e dei file delle posizioni. Il formato dei dati, infatti, non è specificato, e l'estensione "xgtf" usata non è nota.

4.1.4 Magglingen 2013

Questo dataset, disponibile in [6], contiene le posizioni dei giocatori e della palla sul campo di gioco con un passo di 100ms tra un dato e l'altro (10Hz).

Le partite sono state registrate in una situazione di reale competizione da giocatori professionisti svizzeri under 19 di calcio. I dati sono stati intenzionalmente anonimizzati, non per ultimo per l'alta densità di informazioni disponibili. Le registrazioni inoltre sono disponibili dall'inizio della seconda metà fino alla sua fine.

I dati sono così organizzati: vi è un file denominato "Game" che contiene le informazioni sulle posizioni e i metadati della partita (struttura definita "Description"), come ad esempio l'appartenenza di ogni giocatore a una precisa squadra per quell'incontro. I dati posizionali sono organizzati in un array di oggetti JSON denominati "frame", che raccolgono tutte le informazioni dei giocatori e della palla in quell'istante di tempo. Un estratto di esempio si trova in Tabella 4.1:

```
{
    "_id": "vc4ZHCQCgH3DfbvRR",
    "game": "mwHuzM6QKphFDw3Sm",
    "ts": 3239387,
    "data": [
        {
            "id": 5,
            "x": -2.122,
            "y": 22.198,
        },
        {
            "id": 11,
            "x": 12.629,
            "y": 4.808,
        },
        ... all players ...
    ],
    "last_id": "jxNvHNquyK8sMsX5Q"
}
```

Tabella 4.1 - Formato dei dati utilizzato nel dataset magglingen.

Questo estratto mostra un oggetto frame, composto nel seguente modo:

- _id: ID associato al frame, univoco per identificarlo;
- game: identificativo di questa partita;
- ts: timestamp espresso in ms che può essere può essere assoluto o relativo;
- data: array delle posizioni dei giocatori e della palla sul campo;
- data.id: ID univoco associato all'elemento di cui si danno le coordinate. Per la palla il valore dell'ID è sempre 5, mentre gli altri ID trovano le loro corrispondenze nella struttura "Description";

- x,y: coordinate x e y dell'elemento (giocatore o palla) con origine degli assi posto al centro del campo di gioco, ed espresso in metri.

Questo dataset aveva delle buone caratteristiche e una durata accettabile per questa tesi. I dati sono infatti completi per entrambe le squadre. Tuttavia, i dati video non sono disponibili, e non sono neppure stati trovati. Dunque, qualsiasi tipo di annotazione manuale sarebbe risultata impossibile scegliendo questo dataset. Inoltre, i dati posizionali non hanno una origine specificata (se interpolati da video o se ottenuti con qualche sensore), pertanto non è possibile desumere quale possa essere la loro accuratezza.

4.1.5 Dataset KTH Multiview Football

Il dataset KTH Multiview Football racchiude le immagini di giocatori professionisti di una partita di calcio. Il focus di questo dataset è sulla posa dei giocatori, ovvero sulla posizione degli arti e del corpo e non sulla posizione sul campo. Sono disponibili due insieme di dati:

- Dataset 3D
  - 800 frame temporali catturati da 3 visuali (2400 immagini);
  - Le visuali sono calibrate e sincronizzate;
  - 3D "ground truth pose" e matrice ortografica di videocamere per ogni frame;
  - 14 giunti annotati;
  - Due giocatori differenti e due sequenze per giocatore.
- Dataset 2D
  - 5907 immagini;
  - 2D "ground truth pose" per ogni immagine;
  - 14 giunti annotati;
  - 3 giocatori differenti.

Questo dataset è limitato per due ragioni fondamentali: i dati fanno riferimento a delle immagini di una scena particolare e molto breve, e come detto non si hanno a disposizione i dati posizionali dei giocatori e tanto meno della palla. Il dataset, infatti, è utile principalmente per effettuare delle analisi sui giunti, osservando un giocatore per volta. Questo dataset, in somma, non risponde alle esigenze di individuare le azioni di più giocatori che interagiscono tra di loro.

4.1.6 Dataset Soccer arc Multiview Sequence by Nagoya University

Questo dataset, reperibile qui [7], ed usato nel paper [8], contiene una sequenza video di 10 secondi, catturata da 12 telecamere disposte a formare un arco di 180 gradi. Le sequenze video sono catturate al Toyota Stadium con una risoluzione full HD ad una frequenza di 30 frame per secondo. In totale dunque sono presenti 300 frame.

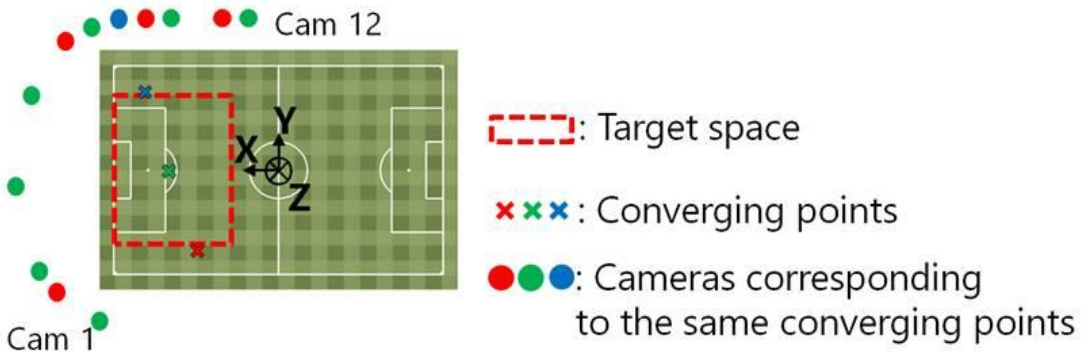


Figura 4.7 - Posizionamento delle telecamere e rispettivi punti focali.

Oltre alle sequenze video, sono disponibili diversi parametri di calibrazione delle telecamere. Tramite questi parametri è quindi possibile mappare una posizione espressa sul video in coordinate relative al campo, secondo il sistema di riferimento mostrato in Figura 4.7.

In particolare, i parametri di calibrazione che sono stati forniti sono i seguenti:

- Matrice intrinseca;
- Vettore di rotazione e la corrispondente matrice di rotazione;
- Vettore di traslazione;
- Coefficienti di distorsione della telecamera.

Figura 4.8 - Esempio di frame di una telecamera che riprende la sequenza video.

Come si sarà intuito questo dataset non ha le peculiarità per essere usato nell'ambito dell'addestramento di un sistema di regole. Infatti, questo dataset non possiede né i dati posizionali relativi ai giocatori e alla palla, né le annotazioni riguardo agli eventi accaduti. Infine, la sequenza ha una durata infima, che non è assolutamente sufficiente per addestrare un sistema.

Questo dataset, come vedremo più avanti nel capitolo 6, è stato usato per confrontare i risultati del sistema di riconoscimento basato su un dataset sintetico rispetto ai risultati ottenuti usando un insieme di dati reali.

4.2 Dataset sintetici

Nessuno dei dataset disponibili fin ora analizzati e presentati soddisfa le peculiarità richieste da un lavoro di addestramento e/o machine learning. Non sono presenti, infatti, i dati posizionali dei giocatori associati ai corrispettivi dati sugli eventi che si sono manifestati durante il gioco. Nella la fase di addestramento, e in seguito di validazione, questi dati sono imprescindibili. Inoltre, la dimensionalità del dataset deve essere sufficiente da garantire generalità e non specificità a pochi casi per il sistema che ne fa uso.

Per ovviare alla mancanza di dati per addestrare il sistema, si è presa in considerazione la possibilità di fare uso di un simulatore di gioco, di modificarlo e di fargli esportare le informazioni necessarie.

In un primo momento sono stati presi in esame diversi simulatori di gioco proprietari, ovvero giochi commerciali, che presentano un motore di gioco e fisico molto evoluti e realistici, ed è stata verificata la possibilità di estrarne i dati, senza dover ricorrere a delle modifiche (per altro non consentite).

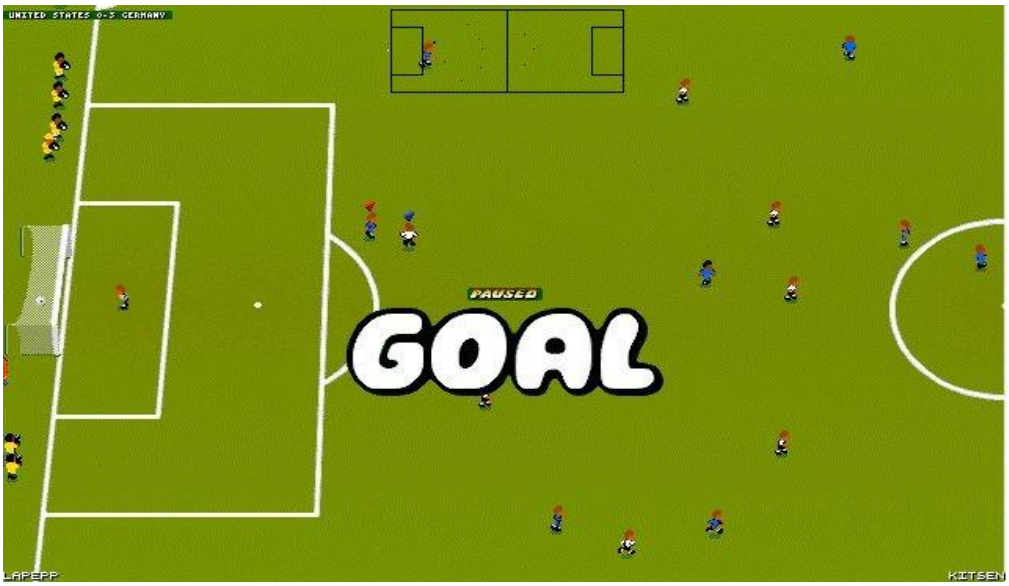
Tuttavia, nessuno dei giochi considerati aveva una tale funzionalità. Pertanto, si è pensato di prendere in esame una serie di simulatori di giochi *open source*, per tentare di modificarne uno e rendere possibile così l'esportazione dei dati.

4.2.1 Giochi open source per generazione sintetica

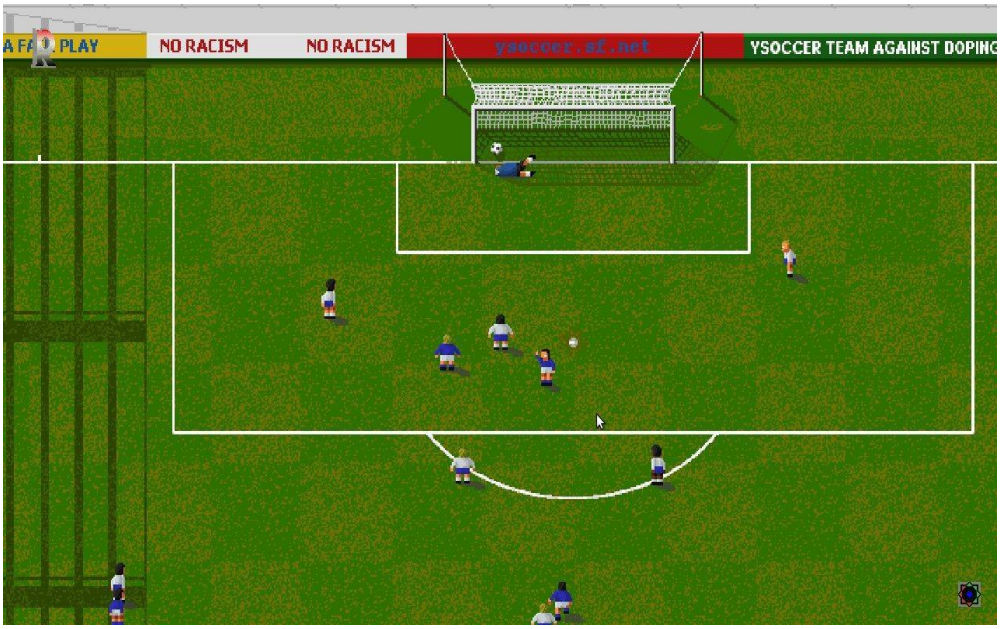
La ricerca di giochi *open-source* ha portato alla luce meno risultati di quelli sperati. Sostanzialmente tutti i giochi trovati, al di fuori di uno, non avevano delle buone caratteristiche in termini di qualità della simulazione, o anche semplicemente a livello grafico. Si menzionano soltanto alcuni di essi come semplice esempio e riferimento:

- **Eat The Whistle**[21]: questo gioco *open-source* è un rifacimento del gioco classico.

[21] Disponibile su http://www.ggsoft.org/etw/



- **Yoda Soccer**[22]: anche in questo caso si tratta di un rifacimento *open-source* di un gioco di calcio classico.



Come si vede anche dalle immagini questi giochi sono dei semplicistici simulatori 2D che non possiedono come detto le caratteristiche ricercate. L'unico gioco *open-source* con un sistema fisico e una grafica realistici è GameplayFootball, sviluppato da Bastian Konings Schuiling. Esso verrà descritto nel prossimo paragrafo.

[22] Disponibile su https://sourceforge.net/p/ysoccer/code/ci/master/tree/

4.2.2 GameplayFootball

GameplayFootball è un gioco multipiattaforma (disponibile sia per Windows che per i sistemi Unix) sviluppato da Bastiaan Konigs Schuiling in C++ e che fa uso a sua volta di un motore grafico denominato *Blunted2* sviluppato anch'esso da Schuiling. Il gioco, sviluppato tra il 2008 e il 2015, presenta un sistema di azione completo e realistico, come si può osservare dalle immagini seguenti. Il motore di gioco è tridimensionale, e presenta un sistema di visione che è del tutto simile a quello di una partita trasmessa su una rete televisiva. Le animazioni, in generale, sono di sufficiente verosimiglianza e l'interazione con i giocatori mossi dall'AI ha una discreta realisticità.

Figura 4.9 - Immagine del gioco GameplayFootball in azione.

Tra le opzioni messe a disposizione dal gioco, c'è la possibilità di giocare in modalità AI vs AI, AI vs giocatore umano oppure giocatore umano vs giocatore umano. Quando vi è un giocatore umano, è possibile optare per l'uso della tastiera come strumento di input, oppure di usare un joystick, o perfino di giocare in due guidando la stessa squadra. Da questo punto di vista GameplayFootball è più che completo.

Il codice, sebbene sia molto lungo e non sempre chiarissimo, è comunque abbastanza ben strutturato e questo fattore, insieme ai precedenti, ha fatto ricadere la scelta su questo gioco.

Le Figure Figura 4.10, Figura 4.11 e Figura 4.12 mostrano alcune delle impostazioni configurabili. Ad esempio, è possibile modificare la camera, nella distanza, nella posizione ecc.

Figura 4.10 - Modifica del valore di zoom per la camera.

Figura 4.11 - Modifica del valore *field of view* per la camera.

Figura 4.12 - Modifica del valore di altezza per la camera.

Le Figura 4.13 invece rappresentano alcune sequenze di gioco, mostrate però in modo ravvicinato durante un replay dell'azione. Come si vede i giocatori sono abbastanza stilizzati e semplificati, ma dal punto di vista posizionale i dati non appaiono così inverosimili.

Figura 4.13 - Immagini ravvicinate del replay di una azione di gioco.

Capitolo 5

Implementazione

In questo capitolo si esporrà, in modo descrittivo, il lavoro svolto, sviluppandolo in tre sezioni differenti. Nella prima sezione si mostrerà il processo di generazione del dataset tramite la modifica di un gioco open-source. La seconda sezione, invece, esporrà il confronto effettuato tra tale dataset, con uno reale al fine di compararne le caratteristiche. Infine, la terza ed ultima parte mostrerà come è stato applicato il sistema di riconoscimento sviluppato nell'ambito della tesi, nei confronti di un dataset reale. Verranno descritti i passi fondamentali e le problematiche riscontrate.

5.1 Generazione dataset con GameplayFootball

Il primo stadio implementato in questo lavoro di tesi consiste nell'impiego e nella modifica del gioco GameplayFootball. Dall'analisi effettuata al Capitolo 4 in merito ai dataset, è emerso che nessun dataset tra quelli disponibili fosse sufficientemente completo e adeguato all'obiettivo di tutto il lavoro. Inoltre, in virtù dei vantaggi di una generazione sintetica descritta in precedenza, si è scelto di utilizzare questo simulatore open-source così da permettere la generazione in via del tutto autonoma il dataset in questione.

5.1.1 Passi preliminari per testare il gioco

Il gioco fa uso di diverse librerie esterne, oltre al motore grafico *Blunted2.* Pertanto, per poter utilizzare il gioco, in modo da testarne le caratteristiche, è stato necessario reperire le librerie che poi il gioco avrebbe caricato all'avvio. Le librerie esterne necessarie sono:

- Boost;
- SDL;
- SDL Image;
- SDL TTF;

- SDL GFX;
- SDL NET;
- SGE;
- OpenGL;
- OpenAL;

Si è usato Linux Ubuntu 18.04 per testare in gioco, potendo sfruttare la maggiore semplicità di installazione delle librerie da riga di comando. Per correggere tutti gli errori di caricamento del loader all'avvio del gioco, sono stati creati tutti i collegamenti simbolici alle varie directory di installazione delle librerie.

5.1.2 Compilazione

Per poter modificare il gioco è stato ovviamente necessario riuscire a compilare i sorgenti. Il gioco è fornito di un set di compilazione che sfrutta il software *CMake*, il quale permette di creare un *makefile*. I passi di compilazione e configurazione sono specificati in un file denominato *CMakeLists.txt*, che è stato modificato in modo da fornire i corretti percorsi di ricerca dei sorgenti delle librerie e dei relativi file ".a" richiesti dal gioco. Il comando lanciato per completare la configurazione è stato: "cmake ." senza ulteriori parametri. A questo punto il gioco è quasi pronto per essere costruito a tutti gli effetti, usando il comando make, il quale si occupa sia di compilare i file sorgente, che di linkare tutti i file oggetto risultanti dalla precedente procedura. In più vengono eseguiti tutti i collegamenti alle librerie statiche, tra cui quelle esterne che sono state inserite in fase di configurazione di CMake, così da avere l'eseguibile.

Per poter completare correttamente la procedura, con la mia configurazione di sistema, e la versione di GCC presente sulla mia macchina, è stato necessario aggiungere il flag "-Wno-narrowing" ai comandi eseguiti dal compilatore GNU per impedire una serie di errori di compilazione. Per poterli aggiungere è stata inserita la stringa in questione in ciascuno dei file *flags.make* generati da CMake.

5.1.3 Analisi della struttura

La struttura del gioco è abbastanza complicata per natura, e distribuita su una grossa quantità di file sorgenti. Il primo passo per comprenderne il funzionamento è stato ovviamente quello di cercare il file main e la funziona main, per capire quali erano i passi operativi logici compiuti dal gioco. In sostanza il main si occupa di far partire il motore grafico, allocare le strutture nella memoria caricando gli eventuali controller e mostrare il menù al giocatore per permettergli di effettuare la propria scelta.

Se la scelta ricade su "Nuova partita", allora viene istanziato un oggetto della classe *Match*, che è la classe fondamentale su cui si basa la logica del gioco. In modo più specifico, questo oggetto a sua volta istanzia a cascata tutti gli "attori" del gioco, ovvero

un oggetto di tipo *Ball* (la palla di gioco), un oggetto di tipo *Referee* (ovvero l'arbitro della partita), due oggetti della classe *Team* (le squadre che si scontreranno). Quando vengono chiamati i costruttori della classe Team, a sua volta vengono invocati i costruttori degli 11 oggetti *Player* fornendogli i dati letti dal database riguardo ai nomi e alle statistiche dei giocatori stessi. Ogni oggetto della classe Player, inoltre, istanzia un oggetto *Humanoid*. Quest'ultimo ha il compito di gestire le animazioni per il componente Player a cui è associato, ed è a questo oggetto demandato l'onere di rappresentare graficamente il giocatore sul campo. La struttura stessa è rappresentata nella **Errore. L'origine riferimento non è stata trovata.** sottostante.

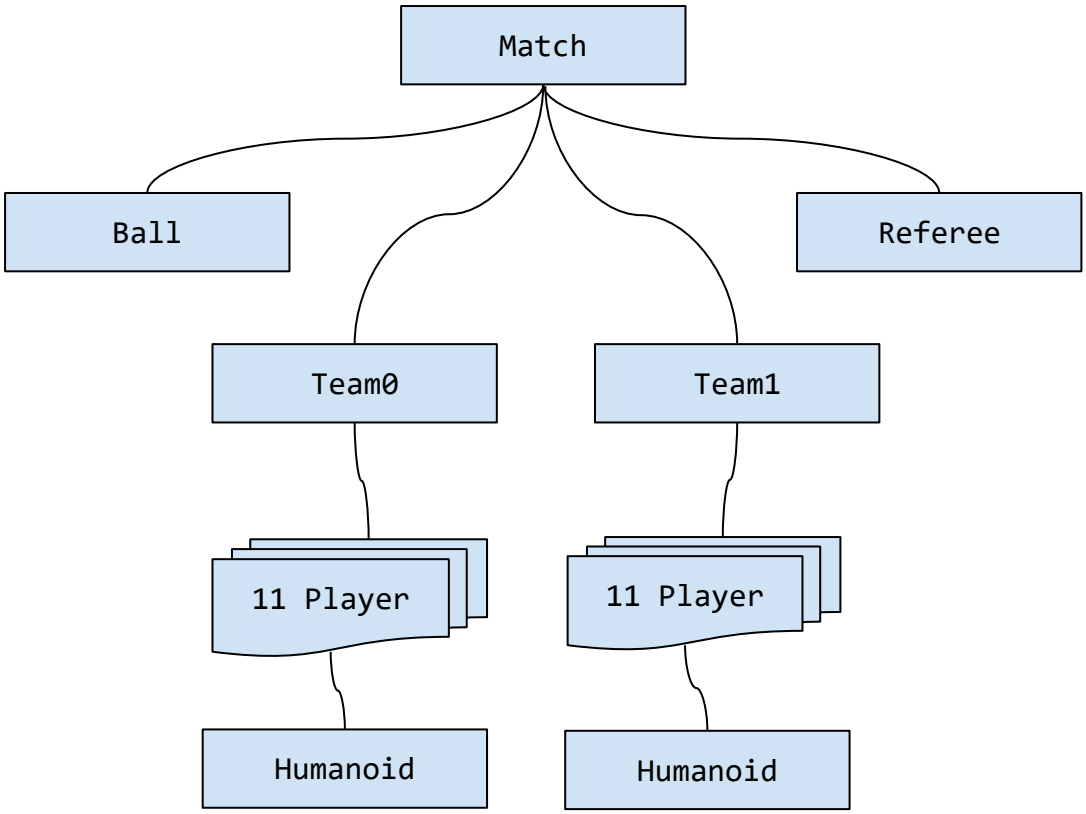


Figura 5.1 - Descrizione della struttura degli oggetti del gioco

Ognuna di queste classi presenta un metodo *Process()*, il quale viene richiamato ciclicamente ad ogni frame del gioco e che si occupa di aggiornare le informazioni dell'oggetto sul quale viene richiamato, in funzione dei cambiamenti che sono avvenuti nel frame corrente. Pertanto, si ha che il metodo *Process()* di Match viene richiamato ad ogni frame, richiamando a sua volta e in successione i metodi *Process()* di *Ball*, *Referee* e dei due *Team*, che infine richiamano quelli di ogni *Player*. In questo modo ognuno di questi componenti della partita può reagire ai cambiamenti di stato degli altri e modellare così la logica del gioco.

5.1.4 Modifiche effettuate al codice sorgente

Questa sezione si occuperà di presentare in modo descrittivo le modifiche apportate al codice sorgente del gioco, al fine di esportare i dati necessari alla generazione di un dataset che rispecchi i requisiti precedentemente formalizzati.

L'idea di base è quella di sfruttare la struttura già presente, in particolar modo i metodi *Process* di ogni classe di interesse, per esportare i dati. In ciascuno di essi, sono disponibili determinati dettagli di ciò che sta accadendo sul campo di gioco, e, poiché nel determinare una certa azione è necessario sapere più dettagli insieme, si è pensato di usare una struttura di *flag* di stato che sono attivati e disattivati in funzione dell'interazione dei vari componenti del gioco. In sostanza la classe *Match* si comporta da orchestratore delle informazioni che arrivano via via dai suoi sottocomponenti e stabilisce, nel ciclo che viene perennemente eseguito durante il gioco, se una data condizione è verificata. Ad esempio, se al momento di un goal il stato relativo l'evento *Shot* è attivo, vuol dire che un evento *ShotThenGoal* si è verificato e quindi i dati associati allo stato sono usati per l'esportazione.

5.1.4.1 Esportazione delle posizioni

Come già detto l'uso del gioco è volto alla costruzione di un dataset sintetico contenente sia le posizioni che gli eventi durante tutta la partita. All'interno del sorgente erano già presenti delle righe di codice atte a riportare su un file testuale le posizioni dei giocatori, ma erano disabilitate usando un valore booleano. La modifica di questo flag e del formato di esportazione ha consentito di ottenere rapidamente la posizione dei giocatori e della palla, ma in coordinate comprese in [0, 110] per la x, e in [0, 72] per la y.

La totalità di questi 23 valori è esportata per ogni singolo frame che viene calcolato dal gioco, corrispondente all'esecuzione del metodo *Process()* dell'oggetto *Match*. Ogni riga del file *positions.log* così generato è nella forma:

<#frame> <ID> <posizione x> <posizione y>

dove il numero frame è un contatore incrementale e ID è il numero identificativo del giocatore o della palla. In particolare, il valore di ID pari a 128 rappresenta la palla, tutti i valori al di sopra di esso rappresentano i giocatori di una squadra, mentre i valori inferiori quelli dell'altra.

5.1.4.2 Esportazione degli eventi

Per esportare i dati delle annotazioni in plain text durante la generazione di una partita, è stata usata una classe definita *Logger*. I dati così prodotti possono essere letti e verificati mentre si simula una partita. L'esportazione vera e propria delle annotazioni in formato *XML*, invece, è stata realizzata usando la libreria di classi definita in precedenza. Si interagisce con questa libreria mediante una interfaccia, passando di volta in volta i dati dell'evento che si vogliono esportare.

Questi sono gli eventi che sono esportati dal gioco in XML mediante le modifiche descritte più avanti, e attraverso l'interfaccia di esportazione presentata al prossimo paragrafo.

TIPOLOGIA	EVENTO	DATI ASSOCIATI
ATOMICO	**BallOut**	**Frame**
	Goal	**Frame, id giocatore, id squadra**
	Tackle	**Frame, id giocatore, posizione, id squadra dell'effettuante e della vittima**
	KickingTheBall	**Frame, id giocatore, id squadra, posizione**
	BallDeflection	**Frame, id giocatore, id squadra, posizione**
	BallPossession	**Frame, id giocatore, id squadra, posizione, id giocatore avversario più vicino alla propria porta**
COMPLESSO	**Pass**	**Frame inizio, frame fine, id giocatore, id giocatore ricevente, id squadra**
	PassThenGoal	**Frame inizio, frame fine, id giocatore, id giocatore ricevente, id squadra**
	FilteringPass	**Frame inizio, frame fine, id giocatore, id giocatore ricevente, id squadra**
	FilteringPassThen Goal	**Frame inizio, frame fine, id giocatore, id giocatore ricevente, id squadra**
	Cross	**Frame inizio, frame fine, id giocatore, id giocatore ricevente, id squadra**
	CrossThenGoal	**Frame inizio, frame fine, id giocatore, id giocatore ricevente, id squadra**
	Tackle	**Frame inizio, frame fine, id giocatore, posizione, id squadra dell'effettuante e della vittima**
	Shot	**Frame inizio, frame fine, id giocatore, id squadra**
	ShotOut	**Frame inizio, frame fine, id giocatore, id squadra**
	ShotThenGoal	**Frame inizio, frame fine, id giocatore, id squadra**
	SavedShot	**Frame inizio, frame fine, id giocatore, id giocatore ricevente, id squadra, id portiere, id squadra portiere, posizione tiro**

Tabella 5.1 - Informazioni associate a ciascun tipo di evento.

5.1.4.3 Interfaccia di esportazioni XML degli eventi

L'interfaccia consente di richiamare i metodi della classe *Annotation* in base all'evento che si sta generando. *Annotation* funge da accumulatore degli eventi effettuandone la

scrittura su disco al termine dalla partita. Precisamente vengono generati due file, uno per gli eventi atomici, ed uno per i complessi, denominati rispettivamente *Annotations_AtomicEvents.xml* e *Annotations_ComplexEvents.xml*. L'interfaccia espone i metodi riportati nella Tabella 5.2.

```
// Eventi atomici

void Annotation::KickingTheBall(int frame, string playerId, string teamId,
double x, double y);

void Annotation::BallPossession(int frame, string playerId, string teamId,
double x, double y, string outermostOtherTeamDefensivePlayerId, string
team2Id);

void Annotation::Tackle(int frame, string player1Id, string team1Id, double
x1, double y1, string player2Id, string team2Id, double x2, double y2);

void Annotation::BallDeflection(int frame, string playerId, string teamId,
double x, double y);

void Annotation::BallOut(int frame);

void Annotation::Goal(int frame, string scorerId, string teamId);


// Eventi complessi

void Annotation::Pass(int startFrame, int endFrame, string senderId, string
teamId, string receiverId);

void Annotation::PassThenGoal(int startFrame, int endFrame, string
senderId, string teamId, string scorerId);

void Annotation::FilteringPass(int startFrame, int endFrame, string
senderId, string teamId, string receiverId);

void Annotation::Cross(int startFrame, int endFrame, string senderId,
string teamId, string receiverId, bool outcome);

void Annotation::CrossThenGoal(int startFrame, int endFrame, string
senderId, string teamId, string scorerId);

void Annotation::Tackle(int startFrame, int endFrame, string victimId,
string teamId, string tacklerId, bool outcome);

void Annotation::SlidingTackle(int startFrame, int endFrame, string
victimId, string teamId, string tacklerId, bool outcome);

void Annotation::Shot(int startFrame, int endFrame, string shooterId,
string teamId);

void Annotation::ShotThenGoal(int startFrame, int endFrame, string
shooterId, string teamId);

void Annotation::SavedShot(int startFrame, int endFrame, string shooterId,
string teamId, string goalkeeperId, string team2Id, double x, double y,
SavedShotCase sscase);
```

Tabella 5.2 - Interfaccia C++ della classe Annotation.

Oltre all'esportazione dei due file relativi agli eventi atomici e complessi, viene esportato anche un terzo file, denominato *Annotations_AtomicEvents_Manual.xml* che riporta alcuni metadati della partita, i goal che si sono verificati e tutti i fuorigioco dell'incontro. I metadati sono: gli id dei portieri delle due squadre e il frame di inizio del

secondo tempo della partita. I goal sono esportati per ovviare all'assenza dell'asse z della palla, che rende difficile determinare se la palla è entrata effettivamente a rete oppure è passata al di sopra di essa. Infine, i fuorigioco sono stati inseriti in una fase avanzata dello sviluppo, per consentire di individuare i momenti della partita in cui il gioco è fermo. I fuorigioco sono stati inseriti in questo file in quanto non sono oggetto di riconoscimento da parte del sistema, ma sono dati per noti a priori.

5.1.4.4 Esportazione degli eventi atomici

Sfruttando un controllo già inserito dall'autore del gioco, è stato possibile esportare sul file testuale l'evento atomico *Goal,* e da successiva verifica è risultato essere correttamente presente su file, insieme ad altre informazioni come il frame a cui l'evento è avvenuto, l'ID del giocatore che ha segnato e l'ID della squadra del giocatore.

L'evento *Tackle* (inteso in senso di contrasto) è stato esportato usando un'altra sezione di codice del metodo *Process()* della classe *match,* in cui l'autore distingue chi subisce il contrasto da chi lo effettua, per scatenare l'esecuzione dell'animazione corretta. In questo contesto è stata aggiunta la chiamata all'interfaccia di esportazione, determinando i parametri a partire dagli oggetti *player* coinvolti nell'azione. In questo modo si riporta nel file xml chi ha subito il contrasto, chi lo ha effettuato, in quale frame e il risultato dello stesso, ovvero se il contrasto sia riuscito o meno.

Per valutare l'evento *BallPossession* è stato utilizzato il valore restituito da un metodo della classe *Player* denominato *HasUniquePossession()* che specifica se il giocatore è l'unico possessore della palla (cosa non vera se più giocatori si trovano ad una distanza inferiore di un certo valore). Valutando il metodo per ogni giocatore una volta per frame, e verificando se per almeno uno di essi il valore restituito fosse *true,* si è esportato sul file degli atomici l'evento. I dati riportati sono i valori dell'ID del giocatore stesso, della sua squadra e il frame in cui è avvenuto il possesso palla in questione. Talvolta il gioco considerava più di un giocatore come possessore della palla. Per evitare l'esportazione multipla si è inserito un controllo sul numero di giocatori considerati in possesso. Nell'eventualità che questi fossero due e appartenenti a squadre avversarie, si è proceduto con l'esportazione di un *Tackle*. Dai dati posizionali risultava, infatti, una vicinanza elevata di entrambi i giocatori con la palla.

BallOut è stato esportato partendo dalla classe *Referee.* Questa classe, nel suo metodo *Process(),* effettua le verifiche sui possibili falli che sono accaduti nel frame corrente. Mediante un controllo sulla posizione della palla, si controlla che essa sia o meno entro i limiti del campo. Nel caso ci fosse un superamento della linea di gioco, si va ad effettuare una chiamata all'interfaccia di esportazione, riportando così i dati dell'evento sul file.

L'esportazione dell'evento *BallDeflection* è stata completata inserendo nell'iterazione che avviene nella classe *Match* per ogni frame, sfruttando una parte di codice già presente. Questa sezione determina l'animazione da lanciare in caso di deflessione della palla. In un primo momento la deflessione veniva calcolata ed emessa per ogni giocatore sul campo. In seguito, per necessità legate alle performance di riconoscimento degli eventi, l'emissione dell'evento deflessione palla è stato limitato ai soli portieri. Dunque, nel

controllo che viene fatto su ogni giocatore, si accerta anche il suo ruolo, in modo da decidere se emettere effettivamente la deflessione della palla o meno.

L'evento *KickingTheBall* è stato esportato modificando il codice presente nella classe *Humanoid*. Questa classe, che è incorporata in una istanza della classe *Player*, si occupa di gestire le animazioni e gli eventi che vengono ricevuti dal controller del giocatore: quando il giocatore preme un pulsante, l'azione associata viene inserita in una coda che viene consumata ad ogni ciclo. A questo punto la classe *Humanoid* interviene nel suo metodo *Process()* decidendo in base al tipo di azione richiesta, che tipo di animazione effettuare sul player associato. Pertanto, è stata attivata l'esportazione dell'evento ogni qual volta l'animazione richiesta fosse stata del tipo *ShortPass*, *LongPass*, *HighPass* o *Shot*, ovvero tutte le tipologie di azione in cui si verifica il calcio alla palla.

5.1.4.5 Esportazione degli eventi complessi

Gli eventi complessi sono caratterizzati dall'avere un momento di inizio e uno di fine. Pertanto, ciascuno di essi è accompagnato dall'informazione del frame iniziale e di quello finale.

Per annotare l'evento *Pass* sono state aggiunte tre variabili globali dell'oggetto match, in modo da essere accessibili facilmente da tutti gli altri oggetti a cui ne era noto il riferimento. La prima variabile è un valore booleano che indica se al momento attuale un passaggio è partito o meno. Questa viene impostata al valore *true* non appena avviene uno tra i seguenti eventi: *ShortPass*, *LongPass*, *HighPass*. A quel punto vengono salvati il frame corrente (frame di inizio passaggio) e il riferimento all'oggetto *Player* che lo ha fatto partire. Se durante il metodo *Match::Process()* uno dei giocatori diventa il possessore della palla e il valore booleano della variabile è a *true*, allora il giocatore in questione è il ricevitore del passaggio e l'evento *Pass* può essere annotato. In questa eventualità, e in tutti i casi in cui la palla non finisce in possesso a un compagno di squadra, il valore booleano viene resettato. Ciò significa che in caso di inizio o fine di un altro evento, invalida l'emissione del passaggio. Non è stato inserito un timeout, in quanto non c'è modo con cui la palla passi a un giocatore della stessa squadra in modo indiretto (ad esempio dopo un un tiro), senza che muti lo stato per via di altri eventi che intercorrono.

Nella Figura 5.2 è visibile un diagramma schematico del processo di emissione degli eventi di tipologia passaggio, a cui appartengono anche il cross e il passaggio filtrante, descritti poco più avanti.

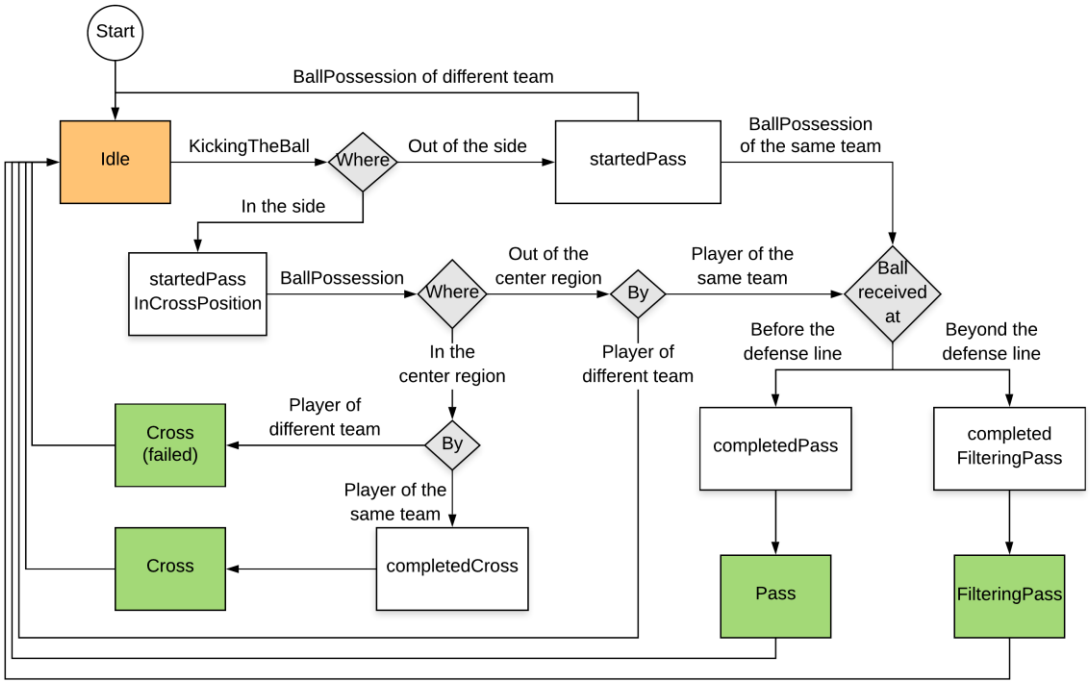


Figura 5.2 - Rappresentazione schematica dell'emissione della famiglia dei passaggi.

L'evento *PassThenGoal*, che indica se un passaggio sfocia in un goal, è verificato sfruttando il precedente evento *Pass*. In particolare, se un passaggio viene completato correttamente, un *flag,* denominato *completedPass*, viene impostato al valore *true*. A questo punto se lo stesso giocatore che ha ricevuto la palla a seguito del passaggio, nei frame successivi effettua un tiro e segna un goal, l'evento può essere annotato. Se nel frattempo avvengono altri eventi che invalidano questa condizione, il valore di *completedPass* è resettato, impedendo lo scaturire dell'evento su file.

FilteringPass è verificato solo quando vi è un *Pass* semplice avviato ma non ancora terminato. Quando la palla va in possesso a un giocatore, viene valutata la posizione di tutti i giocatori in un ciclo, e si trova la maggiore (o la minore, in base a qual è la metà campo avversaria) escludendo il portiere. Se il giocatore che sta più a destra o più a sinistra è il ricevitore della palla, l'evento è un *FilteringPass*. In questa eventualità, viene annotato sia l'evento *Pass* che il *FilteringPass*. Bisogna notare come le condizioni di passaggio filtrante sono facilmente replicate in condizione di fuorigioco. Per evitare allora che venga emesso un passaggio filtrante poco prima che venga fischiato dall'arbitro il fuorigioco. Per farlo si controlla la posizione del giocatore che sta per ricevere la palla poco all'istante in cui parte il passaggio. Se il giocatore è più avanti della linea difensiva, allora il passaggio filtrante non viene emesso.

Analogamente all'evento *PassThenGoal*, anche il *FilteringPassThenGoal* viene rilevato solo a condizione che in precedenza sia accaduto un particolare evento, il *FilteringPass* in questo caso. Se il valore della variabile *completedFilteringPass* è uguale a *true* e il giocatore che segna è lo stesso che ha ricevuto la palla, allora questo evento viene esportato, insieme al più semplice *PassThenGoal*. Uno schema della successione di condizioni per la famiglia dei passaggi sfociati in goal è presente in Figura 5.3.

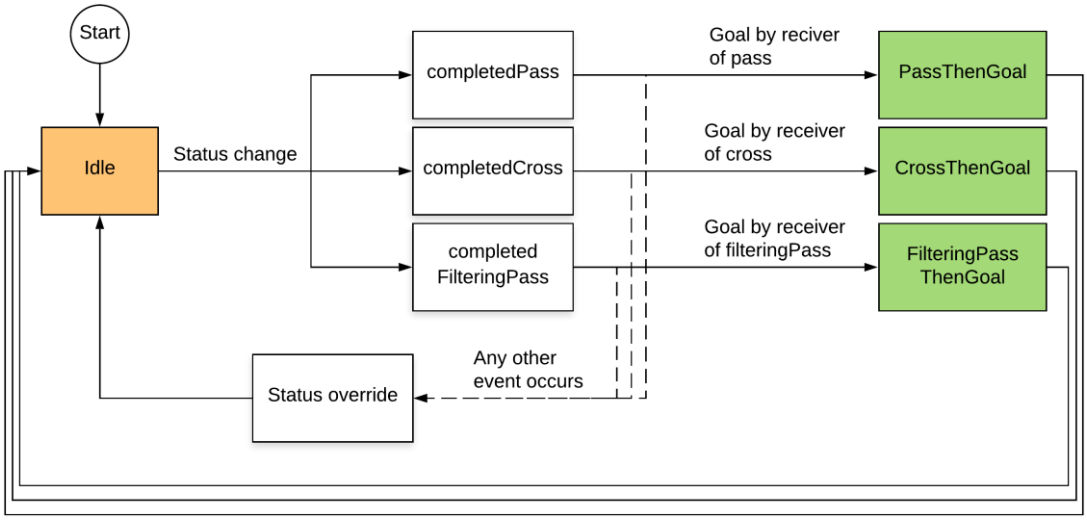


Figura 5.3 - Rappresentazione schematica della famiglia dei passaggi sfociati in goal.

Il *Cross* è definito considerando il punto dal quale parte un passaggio. Si controlla all'inizio del passaggio se il giocatore si trova nella metà campo avversaria e sulla fascia (la cui ampiezza è definita da alcune costanti numeriche). Se queste condizioni sono verificate con successo, allora viene impostato al valore *true* una variabile booleana, *startedPassInCrossPosition*. Quando un giocatore della stessa squadra diventa il possessore della palla (verificato nel solito ciclo di *Match::Process()*), se egli si trova nella posizione denominata come area di ricezione cross (definita anch'essa mediante costanti numeriche) e *startedPassInCrossPosition* ha valore *true*, allora si tratta di un cross riuscito, e in quanto tale viene annotato. Se la squadra del giocatore che riceve la palla, invece, non è la stessa di chi ha iniziato il cross, allora viene annotato un evento *Cross*, ma con il valore di *outcome* a *false* (indicando, dunque, un cross fallito).

L'evento *CrossThenGoal* viene controllato solo quando in precedenza è accaduto un cross riuscito. In tal caso, la variabile *crossCompleted* avrà valore *true* e se il giocatore che ha ricevuto il cross è colui che segna un goal, questo evento sarà esportato sul file. Anche in questo caso il più elementare *PassThenGoal* (di cui *CrossThenGoal* è una specializzazione) verrà annotato. Un possibile caso è quello di un goal successivo a un passaggio che verifica sia le condizioni di un cross che di un passaggio filtrante. In questa situazione, il sistema annoterà i tre eventi contemporaneamente.

L'evento complesso *Tackle* racchiude, come detto, una serie di eventi atomici *Tackle*. Al primo frame in cui viene annotato un evento tackle atomico, il flag *startedTackle* viene impostato a *true* e vengono salvati i riferimenti agli oggetti *Player*, (attori nell'azione) e il frame di inizio. Fintanto che la serie di eventi atomici non termina, non viene emesso alcun evento *Tackle* complesso. Quando la serie è interrotta, viene allora annotato l'evento complesso, con inizio e fine corrispondenti al primo e all'ultimo dei Tackle atomici della serie. Più in dettaglio, il valore del flag *stillInTackle* indica lo stato della serie di Tackle atomici. Se a un dato frame si ha *stillInTackle* uguale a *false* e *startedTackle* pari a *true,* allora viene annotato il Tackle complesso, riportando i dati che in precedenza erano stati memorizzati nelle variabili. Il successo del Tackle è determinato

invece dal giocatore che aveva il possesso prima e dopo la serie di Tackle atomici. Se il possesso cambia, allora il Tackle ha avuto successo.

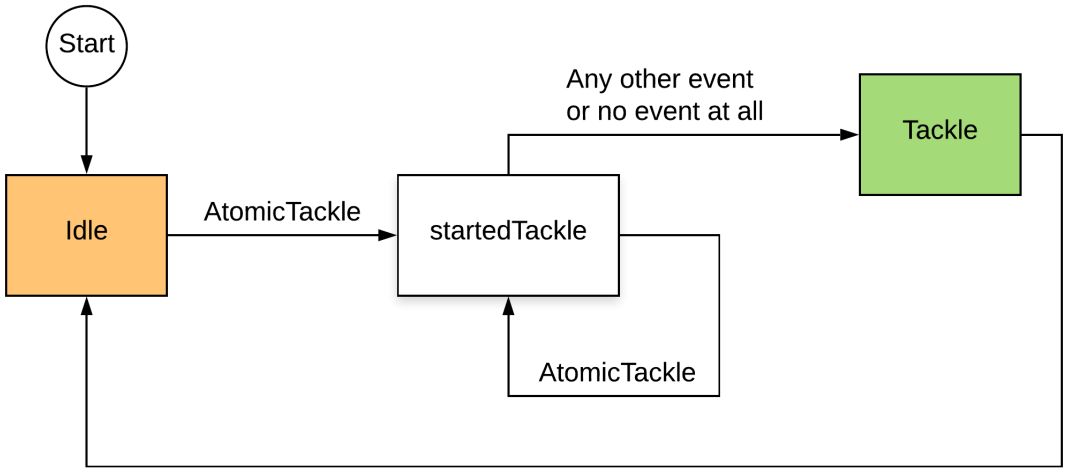


Figura 5.4 - Rappresentazione dei passaggi per l'emissione dell'evento complesso tackle.

Shot è annotato ogni qual volta si verifica un tiro. In particolare, quando un oggetto di tipo *Humanoid* invoca l'animazione di un tiro, viene impostato a *true* il flag *startedShot*, e vengono salvati il riferimento al giocatore e al frame di inizio tiro. Per la condizione di terminazione dell'evento, si verifica il successivo evento. Ad esempio, in caso di *BallPossession* il tiro ha necessariamente terminato il suo corso. Si controlla che la condizione di tiro iniziato sia vera, e in tal caso si annota l'evento *Shot* con frame finale pari all'inizio del successivo evento.

L'evento *ShotThenGoal* è annotato quando, a seguito di un goal, il flag *startedShot* è *true* e il giocatore che ha effettuato il tiro è lo stesso che è segnato. Vengono salvati su file come frame di inizio, quello di inizio del tiro, mentre come fine quello del goal.

Un evento più specifico per il tiro, è lo *ShotOut*, ovvero il tiro che finisce fuori dalla linea di fondo campo. Per valutare questo evento si fa uso dei controlli effettuati nel metodo *Process()* nell'oggetto *Referee*. Se la palla termina fuori e il flag *startedShot* ha valore *true* allora viene scritto sul file l'evento, riportando i dati di inizio tiro, il frame di uscita della palla, e i dati del giocatore che ha tirato.

SavedShot è l'evento tiro parato, e anche questo è un sotto caso del tiro (*Shot*). Un tiro parato ha diversi modi di avvenire, ognuno con delle condizioni specifiche. Il caso tipico prevede la deflessione della palla ad opera del portiere, preceduta da un tiro di un giocatore avversario. In tal caso vengono salvati i riferimenti dei giocatori in azione e il frame della parata. Inoltre, viene settato al valore *true* la variabile *startedSavedShot*. I tre casi possibili sono i seguenti:

1) *startedShot* con valore *true* (quindi quando è partito un tiro) e possesso della palla al portiere della squadra avversaria (in queste condizioni il portiere ha parato la palla riuscendo a bloccarla, e non c'è alcuna *BallDeflection*);

2) *startedSavedShot* con valore *true*, e il possesso palla a un qualsiasi altro giocatore (a seguito della parata la palla è finita in possesso ad un altro giocatore, c'è stata una *BallDeflection*);

3) *startedSavedShot* con valore *true*, e palla terminata fuori dalle linee di gioco (portiere para la palla e la manda fuori campo).

In tutti questi casi vengono annotate tutte le informazioni sui giocatori coinvolti (portiere e giocatore che tira), sul frame di inizio e quello frame di fine (momento della parata) e sul luogo in cui è partito il tiro (da fuori area, nell'area grande o in quella piccola).

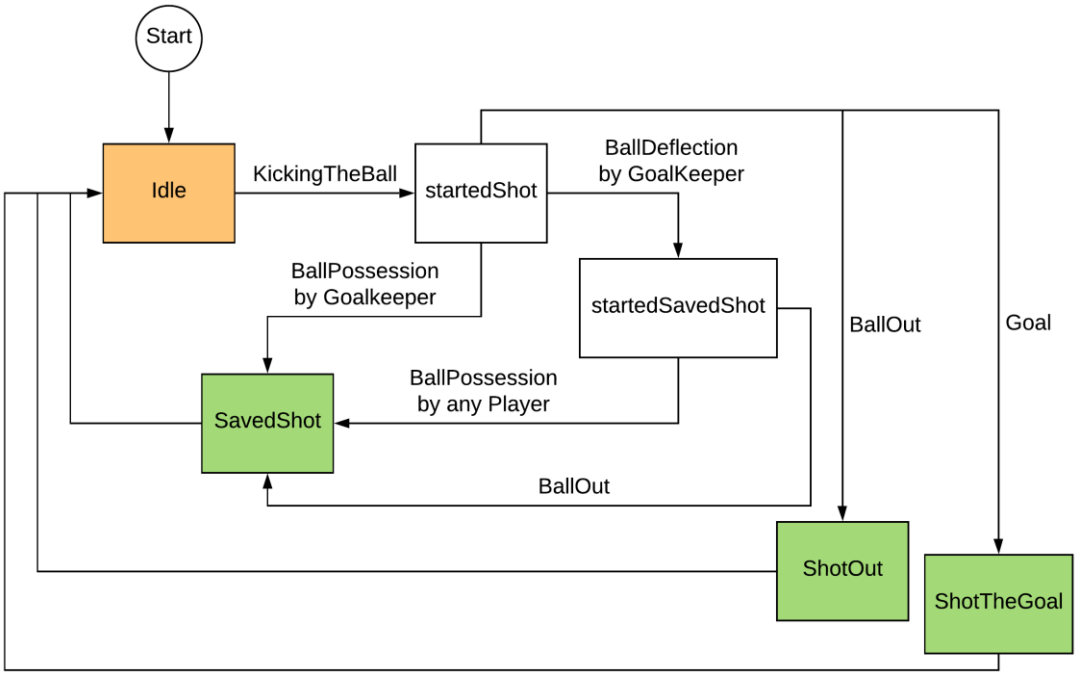


Figura 5.5 - Rappresentazione schematica dell'emissione della famiglia degli Shot.

5.1.5 File generati e loro formato

Durante ogni partita vengono generati automaticamente una serie di file XML che costituiscono le annotazioni vere e proprie, un file video, che racchiude tutti i minuti di gioco, un file testuale contenente le posizioni dei giocatori e della palla e infine un file testuale che contiene le annotazioni in modo leggibile e discorsivo. A questi file poi viene aggiunto un ulteriore file di informazioni scritto a mano che contiene alcuni dati inerenti alla partita e ai dati ad essa associati.

1. Annotations_AtomicEvents.xml
2. Annotations_AtomicEvents_Cvat.xml
3. Annotations_AtomicEvents_Manual.xml
4. Annotations_ComplexEvents.xml
5. Info
6. positions.log
7. Textual_annotations

8. Video.mkv

5.1.5.1 File delle posizioni

Il file *positions.log* contiene la posizione, frame per frame, di tutti i giocatori e della palla per la durata dell'intera partita. Il formato usato è il seguente:

#frame #ID x_pos y_pos

dove #frame è un numero intero, #ID è un intero pari a 128 nel caso della palla, maggiore di 128 se si tratta di un giocatore della prima squadra, minore di 128 se è un giocatore della seconda squadra. Infine, x_pos e y_pos rappresentano le coordinate della posizione del giocatore o della palla in un range compreso tra [0, 0] e [110, 72].

Qui di seguito vi è una rappresentazione di esempio che descrive le posizioni assunte dai giocatori e dalla palla al frame 1.

```
1 128 55.000000 36.000000
1 39 1.099998 36.000000
1 40 31.790001 54.143288
1 41 27.500000 40.067329
1 42 27.500000 32.015499
1 43 31.790001 18.496141
1 44 55.000000 36.299999
1 45 41.832119 30.908035
1 46 50.930000 55.328522
1 47 55.000000 32.599998
1 48 50.930000 14.562845
1 49 52.317490 45.009113
1 185 108.900002 36.000000
1 186 78.209999 21.363310
1 187 82.500000 31.270725
1 188 82.500000 40.920780
1 189 78.209999 52.884415
1 190 65.131714 26.550495
1 191 69.716583 35.645824
1 192 65.131714 45.931969
1 193 60.060001 19.287315
1 194 59.151516 27.566441
1 195 60.060001 55.894096
1 195 60.060001 55.894096
```

Tabella 5.3 – Estratto del file che riporta le posizioni nella partita.

5.1.5.2 Annotazione degli eventi atomici

Il primo file, ovvero *Annotations_AtomicEvents.xml*, contiene tutta la lista di eventi atomici che sono generati dal gioco. Gli eventi che è possibile individuare in questo file sono i seguenti:

- KickingTheBall;
- BallPossession;
- Tackle;
- BallDeflection;
- BallOut;
- Goal;
- Foul;
- Penalty.

Questi eventi contengono un solo riferimento temporale, ovvero il frame a cui sono stati individuati, oltre alle informazioni tipiche di ogni particolare evento. Di seguito viene riportato un esempio circa la rappresentazione in formato dati XML per l'evento KickingTheBall.

```
<track id="168741" label="KickingTheBall">
    <box frame="281161" keyframe="1" occluded="0" outside="0"
        xbr="1.24157999999999993e+03"
        xtl="1.20052999999999997e+03"
        ybr="1.68740000000000009e+02"
        ytl="9.17900000000000063e+01">
            <attribute name="playerId">47</attribute>
            <attribute name="teamId">0</attribute>
            <attribute name="x">54.9629</attribute>
            <attribute name="y">35.7232</attribute>
    </box>
</track>
```

Tabella 5.4 - Estratto di un possibile file di annotazione di eventi atomici.

Il secondo file, *Annotations_AtomicEvents_Cvat.xml*, è fondamentalmente una copia del primo, ma con una serie di attributi in meno nella struttura XML, i quali sono stati rimossi per renderlo compatibile con il software *Cvat* (utilizzabile per visionare ed eventualmente correggere i "bounding box" degli eventi esportati sul video).

5.1.5.3 Annotazione degli eventi complessi

Annotations_ComplexEvents.xml contiene tutti gli eventi complessi che si sono verificati nella partita con lo stesso formato XML. Gli eventi che vi si possono trovare sono i seguenti:

- Pass;
- PassThenGoal;
- FilteringPass;
- FilteringPassThenGoal;
- Cross;
- CrossThenGoal;
- Tackle;
- SlidingTackle;
- Shot;
- ShotOut;

- ShotThenGoal;
- SavedShot.

Questi eventi, a differenza degli atomici, sono corredati da due riferimenti temporali: il frame di inizio e il frame di fine dell'evento. Qui di seguito un esempio per l'evento Cross che intercorre tra il frame 212757 e 212835. Si osservi come l'elemento *track* comprenda una lista di elementi di tipo *box* contenenti le informazioni su ogni singolo frame dell'evento. L'elemento track, che indica il tipo di evento, racchiude tanti box quanti sono il numero di frame in cui si svolge l'evento complesso. Ogni box è dunque la rappresentazione di ogni frame, di cui indica le informazioni dell'evento.

```
<track id="545" label="Cross">
    <box frame="212757" keyframe="1" occluded="0" outside="0"
        xbr="1.24157999999999993e+03"
        xtl="1.20052999999999997e+03"
        ybr="1.68740000000000009e+02"
        ytl="9.17900000000000063e+01">

        <attribute name="sender">66</attribute>
        <attribute name="teamId">1</attribute>
        <attribute name="receiver">41</attribute>
        <attribute name="outcome">false</attribute>
    </box>
    <box>
    ...
    </box>
    <box frame="212835" keyframe="0" occluded="0" outside="0"
        xbr="1.24157999999999993e+03"
        xtl="1.20052999999999997e+03"
        ybr="1.68740000000000009e+02"
        ytl="9.17900000000000063e+01">

        <attribute name="sender">66</attribute>
        <attribute name="teamId">1</attribute>
        <attribute name="receiver">41</attribute>
        <attribute name="outcome">false</attribute>
    </box>
</track>
```

Tabella 5.5 - Estratto di un possibile file di eventi complessi.

Sebbene vi sia un numero elevato di dati ripetuti in questo formato, si è deciso comunque di procedere per questa soluzione per avere dati conformi allo standard del software di annotazione *Cvat*.

5.1.5.4 Annotazioni testuali

Oltre alle annotazioni XML, viene prodotto un file testuale semplice che contiene le informazioni sugli eventi in formato non rigoroso ma leggibile. Questo è prodotto, a differenza dei file XML, durante la partita e non soltanto al termine. In questo modo è possibile mettere in pausa la partita e verificare come si sta comportando la logica

implementata. Inoltre, alla fine della partita viene esportata la lista di tutti gli eventi con il relativo numero di eventi che si sono verificati. Non essendo un file strutturato, non è utilizzabile come fonte rigorosa per i dati del match.

5.1.5.5 Video

Come si accennava in precedenza, il video è stato registrato utilizzando il software *OBS studio*. Il video catturato dalla finestra del gioco è stato registrato con le seguenti impostazioni:

- Risoluzione video: 1920 x 1080 px (Full HD);
- Framerate: 25 fps;
- Bitrate: elevato (qualità alta);
- Campionamento audio: 44.1 KHz.

Il *framerate* è stato impostato a 25 fps, invece dei tipici 30 fps, per consentire allo strumento di visualizzazione prodotto in questo lavoro di tesi di avere una sincronizzazione migliore rispetto ai dati di gioco. Il gioco infatti lavora con una frequenza di 100Hz e avere il video ad un sottomultiplo esatto ha semplificato alcune operazioni che vedremo più avanti.

5.1.6 Passi operativi per la generazione di una partita

I passi effettuati per generare una partita sono i seguenti:

1. si lancia il gioco da riga di comando;
2. si lancia *OBS studio* per registrare il video della finestra del gioco;
3. dopo aver scelto le squadre, la durata della partita e gli eventuali controller si fa partire l'incontro;
4. si avvia la registrazione da OBS studio e si gioca, oppure si osserva che non ci siano pause durante la partita se la modalità di gioco è AI vs AI;
5. al termine della partita si avvia in automatico l'esportazione di tutti i file XML e nel frattempo è possibile terminare la registrazione dal video;
6. dopo aver atteso la fine dell'esportazione si genera il file di info in modo manuale, specificando il tipo di registrazione (qualità del video ecc.), la durata della partita, modalità di gioco ed eventuali altre informazioni utili.

5.2 Confronto con il dataset reale Alfheim

Uno dei principali problemi dell'utilizzo di un dataset sintetico per addestrare un sistema di riconoscimento degli eventi, è l'incertezza della validità del sistema quando applicato a un dataset reale. Il gioco infatti, può avere delle caratteristiche che non sono riscontrabili poi nella realtà, invalidando o comunque diminuendo le performance del sistema in modo anche significativo.

Allora, per validare le caratteristiche del dataset generato, si è pensato di valutarne alcune metriche e di confrontarle con le stesse di un dataset reale. Nella fattispecie, si è optato per l'unico dataset disponibile, e che avesse una dimensione sufficiente per lo scopo, ovvero il dataset Alfheim, presentato al paragrafo 4.1.1.

5.2.1 Metriche in esame

Le metriche usate sono state desunte da [16] e, in particolare, si fa riferimento alle seguenti metriche fisiche: distanza totale percorsa, distanza totale percorsa ad alta intensità, distanza totale percorsa a bassa intensità e infine velocità media. Per distanza percorsa a bassa intensità si intende la distanza percorsa con una velocità al di sotto della velocità media e, in maniera speculare, per distanza percorsa ad alta intensità si intende la distanza percorsa a velocità superiore la velocità media.

5.2.2 Procedimento

Per calcolare i valori della distanza percorsa e della velocità dei giocatori a partire dal dataset sintetico, è stato necessario utilizzare il file di *features* estratto dal riconoscitore degli eventi atomici. Grazie ai dati in esso presenti, è stato possibile calcolare la distanza percorsa da ogni singolo giocatore dell'incontro di calcio, la media della velocità, la distanza percorsa a bassa intensità e quella ad alta intensità.

Inizialmente si procede alla lettura di tutto il file, per acquisire i dati. Come prima operazione si effettua la somma di tutti i valori delle velocità per ogni istante di gioco e per ogni frame, in modo da ottenere la velocità media dei giocatori nella partita analizzata. A questo punto si vanno a leggere, istante per istante, i dati posizionali di ogni giocatore, al frame corrente di gioco e poi a quello successivo. Mediante le due coordinate si calcola la distanza tra i due punti e la si somma a un contatore universale. Allo stesso tempo, si confronta il valore della velocità per quel frame con la velocità media precedentemente calcolata, e si somma la distanza percorsa al contatore della distanza percorsa a bassa intensità, o a quello dell'alta intensità. Iterando così su tutti i dati, si ottengono le metriche cercante.

Per il calcolo degli stessi valori, ma a partire dal dataset Alfheim si sono seguiti dei passaggi analoghi a quelli appena descritti. La principale differenza sta ovviamente nel modo in cui i dati sono organizzati, ma un altro problema da affrontare è stato quello delle sostituzioni. Infatti, durante l'incontro registrato nel dataset, alcuni giocatori sono stati sostituiti, rendendo il calcolo della distanza percorsa in totale e in media leggermente più complicato. Inoltre, questi cambi hanno sicuramente introdotto un errore, sebbene auspicabilmente piccolo, nei valori risultanti.

Il terzo ed ultimo passaggio è stato quella della generazione del grafico dei risultati. Tramite uno script python, sono stati presi in input i valori di uscita delle due computazioni precedenti, e sono stati generati degli istogrammi rappresentanti i valori per le metriche prese in esame.

5.3 Sistema di visualizzazione del dataset

Individuare i problemi relativi al detector e al dataset è stata una sfida ardua a causa dell'assenza di un tool grafico che evidenziasse tutti i dati necessari insieme e in modo rapido e intuitivo. In un primo momento, la ricerca dei problemi è stata portata avanti leggendo i diversi file, confrontando i numeri dei frame, degli id dei giocatori e così via. Questa strada è evidentemente poco efficace e non sostenibile. Pertanto, si è proceduto verso la creazione di uno strumento che consentisse di visualizzare sul video, in modo immediato, tutti i dati del dataset e dei detector disponibili.

Nei prossimi paragrafi si illustreranno le modifiche che si sono rese necessarie per poter completare un simile tool, ed integrarlo con il gioco.

5.3.1 Modifiche al gioco e dati esportati

Per ottenere le posizioni dei giocatori sullo schermo, è stato necessario utilizzare alcuni metodi esposti dal motore grafico *blunted2*, in modo da ottenere a partire dalla posizione (x, y) del giocatore, le sue coordinate sullo schermo. Queste sono state ottenute mediante il metodo *GetProjectedCoord*, applicato sugli oggetti Player e Ball.

Insieme ai dati posizionali su schermo, è stato esportato anche un valore temporale relativo al calcio di inizio, che fosse sincrono con il video e che avesse consentito di avere un mapping univoco tra informazione posizionale e istante di tempo sul video.

Alla fine della generazione di una nuova partita, il sistema mette a disposizione un ulteriore file, denominato "*positions_timestamp.log*" (simile al file *positions.log*) che ha il seguente formato:

<timestamp_relativo> <tempo_di_gioco> <ID> <pos_x> <pos_y>

In questi dati, timestamp_relativo è un valore in millisecondi che rappresenta il tempo trascorso dal calcio d'inizio; il tempo_di_gioco è il valore del frame di gioco trascorsi (già presente in positions.log); ID corrisponde all'identificativo dell'entità; pos_x e pos_y sono le posizioni in pixel dell'entità rispetto allo schermo.

5.3.2 Preparazione dei dati

Per usare il sistema è necessario innanzitutto estrarre i frame a partire dalla sorgente video. Pertanto, mediante uno script *python*, si può processare il file "*recording.mkv*" per ottenere la lista di file in formato "*jpg*" che rappresentano frame per frame l'intera sequenza di gioco. Così facendo, il programma per visualizzare il dataset può caricare i dati con una lettura rapida e non sequenziale, consentendo salti da una parte all'altra dell'incontro. Alla fine delle operazioni di questa fase (da eseguire solo la prima volta che si vuole osservare un particolare dataset) viene generato anche un file che indica il

numero di frame effettivamente estratti. Anche questo dato sarà utile nell'esecuzione del tool di visualizzazione.

Un ulteriore parametro che deve essere modificato, questa volta in modo manuale, è il numero di frame che intercorrono dalla prima immagine esportata al momento dell'effettivo calcio di inizio. È dunque necessario trovare quel numero e inserirlo nel sistema. Questo passaggio è necessario per quei casi in cui il video non è stato preventivamente tagliato (rimuovendo quella parte iniziale di video di durata variabile che non fa parte dei minuti di gioco) e serve ad eliminare la traslazione iniziale che altrimenti resterebbe per tutta la durata della partita e a sincronizzare il video con i dati.

5.3.3 Funzionamento del sistema

Il sistema, sviluppato in *python*, sfruttando il suddetto framework grafico *openCV*, legge e carica in memoria principale tutto il file positions_timestamp.log appena definito e memorizza anche una finestra di 20 frame per volta. Inoltre, cerca i file XML del dataset contenenti le annotazioni degli eventi e ne effettua il caricamento. In breve, questa è la serie di operazioni svolte dal sistema:

1. estrazione del video in singoli frame (solo la prima volta);
2. caricamento del file delle posizioni in memoria;
3. caricamento delle annotazioni degli eventi, sia del ground truth che eventualmente quelle riconosciute;
4. utilizzo del sistema da parte dell'utente finale.

Più precisamente, il programma opera le seguenti operazioni: carica l'immagine centrale della finestra dei frame; successivamente, mediante accesso diretto con il dato temporale del momento corrente, carica le informazioni delle entità presenti nell'area visiva (ovvero le coordinate (x, y)) e le mostra sopra l'immagine sottoforma di *bounding box*; infine, carica le annotazioni degli eventi avvenuti in quel frangente temporale e li disegna in basso in 4 colonne rispettivamente.

L'ultimo passaggio è compiuto scandendo i dizionari che il sistema ha generato internamente. In tutto sono presenti 4 dizionari: il primo per i dati degli eventi atomici del *ground truth*, il secondo per gli eventi complessi del *ground truth*, il terzo per i dati degli eventi atomici riconosciuti dal sistema e il quarto per gli eventi complessi riconosciuti. In realtà è possibile lanciare il sistema anche in assenza dei dati degli eventi riconosciuti, visualizzando così solo i dati propri del dataset.

Il video, come detto in precedenza, ha un *framerate* di 25 frame al secondo. La scelta di questo valore non è casuale, ma è anzi dettata dal fatto che il gioco esporta i dati con una frequenza di 100 frame al secondo. La conseguenza di questo valore è che potenzialmente, in una sola immagine della partita, possono essere presenti fino a 4 eventi atomici, sebbene se ne possa effettivamente osservare a video solo uno. Si hanno infatti, nei 40 millisecondi che intercorrono tra un'immagine e l'altra, fino a 4 eventi distanziati di 10 millisecondi.

5.3.4 Interfaccia utente

L'interfaccia di interazione, mostrata nella Figura 5.6, è composta da una area di visualizzazione, che occupa gran parte della finestra e di una *trackbar* con cui spostarsi all'interno del dataset nella posizione di interesse della partita.

Figura 5.6 - Visualizzazione dei *Bounding Box* e di controlli sul video della partita.

Oltre alla *trackbar*, è possibile utilizzare le *frecce direzionali* per andare avanti e indietro di un frame alla volta, oppure di avviare l'avanzamento automatico a velocità naturale utilizzando la *barra spaziatrice*. Con il tasto *Esc* invece si può uscire dall'applicazione.

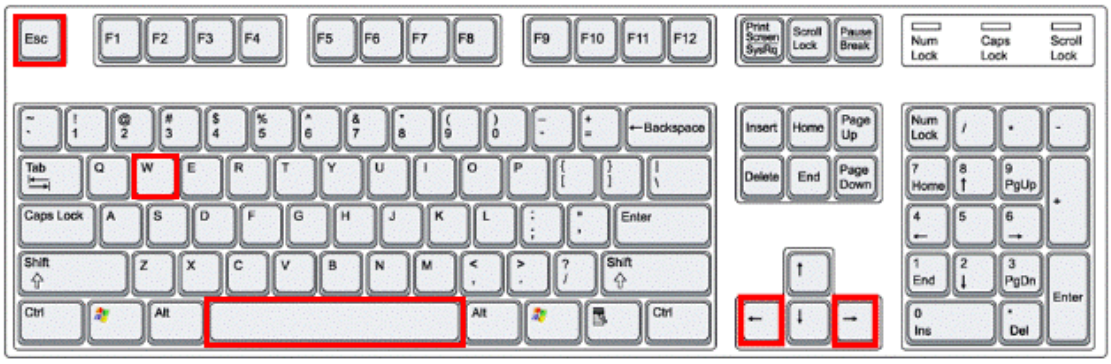


Figura 5.7 - Tasti associati ad una funzione del programma.

Un'ultima, utile, funzione è la possibilità di esportare sottoforma di video, tutto il flusso di immagini e dati che è mostrato, così da non avere la necessità di aprire il tool e di avere il costo di caricare nuovamente i dati da zero, potendo poi riprodurre l'esportazione come un normalissimo video.

Nell'immagine vi sono, come già detto, le quattro colonne degli eventi. Partendo da sinistra e proseguendo verso destra, esse mostrano rispettivamente nelle prime due colonne i dati del *ground truth*, nelle altre due colonne i dati eventualmente riconosciuti dai sistemi. In entrambi i casi la pima colonna è quella degli eventi atomici, la seconda è quella dei complessi. Un esempio di evento visualizzato da questo programma è visibile nell'immagine seguente:

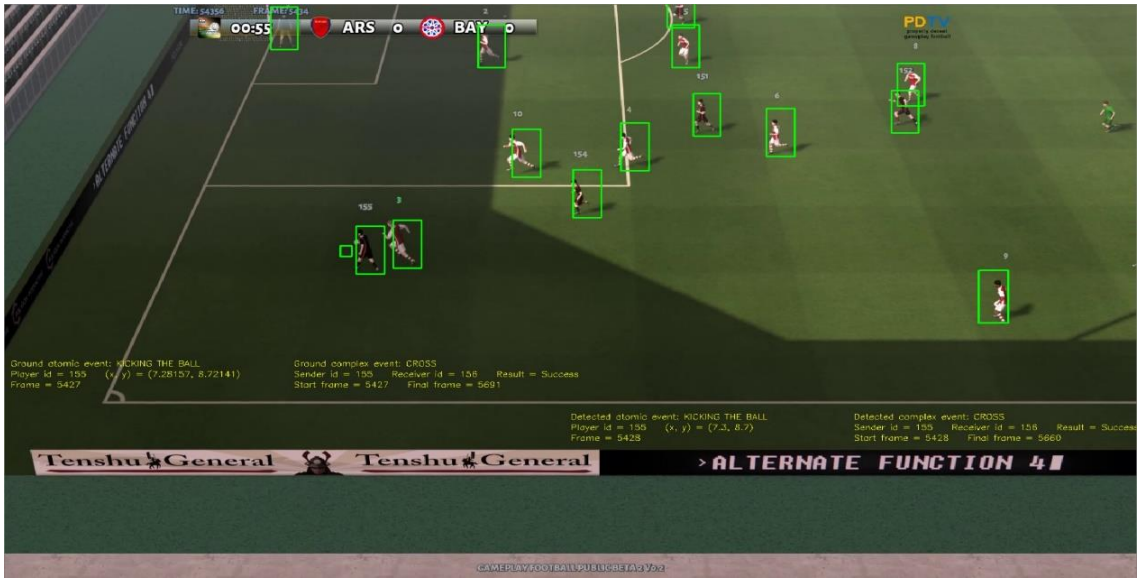


Figura 5.8 - Visualizzazione dei *Bounding Box* e delle annotazioni degli eventi.

L'evento mostrato in questa cattura è un *cross*. Il preciso instante è quello dell'inizio dell'evento, quando si verifica il calcio alla palla, che infatti viene mostrato nella prima colonna di sinistra e nella terza colonna. Nella seconda e nella quarta colonna a partire da sinistra, invece, sono visibili le informazioni inerenti all'evento complesso (cross). Questo evento, così come tutti gli eventi complessi, restano visibili nella loro colonna per tutta la durata dell'evento stesso. Si può osservare, inoltre, che gli eventi mostrati risiedono su piani diversi: le colonne di destra sono leggermente più in basse rispetto quelle di sinistra. Le colonne sono pensate per avere fino a quattro eventi da mostrare per l'immagine, e ogni "riga" si trova più in alto o più in basso in base a quando viene rilevato nei 4 possibili frame dell'immagine. In questo caso le colonne di sinistra mostrano gli eventi una riga più sotto perché i riconoscitori hanno individuato gli eventi con un frame di ritardo da quello effettivo.

5.4 Correzioni applicate al gioco e generazione di un nuovo dataset

Sfruttando il sistema di visualizzazione, è stato possibile osservare e verificare la corretta esportazione degli eventi e il loro riconoscimento in maniera enormemente più efficace. In questo modo sono stati rilevati diversi problemi sia dal punto di vista del gioco (ovvero a livello di esportazione del *ground truth*), ma soprattutto a livello di eventi riconosciuti

[23]. Dopo una breve fase di analisi delle performance dell'intero sistema, si è deciso di operare le dovute modifiche correttive, e di rigenerare un nuovo dataset, in modo da poterne beneficiare.

È stato infatti necessario generare un nuovo dataset a causa del numero di errori trovati, un numero troppo grande per effettuare una correzione manuale sui dati. In più alcuni di questi errori prevedevano la necessità di informazioni non più reperibili neppure attraverso il video.

Questa fase di valutazione è stata effettuata ovviamente su alcune partite generate a seguito delle modifiche attuate per il sistema di visualizzazione. Non è stato possibile invece agire sulle partite del primo dataset, a causa della sua incompatibilità con il sistema di visualizzazione.

5.4.1 Emissione dei cross

Uno dei casi riconosciuti dal sistema è il cross fallito. Il gioco tuttavia, nella prima generazione, non emetteva questo evento. Il risultato è stato un abbassamento elevato della *precision* per l'evento in questione. Dall'analisi del codice del gioco è risultata una chiamata mancante alla funzione di esportazione dell'evento cross fallito. Questa chiamata doveva avvenire nel caso in cui ad avere il possesso palla a seguito di un lancio fosse un giocatore della squadra avversaria. È stato allora sufficiente aggiungere questa chiamata, in condizioni di posizionamento dei giocatori corretto, per ovviare al problema.

Questa modifica non poteva essere ovviata in un qualsiasi modo manuale, a causa della numerosità dei casi. Pertanto, essa ha costituito una delle modifiche che hanno reso necessaria la rigenerazione di un nuovo dataset in luogo del primo generato.

5.4.2 Pass, Cross, FilteringPass

Questi tre eventi seguivano una dinamica di esportazione non pienamente coerente. La regola inizialmente decisa prevedeva l'emissione esclusiva dell'evento più specifico. Se, ad esempio, si fosse verificato un cross sarebbe stato emesso solo l'evento cross e non l'evento più generico passaggio. Identico discorso avveniva per passaggio filtrante. Tuttavia, nel caso di un cross che fosse anche in condizioni di passaggio filtrante, allora sarebbero stati emessi entrambi gli eventi. Nei software commerciali e nelle loro statistiche, infatti, si rapporta sia la casistica generica che quella specifica. Per completezza ed allineamento nei confronti dei software commerciali, si è deciso di emettere assieme all'evento più specifico, anche l'evento base. Pertanto, un caso di passaggio filtrante avrà come risultato l'emissione contemporanea dell'evento *Pass* e dell'evento *FilteringPass*, e così via.

[23] Come spiegato più avanti, nel capitolo 6.

5.4.3 Shot, ShotOut, ShotThenGoal, SavedShot

Analogamente al caso descritto precedentemente, anche la famiglia degli eventi “tiro” presentava la medesima ambiguità circa l’emissione di eventi generici e specifici. Se durante una simulazione si fosse verificato un evento specifico come SavedShot, la sua versione meno specifica non sarebbe stata emessa. Allora anche in questo caso si è proceduto all’inserimento dell’annotazione contemporanea di uno dei tre eventi specifici (tiro fuori, tiro goal e tiro parato) insieme al semplice evento tiro. In questo caso però, i tre eventi sono in mutua esclusione, quindi non possono occorrere casi in cui si emetta più di una coppia di annotazioni per evento.

5.4.4 Problema del passaggio

Un problema individuato in un solo caso consisteva nell’emissione di un passaggio che in realtà non si era verificato. Se un giocatore avesse passato la palla a un giocatore avversario, e quest’ultimo l’avesse passata a sua volta a un giocatore avversario, sarebbe stato esportato dal simulatore un passaggio tra il primo giocatore e l’ultimo. Ovviamente questa situazione è errata e non dovrebbe portare all’esportazione di alcun passaggio. L’errore a livello di gioco è stato individuato in un errato reset della condizione “passaggio in corso”, che non avveniva quando il secondo giocatore (quello avversario) calciava rapidamente la palla.

Capitolo 6

Sistema di riconoscimento degli eventi

L'architettura del sistema di riconoscimento è formata dal riconoscitore degli eventi atomici e quello degli eventi complessi. Il primo sistema genera gli eventi atomici a partire dai dati posizionali del dataset. Il secondo, invece, prende in input le annotazioni sugli eventi atomici generate e genera gli eventi complessi. La Figura 6.1 in modo molto schematico rappresenta l'architettura nel suo insieme. La parte evidenziata in azzurro rappresenta la sfera degli eventi atomici, mentre quella verde gli eventi complessi. Sono presenti anche dei moduli software che valutano i risultati del riconoscimento confrontando le annotazioni prodotte con quelle del dataset di input.

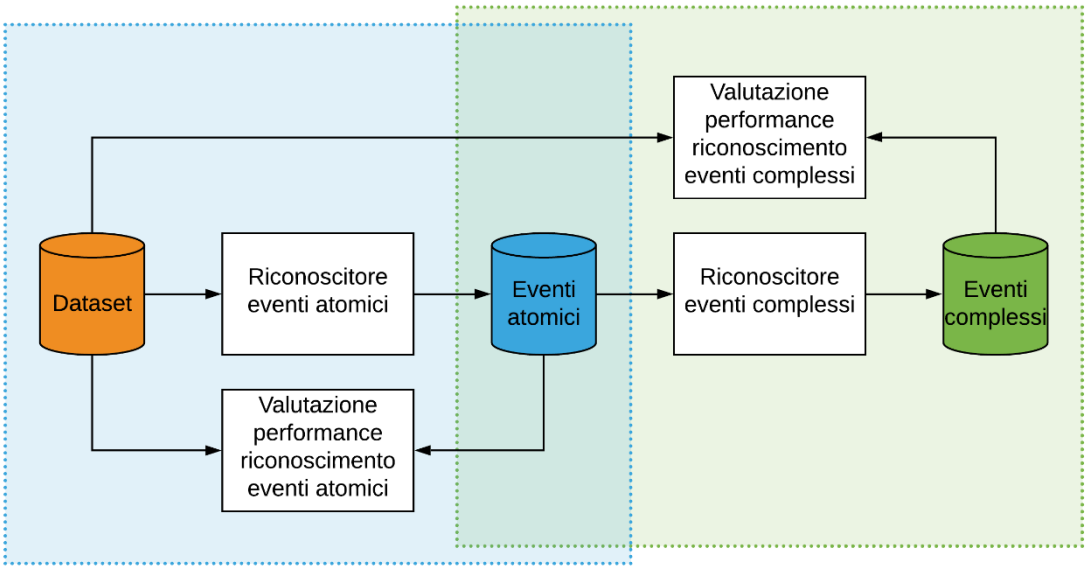


Figura 6.1 - Architettura del sistema di riconoscimento ad alto livello.

Nei prossimi paragrafi verrà descritta in maniera più specifica la struttura dei singoli componenti software, come anche le modifiche che ne sono state fatte per cercare di migliorarne le prestazioni, che invece sono descritte nel paragrafo 7.4.

6.1 Riconoscitore degli eventi atomici

Come si accennava brevemente nell'introduzione, uno dei componenti dell'intero sistema sviluppato nell'ambito di un altro lavoro di tesi, è il rilevatore degli eventi atomici. Esso è scritto in *python* ed è composta da alcune parti funzionali, ognuna delle quali si occupa di portare a termine un certo compito. I componenti del sistema di riconoscimento di eventi atomici sono i seguenti:

- Riconoscitore degli eventi vero e proprio;
- Ottimizzatore delle soglie per gli eventi;
- Validatore.

Il primo di questi componenti può prendere in input il file *positions.log* generato da una qualsiasi partita di GameplayFootball, ed estrarne le *feature* su un file, oppure effettuare immediatamente la rilevazione degli eventi atomici. In questo ultima eventualità, il sistema restituisce in output un file con lo stesso identico formato di quello generato dal gioco per gli eventi atomici. Dunque, un file del tipo *Annotations_AtomicEvent.xml*, che nel caso ideale dovrebbe coincidere con il *ground truth.* Questa fase operativa richiede l'utilizzo di alcuni parametri variabili. Ne sono un esempio la dimensione della finestra di riconoscimento di un evento, le soglie di velocità e di distanza. Poiché la configurazione manuale di questi parametri non è affatto banale e può portare ai risultati più variabili, si è pensato ad una soluzione che ottimizzasse questi parametri in modo automatico per ottenere la soluzione migliore possibile.

Il secondo componente è l'ottimizzatore delle soglie. Si tratta di un algoritmo genetico che lancia innumerevoli volte il sistema degli atomici, ogni volta con parametri e soglie diverse, verificando alla fine di ogni esecuzione, quali sono stati i risultati. Dopo un certo numero di iterazioni, l'algoritmo restituisce i parametri che hanno portato alla soluzione migliore, così che essi possano essere usati in via definitiva per le partite successive. Ovviamente, per addestrare correttamente il sistema e poi verificarne le effettive performance, è stato necessario suddividere il dataset iniziale in un *training set* e in un *validation set*.

L'ultima parte di sistema è composta dal validatore. Questo componente è costituito da un algoritmo che legge sia i dati del *ground truth* che quelli generati dal sistema, e verifica la corrispondenza di eventi nei due file. In particolare, questo software genera un grafico a istogramma, e per ogni evento genera due colonne: la *precision* e la *recall*. Questi due valori percentuali danno a colpo d'occhio un'idea delle effettive prestazioni del sistema su quella specifica partita. Il validatore, inoltre, è usato dall'ottimizzatore per verificare quanto buona sia ogni soluzione generata dal sistema di riconoscimento, e proprio per questo ha una grossa importanza.

6.2 Riconoscitore degli eventi complessi basato su ETALIS

L'ultimo sistema che completa tutta l'architettura costruita nell'ambito delle tre tesi, è il riconoscitore degli eventi complessi. Questo strumento è stato costruito a paritre da un *framework* logico, chiamato ETALIS. ETALIS è un processatore di regole logico-temporali che fa uso di un interprete PROLOG per determinare uno specifico pattern. Su questo strumento, è stata costruita una serie di processi mediante il linguaggio *python*, che, a partire da un file di eventi atomici (*Annotations_AtomicEvent.xml*), restiruisce un file di eventi complessi riconosciuti. Il motore di inferenza del sistema legge un file di regole e le usa per determinare il risultato in funzione dell'input. La regola principalmente usata in questo lavoro è l'operato SEQ. Quest'ultimo indica che, data una successione particolare di entità, deve essere dato in uscita a sua volta una precisa entità. Ogni entità inoltre è soggetta a delle restrizioni che devono essere rispettate per ottenere l'emissione della nuova entità. In questo caso le entità sono degli eventi, con i loro relativi attributi.

Analogamente al riconoscitore di eventi atomici, anche questo strumento ha un validatore, il quale confronta il *ground truth* e il file riconosciuto, per valutare le prestazioni di riconoscimento degli eventi. Viene generato, anche in questo caso, un grafico a istogrammi che mostra *precison* e *recall* per ogni evento complesso. Viene generato inoltre un file XML che mostra quali eventi del *ground truth* non sono stati classificati correttamente. In particolare, viene generata una lista di *falsi positivi* e una lista di *falsi negativi*, contenenti la tipologia di evento e il riferimento temporale al dataset rispetto al quale si è verificata la classificazione.

6.3 Modifiche ai sistemi di riconoscimento

Avvantaggiandosi dello strumento di visualizzazione del dataset, è iniziata una fase volta a individuare i problemi di ognuna delle parti che compongono l'intera architettura. Nei prossimi paragrafi si descriveranno quindi i problemi individuati e le modifiche attuate ai singoli componenti per correggere tali errori. Tutte le verifiche fatte sono state eseguite utilizzando come input soltanto dati appartenenti al *validation set*, in modo da avere dei risultati quanto più veritieri possibile, lanciando poi il sistema modificato sul *testing set*. In totale sono stati usati 250 minuti per il *validation* e altrettanti minuti per il *testing*.

6.3.1 Modifiche al riconoscitore degli eventi atomici

Il sistema non ha dei risultati ottimali per ogni classe di evento. Escludendo l'evento KickingTheBall, l'evento BallPossession e l'evento BallOut, le performance di BallDeflection e di Tackle sono molto basse. Inoltre, sebbene in una prima fase i risultati per l'evento palla fuori fossero stati giudicati ottimi (ovvero il 100% di *precision* di *recall*), in realtà ci si è resi conto di un errore in fase di valutazione dei

risultati e di generazione del grafico finale. Il vero risultato, in sostanza, era di circa 5 punti percentuale in meno sulla *precision* e sulla *recall.*

6.3.1.1 Evento BallOut

Per migliorare l'emissione di questo evento, che sicuramente sulla base dei dati posizionali dovrebbe avere un valore ottimale, si è proceduto con una fase estesa di verifica di quali eventi fossero emessi in più e quali in meno rispetto al *ground truth*, in modo da correggere gli eventuali errori sul *validation set* e confermare i risultati sul *testing set*.

Il primo errore individuato consiste in un margine che è stato aggiunto in fase di controllo, in prossimità dei limiti della porta. Esso è stato inserito per evitare che un goal potesse essere segnato come palla fuori. Di fatto quello che accadeva era che quando la palla usciva fuori, ma a lato dei pali delle porte, il *BallOut* non veniva correttamente emesso. Rimuovendo questi margini, smisuratamente sovradimensionati al valore di un metro, le performance sono migliorate leggermente (appena qualche decimo percentuale nella *recall*).

Procedendo nell'osservare le casistiche di falsi positivi per il *BallOut*, il secondo errore individuato risiedeva nel non aver considerato la dimensione della linea di bordo campo. Infatti, i limiti del campo di gioco sono compresi tra 0 e 110 metri per la x, e 0 e 72 metri per la y. Senza considerare i 10 cm di spessore della linea di campo, alcuni casi in cui la palla andava oltre i limiti appena menzionati, ma restando sempre sopra la linea, venivano annotati come palla fuori. Aggiungendo al controllo sui limiti del campo questo ulteriore valore, sono stati recuperati altri punti percentuali di precisione.

Un terzo caso di errore era causato dall'uscita della palla al di sopra della porta. Se la palla usciva al di sopra della traversa, e impiegava più della finestra temporale ad oltrepassare il limite della porta, allora il *BallOut* non veniva emesso. Un caso del genere è stato individuato nella seguente dinamica: un tiro colpisce la traversa, e si impenna notevolmente, deviando la sua velocità orizzontale in velocità verticale. A questo punto la palla impiega troppo tempo per uscire dai limiti della porta in relazione alla finestra temporale usata, e questo causa la non emissione dell'evento. Per correggere questo caso di *true negative* è stato inserito un controllo sull'uscita della palla a seguito di uno stazionamento sopra l'area della rete.

L'ultimo caso, ovvero quello più costoso in termini di performance, è stato un problema dovuto alla presenza di fuori gioco, falli e rigori. Durante uno di questi eventi, il gioco viene fermato dall'arbitro, ma la palla continua a muoversi ancora per circa due secondi di tempo. Se in questo frangente la palla oltrepassava i limiti del campo di gioco, allora il sistema di rilevamento annotava il *BallOut*. D'altronde dal punto di vista dei dati posizionali l'annotazione sarebbe stata del tutto corretta. Tuttavia, il dataset non possedeva un corrispondente evento, e questo faceva perdere prestazioni. Per ovviare al problema è stato necessario aggiungere un controllo prima dell'emissione di ogni evento di tipo *BallOut*, che verificasse se un caso di fuorigioco o fallo fosse stato rilevato in precedenza. La distanza temporale usata è pari alla dimensione della finestra di

riconoscimento dell'evento palla fuori. Il problema di questa soluzione è la mancanza di informazioni circa i fuori gioco. Mentre i falli e i rigori erano già presenti nel dataset generato, per il fuori gioco questo non era vero. È stato allora necessario annotare manualmente tutti i casi di fuori gioco, e inserirli nel dataset, controllando dai video la loro effettiva presenza.

A seguito di tutti i cambiamenti effettuati sul *validation set*, è stato riscontrato un miglioramento massimo per la classe *BallOut*. In seguito, l'intero processo è stato lanciato sul *testing set*, dove è stato congruamente riscontrato un miglioramento fino al 100% di *precision e recall*.

6.3.1.2 Problema sugli eventi KickingTheBall e BallDeflection

Questi due eventi hanno una forte sovrapposizione in fase di riconoscimento. Infatti, le caratteristiche fisiche della palla hanno delle analogie evidenti sia per la deviazione alla palla ad opera di un giocatore, che nel caso del tiro. Il calcio alla palla, ad esempio, avviene quando un giocatore accelera fortemente la palla stessa. Questa accelerazione, tuttavia, non necessariamente ha una direzione uguale alla velocità della palla. Questo vuol dire che la palla può procedere in una direzione, e successivamente essere accelerata in un'altra, portando a quello che può essere considerata una deflessione. Nella deflessione effettivamente non si dovrebbe verificare un cambiamento di velocità (dunque l'accelerazione assoluta dovrebbe essere prossima allo zero), ma solo un cambiamento di direzione. In gran parte dei casi, però questo non succede.

Il risultato è avere una casistica abbastanza grande di casi riconosciuti come *BallDeflection* in luogo di *KickingTheBall* e viceversa. Poiché il primo evento ha una ricaduta sugli eventi complessi piuttosto bassa (ovvero è utile a riconoscere un solo evento complesso, il *SavedShot*), si è deciso in un primo momento di eliminarlo. In questo modo le performance di *KickingTheBall* sono salite di diversi punti percentuali. Successivamente, per tentare di riconoscere in maniera diversa il *SavedShot*, si è proceduto a una ridefinizione dell'evento atomico. Pur lasciando il nome uguale, l'evento ha cambiato natura, divenendo non una deflessione della palla per opera di un qualsiasi giocatore ma esclusivamente dei portieri. In questo modo ci si è concentrati sui soli casi di deflessione interessanti ai fini del riconoscimento.

6.3.2 Modifiche al riconoscitore degli eventi complessi

Parte del lavoro di tesi è stata la modifica del sistema di riconoscimento degli eventi complessi, al fine di migliorare quanto più possibile i risultati. Utilizzando il sistema di visualizzazione, si è pertanto proceduto a verificare quali regole non stessero funzionando correttamente, o se queste fossero inadeguate in specifiche circostanze.

6.3.2.1 FilteringPass

Analogamente a quanto visto per l'evento *BallOut*, il caso del fuorigioco costituiva un problema evidente anche per i passaggi filtranti. Se la passa fosse partita in posizione di fuorigioco, e il giocatore l'avesse ricevuta posizionandosi oltre la linea di difesa avversaria, il sistema avrebbe annotato un passaggio filtrante. Il rilevamento del passaggio filtrante, infatti, non tiene conto della posizione del giocatore in ricezione al momento dell'inizio del passaggio, ma solo all'arrivo.

Le informazioni possedute dagli eventi atomici utilizzati al fine del riconoscimento (in questo caso *KickingTheBall* e *BallPossession*), non comprendono la posizione del giocatore al momento dell'inizio passaggio. In questo modo non è stato possibile introdurre direttamente il controllo sul fuorigioco, a meno di una modifica dell'evento *KickingTheBall*. Data l'impraticabilità di questa strada, si è deciso di procedere con l'annotazione diretta degli eventi di fuorigioco. Sfruttando così la modifica resasi necessaria anche per i *BallOut*, è stato possibile cancellare tutti quei casi riconosciuti in modo erroneo di *FilteringPass*, e avere un incremento di prestazioni nella *precision* di circa il 20%.

6.3.2.2 PassThenGoal, FilteringPassThenGoal, CrossThenGoal

Questa serie di eventi, tutti caratterizzati dall'essere una forma di passaggio che termina in un goal ad opera di chi riceve la palla, è stata soggetta al medesimo problema di emissione. La regola base di questi eventi è del tipo: *KickingTheBall* seguito da *BallPossession* seguito da *ShotThenGoal*. Questa serie, consente tuttora di riconoscere correttamente molti dei casi, ma falliva in presenza di azioni molto rapide in cui il giocatore che riceveva l'assist calciava immediatamente. In tali circostanze, infatti, non veniva emesso alcun *BallPossession*, di fatto rendendo vana la regola. Per aggiungere tutti quei casi non riconosciuti, e fare alzare quindi la *recall*, è stato sufficiente aggiungere la regola *KickingTheBall* seguito da *KickingTheBall* seguito da *ShotThenGoal*. In questo modo la regola viene correttamente individuata in quei casi in cui il giocatore calcia la palla al volo.

Capitolo 7

Risultati

Le annotazioni prodotte nei dataset rispettano quasi nella sua interezza i requisiti sulla tassonomia richiesta in principio. Gli eventi che non sono parte del dataset, ma che erano presenti nella tassonomia iniziale sono rispettivamente le scivolate, i dribbling, gli stop riusciti e infine alcuni valori aggregati riguardo la distanza percorsa e il numero di tocchi della palla.

EVENTO	TIPOLOGIA	NOME EVENTO DATASET
PASSAGGI	Effettuati	Pass
	Filtranti effettuati	FilteringPass
	Ricevuti	Pass
	Filtranti ricevuti	FilteringPass
	Ricevuti che sono sfociati in goal	PassThenGoal
	Filtranti ricevuti che sono sfociati in goal	FilteringPassThenGoal
CROSS	Effettuati	Cross
	Corretti effettuati	Cross
	Ricevuti	Cross
	Effettuati che sono sfociati in goal	CrossThenGoal
	Effettuati che sono sfociati in goal	Cross
SCIVOLATE	Fatte	-
	Vinte	-
CONSTRASTI	Fatti	Tackle
	Vinti	Tackle
DRIBBLING	Fatti	-
TIRI	Totali	Shot
	Nello specchio della porta	Shot
	Goal fatti	Goal
	Parati	SavedShot

PARATE	Su tiri dentro area piccola	SavedShot
	Su tiri dentro area di rigore	SavedShot
	Su tiri da fuori area	SavedShot
	Su rigori	SavedShot
	Su punizioni	SavedShot
ALTRI	Stop riusciti	-
	Distanza percorsa senza palla	-
	Distanza percorsa con palla al piede	-
	Numero tocchi palla prima di passaggio	-

Si noti che alcuni eventi sono corredati da degli attributi che consentono di definire diverse tipologie di evento allo stesso tempo. Ad esempio, l'evento *SavedShot* consente di definire l'insieme di parate effettuate da tiri fuori area, da dentro l'area grande o da quella piccola semplicemente osservando il

7.1 Risultati della prima generazione

Come detto nel capitolo Capitolo 5, sono state eseguite due differenti generazioni di dataset, ciascuna di 500 minuti circa. La prima generazione contiene un non quantificato numero di errori, della tipologia descritta al paragrafo 5.4. Non tutti gli eventi contenuti in questo dataset sono invalidi, anzi, la maggior parte di essi è corretta e pertanto viene lasciato qui il resoconto degli eventi che sono presenti.

Sono state giocate e registrate un totale di 8 partite, in diverse modalità e per durate differenti. Alcune di esse hanno sia il primo che il secondo tempo, altre invece presentano esclusivamente un primo tempo di gioco. I dati nella tabella e i grafici seguenti riassumono l'intero dataset.

TIPOLOGIA	EVENTO	NUMERO
ATOMICI	KickingTheBall	6870
	BallPossessionm	1485255
	Tackle	80495
	BallDeflection	906
	BallOut	354
	Goal	60
	Foul	29
	Penalty	5
COMPLESSI	Pass	4771

	PassThenGoal	32
	FilteringPass	73
	FilteringPassThenGoal	2
	Cross	258
	CrossThenGoal	10
	Tackle	3001
	Shot	340
	ShotThenGoal	55
	SavedShot	221

Tabella 7.1 - Numero di eventi per classe nel dataset 1.

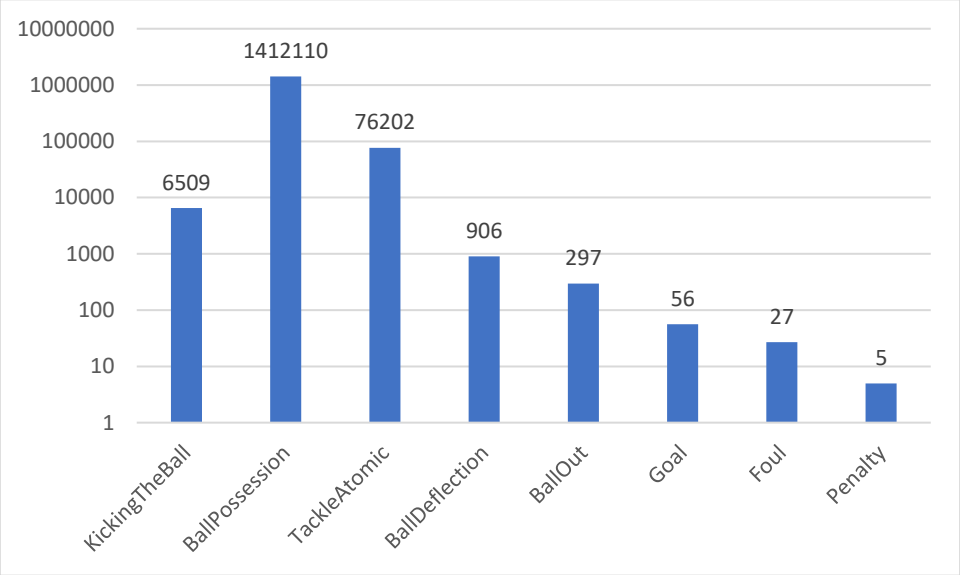


Figura 7.1 - Numero eventi atomici annotati per classe nel dataset 1.

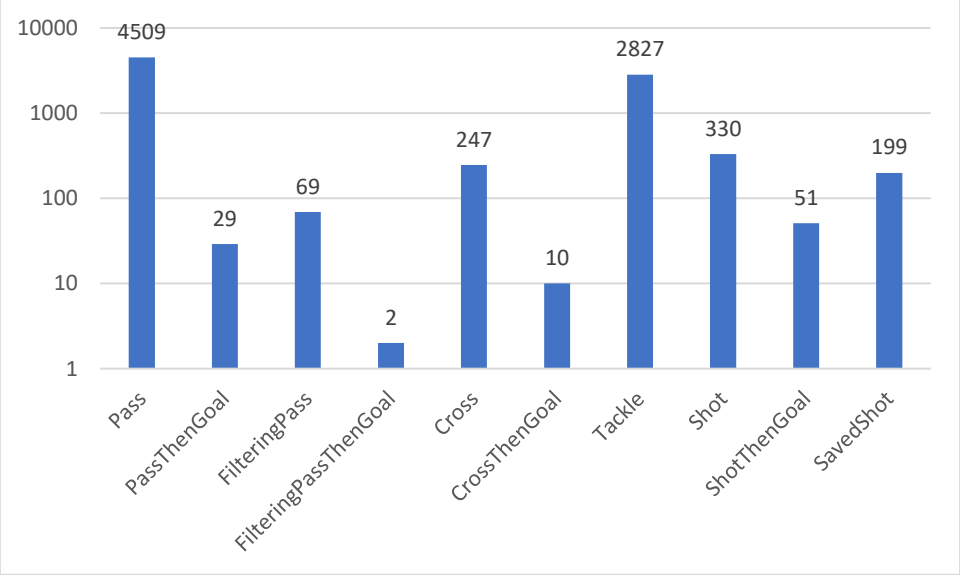


Figura 7.2 - Numero eventi complessi annotati per classe nel dataset 1.

7.2 Risultati della seconda generazione

Come evidenziato nel paragrafo 5.4, alcuni errori di emissione degli eventi, individuati grazie al sistema di visualizzazione, andavano corretti. Per questa ragione è stato generato un nuovo dataset che avesse applicate queste modifiche. Anche in questo caso è stato generato un dataset di 500 minuti di gioco circa. Sono state generate un totale di 17 partite, ognuna di circa 30 minuti. In totale i minuti di gioco sono stati raccolti nelle seguenti modalità:

1. 90 minuti in modalità giocatore umano contro giocatore umano;
2. 150 minuti in modalità giocatore umano contro AI;
3. 270 minuti in modalità AI contro AI;

In totale sono state generate 1591199 annotazioni, distribuite in 19 classi tra eventi atomici ed eventi complessi, così come indicato nelle figure sottostanti. Per la grande differenza di numero tra le varie classi si è usata una scala logaritmica per generare i grafici.

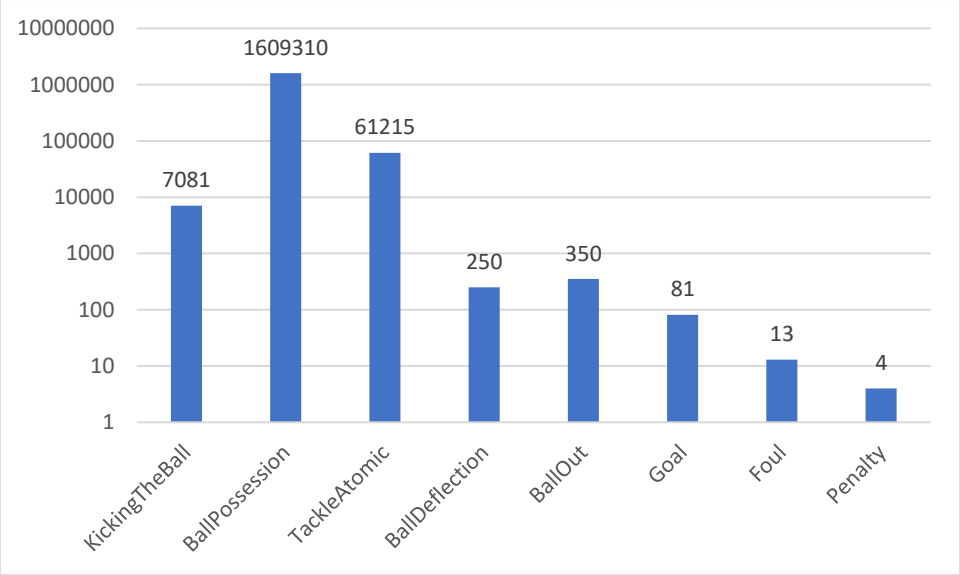


Figura 7.3 - Numero eventi atomici annotati per classe nel dataset 2.

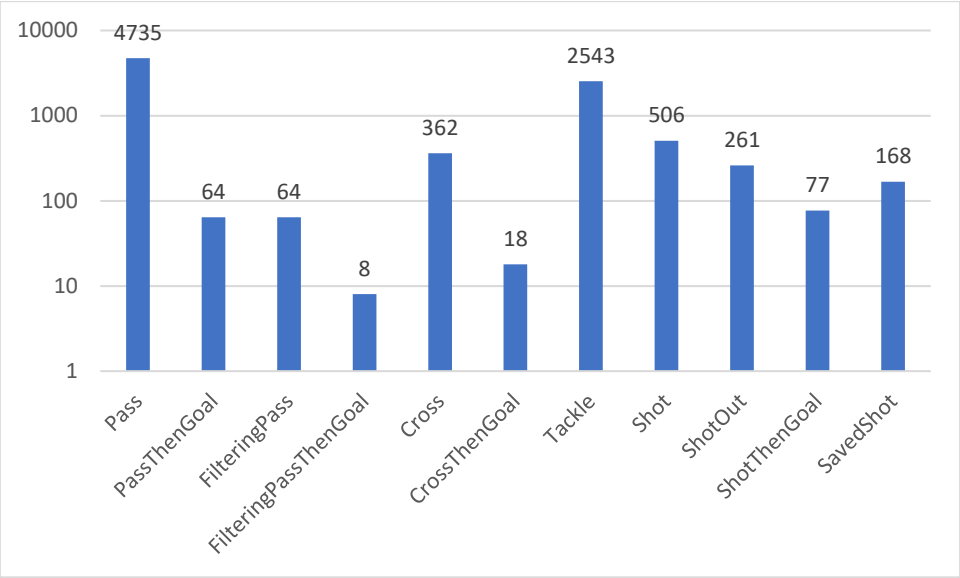


Figura 7.4 – Numero eventi complessi annotati per classe nel dataset 2.

Si può osservare che molti eventi presentano una casistica piuttosto bassa. In particolar modo gli eventi della classe *FilteringPassThenGoal* sono in un numero oltremodo limitato. Questo può risultare come un grosso problema in fase di addestramento di un sistema di *machine learning*, perché richiederebbe sicuramente un quantitativo di dati molto più grande per eseguire un addestramento adeguato.

7.3 Risultati del confronto con il dataset reale Alfheim

Nella generazione dei risultati, sono stati presi in esame dapprima 45 minuti di gioco, sia per il caso reale che per quello sintetico. Il primo valore analizzato è quello della velocità media dei giocatori durante la partita. Come si vede dalla Figura 7.5 il gioco ha generato una velocità media pari a 3.8 m/s, mentre il dataset reale presenta una velocità media molto inferiore, di circa 2.2 m/s.

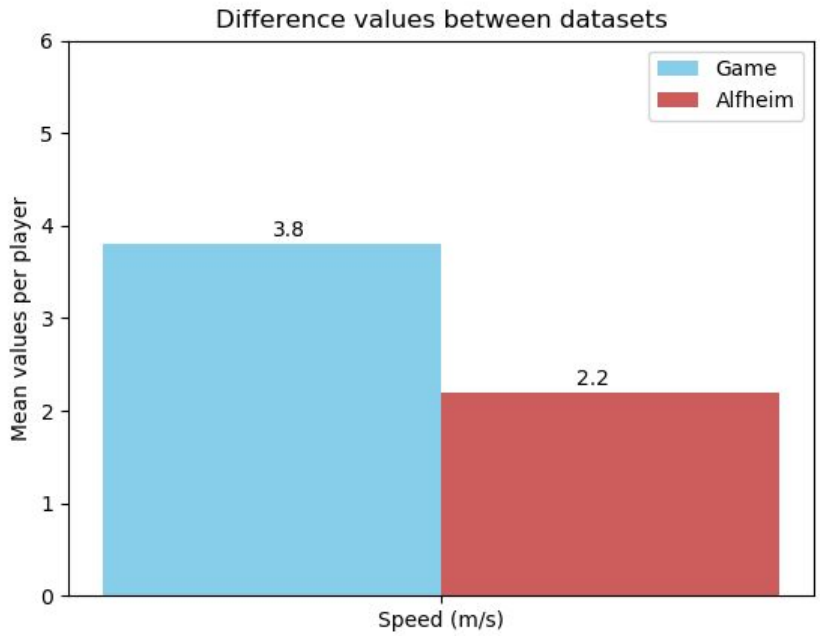


Figura 7.5 - Velocità media dei giocatori nel dataset sintetico e in Alfheim.

La seconda figura rappresenta invece le distanze percorse in totale e ai livelli di intensità alto e basso. Anche in questo caso i valori di GameplayFootball sono più grandi di quelli del dataset Alfheim, confermando quindi la tipicità del primo di avere una modalità di gioco più veloce, con giocatori che mediamente si muovono di più e a velocità più elevate.

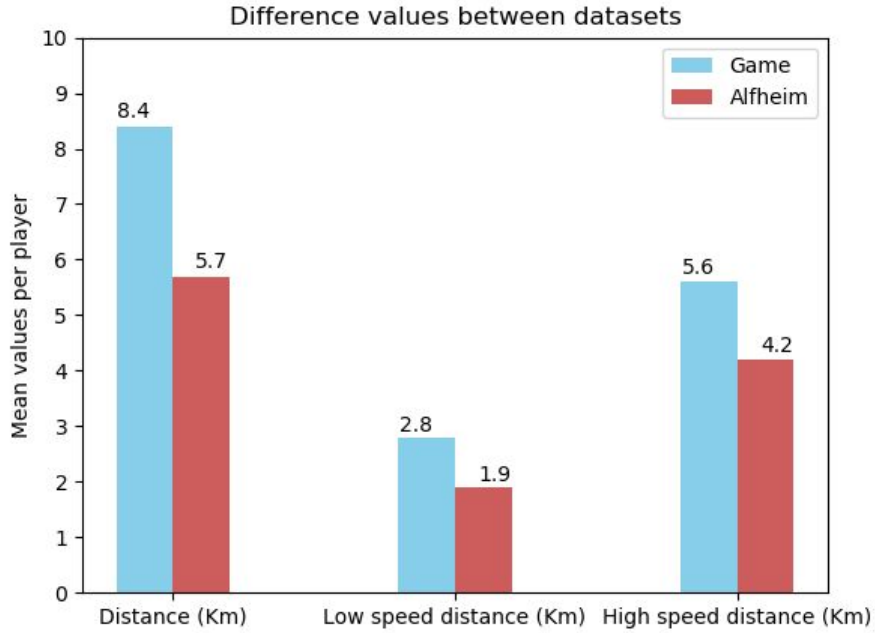


Figura 7.6 - Distanze percorse dai giocatori per intensità nel confronto.

In parte però questi valori sono giustificabili da un semplice fatto: durante una partita reale, il tempo di gioco continua a scorrere anche in caso di falli, goal, esultanze ecc., mentre nel gioco questo tempo viene fermato. Ciò significa che in quella fase i giocatori camminano o sono fermi, mentre nel gioco si ha un salto quasi istantaneo alla nuova azione, in cui i giocatori correranno subito.

Poiché durante una partita generata dal gioco si ha una frequenza elevata di interruzioni di gioco (palla fuori, falli ecc.), per effettuare un confronto più equo e sensato si è presa in considerazione una sola azione della durata di un minuto in cui non si sono verificate interruzioni. D'altro canto, i riconoscitori agiscono a livello più microscopico che macroscopico, pertanto è più indicativo un confronto dei dataset su piccoli frammenti (quelli che appunto vengono analizzati) che su larga scala come fatto in precedenza.

In questo nuovo confronto è stato usato nuovamente il dataset Alfheim, considerando solo il primo minuto di gioco, mentre per i dati sintetici è stato usato Match_2019_02_13_#002. Per il dataset sintetico, si è fatto uso dei dati a partire dal frame 4592 fino a 10592, per un totale di 60 secondi, corrispondenti alla sezione del video che va da 00:50s a 01:50s. Con uno script *python* è stata estratta la porzione di dati richiesti dal file di feature. A seguito, questa volta in modo manuale, sono stati estratti i dati in *csv* dal dataset Alfheim. Infine, lanciando gli script precedenti con le dovute modifiche sono stati generati i grafici e i risultati delle immagini sottostanti.

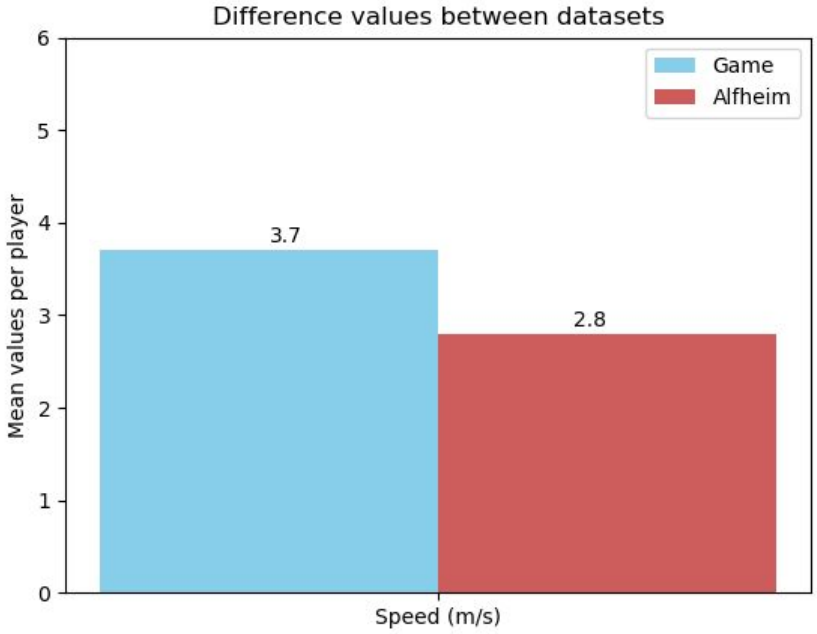


Figura 7.7 - Velocità media dei giocatori considerando solo 60 secondi.

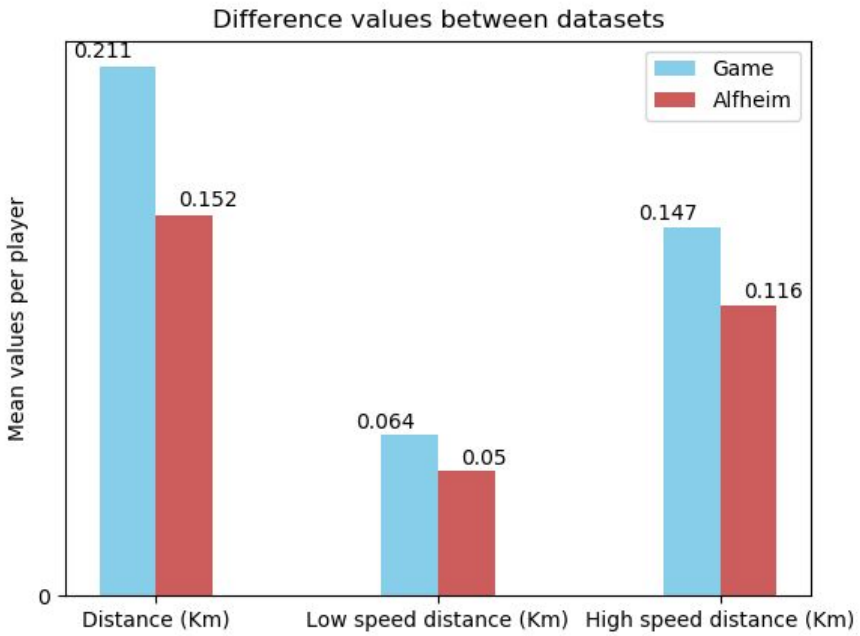


Figura 7.8 - Distanze percorse dai giocatori per intensità considerando solo 60 secondi.

Le differenze tra dataset sintetico e dataset reale sono effettivamente diminuite. Nel caso della velocità media la differenza è più marcata che nelle metriche sulla distanza.

Il risultato di questa analisi è che le dinamiche fisiche dei giocatori nei due dataset presentano delle differenze abbastanza marcate. Ciò significa che alcuni parametri del riconoscitore degli eventi atomici non saranno i migliori per il dataset Alfheim. Con particolare riferimento alle soglie sulla velocità e sulla accelerazione, per avere dei risultati in linea con quelli del dataset sintetico è probabilmente necessario lanciare nuovamente l'ottimizzatore. In questo modo si possono individuare quei parametri ottimali nelle diverse condizioni di gioco.

7.4 Risultati dei sistemi di riconoscimento sul dataset sintetico

Di seguito vengono riportati i risultati finali delle performance di entrambi i software di riconoscimento sviluppati e migliorati nell'ambito di questa tesi. Sebbene sia stato usato per larga parte dello sviluppo il primo dataset generato, si mostreranno solo i risultati relativi al secondo dataset. Generalmente gli eventi sono riconosciuti con performance migliori in entrambi i sistemi, sia in virtù dei miglioramenti al dataset, che delle migliorie applicate al codice stesso.

In entrambi i casi sono stati usati un *validation set* e un *testing set* pari a metà dell'intero dataset: in totale, dunque, 250 minuti per il primo e 250 minuti per il secondo set. La selezione delle partite da utilizzare in ciascuno dei due insieme di dati invece è

stata fatta in modo casuale, cercando comunque di inserire un numero di partite equo in base alla modalità di esecuzione delle stesse (modalità AI vs AI ecc.).

I grafici riportati di seguito rappresentano il numero di eventi presenti rispettivamente nel *validation set* e nel *testing set*.

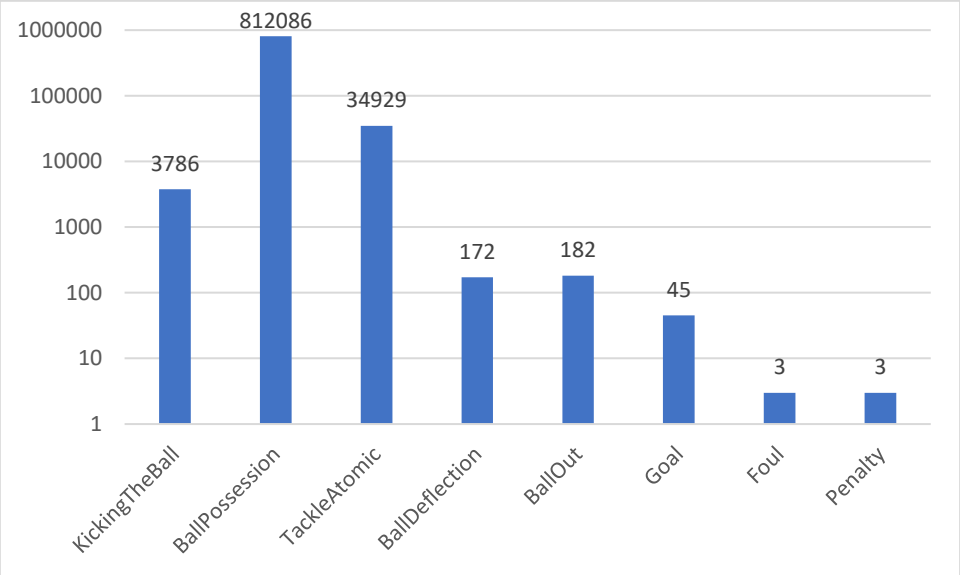


Figura 7.9 - Eventi atomici presenti nel *validation set*.

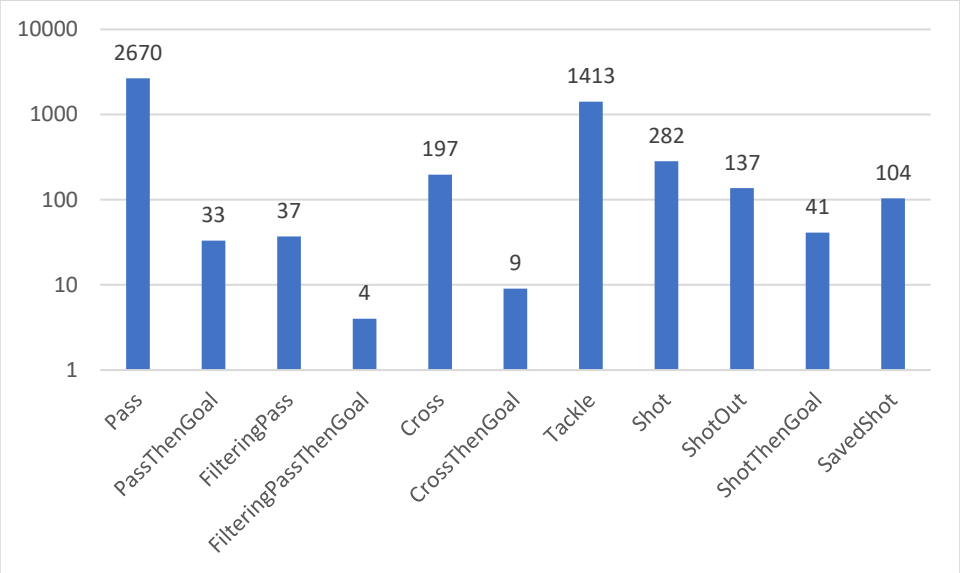


Figura 7.10 - Eventi complessi presenti nel *validation set*.

I prossimi grafici sono invece quelli utilizzati per produrre i risultati finali, ovvero per valutare le prestazioni totali dei sistemi di riconoscimento.

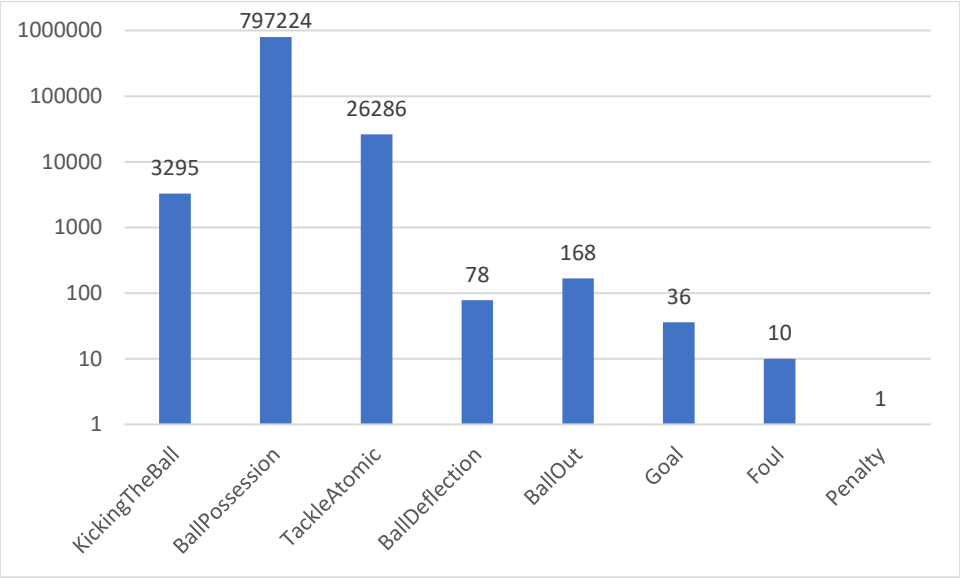


Figura 7.11 - Eventi atomici presenti nel *testing set*.

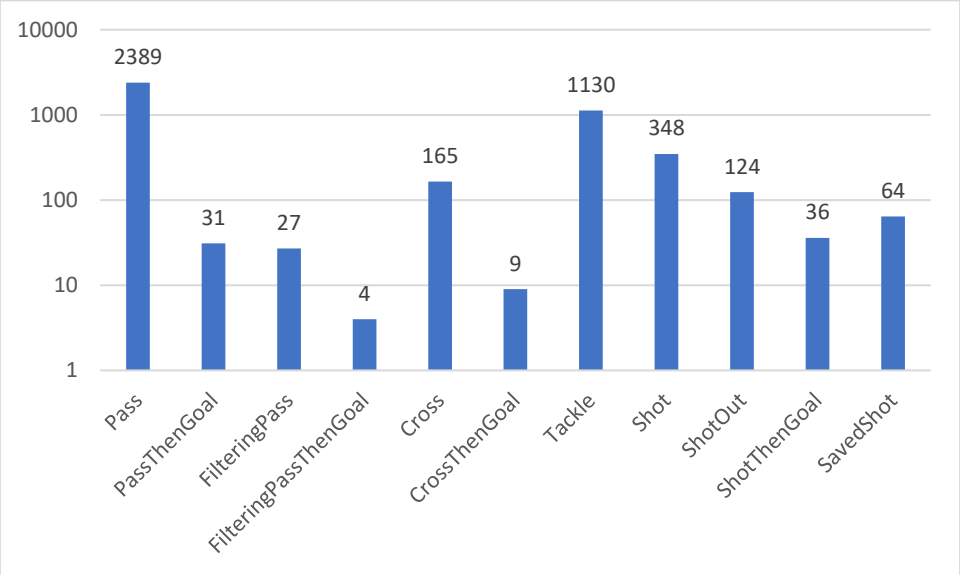


Figura 7.12 - Eventi complessi presenti nel *testing set*.

7.4.1 Riconoscitore degli eventi atomici

Il sistema di riconoscimento per gli eventi atomici ha subito poche modifiche rispetto allo stato di sviluppo precedente, se non per l'evento *BallDeflection* e per l'evento *BallOut*. I risultati, dunque, non sono variati di molto nel complesso.

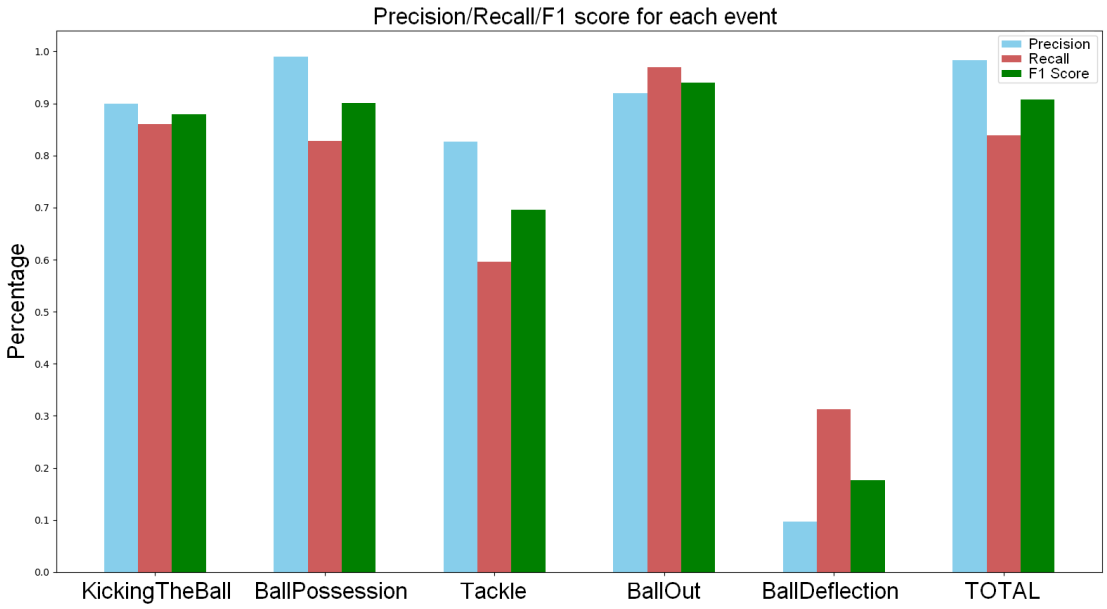


Figura 7.13 - Prestazioni del sistema degli atomici sul testing set, prima delle modifiche.

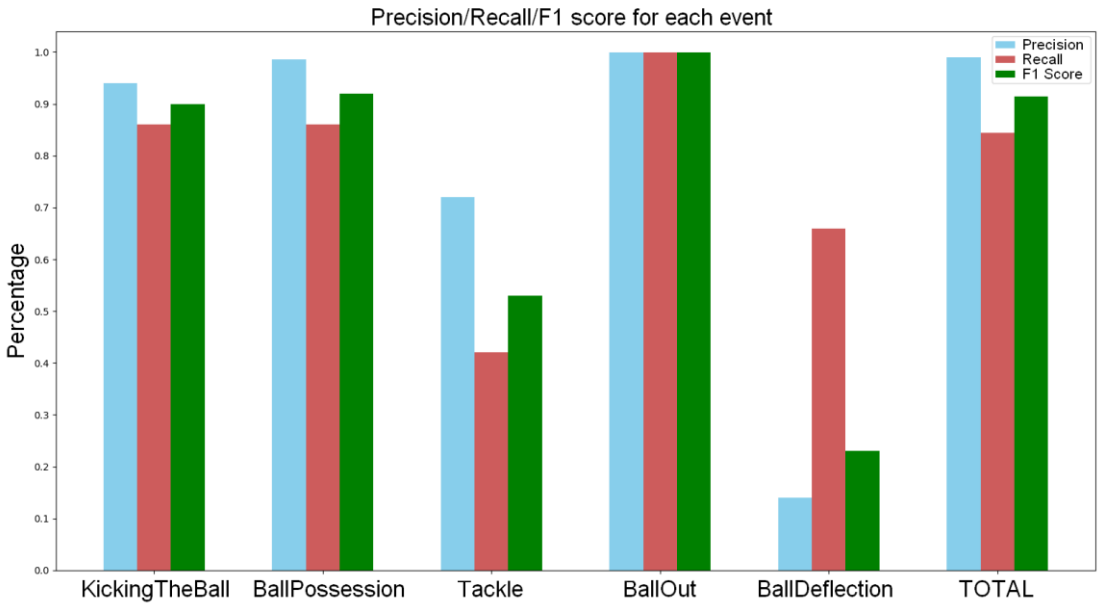


Figura 7.14 - Prestazioni del sistema degli atomici sul testing set, dopo le modifiche.

Grazie alle modifiche effettuate è stato possibile far aumentare le prestazioni dell'evento *BallOut* fino al 100% di *precision* e di *recall* sul testing set. Il cambiamento della regola per l'evento *BallDeflection* ha consentito il miglioramento della *recall* per l'evento, che si attesta però su dei valori fin troppo bassi, soprattutto se si considera la precisione che non è minimamente soddisfacente. Tuttavia, la ridefinizione dell'evento ha consentito un lieve miglioramento sull'evento *KickingTheBall*, il quale ha una notevole importanza in ottica di riconoscimento degli eventi complessi. A parte un lieve abbassamento dell'evento *Tackle*, non ci sono stati altri cambiamenti significativi.

7.4.2 Riconoscitore degli eventi complessi

Prendendo i dati generati dal riconoscitore degli atomici sul *testing set*, e utilizzandoli come input per il sistema dei complessi, vengono generate le annotazioni per gli eventi complessi. In modo automatico il software esegue l'algoritmo di valutazione dei risultati e li confronta con il *ground truth*. Da questo confronto si produce in automatico il grafico che mostra *precision* e *recall* di ciascuno degli eventi complessi.

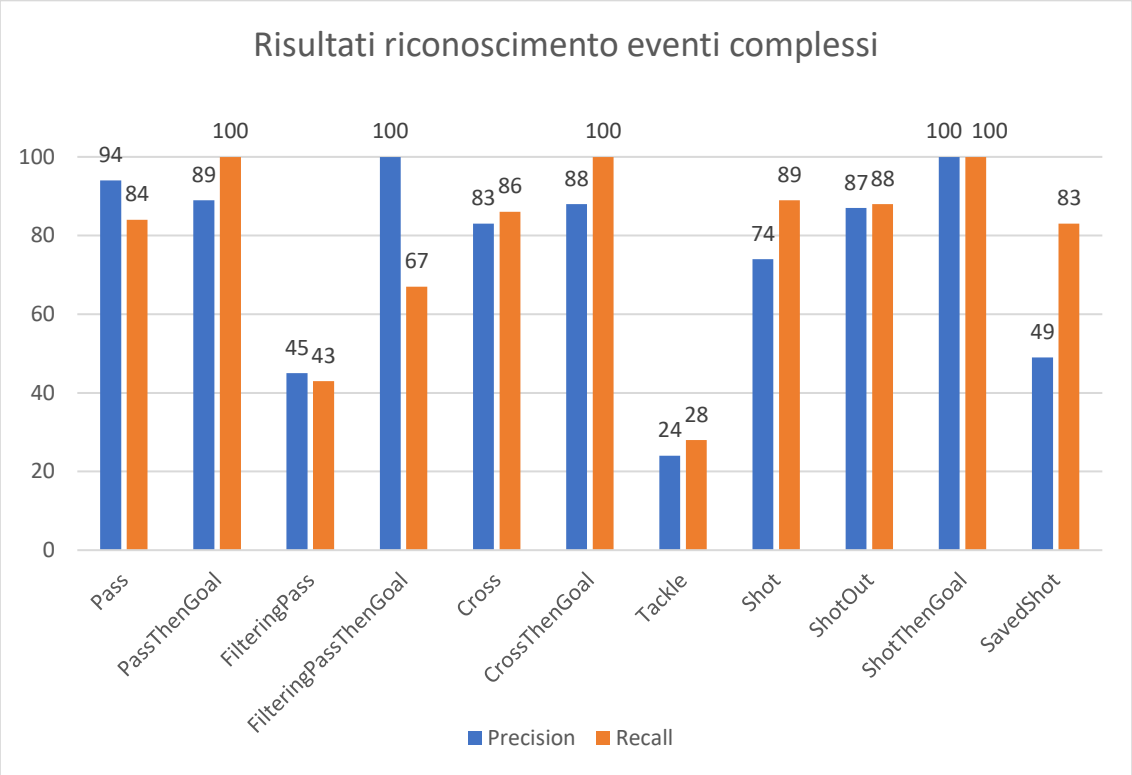


Figura 7.15 - Risultati del riconoscimento degli eventi complessi sul testing set

Nella tabella si riportano i valori del grafico, insieme al supporto di ciascun evento e la numerosità di *true positive*, *false positive* e *true negative* che sono stati generati.

Evento	*Prec.*	*Recall*	*F-score*	*Supporto*	*TP*	*FP*	*FN*
Pass	94	84	89	2426	2046	128	380
PassThenGoal	89	100	94	31	31	4	0
FilteringPass	45	43	44	23	10	12	13
FilteringPassThenGoal	100	67	80	3	2	0	1
Cross	83	86	84	176	152	32	24
CrossThenGoal	100	88	94	7	7	1	0
Tackle	24	28	26	1065	236	1038	829
Shot	74	89	81	230	204	70	26
ShotOut	87	88	87	130	114	17	16
ShotThenGoal	100	100	100	36	36	0	0
SavedShot	49	83	62	64	53	56	11

Tabella 7.2 - Risultati sul *validation set* del sistema dei complessi.

Alcuni eventi hanno valori elevati di *precision* e di *recall*, mentre altri eventi hanno valori molto più bassi. In particolar modo l'evento meno performante, che non viene riconosciuto in modo adeguato è l'evento *Tackle*. Quest'ultimo infatti dipende quasi in modo totale dall'emissione di eventi *Tackle* atomici, i quali a loro volta hanno pessimi risultati, motivo per cui il riconoscimento dell'associato evento complesso ha così basse performance. L'evento *FilteringPass*, invece, ha un problema significativo ma sottile: se la palla viene ricevuta da un giocatore che ha superato la linea della difesa avversaria, può accadere che egli tiri al volo, oppure sia in contrasto per il possesso palla. In entrambe le situazioni il sistema degli eventi atomici non emette alcun evento *BallPossession*, evento necessario affinché la regola del passaggio filtrante sia rispettata. La perdita di tutti questi casi, ovviamente, fa abbassare notevolmente la recall.

L'evento *SavedShot* ha un valore molto basso di precision a causa dell'emissione poco precisa dell'evento atomico *BallDeflection*. La regola del *SavedShot* prevede infatti che ci sia una deflessione della palla ad opera del portiere. Tuttavia, il sistema degli atomici emette un numero di falsi positivi molto grande, facendo sì che il sistema dei complessi ritrovi il pattern dell'evento tiro parato molte più volte del *ground truth*.

Capitolo 8

Conclusioni e sviluppi futuri

Nella sua interezza il sistema ha diversi punti di forza. Innanzitutto, vi è la possibilità di generare dati in modo virtualmente illimitato, in base alle proprie necessità. Quindi vi è una larga possibilità di uso nell'ambito di ricerca e sviluppo di sistemi di riconoscimento in cui è necessario un dataset. Il gioco è anche estendibile per estrarre eventualmente altri eventi o dati che sono utilizzati dal motore di gioco. Nel farlo si può usare la distinzione fatta in fase di progettazione tra eventi atomici e complessi, oppure rompere questa netta separazione, e determinare gli eventi in categorie differenti. Il limite principale del sistema, invece, risiede nel basso fotorealismo. Le scene, infatti, non risultano molto realistiche, soprattutto se si osservano i giocatori da vicino. Alcune azioni dei giocatori, inoltre, non sono molto dettagliate in termini di animazione, e i modelli 3D dei giocatori stessi sono abbastanza schematici. Un sistema di riconoscimento che fa direttamente uso delle immagini generate dal gioco per l'addestramento non sarebbe probabilmente in grado di generalizzare a sufficienza sul mondo reale. Almeno per questo aspetto, quindi, il gioco si discosta troppo dalla realtà per poter sostituire integralmente un dataset di sequenze reali con uno sintetico.

Il sistema di visualizzazione consente di visionare l'output eventualmente generato da qualsiasi sistema di riconoscimento che dia in uscita dei dati conformi allo standard adottato in questo lavoro. In questo modo si può controllare graficamente, sul video della partita, quali sono le performance e gli eventuali errori di un tale sistema di riconoscimento.

I sistemi di riconoscimento hanno mostrato diverse debolezze nei confronti di alcuni eventi. Con riferimento a *Tackle* e *BallDeflection* rispetto agli eventi atomici, e a *SavedShot*, *FilteringPass*, *Tackle* per gli eventi complessi, si è evidenziata la necessità di operare delle modifiche sostanziali, o sostituire integralmente con un sistema di riconoscimento alternativo. La principale possibilità è quella di sostituire il sistema a regole con un sistema di *Machine Learning*, ad esempio un classificatore, che a partire dalle features calcolate riesca a classificare il tipo di evento. Più di metà degli eventi, invece, presentano delle performance abbastanza elevate, il che dimostra l'efficacia sia del sistema di riconoscimento degli eventi atomici che degli eventi complessi per almeno una selezione di eventi.

Bibliografia

[1] S. A. Pettersen, D. Johansen, H. Johansen, V. Berg-Johansen, V. R. Gaddam, A. Mortensen, R. Langseth, C. Griwodz, H. K. Stensland, and P. Halvorsen, "Soccer video and player position dataset", International Conference on Multimedia Systems (MMSys), Singapore, March 2014, pp. 18 – 23, DOI:

[2] Soccer Video and Player Position Dataset, http://home.ifi.uio.no/paalh/dataset/alfheim/

[3] S. Giancola, M. Amine, T. Dghaily and B. Ghanem, "SoccerNet: A Scalable Dataset for Action Spotting in Soccer Videos", 2018 IEEE/CVF Conference on Computer Vision and Pattern Recognition Workshops (CVPRW), Salt Lake City, UT, 2018, pp. 1792-179210, DOI: 10.1109/CVPRW.2018.00223

[4] SoccerNet, https://silviogiancola.com/publication/2018-04-soccernet/details/

[5] Issia CNR, http://www.issia.cnr.it/wp/dataset-cnr-fig/

[6] Magglingen 2013, https://old.datahub.io/dataset/magglingen2013

[7] Soccer arc Multi-view Sequence by Nagoya University, http://www.fujii.nuee.nagoya-u.ac.jp/multiview-data/Soccer_arc_page/soccer_arc.htm

[8] Tetta Maeda, Ryo Suenaga, Kazuyoshi Suzuki, Mehrdad Panahpour Tehrani, Keita Takahashi, and Toshiaki Fujii, "Free Viewpoint Video for Sport Events Using Multi-resolution Visual Hull and Micro-facet Billboarding", Proc. of International Workshop on SISA, Thailand, Sep. 2016.

[9] Opta, https://www.optasports.com/

[10] STATS, https://www.stats.com/football/

[11] Håkon Kvale Stensland, Vamsidhar Reddy Gaddam, Marius Tennøe, Espen Helgedagsrud, Mikkel Næss, Henrik Kjus Alstad, Asgeir Mortensen, Ragnar Langseth, Sigurd Ljødal, Østein Landsverk, Carsten Griwodz, Pål Halvorsen, Magnus Stenhaug, and Dag Johansen. 2014. "Bagadus: An integrated real-time system for soccer analytics". ACM Trans. Multimedia Comput. Commun. Appl. 10, 1s, Article 14 (January 2014), 21 pages. DOI: 10.1145/2541011

[12] C. Dietrich, D. Koop, H. T. Vo and C. T. Silva, "Baseball4D: A tool for baseball game reconstruction & visualization", *2014 IEEE Conference on Visual Analytics Science and Technology (VAST)*, Paris, 2014, pp. 23-32. DOI: 10.1109/VAST.2014.7042478

[13] Brath, Richard & Moon, Bo. (2013). "Bloomberg Sports Visualization for Pitch Analysis"

[14] Perin, C., Vuillemot, R., Stolper, C. D., Stasko, J. T., Wood, J. and Carpendale, S. (2018), “State of the Art of Sports Data Visualization”. Computer Graphics Forum, 37: 663-686. doi:10.1111/cgf.13447

[15] Demaj, Damien. “Geovisualizing spatio - temporal patterns in tennis: An alternative approach to post - match analysis.” (2013).

[16] Ankur Handa, Viorica Patraucean, Vijay Badrinarayanan, Simon Stent, Roberto Cipolla, “Understanding Real World Indoor Scenes With Synthetic Data”. The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2016, pp. 4077-4085.

[17] Alexey Dosovitskiy, Philipp Fischer, Eddy Ilg, Philip Hausser, Caner Hazirbas, Vladimir Golkov, Patrick van der Smagt, Daniel Cremers, Thomas Brox, “FlowNet: Learning Optical Flow With Convolutional Networks”. The IEEE International Conference on Computer Vision (ICCV), 2015, pp. 2758-2766

[18] Qi Wang, Junyu Gao, Wei Lin, Yuan Yuan, “Learning From Synthetic Data for Crowd Counting in the Wild”. The IEEE Conference on Computer Vision and Pattern Recognition (CVPR), 2019, pp. 8198-8207

[19] A, Dellal & Chamari, Karim & Wong, Del P. & Ahmaidi, Said & Keller, Dominique & Barros, Ricardo & Bisciotti, Gian Nicola & Carling, Christopher. (2011). Comparison of physical and technical performance in European professional soccer match-play: The FA Premier League and La LIGA. European Journal of Sport Science. 11. 51-59. 10.1080/17461391.2010.481334.

[20] Darko Anicic, Paul Fodorm Sebastian Rudolphm Roland Stühmer, Nenad Stojanovic, Rudi Studer, "ETALIS: Rule-Based Reasoning in Event Processing" in "Reasoning in Event-Based Distributed Systems" edited by Sven Helmer, Alexandra Poulovassilis, Fatos Xhafa, Springer, 2011, pp. 99-124, DOI 10.1007/978-3-642-19724-6 52