Alma Mater Studiorum
Università di Bologna
Campus di Cesena

DIPARTIMENTO DI INFORMATICA – SCIENZA E INGEGNERIA
Corso di Laurea Magistrale in Ingegneria e Scienze Informatiche

BallVisionAI: Un'applicazione di Visione Artificiale per l'analisi automatizzata di partite di Beach Volley.

Elaborato in:
Analisi Di Immagini 3d E Sistemi Di Computer Vision

Relatore:
Prof.ssa
Alessandra Lumini

Presentata da:
Andrea Zammarchi

Sessione III
Anno Accademico 2023-2024

"Ci sono momenti in cui bisogna prendere in mano il timone, tracciare la propria rotta e seguirla, anche in caso di burrasca. Solo così si potrà mettere alla prova la forza delle proprie vele e dimostrare di che pasta si è fatti. E quando arriverà il giorno in cui si brillerà, la luce emanata sarà il riflesso delle proprie scelte e del coraggio dimostrato."

- John Silver, Il Pianeta del Tesoro

Introduzione

L'analisi automatizzata di eventi sportivi è un settore in rapida crescita nell'ambito della visione artificiale, con applicazioni che vanno dall'analisi delle performance degli atleti fino alla generazione di report utili per allenatori e scoutman. Nel contesto del beach volley, l'analisi manuale delle partite attraverso video è un processo lungo e soggetto a errori, poiché richiede la revisione di interi filmati per identificare gesti tecnici fondamentali, come servizi, attacchi e difese, e per determinare i momenti di gioco effettivo. Questa metodologia non solo richiede molto tempo, ma può anche compromettere la qualità e la coerenza dei dati raccolti.

Automatizzare questo processo rappresenta un contributo significativo per migliorare le prestazioni e le strategie degli atleti. Gli allenatori e gli scoutman hanno bisogno di strumenti che consentano di analizzare rapidamente i momenti più rilevanti della partita, eliminando i tempi morti e concentrandosi sulle azioni di gioco. Un sistema automatizzato può non solo velocizzare questo processo, ma anche standardizzare i risultati, riducendo gli errori umani e garantendo una maggiore precisione nell'analisi delle prestazioni.

Il lavoro presentato in questa tesi ha come obiettivo la realizzazione di un'applicazione che utilizza tecniche di visione artificiale per analizzare automaticamente i video delle partite di beach volley. Questo sistema è in grado di riconoscere in tempo reale i principali gesti tecnici e di visualizzarli su una timeline interattiva, fornendo agli utenti la possibilità di concentrarsi esclusivamente sulle fasi di gioco attivo. L'applicazione permette inoltre di

esportare video contenenti solo i momenti salienti, eliminando i tempi morti, e fornisce un'indicazione del punteggio in tempo reale, basata sull'analisi delle azioni di gioco.

Il contributo principale di questo progetto risiede nello sviluppo di uno strumento che rende l'analisi video delle partite di beach volley più rapida, efficiente e precisa. Questa applicazione non solo riduce significativamente il tempo richiesto per analizzare una partita, ma rappresenta anche una base solida per ulteriori espansioni, come il riconoscimento individuale dei giocatori e la generazione di report dettagliati sulle loro prestazioni. I risultati ottenuti dimostrano che l'applicazione è in grado di identificare correttamente i gesti tecnici e di gestire con successo l'eliminazione automatica dei momenti di inattività, migliorando la velocità e l'efficacia del processo di analisi.

Il lavoro di tesi è organizzato in sei capitoli principali, ciascuno dei quali affronta un aspetto specifico dello sviluppo del sistema e della sua valutazione:

- **Capitolo 1: Problema affrontato e Sistemi esistenti** Introduce le problematiche principali, le applicazioni pratiche e i requisiti del problema. Presenta una panoramica dei sistemi esistenti e delle tecnologie attuali, con un approfondimento sui principali gesti tecnici del beach volley e sugli obiettivi dello scout. Si definiscono gli obiettivi specifici della tesi e l'approccio adottato.

- **Capitolo 2: Reti Convoluzionali per Detection** Descrive l'applicazione delle reti convoluzionali (CNN) nella detection, illustrando il processo di localizzazione e classificazione degli oggetti, le architetture più comuni (YOLO, Faster R-CNN, SSD) e le tecniche di ottimizzazione, come il data augmentation e il transfer learning. Si discutono le principali metriche di valutazione e le sfide legate al rilevamento in tempo reale.

- **Capitolo 3: Progettazione dell'Applicazione** Approfondisce il preprocessing e la data augmentation applicati al dataset, descrivendo l'uso di Roboflow e le operazioni specifiche implementate. Viene

illustrata l'architettura del modello YOLO utilizzata, con la configurazione dei parametri per l'addestramento e le ottimizzazioni adottate per massimizzare le performance.

- **Capitolo 4: Estrazione delle Informazioni** Analizza il riconoscimento dei gesti tecnici e la strutturazione dei dati tramite timepoints. Si approfondisce la rilevazione dei momenti di gioco e la distinzione tra azioni consecutive per garantire una coerenza temporale. Il capitolo include diagrammi che descrivono la struttura generale dell'applicazione, i package e le classi principali.

- **Capitolo 5: Video Player** Descrive il video player sviluppato, illustrandone il flusso operativo e le funzionalità principali, come il caricamento e l'analisi dei video, il filtraggio dei gesti e la riproduzione dei soli momenti di azione. Sono approfondite funzionalità avanzate, tra cui il tracking dei giocatori e della palla, il calcolo delle traiettorie e l'esportazione dei video. Include inoltre un diagramma degli stati per rappresentare il flusso dell'interfaccia utente.

- **Capitolo 6: Risultati e Valutazione** Presenta i risultati ottenuti, descrivendo il dataset utilizzato, il protocollo di testing e le metriche di valutazione adottate (F1-Score, mAP, coverage). Vengono analizzati i risultati per ogni partita e complessivi, con una discussione approfondita sui punti di forza e debolezze del sistema. Il capitolo include un'analisi degli errori e propone prospettive per miglioramenti futuri.

Indice

Elenco delle figure

Elenco delle tabelle

Capitolo 1

Problema affrontato e Sistemi esistenti

1.1 Problematiche

L'analisi manuale dei video sportivi, in particolare delle partite di beach volley, presenta diverse sfide che limitano l'efficacia e l'efficienza del processo di scouting e valutazione delle performance. Gli allenatori e gli scoutman sono spesso costretti a rivedere interamente le registrazioni delle partite per identificare momenti di gioco rilevanti, gesti tecnici specifici (come il servizio, l'attacco o la difesa), e altri eventi chiave. Questo processo è non solo lungo e laborioso, ma anche suscettibile a errori umani, con il rischio di perdita di informazioni importanti.

Un problema fondamentale risiede nella quantità di tempo necessaria per revisionare le registrazioni. Una partita di beach volley può includere molte interruzioni e momenti di inattività che devono essere filtrati manualmente. Senza strumenti automatizzati, l'osservazione continua del video è necessaria per identificare le azioni significative, il che può richiedere ore di lavoro per singolo video. Questo porta a un aumento significativo del carico di lavoro e rende l'analisi meno sostenibile, specialmente in contesti dove è richiesto il monitoraggio costante di più atleti o squadre.

Inoltre, l'analisi manuale è influenzata dalla soggettività dell'osservatore, che può interpretare le azioni e i momenti della partita in modo diverso a seconda della propria esperienza e attenzione. Questo introduce una variabilità nei dati raccolti, che rende difficile confrontare o standardizzare le osservazioni tra diversi osservatori e partite. L'assenza di criteri di rilevamento uniformi porta a dati potenzialmente incoerenti, limitando l'accuratezza dei report sulle performance sportive.

Un'altra sfida è rappresentata dall'ampia varietà di informazioni che sarebbe ideale raccogliere da ogni partita. Per un'analisi completa, sarebbe utile tracciare non solo i gesti tecnici di ciascun giocatore, ma anche le sequenze di gioco, i tempi di azione e di pausa, e il punteggio aggiornato in tempo reale. Tuttavia, raccogliere e strutturare manualmente queste informazioni in modo coerente è impraticabile e richiede un elevato impegno di tempo e risorse.

Queste problematiche sottolineano la necessità di un sistema automatizzato capace di analizzare i video sportivi in maniera rapida, precisa e standardizzata. Un tale sistema non solo ridurrebbe il carico di lavoro degli scoutman e degli allenatori, ma aumenterebbe anche la qualità e la coerenza dei dati raccolti, permettendo di ottenere risultati comparabili e di facilitare il miglioramento delle prestazioni degli atleti.

1.2 Applicazioni pratiche

L'automazione dell'analisi video in ambito sportivo ha diverse applicazioni pratiche che possono migliorare le prestazioni degli atleti e ottimizzare le strategie di gioco. La possibilità di identificare automaticamente gesti tecnici e fasi di gioco fornisce ai coach e agli analisti dati precisi in tempi ridotti, supportando decisioni basate su dati oggettivi.

Tra le applicazioni principali vi sono la creazione di report statistici e la generazione di insight che consentono agli allenatori di rilevare tendenze e schemi di gioco. Per esempio, un sistema automatizzato può identificare qua-

li gesti tecnici sono più frequentemente utilizzati da una squadra avversaria o rilevare le aree di debolezza di un giocatore. Queste informazioni possono essere sfruttate per migliorare l'allenamento personalizzato e pianificare strategie più efficaci contro avversari specifici.

Un'altra applicazione pratica è la valutazione delle prestazioni degli atleti in tempo reale. Con un sistema di analisi video automatizzato, è possibile monitorare l'andamento di una partita e ottenere feedback istantanei su variabili chiave, come il punteggio, la durata delle azioni e il tipo di gesto tecnico. Questa capacità di analisi immediata rappresenta un vantaggio competitivo, poiché consente di apportare modifiche strategiche durante la partita basate su dati accurati.

1.3 Sistemi esistenti e tecnologie attuali

Attualmente, sono stati sviluppati diversi sistemi per l'analisi video in ambito sportivo, molti dei quali utilizzano tecniche avanzate di visione artificiale e machine learning. Tuttavia, la maggior parte dei sistemi esistenti si concentra su sport popolari come il calcio e il basket, lasciando meno attenzione a discipline come il beach volley.

Uno dei sistemi più noti è l'*Hawk-Eye*[1] (Occhio di Falco), utilizzato in sport come il tennis, il cricket e il calcio. Hawk-Eye impiega un insieme di telecamere ad alta velocità per tracciare la traiettoria della palla con estrema precisione, permettendo di rivedere azioni contestate e verificare se la palla è dentro o fuori. Nel tennis, ad esempio, le telecamere catturano immagini della palla in movimento e un software dedicato elabora i dati per generare una ricostruzione virtuale della traiettoria, fornendo ai giudici e al pubblico una visualizzazione immediata dell'esito (Figura 1.1).

Un altro esempio è il sistema *VAR*[2] (Video Assistant Referee), implementato nel calcio per assistere gli arbitri nelle decisioni critiche. Il VAR utilizza diverse telecamere distribuite sul campo per catturare prospettive multiple dell'azione. Gli assistenti video analizzano le immagini in tempo

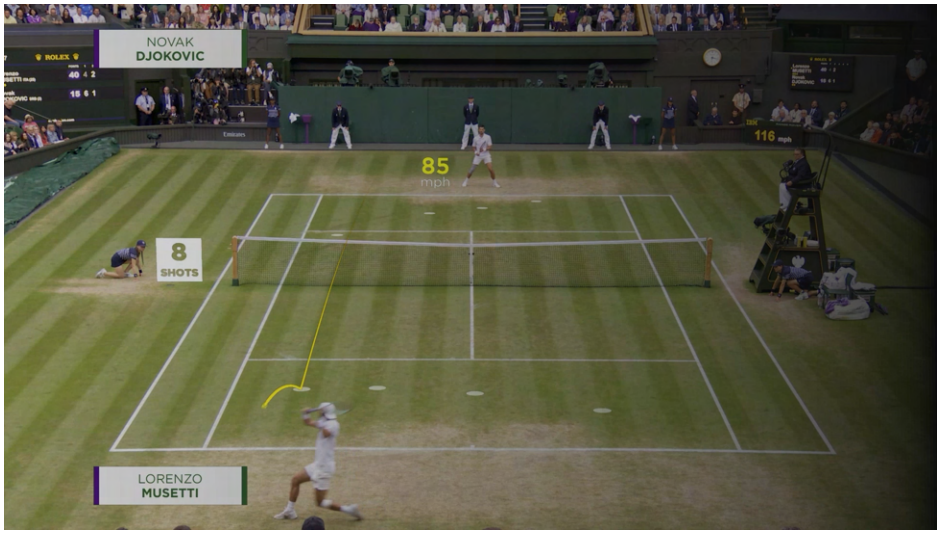


Figura 1.1: Rappresentazione del sistema Hawk-Eye nel tennis.
Fonte: *hawkeyeinnovations.com*[1]

reale, rivedendo situazioni come goal, rigori e decisioni disciplinari per garantire una maggiore accuratezza delle decisioni arbitrali (Figura 1.2 e Figura 1.3).

Nel basket, sistemi come *Second Spectrum*[3] offrono una tecnologia avanzata per l'analisi delle partite. Second Spectrum utilizza telecamere ad alta risoluzione per tracciare i movimenti di ogni giocatore e della palla, generando statistiche avanzate e visualizzazioni grafiche che supportano gli allenatori nell'analisi tattica e nella preparazione delle partite. Il sistema fornisce una rappresentazione digitale dettagliata del gioco, che consente di comprendere ogni movimento e interazione dei giocatori sul campo (Figura 1.4 e Figura 1.5).

Questi sistemi, utilizzati a livello professionale, impiegano configurazioni di più telecamere posizionate intorno al campo per tracciare i movimenti dei giocatori e della palla in tempo reale. Grazie alla visione 3D, è possibile ottenere informazioni dettagliate, come la velocità e la traiettoria della palla, oltre alla posizione precisa dei giocatori durante le fasi di gioco. Tali sistemi sono utilizzati in competizioni internazionali per migliorare le strategie di

Figura 1.2: Sala VAR della FIFA durante la Coppa del Mondo 2022, utilizzata per monitorare e analizzare le azioni di gioco in tempo reale.
Fonte: *fifa.com*[2]

squadra, supportare l'arbitraggio e fornire dati utili per l'analisi post-partita.

Tuttavia, questi sistemi multicamera, sebbene estremamente precisi, richiedono un'infrastruttura complessa e costosa, che li rende difficilmente applicabili in contesti al di fuori dell'ambito professionale.

1.4 Requisiti del problema e approccio adottato

L'analisi automatizzata si basa su video acquisiti da una singola telecamera. Questa scelta è stata determinata dalla disponibilità di un ampio database di video fornito dalla FIVB (Fédération Internationale de Volleyball)[4], contenente registrazioni ufficiali di partite di beach volley svolte in diversi tour mondiali. L'utilizzo di una singola telecamera è motivato da esigenze

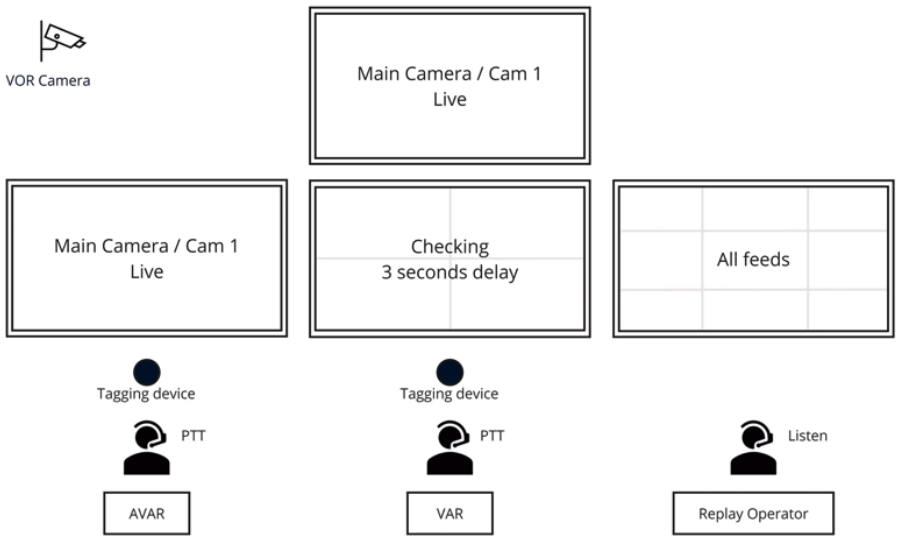


Figura 1.3: Schema di funzionamento del sistema VAR, che mostra la configurazione delle telecamere e le postazioni degli operatori.
Fonte: *fifa.com*[2]

di semplicità e portabilità, rendendo il sistema accessibile anche in contesti non professionali, senza compromettere l'efficacia dell'analisi.

La telecamera nei video FIVB è posizionata in fondo al campo, lungo il lato corto, offrendo una prospettiva che include entrambe le metà campo. Tale configurazione è stata adottata poiché consente di inquadrare l'intero campo senza la necessità di telecamere con un ampio raggio d'azione. Questo posizionamento, tradizionalmente utilizzato nelle riprese delle partite di beach volley, garantisce una visione chiara delle azioni di gioco e un'inquadratura completa dei movimenti dei giocatori e della traiettoria della palla.

Sebbene l'utilizzo di una singola telecamera comporti alcune limitazioni, come l'impossibilità di acquisire informazioni tridimensionali, le tecniche avanzate di visione artificiale e machine learning adottate dimostrano che è possibile ottenere risultati affidabili.

Questa configurazione rispetta lo standard delle riprese fornite dalla FIVB e si dimostra particolarmente adatta per applicazioni basate su database video preesistenti. Inoltre, la scelta di utilizzare un sistema con una sin-

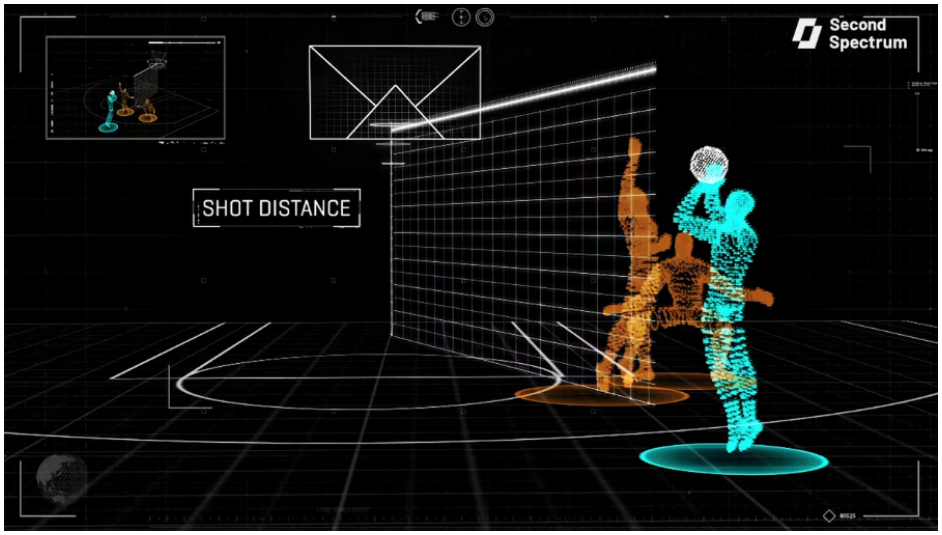


Figura 1.4: Rappresentazione grafica del sistema Second Spectrum che traccia ogni movimento dei giocatori e la posizione della palla. Questo sistema offre una comprensione completa del gioco.
Fonte: *secondspectrum.com*[3]

gola telecamera facilita l'implementazione in diversi contesti sportivi senza necessità di infrastrutture aggiuntive.

1.4.1 Principali gesti tecnici del beach volley

Nel beach volley, i principali gesti tecnici includono:

1. Serve (Servizio)
    - **Float serve**: servizio piatto, senza rotazione.
    - **Jump serve**: servizio in salto, generalmente potente e con spin.
    - **Topspin serve**: servizio in salto con rotazione per maggiore velocità e controllo.
2. Pass (Ricezione)
    - **Bump pass**: ricezione o passaggio effettuato con l'avambraccio.

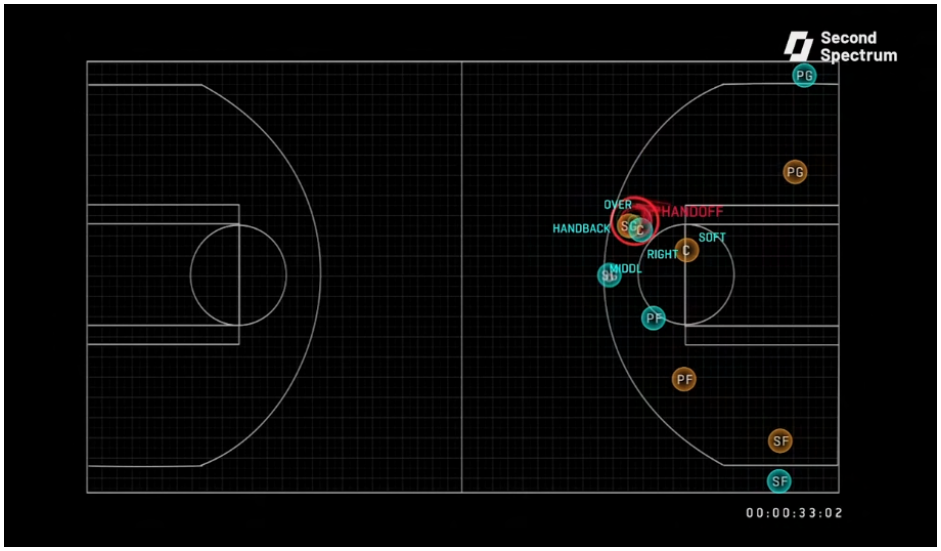


Figura 1.5: Schema tattico generato dal sistema Second Spectrum, che mostra la disposizione e i movimenti dei giocatori.
Fonte: *secondspectrum.com*[3]

- **Overhand pass**: passaggio o ricezione eseguito con le mani sopra la testa.

3. Set (Alzata)
   - **Hand set**: alzata tradizionale eseguita con le mani sopra la testa.
   - **Bump set**: alzata eseguita con l'avambraccio, utile in situazioni difficili.
4. Attack (Attacco)
   - **Spike**: colpo d'attacco potente, spesso in salto.
   - **Roll shot**: attacco morbido e arcuato per evitare il muro.
   - **Cut shot**: colpo diagonale, generalmente vicino alla rete.
   - **Line shot**: colpo lungo e parallelo alla linea laterale.
5. Block (Muro)

- **Stuff block**: muro diretto che respinge la palla nel campo avversario.
- **Soft block**: muro che smorza la palla per facilitarne il recupero.

6. Defense (Difesa)
   - **Dig**: salvataggio su un attacco, spesso basso e vicino al suolo.
   - **Pancake**: tecnica in cui la mano piatta viene posizionata sul terreno per far rimbalzare la palla.
   - **Sprawl**: tuffo rapido per raggiungere una palla difficile.
7. Other (Altri gesti)
   - **Tip**: tocco morbido per piazzare la palla nel campo avversario.
   - **Joust**: contesa della palla sopra la rete da parte di due giocatori.

   Questi termini sono usati internazionalmente e sono comuni sia tra giocatori che arbitri.

1.4.2 Obiettivi e annotazioni dello scout nel beach volley

Lo scout in una partita di beach volley ha il compito di raccogliere dati oggettivi per supportare decisioni tattiche e valutazioni tecniche. Gli obiettivi principali e ciò che viene annotato sono descritti di seguito.

Gli Obiettivi dello scout di una partita di beach volley includono:

- **Analisi delle performance individuali**: Valutare l'efficacia dei gesti tecnici (es. servizi, attacchi, ricezioni) e identificare punti di forza e debolezze dei giocatori.
- **Analisi delle strategie di squadra**: Monitorare le scelte tattiche (es. zone di attacco preferite, tipo di servizio) e verificare la coesione e la comunicazione tra i giocatori.

- **Studio dell'avversario**: Rilevare schemi ricorrenti negli attacchi o nei servizi e individuare i punti deboli degli avversari.
- **Supporto alla pianificazione tattica**: Fornire informazioni in tempo reale per aggiustamenti durante la partita e preparare strategie per match futuri basate sui dati raccolti.

Lo scout può essere in presa diretta o visionando un filmato di una partita e include l'annotazione metodica dei seguenti gesti:

1. **Servizio**
   - Tipo di servizio (float, topspin, salto).
   - Zona di atterraggio nel campo avversario.
   - Efficacia (ace, errore, ricezione positiva o negativa).
2. **Ricezione**
   - Posizionamento del giocatore in campo.
   - Qualità della ricezione (positiva, neutra, negativa).
   - Direzione del passaggio seguente.
3. **Attacco**
   - Tipo di attacco (spike, roll shot, cut shot, line shot).
   - Zona del campo avversario colpita.
   - Risultato (punto, difesa avversaria, errore, muro subito).
4. **Muro**
   - Numero di muri effettuati.
   - Efficacia (muro punto, tocco positivo, errore).
   - Posizionamento rispetto all'attacco avversario.
5. **Difesa**

- Numero di dig effettuati.
- Qualità del recupero (positivo o negativo).
- Successo nella conversione in un contrattacco.

6. **Errori**

   - Errori non forzati (servizio sbagliato, attacco fuori campo).
   - Errori forzati dall'avversario (es. colpi di precisione che obbligano a difese difficili).

7. **Pattern tattici**

   - Frequenza e zone di attacco preferite dagli avversari.
   - Strategie di servizio e ricezione delle squadre.
   - Adattamenti tattici nel corso del set.

I dati raccolti vengono analizzati per creare report dettagliati post-partita, fornire feedback ai giocatori, identificare aree di miglioramento specifiche, pianificare sessioni di allenamento mirate, preparare un incontro contro un avversario mai affrontato. Lo scouting efficace richiede strumenti digitali o schede cartacee ben organizzate, insieme a una comprensione approfondita del gioco e delle sue dinamiche.

1.4.3 Obiettivi della Tesi

L'obiettivo di questa tesi è automatizzare parte del processo di scouting utilizzando tecniche di visione artificiale e intelligenza artificiale. In particolare, il sistema proposto mira a:

- **Rilevazione automatica dei gesti tecnici**: Identificare e classificare con precisione i gesti come servizio, bagher, attacco e muro, riducendo il carico di lavoro manuale dello scout.

- **Analisi delle traiettorie della palla**: Calcolare e visualizzare in tempo reale le traiettorie di battute e attacchi, stimando la velocità e fornendo informazioni utili per l'analisi tattica.
- **Tracciamento dei giocatori**: Monitorare i movimenti dei giocatori durante l'intera partita, visualizzandoli su una minimappa 2D per supportare l'analisi delle strategie di gioco.
- **Creazione automatica di report**: Generare report dettagliati post-partita contenenti statistiche come velocità media delle battute, distribuzione degli attacchi e percentuale di successo delle ricezioni.
- **Riduzione del tempo di scouting**: Automatizzare i compiti più ripetitivi e dispendiosi in termini di tempo, consentendo agli allenatori di concentrarsi sull'interpretazione dei dati piuttosto che sulla loro raccolta.

Il sistema proposto rappresenta un passo avanti nella digitalizzazione dello scouting, fornendo strumenti avanzati che migliorano l'efficienza e l'accuratezza dell'analisi delle partite. Alcuni aspetti più avanzati o personalizzati verranno discussi nella sezione dedicata agli sviluppi futuri.

Capitolo 2

Reti Convoluzionali per Detection

2.1 Applicazione delle CNN nella detection

L'object detection è una tecnica di visione artificiale che combina la localizzazione e la classificazione di oggetti all'interno delle immagini. A differenza della semplice classificazione delle immagini, che assegna un'etichetta all'intera immagine, l'object detection mira a localizzare specifici oggetti delineandoli con una "bounding box" e assegnando a ciascuno una categoria semantica. Ciò consente non solo di identificare il tipo di oggetti presenti, ma anche di determinarne la posizione all'interno dell'immagine.

2.1.1 Localizzazione e classificazione degli oggetti

L'object detection si compone di due compiti principali: la localizzazione e la classificazione degli oggetti. La localizzazione mira a determinare la posizione degli oggetti in un'immagine mediante bounding box, mentre la classificazione identifica la categoria di ciascun oggetto. La combinazione di questi due processi consente a un modello di object detection di stimare simultaneamente la posizione e la tipologia degli oggetti in una o più immagini.

La Figura 2.1 illustra la differenza tra classificazione delle immagini, object detection e segmentazione semantica, mostrando come queste tecniche si distinguano in termini di livello di dettaglio e di localizzazione degli oggetti.

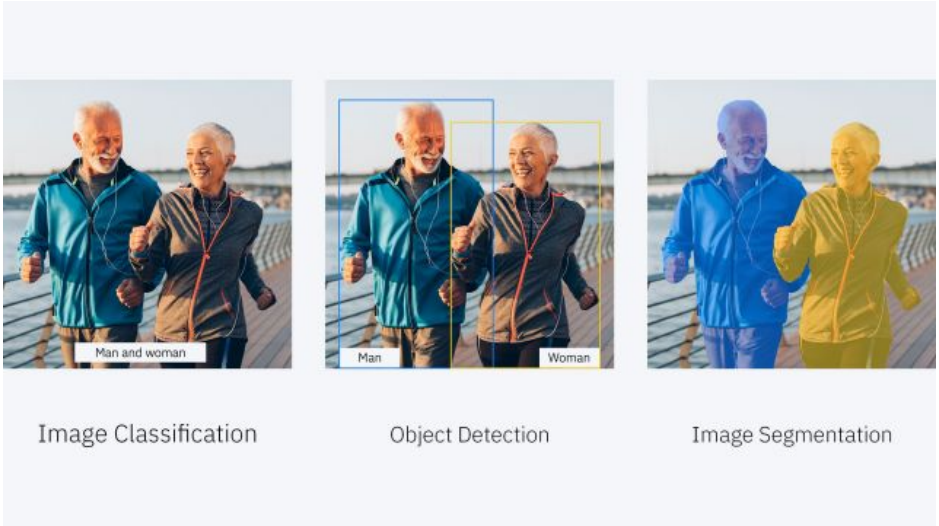


Figura 2.1: Differenza tra classificazione delle immagini, object detection e segmentazione.

Fonte: *ibm.com*[5]

2.1.2 Processo di elaborazione delle immagini

Nella visione artificiale, le immagini vengono rappresentate come funzioni continue su un piano di coordinate 2D, generalmente rappresentate come $f(x, y)$. Durante la digitalizzazione, le immagini subiscono due processi principali: il *campionamento* e la *quantizzazione*. Questi processi trasformano la funzione continua dell'immagine in una griglia discreta di pixel, permettendo al computer di segmentare l'immagine in regioni discrete in base alla somiglianza visiva e alla vicinanza dei pixel.

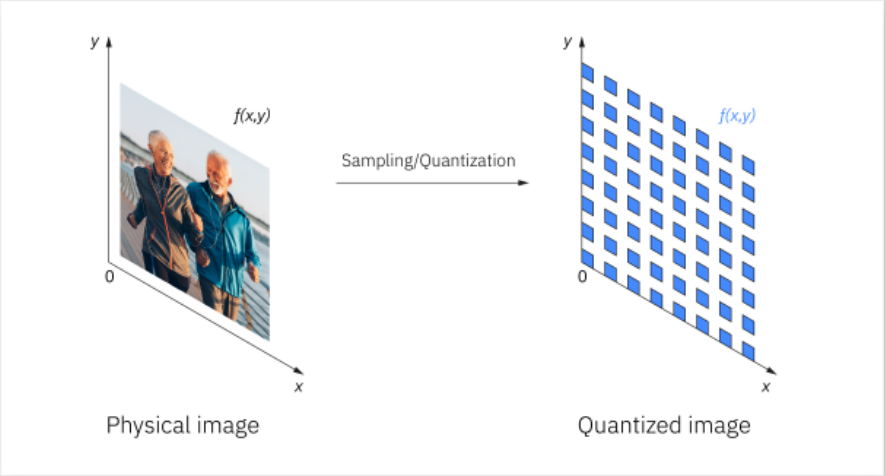


Figura 2.2: Conversione da immagine fisica a immagine quantizzata tramite campionamento e quantizzazione.

Fonte: *ibm.com*[5]

2.1.3 Architettura dei modelli di object detection

I modelli di object detection basati su CNN seguono generalmente un'architettura costituita da tre componenti principali: *backbone*, *neck* e *head*. Il **backbone** è responsabile dell'estrazione delle feature dall'immagine di input, producendo mappe di feature a varie risoluzioni. Il **neck** aggrega queste mappe per creare rappresentazioni più complesse, che vengono poi inviate al **head**, il quale genera le bounding box e le categorie per ciascun oggetto rilevato.

La Figura 2.3 mostra un esempio di architettura per l'object detection, che comprende le fasi di estrazione delle feature, aggregazione e predizione finale.

2.1.4 Metriche di valutazione

Le prestazioni dei modelli di object detection sono misurate tramite metriche come l'*Intersection over Union* (IoU). L'IoU confronta l'area di so-

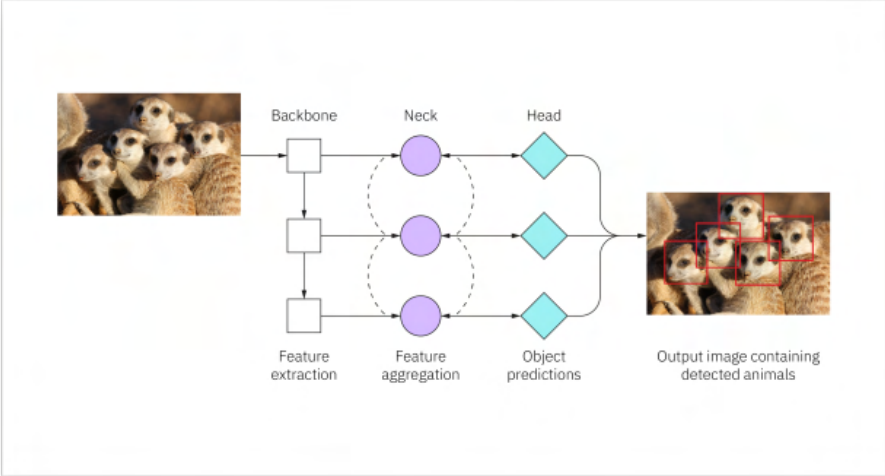


Figura 2.3: Architettura di base di un modello di object detection, composta da backbone, neck e head.

Fonte: *ibm.com*[5]

vrapposizione tra la bounding box predetta e la bounding box di riferimento (ground truth) con l'unione delle due aree, come mostrato nella Figura 2.5.

$$\text{IoU} = \frac{\text{Area di sovrapposizione}}{\text{Area di unione}} \tag{2.1}$$

Questa metrica consente di valutare l'accuratezza della localizzazione del modello.

Esistono altre metriche comunemente utilizzate per valutare le prestazioni dei modelli di object detection, in particolare la *mean Average Precision* (mAP) e il *recall*.

2.1.4.1 Recall

Il **recall** è la misura della capacità del modello di rilevare tutti gli oggetti rilevanti (ovvero, gli oggetti che dovrebbero essere identificati). Si calcola come:

$$\text{Recall} = \frac{\text{True Positives}}{\text{True Positives} + \text{False Negatives}}$$

dove:

- *True Positives (TP)* indica il numero di oggetti correttamente rilevati dal modello.
- *False Negatives (FN)* rappresenta il numero di oggetti reali non rilevati dal modello.

Per comprendere meglio il concetto di recall, è utile fare riferimento alla **matrice di confusione** (confusion matrix), che rappresenta le prestazioni di un modello classificando ogni previsione in una delle seguenti categorie:

- **True Positives (TP)**: Oggetti rilevati correttamente.
- **False Positives (FP)**: Oggetti rilevati ma errati.
- **False Negatives (FN)**: Oggetti presenti ma non rilevati.
- **True Negatives (TN)**: Oggetti correttamente ignorati (non rilevanti e non rilevati).

La matrice di confusione consente di visualizzare il bilanciamento tra queste categorie, come mostrato nella Tabella 2.1.

	Predicted Positive	**Predicted Negative**
Actual Positive	True Positives (TP)	False Negatives (FN)
Actual Negative	False Positives (FP)	True Negatives (TN)

Tabella 2.1: Esempio di matrice di confusione utilizzata per calcolare metriche come recall e precision.

Un valore alto di **recall** indica che il modello è in grado di rilevare la maggior parte degli oggetti rilevanti, riducendo al minimo gli oggetti persi (FN). Tuttavia, un alto recall da solo non garantisce una buona precisione. È quindi importante considerare anche la *Precision*, che valuta la proporzione di oggetti correttamente rilevati rispetto al totale delle previsioni fatte.

2.1.4.2 mean Average Precision (mAP)

La **mean Average Precision** (mAP) è una misura che combina *precision* e *recall* su più soglie di confidenza e diverse classi. Essa viene calcolata come media delle *Average Precision* (AP) per ciascuna classe. L'**Average Precision** rappresenta l'area sotto la curva Precision-Recall per una determinata classe, come mostrato nella Figura 2.4.

$$\text{AP} = \int_0^1 \text{Precision}(r)\, dr$$

dove Precision(r) è la precisione al livello di *recall* r. La **mAP** è quindi la media delle AP di tutte le classi:

$$\text{mAP} = \frac{1}{N} \sum_{i=1}^{N} \text{AP}_i$$

dove N è il numero di classi e AP_i è l'Average Precision per la classe i. In pratica, la mAP valuta la capacità del modello di predire correttamente la posizione e la classe degli oggetti, e valori più alti di mAP indicano prestazioni migliori.

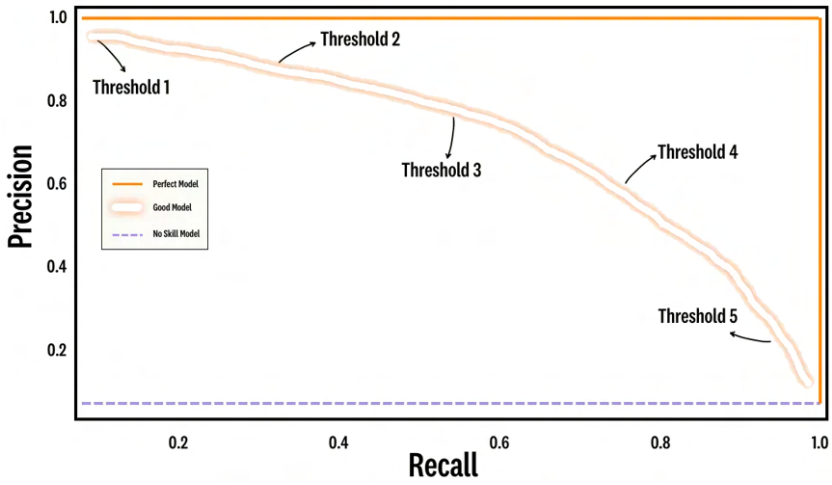


Figura 2.4: Esempio di curva Precision-Recall per una determinata classe. L'area sotto la curva rappresenta l'Average Precision (AP) per quella classe. Fonte: *kili-technology.com*[6]

Per riassumere, sia la **mAP** che il **recall** sono metriche cruciali per valutare l'efficacia dei modelli di object detection, specialmente in scenari complessi e con oggetti sovrapposti. La mAP permette di comprendere la performance globale del modello su più classi e a diversi livelli di confidenza, mentre il recall indica la capacità del modello di rilevare la totalità degli oggetti presenti.

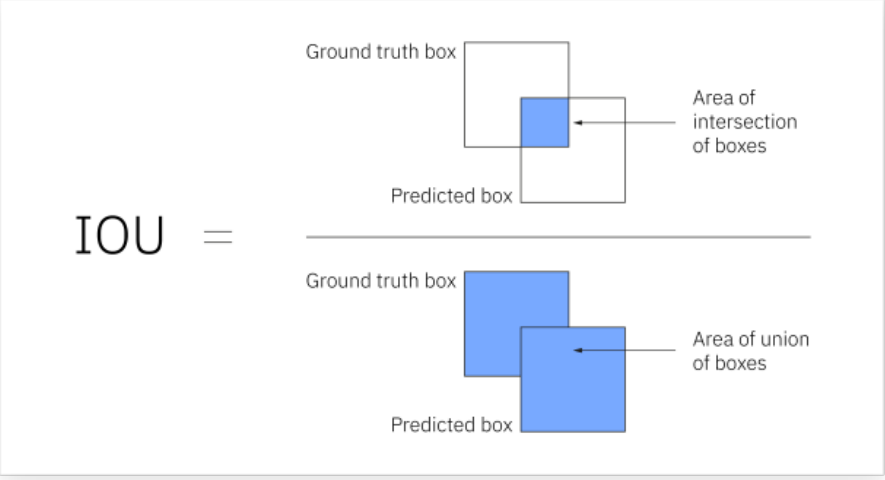


Figura 2.5: Illustrazione della metrica IoU per valutare l'accuratezza della bounding box predetta rispetto alla ground truth.

Fonte: *ibm.com*[5]

2.1.5 Applicazioni pratiche

L'object detection trova applicazione in numerosi settori. Nelle auto a guida autonoma, modelli come YOLO[7] sono preferiti per la loro rapidità, permettendo di identificare veicoli, pedoni e segnali stradali in tempo reale. Nella sorveglianza, l'object detection viene utilizzata per monitorare oggetti sospetti, come armi o veicoli non autorizzati, contribuendo a migliorare la sicurezza. In campo medico, modelli di object detection assistono nella rilevazione di anomalie in immagini diagnostiche, aiutando a identificare condizioni mediche critiche.

2.2 Architetture comuni

Nel campo dell'object detection, diverse architetture di reti neurali convoluzionali (CNN) sono state sviluppate per bilanciare l'accuratezza e la velocità del rilevamento. Tra le architetture più comunemente utilizzate troviamo **YOLO**[7] (You Only Look Once), **Faster R-CNN**[8] e **SSD**[9] (Single Shot MultiBox Detector). Ciascuna di queste architetture presenta caratteristiche uniche che la rendono adatta a specifiche applicazioni di object detection.

2.2.1 YOLO (You Only Look Once)

YOLO[7] (You Only Look Once) è una delle architetture di object detection più popolari e avanzate, progettata per eseguire il rilevamento in un'unica fase, combinando la localizzazione e la classificazione degli oggetti in un'unica rete neurale. Questo approccio consente a YOLO di eseguire il rilevamento in tempo reale, il che lo rende particolarmente adatto per applicazioni che richiedono velocità elevate, come la guida autonoma, la sorveglianza in tempo reale e altre applicazioni critiche.

L'architettura YOLO divide un'immagine in una griglia e, per ciascuna cella, predice un numero fisso di bounding box e i punteggi di confidenza associati, che rappresentano la probabilità che una bounding box contenga un oggetto. Ogni nuova versione di YOLO ha introdotto miglioramenti in termini di accuratezza e velocità, con YOLO11 come l'ultima versione attualmente disponibile. YOLO11 presenta ulteriori ottimizzazioni sia nell'architettura che nelle tecniche di addestramento, continuando a migliorare le prestazioni rispetto alle versioni precedenti, come mostrato in Figura 2.6.

- **YOLOv2 e YOLOv3**: Queste versioni hanno introdotto miglioramenti nella localizzazione e nella capacità di rilevare oggetti di diverse dimensioni, grazie all'uso di anchor boxes multipli.

- **YOLOv4 e YOLOv5**: Queste versioni hanno ulteriormente migliorato l'accuratezza e la velocità, introducendo tecniche avanzate co-

me il bag of freebies e il bag of specials, che includono strategie di ottimizzazione e data augmentation.

- **YOLOv7 e YOLOv8**: Hanno portato miglioramenti nell'architettura della rete per una maggiore efficienza computazionale, ottimizzando le prestazioni anche su dispositivi con risorse limitate.

- **YOLOv10 e YOLO11**: Le ultime versioni di YOLO hanno ulteriormente incrementato l'accuratezza e ridotto la latenza, rendendolo uno degli algoritmi di object detection in tempo reale più efficienti disponibili attualmente.

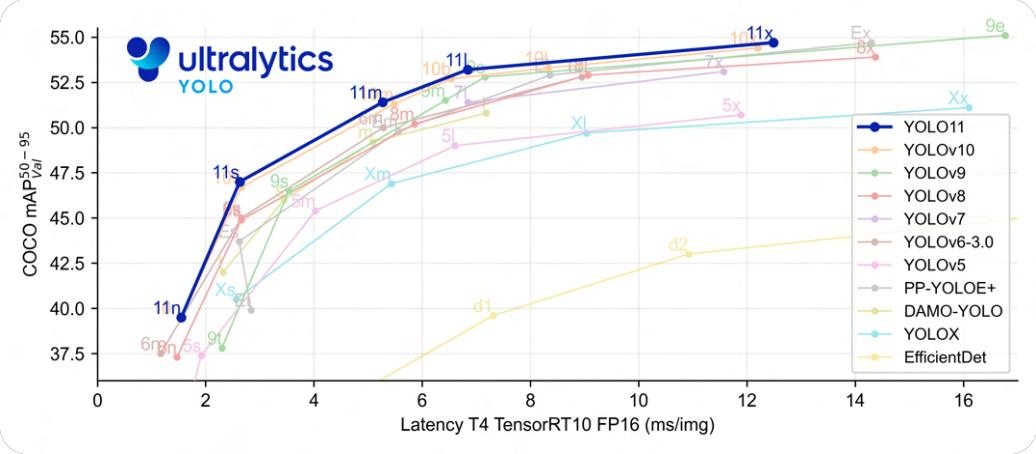


Figura 2.6: Prestazioni delle diverse versioni di YOLO in termini di mAP (COCO) e latenza su TensorRT.

Fonte: *docs.ultralytics.com*[10]

YOLO è un'architettura versatile e flessibile, in grado di eseguire una varietà di task oltre alla semplice object detection:

- **Detect**: Il task principale di YOLO è la rilevazione di oggetti, identificando la posizione e la classe di ciascun oggetto in un'immagine. Questo task è particolarmente utile in applicazioni come la sorveglianza, la sicurezza e la guida autonoma.

Figura 2.7: Esempio di object detection con YOLO.
Fonte: *docs.ultralytics.com*[11]

- **Segment**: YOLO supporta anche la segmentazione, che permette di distinguere le aree specifiche occupate dagli oggetti nell'immagine a livello di pixel. Questo task è utile in applicazioni mediche, robotica e mappatura autonoma.

Figura 2.8: Esempio di segmentazione con YOLO.
Fonte: *docs.ultralytics.com*[12]

- **Classify**: YOLO può essere utilizzato per la classificazione di immagini intere, assegnando un'etichetta all'immagine senza la necessità di bounding box. La classificazione è utilizzata in sistemi di analisi per catalogare immagini su larga scala.

- **Pose**: Con YOLO è possibile eseguire il pose estimation, ovvero la stima della posa degli oggetti, che identifica le posizioni di punti chiave (come articolazioni o componenti di un oggetto) per applicazioni in realtà aumentata, animazione e sport.

Figura 2.9: Esempio di classificazione con YOLO.
Fonte: *docs.ultralytics.com*[13]

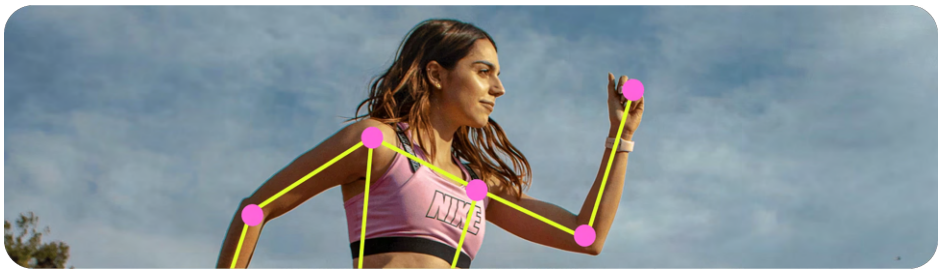


Figura 2.10: Esempio di pose estimation con YOLO.
Fonte: *docs.ultralytics.com*[14]

- **OBB (Oriented Bounding Boxes)**: YOLO supporta anche il rilevamento tramite bounding box orientati (OBB), che permette di rilevare oggetti in immagini dove la rotazione è rilevante, come nelle immagini aeree o nelle applicazioni di sorveglianza.

Nell'applicazione descritta in questo documento, si è scelto di utilizzare l'architettura YOLO per eseguire l'object detection, sfruttando le sue capacità di rilevamento in tempo reale. Sebbene i dettagli implementativi specifici saranno discussi in un capitolo successivo, questa scelta consente di ottenere prestazioni elevate e un'efficace identificazione degli oggetti rilevanti nelle immagini analizzate.

2.2.2 Faster R-CNN

Faster R-CNN è un'architettura di object detection a due fasi sviluppata per migliorare la velocità del modello R-CNN originale. Nella prima fase,

Figura 2.11: Esempio di Oriented Bounding Boxes (OBB) con YOLO.
Fonte: *docs.ultralytics.com*[15]

una rete chiamata *Region Proposal Network* (RPN) genera una serie di region proposals (regioni candidate) che potrebbero contenere oggetti. Nella seconda fase, queste region proposals vengono passate attraverso una rete convoluzionale per la classificazione e la raffinazione delle bounding box.

L'uso della RPN rende Faster R-CNN significativamente più veloce rispetto alle versioni precedenti come R-CNN e Fast R-CNN, mantenendo un'alta accuratezza. Tuttavia, essendo una rete a due fasi, Faster R-CNN è generalmente più lento rispetto a YOLO, rendendolo più adatto a compiti dove l'accuratezza è prioritaria rispetto alla velocità, come l'analisi di immagini statiche.

2.2.3 SSD (Single Shot MultiBox Detector)

SSD (Single Shot MultiBox Detector) è un'altra architettura di object detection a singola fase, progettata per bilanciare velocità e accuratezza. A differenza di YOLO, SSD utilizza feature maps di diverse risoluzioni per rilevare oggetti di varie dimensioni, aumentando la capacità del modello di identificare oggetti piccoli e grandi in una singola immagine.

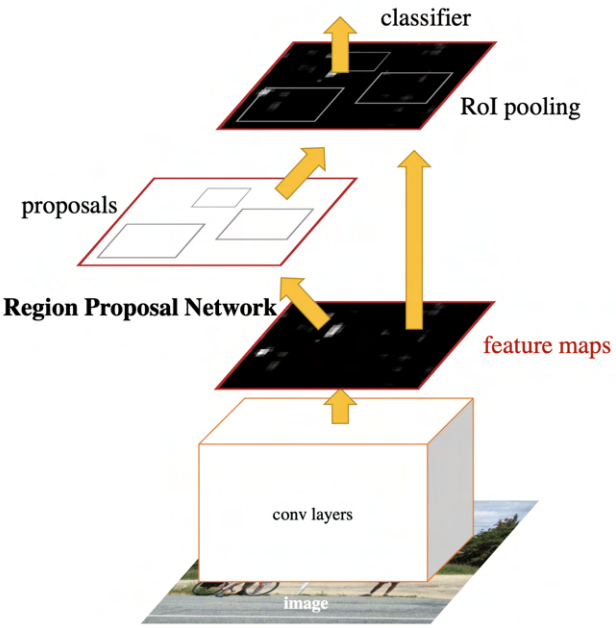
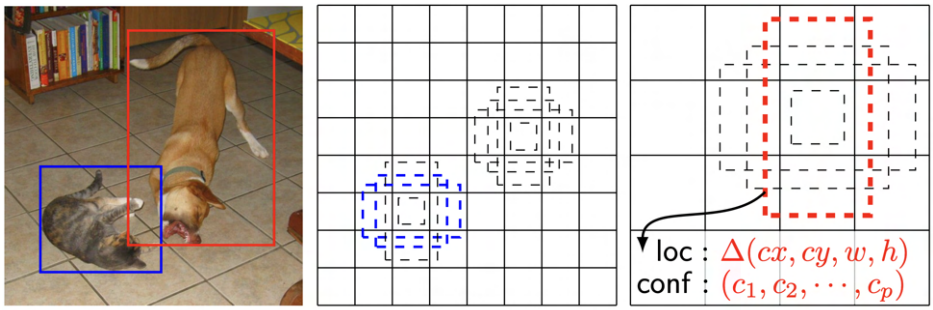


Figura 2.12: Schema dell'architettura Faster R-CNN con la Region Proposal Network.
Fonte: *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks*[8]

SSD produce un insieme di bounding box e categorie per ogni cella della griglia in ciascuna delle sue feature maps. Questa struttura multiscalare permette a SSD di ottenere buone prestazioni in termini di precisione e velocità, rendendolo particolarmente utile per applicazioni mobili o in tempo reale. Versioni migliorate di SSD, come MobileNet-SSD, sono state sviluppate per funzionare su dispositivi con risorse limitate, mantenendo prestazioni elevate.

2.3 Conclusione sulle architetture

Le architetture presentate in questa sezione, **YOLO**, **Faster R-CNN** e **SSD**, rappresentano le principali soluzioni per l'object detection. Ciascuna di esse offre vantaggi e limitazioni specifiche, rendendole più o meno adatte a determinati scenari.



(a) Image with GT boxes (b) 8×8 feature map (c) 4×4 feature map

Figura 2.13: Schema dell'architettura SSD che utilizza feature maps di diverse risoluzioni per la detection.

Fonte: *SSD: Single Shot MultiBox Detector*[9]

La scelta dell'architettura **YOLO** per l'applicazione descritta in questo documento è motivata da diverse considerazioni:

- **Velocità**: YOLO è stato progettato per l'object detection in tempo reale, offrendo tempi di inferenza estremamente rapidi rispetto a Faster R-CNN, che richiede una fase aggiuntiva di region proposal. Questa caratteristica è essenziale per applicazioni come l'analisi video in tempo reale nel contesto delle partite di beach volley.

- **Semplicità e flessibilità**: La natura a singola fase di YOLO semplifica l'integrazione del modello nell'applicazione, riducendo la complessità computazionale e ottimizzando l'uso delle risorse hardware disponibili.

- **Prestazioni competitive**: Sebbene SSD offra anch'essa una struttura a singola fase, YOLO si distingue per un miglior equilibrio tra velocità e accuratezza, specialmente nelle sue versioni più recenti come YOLO11[10].

- **Supporto per task aggiuntivi**: YOLO non solo eccelle nell'object detection, ma supporta anche task avanzati come segmentation, pose estimation e OBB (Oriented Bounding Boxes), ampliando le possibilità di sviluppo futuro per l'applicazione.

In conclusione, la scelta di YOLO risponde in modo ottimale alle esigenze del progetto, dove la velocità e la precisione sono fattori critici. La sua flessibilità e la continua evoluzione attraverso nuove versioni lo rendono una soluzione ideale per applicazioni che richiedono alta efficienza e scalabilità.

2.4 Ottimizzazione e precisione delle CNN

L'ottimizzazione delle reti neurali convoluzionali (CNN) è fondamentale per migliorare l'accuratezza e l'efficienza dei modelli di object detection. Per ottenere risultati di alta qualità nella rilevazione degli oggetti, è possibile applicare diverse tecniche di ottimizzazione. Queste includono la **data augmentation**, la modifica degli **iperparametri** e il **transfer learning**. Ogni tecnica svolge un ruolo chiave nel migliorare la performance del modello, aumentando la capacità della rete di generalizzare a nuovi dati e riducendo l'overfitting.

2.4.1 Data Augmentation

La **data augmentation** è una tecnica che consiste nel generare nuove immagini a partire da quelle già esistenti, applicando trasformazioni come rotazioni, traslazioni, variazioni di luminosità e riflessioni. Queste modifiche aumentano la varietà del dataset senza la necessità di raccogliere nuove immagini, migliorando così la capacità del modello di riconoscere gli oggetti in condizioni diverse.

Nell'ambito della detection, alcune tecniche di data augmentation specifiche includono:

- **Random Cropping**: Tagliare sezioni casuali dell'immagine per simulare la presenza di oggetti parzialmente visibili.
- **Scale Jittering**: Modificare la scala degli oggetti per aiutare il modello a riconoscere oggetti di diverse dimensioni.

- **Color Jittering**: Alterare casualmente la luminosità, il contrasto e la saturazione dei colori per rendere il modello robusto a variazioni di luce.
- **Flip and Rotation**: Applicare riflessioni e rotazioni per aumentare la variabilità delle posizioni degli oggetti.

La data augmentation è particolarmente efficace nel ridurre l'overfitting, poiché introduce variazioni nei dati di addestramento che impediscono alla rete di memorizzare specificamente le immagini originali.

2.4.2 Modifica degli iperparametri

Gli **iperparametri** sono valori che influenzano l'apprendimento della rete e devono essere selezionati prima dell'addestramento. Tra gli iperparametri comuni troviamo il **learning rate**, la **dimensione del batch**, il **numero di epoche**, e i parametri specifici del modello come il numero di filtri nelle convolutional layers. La modifica degli iperparametri può avere un impatto significativo sulla precisione e la stabilità del modello.

- **Learning Rate**: Un valore di learning rate più alto accelera l'apprendimento, ma può causare instabilità e far divergere il modello; un valore più basso consente una convergenza più stabile ma richiede più tempo.
- **Batch Size**: Dimensioni del batch maggiori possono stabilizzare l'aggiornamento dei pesi, mentre dimensioni più piccole possono aumentare la variabilità e consentire al modello di esplorare diverse direzioni durante l'apprendimento.
- **Number of Epochs**: L'aumento del numero di epoche consente al modello di apprendere più a lungo dai dati, ma aumenta anche il rischio di overfitting.

La regolazione fine degli iperparametri è spesso eseguita tramite tecniche come la **grid search** o la **random search**, che testano diverse combinazioni di valori per identificare i parametri ottimali.

2.4.3 Transfer Learning

Il **transfer learning** è una tecnica che consente di sfruttare i pesi pre-addestrati di un modello su un grande dataset, come ImageNet, e applicarli a un nuovo compito di detection con un dataset più specifico e ridotto. In questo modo, la rete può beneficiare delle rappresentazioni apprese da un grande corpus di immagini generiche, migliorando l'accuratezza anche con dati limitati.

Nel transfer learning per la detection, di solito si "congela" (blocca) una parte della rete, come le prime convolutional layers, e si addestra solo la parte finale. Questo approccio permette di ridurre i tempi di addestramento e di ottenere una migliore generalizzazione. Per i modelli di detection, il transfer learning è particolarmente efficace quando il nuovo dataset ha somiglianze con il dataset originale usato per l'addestramento iniziale, in quanto le feature apprese sono trasferibili.

L'uso del transfer learning è comune nei modelli di object detection avanzati e rappresenta un metodo efficiente per ottenere alta precisione senza la necessità di grandi risorse computazionali.

2.5 Sfide e limitazioni

Nonostante i progressi significativi delle reti neurali convoluzionali (CNN) applicate alla object detection, esistono ancora diverse sfide e limitazioni. Queste problematiche influenzano le prestazioni e l'applicabilità dei modelli di detection in scenari reali, dove spesso sono richiesti tempi di elaborazione rapidi, alta precisione e un uso efficiente delle risorse hardware. Di seguito si esplorano le principali sfide e limitazioni legate all'uso delle CNN per la detection.

2.5.1 Rilevamento in tempo reale

Una delle sfide principali nell'applicazione delle CNN per la detection è il **rilevamento in tempo reale**. Per molte applicazioni, come la guida autonoma, la sorveglianza e l'analisi sportiva, è essenziale che il modello rilevi e classifichi gli oggetti istantaneamente, senza ritardi significativi.

Per ottenere rilevamenti in tempo reale, è necessario bilanciare la complessità dell'architettura con la velocità di esecuzione. Modelli avanzati, come YOLO, sono stati progettati specificamente per questo scopo, ma ottenere velocità elevate richiede compromessi in termini di accuratezza. Inoltre, l'uso di hardware specializzato, come le GPU o i dispositivi edge (es. TPU o VPU), diventa fondamentale per eseguire le operazioni di inferenza con bassa latenza. Tuttavia, anche con questi strumenti, la latenza e il consumo energetico restano sfide importanti in contesti dove le risorse computazionali sono limitate.

2.5.2 Riconoscimento di azioni complesse

Il **riconoscimento di azioni complesse** rappresenta un'altra sfida significativa per i modelli di detection basati su CNN. Mentre le CNN eccellono nel rilevare oggetti statici, il riconoscimento di azioni complesse o sovrapposte è molto più difficile.

Per azioni che coinvolgono sequenze di movimenti (es. sport, interazioni sociali o gesti), i modelli di detection devono spesso integrare informazioni temporali oltre alla singola immagine. Questo richiede l'uso di reti ricorrenti (RNN) o modelli basati su Transformer, oppure tecniche di analisi temporale come LSTM (Long Short-Term Memory). Tuttavia, combinare queste tecniche con le CNN aumenta la complessità computazionale e può introdurre ritardi significativi. Inoltre, la sovrapposizione di più azioni e la presenza di occlusioni possono ridurre ulteriormente l'accuratezza del modello.

2.5.3 Limitazioni hardware

Le **limitazioni hardware** rappresentano una barriera importante per l'implementazione di modelli di detection ad alte prestazioni. Le CNN richiedono grandi quantità di potenza computazionale e memoria per elaborare i dati, rendendo difficile l'implementazione su dispositivi con risorse limitate, come smartphone, telecamere di sorveglianza o dispositivi IoT.

L'esecuzione di modelli di detection su dispositivi con hardware limitato spesso richiede compromessi, come l'uso di modelli compressi (es. pruned o quantized models), che riducono la complessità della rete a scapito della precisione. Anche l'utilizzo di framework ottimizzati per l'inferenza, come TensorRT o TFLite, può aiutare a migliorare le prestazioni su dispositivi con risorse limitate. Tuttavia, questi approcci non sono sempre sufficienti per applicazioni che richiedono alta accuratezza e tempi di risposta rapidi.

Inoltre, i costi associati a hardware avanzato come le GPU e i chip specializzati (TPU, VPU) limitano la scalabilità di queste soluzioni. Di conseguenza, la scelta dell'hardware rappresenta spesso un compromesso tra costo, prestazioni e scalabilità, specialmente per applicazioni su larga scala.

Capitolo 3

Progettazione dell'Applicazione

3.1 Preprocessing e Data Augmentation

Il dataset iniziale è composto da 230 immagini, con un totale di 1142 annotazioni distribuite su 7 classi: **player**, **ball**, **serve**, **bagher**, **set**, **spike** e **block**. Prima dell'applicazione delle tecniche di preprocessing e data augmentation, tutte le immagini presentavano annotazioni complete, senza esempi mancanti o nulli.

Numero di Immagini	**Annotazioni Totali**	**Classi**
230	1142	7

Tabella 3.1: Statistiche del dataset iniziale

3.1.1 Utilizzo di Roboflow

Per la preparazione e la gestione del dataset, è stato utilizzato *Roboflow*[16], una piattaforma che consente di applicare operazioni di preprocessing e data augmentation in modo rapido ed efficiente. Roboflow è stato scelto per la sua compatibilità con YOLO, il framework adottato per l'addestramento del modello, e per la possibilità di esportare direttamente i dati in formati supportati da YOLO.

3.1.2 Operazioni di Preprocessing e Data Augmentation

Per migliorare la varietà e la robustezza del dataset, sono state applicate le seguenti operazioni tramite Roboflow:

- **Preprocessing**:
  - Auto-orientamento delle immagini per garantire una corretta disposizione.
  - Ridimensionamento uniforme delle immagini a una dimensione standard di 640×640 pixel, scelto ottimizzare il bilanciamento tra la risoluzione delle immagini e le risorse computazionali richieste per l'addestramento e l'inferenza.
- **Data Augmentation**:
  - **Rotazione**: le immagini sono state ruotate casualmente tra $-15°$ e $+15°$ per simulare diverse angolazioni della telecamera.
  - **Modifiche di luminosità**: le immagini sono state elaborate per includere variazioni nella luminosità tra il -15% e il $+15\%$.
  - **Aggiunta di rumore**: è stato introdotto rumore fino all'1.01% dei pixel per simulare condizioni di acquisizione reali.

3.1.3 Esempio di Immagine Annotata

Un esempio di immagine annotata dal dataset utilizzato è mostrato in Figura 3.1. Le annotazioni includono bounding box per diverse classi rilevanti, come **player**, **ball**, **serve**, **bagher**, **set**, **spike**, e **block**. Questa rappresentazione fornisce al modello le informazioni necessarie per addestrarsi efficacemente a riconoscere oggetti e azioni specifiche.

Figura 3.1: Esempio di immagine annotata dal dataset. Le bounding box evidenziano le classi di interesse, in questo caso: player, ball, spike e block.

3.2 Architettura del Modello e Parametri

3.2.1 Descrizione dell'architettura

Per l'addestramento e la rilevazione degli oggetti, è stata utilizzata la versione **YOLO11l** del framework YOLO, una delle architetture più avanzate e ottimizzate per compiti di object detection in tempo reale. L'architettura YOLO (You Only Look Once) è progettata per eseguire rilevazioni rapide e precise combinando i processi di localizzazione e classificazione in un'unica rete convoluzionale.

La versione **YOLO11l** offre i seguenti vantaggi:

- Maggiore capacità di rappresentazione grazie alla struttura di rete estesa (*large variant*).
- Prestazioni migliorate su dataset complessi grazie al supporto per l'elaborazione di immagini a risoluzione più alta.

- Utilizzo di tecniche avanzate come *Cross Stage Partial Networks (CSP)*[17] per una migliore efficienza computazionale.

L'architettura YOLO11 si basa su una struttura a tre componenti principali:

- **Backbone**: responsabile dell'estrazione delle feature dalle immagini input.
- **Neck**: aggrega e raffina le feature estratte per ottimizzare il processo di rilevamento.
- **Head**: predice le bounding box e le relative classi per ciascun oggetto identificato.

Questa versione è stata scelta per la capacità di eseguire rilevazioni accurate anche in contesti complessi come quello del beach volley, dove gli oggetti rilevati (giocatori, palla, gesti tecnici) sono spesso in movimento e sovrapposti.

3.2.2 Configurazione dei parametri

Il training del modello è stato configurato inizialmente su *Ultralytics Hub*[18], una piattaforma che consente la configurazione intuitiva dei parametri di addestramento e la generazione del codice necessario per eseguire il training su altri ambienti. Successivamente, il training vero e proprio è stato eseguito su **Google Colab**[19], sfruttando le sue risorse computazionali, in particolare le GPU fornite gratuitamente, per accelerare il processo di addestramento.

3.2.2.1 Configurazione iniziale su Ultralytics Hub

Ultralytics Hub è stato utilizzato per definire i parametri principali di addestramento e per generare il codice necessario per l'esecuzione su ambienti esterni. La configurazione dei parametri iniziali è riportata in Tabella 3.2.

Parametro	Valore
Versione modello	YOLO11l
Pretrained	Sì
Numero di epoche	100
Dimensione immagini	640 × 640
Patience	100
Cache	None
Batch size	Auto
Device	GPU

Tabella 3.2: Parametri di training utilizzati per YOLO11l

3.2.2.2 Esecuzione del training su Google Colab

Il codice fornito da Ultralytics Hub è stato successivamente eseguito su **Google Colab**, un ambiente cloud-based che offre risorse computazionali avanzate, tra cui l'accesso a GPU e TPU, ideali per il training di modelli di deep learning. Google Colab è stato scelto per la sua capacità di eseguire training intensivi senza la necessità di hardware locale dedicato, riducendo i tempi di addestramento e migliorando la capacità del modello di convergere verso una soluzione ottimale.

I parametri configurati inizialmente su Ultralytics Hub sono stati mantenuti durante il training su Colab, con alcune ottimizzazioni per adattare l'addestramento al dataset limitato utilizzato:

- **Loss Function**: è stata utilizzata la *Composite Loss* di YOLO, che combina la *Bounding Box Loss*, la *Classification Loss* e la *Objectness Loss*.

- **Fine-Tuning**: è stato adottato un approccio di *fine-tuning* con il congelamento (*freeze*) dei primi layer dell'encoder, derivati dai pesi pre-addestrati, per sfruttare le caratteristiche generali apprese in pre-

cedenza. Gli strati dell'*head* sono stati lasciati liberi di aggiornarsi, consentendo un apprendimento specifico sul dataset di riferimento.

- **Learning Rate**: è stato impostato un *learning rate* basso per l'encoder (strati congelati) e uno più elevato per l'*head*, al fine di migliorare le prestazioni e favorire l'adattamento ai dati specifici del progetto.
- **Batch Size e Anchor Boxes**: configurati automaticamente per adattarsi al dataset specifico.

L'integrazione tra Ultralytics Hub e Google Colab ha permesso di combinare la semplicità della configurazione con la potenza di calcolo necessaria per eseguire il training su un dataset complesso come quello utilizzato in questo progetto. Grazie all'utilizzo di tecniche di *fine-tuning*, è stato possibile massimizzare l'efficienza dell'addestramento, migliorando la capacità del modello di generalizzare su nuove immagini.

Capitolo 4

Estrazione delle Informazioni

4.1 Informazioni Estratte

L'applicazione sviluppata è in grado di analizzare i video di partite di beach volley per estrarre informazioni utili al miglioramento delle performance degli atleti e all'ottimizzazione delle strategie di gioco. In particolare, l'applicazione si concentra sul riconoscimento dei gesti tecnici eseguiti dai giocatori e sull'identificazione dei momenti chiave del gioco, fornendo dati dettagliati e strutturati.

4.1.1 Riconoscimento dei gesti tecnici

Uno degli aspetti principali dell'applicazione è il riconoscimento automatico dei gesti tecnici eseguiti durante la partita. Grazie all'utilizzo di un modello YOLO addestrato su un dataset annotato manualmente, l'applicazione è in grado di identificare con precisione i seguenti gesti tecnici:

- **Bagher** (`BAGHER`): il gesto utilizzato per ricevere o difendere il pallone, caratterizzato dal contatto con l'avambraccio.

- **Alzata** (`SET`): il gesto che prepara l'attacco, eseguito con entrambe le mani sopra la testa.

- **Schiacciata** (`SPIKE`): il gesto d'attacco, in cui il giocatore colpisce la palla con forza per inviarla nel campo avversario.
- **Muro** (`BLOCK`): il tentativo di intercettare l'attacco avversario saltando a rete.
- **Battuta** (`SERVE`): il gesto d'inizio dell'azione, con cui il pallone viene lanciato dal giocatore per iniziare il gioco.

Oltre ai gesti tecnici, il modello rileva la posizione dei **giocatori** (`PLAYER`) e della **palla** (`BALL`) all'interno del campo, associando ciascun evento a coordinate spaziali precise.

4.1.2 Strutturazione dei dati con Timepoints

Per organizzare e memorizzare i dati estratti durante l'analisi video, l'applicazione utilizza una struttura dati chiamata **timepoints**. Ogni timepoint rappresenta un frame o un insieme di frame del video, contenente informazioni dettagliate sui gesti tecnici rilevati, i bounding box associati, la posizione dei giocatori e della palla. Questa struttura consente di mappare temporalmente le azioni chiave e di fornire un output ben organizzato per ulteriori analisi.

Dopo che il modello YOLO analizza un video frame per frame, i risultati vengono convertiti in timepoints. Un `timepoint` è un dizionario che memorizza i seguenti elementi per ciascun frame:

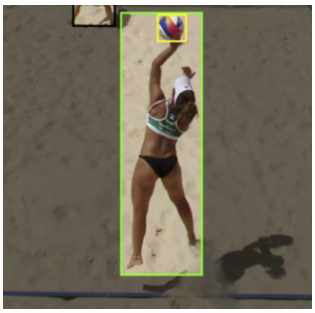
- `frame_number`: il numero del frame analizzato.
- `gestures`: un elenco di gesti tecnici rilevati nel frame.
- `gesture_boxes`: le bounding box associate ai gesti tecnici rilevati.
- `players`: le bounding box dei giocatori rilevati.
- `balls`: le bounding box della palla rilevata. Il nome `balls` è stato scelto perché, tecnicamente, in ogni frame il sistema di rilevamento potrebbe individuare più oggetti interpretati come una palla. Questo può

(a) BAGHER

(b) SET

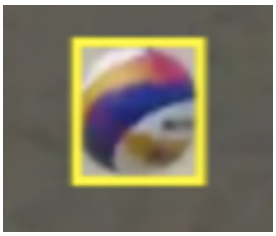
(c) SPIKE

(d) BLOCK



(e) SERVE

(f) PLAYER



(g) BALL

Figura 4.1: Esempi di rilevazione dei gesti tecnici e degli oggetti nel video. Ogni immagine evidenzia il gesto tecnico riconosciuto o gli oggetti rilevati con una bounding box.

accadere a causa di errori di rilevamento, oggetti confondibili (es. loghi circolari o ombre), oppure variazioni rapide nella posizione della palla che portano a rilevazioni multiple. Successivamente, il sistema applica filtri e logiche per selezionare la palla effettiva, eliminando eventuali falsi positivi.

Un esempio di struttura di un timepoint è mostrato di seguito:

Listing 4.1: Esempio di struttura dati di un `timepoint`.

```
{
    "frame_number": 150,
    "gestures": ["SPIKE"],
    "gesture_boxes": [[100, 200, 150, 250]],
    "players": [[50, 100, 90, 150], [200, 250, 240, 300]
        ↪ , [300, 400, 350, 450]],
    "balls": [[400, 450, 420, 470]]
}
```

Questa struttura permette di:

- Associare ogni gesto tecnico e oggetto rilevato al corrispondente frame temporale.
- Fornire un output ben organizzato che può essere utilizzato per analisi successive, come il calcolo di statistiche o la generazione di report.
- Integrare facilmente i dati in pipeline di analisi personalizzate.

L'approccio basato sui timepoints garantisce una rappresentazione efficiente e scalabile dei dati, semplificando l'interpretazione dei risultati ottenuti dall'applicazione.

4.1.3 Rilevazione dei momenti di gioco

La rilevazione automatica dei momenti di gioco (azione) e dei momenti di pausa rappresenta un aspetto cruciale per l'analisi delle partite di beach

volley. L'applicazione utilizza i dati generati dal modello YOLO e li struttura in *timepoints* per distinguere le fasi di gioco rilevanti.

Un'**azione** è definita come una sequenza continua di frame in cui vengono rilevati gesti tecnici o la presenza della palla in movimento. Al contrario, un momento di pausa è caratterizzato dall'assenza di rilevazioni significative per una serie di frame consecutivi.

Tuttavia, i risultati grezzi ottenuti dal modello YOLO non sono immediatamente utilizzabili per identificare in modo accurato le fasi di gioco. Essi possono includere rilevazioni errate, falsi positivi e sequenze frammentate che necessitano di ulteriori elaborazioni. Per questo motivo, vengono applicate una serie di elaborazioni:

- **Riempimento di gap nelle sequenze**: Eventuali interruzioni brevi all'interno di una sequenza di frame con rilevazioni vengono riempite per garantire la coerenza dell'azione. Questo processo utilizza una soglia temporale (*max_gap*) per determinare quando considerare una pausa come parte della stessa azione.

- **Rimozione di rilevazioni isolate**: Gesti o rilevazioni che appaiono in un singolo frame, circondati da frame senza rilevazioni, vengono trattati come falsi positivi e rimossi. Questi dati vengono invece riclassificati come giocatori o ignorati per preservare la coerenza.

- **Raggruppamento in blocchi di azione**: I momenti di gioco vengono raggruppati in *action blocks*, che iniziano e terminano in corrispondenza di frame chiave definiti rispettivamente come *start_action* e *end_action*.

Questi processi consentono di distinguere chiaramente i momenti di azione dalle pause, rendendo l'analisi delle partite più efficiente e riducendo il rumore nei dati.

4.1.4 Distinzione tra azioni consecutive

La gestione delle azioni consecutive è fondamentale per evitare sovrapposizioni o ambiguità nella rilevazione dei gesti tecnici e delle sequenze di gioco. L'applicazione utilizza un approccio strutturato per garantire che ogni gesto o sequenza sia correttamente identificato e distinto. Le seguenti tecniche sono state implementate:

- **Pulizia di gesti multipli nel medesimo frame**: Quando più gesti tecnici vengono rilevati nello stesso frame, l'applicazione conserva solo quello più rilevante. La scelta avviene considerando la distanza tra il gesto e la palla, oppure privilegiando gesti di inizio azione come la *battuta (serve)*.

- **Conversione di gesti per coerenza temporale**: Alcuni gesti, come la *schiacciata (spike)*, possono essere erroneamente rilevati subito dopo una battuta. In questi casi, il sistema converte tali gesti in *battuta (serve)* per preservare la coerenza logica e temporale della sequenza.

- **Selezione del frame centrale**: Per sequenze di gesti ripetuti, viene conservato solo il frame centrale, eliminando i frame iniziali e finali della sequenza per garantire una rappresentazione sintetica e accurata dell'azione.

- **Pulizia generale e validazione**: Per gestire scenari complessi, come il rilevamento di gesti che non rispettano l'ordine logico (ad esempio, *muro (block)* senza una *schiacciata (spike)*), vengono applicate regole di pulizia che spostano le rilevazioni incoerenti nella categoria *giocatori (players)*.

Questi metodi garantiscono che ogni azione sia accuratamente separata e che i dati risultino coerenti e pronti per ulteriori analisi o visualizzazioni. L'output finale è un set di *timepoints* organizzati in blocchi di azione ben definiti, pronti per essere analizzati.

Questi metodi garantiscono che ogni azione sia accuratamente separata e che i dati risultino coerenti e pronti per ulteriori analisi o visualizzazioni. L'output finale è un set di *timepoints* organizzati in blocchi di azione ben definiti, pronti per essere analizzati.

4.1.4.1 Esempio di Rilevazione e Pulizia dei Timepoints

Di seguito vengono mostrati due esempi: un JSON iniziale ottenuto direttamente dal modello e il JSON risultante dopo l'applicazione delle tecniche di pulizia e organizzazione. Ogni step è spiegato per mostrare come i dati vengono elaborati.

Listing 4.2: JSON iniziale con rilevazioni multiple e incoerenze.

```
[
    {
        "frame_number": 150,
        "gestures": ["SERVE", "SPIKE"],
        "gesture_boxes": [[100, 50, 150, 100], [120, 90
            ↪ , 160, 140]],
        "balls": [[110, 55, 120, 65], [115, 60, 125, 70]
            ↪ ],
        "players": [[50, 200, 100, 250], [150, 200, 200
            ↪ , 250]]
    },
    {
        "frame_number": 170,
        "gestures": ["SPIKE", "BAGHER"],
        "gesture_boxes": [[120, 90, 160, 140], [120, 80
            ↪ , 160, 130]],
        "balls": [[125, 95, 135, 105], [115, 60, 125, 70
            ↪ ]],
        "players": [[50, 200, 100, 250], [150, 200, 200
            ↪ , 250]]
    },
```

```
    {
        "frame_number": 190,
        "gestures": ["SPIKE"],
        "gesture_boxes": [[120, 80, 160, 130]],
        "balls": [[125, 85, 135, 95], [115, 60, 125, 70]
            ↪ ],
        "players": [[50, 200, 100, 250], [150, 200, 200
            ↪ , 250], [50, 300, 100, 350]]
    }
]
```

Listing 4.3: JSON risultante con dati puliti e organizzati.

```
[
    {
        "frame_number": 150,
        "gestures": ["SERVE"],
        "gesture_boxes": [[100, 50, 150, 100]],
        "balls": [[110, 55, 120, 65]],
        "players": [[50, 200, 100, 250], [150, 200, 200
            ↪ , 250], [120, 90, 160, 140]],
        "start_action": true
    },
    {
        "frame_number": 170,
        "gestures": ["BAGHER"],
        "gesture_boxes": [[120, 90, 160, 140]],
        "balls": [[125, 95, 135, 105]],
        "players": [[50, 200, 100, 250], [150, 200, 200
            ↪ , 250], [120, 90, 160, 140]]
    },
    {
        "frame_number": 190,
        "gestures": ["SPIKE"],
```

```
20          "gesture_boxes": [[120, 80, 160, 130]],
21          "balls": [[125, 85, 135, 95]],
22          "players": [[50, 200, 100, 250], [150, 200, 200
                ↪ , 250], [50, 300, 100, 350]],
23          "end_action": true
24      }
25  ]
```

Step per l'Ottenimento del Risultato Per ottenere il JSON risultante, sono stati applicati i seguenti step:

1. **Pulizia dei Gesti Multipli:** Quando più gesti sono rilevati nello stesso frame, viene mantenuto solo quello più rilevante, ad esempio il gesto più vicino alla palla o il gesto di inizio azione (`SERVE`).

2. **Rimozione di Palloni Statici:** Viene mantenuto un solo pallone per frame, eliminando quelli con movimenti incoerenti o statici.

3. **Aggiunta di `start_action` e `end_action`:** Il primo frame di un'azione viene marcato con `start_action = true` e l'ultimo frame con `end_action = true`.

4. **Validazione Generale:** Eventuali gesti incoerenti, come una `SPIKE` dopo un `SERVE`, vengono spostati nella categoria `players`.

4.2 Struttura Generale dell'Applicazione

In questa sezione vengono presentati i diagrammi che rappresentano la struttura generale dell'applicazione, con una panoramica dei package principali, la struttura delle classi del modello utilizzato e il flusso dell'analisi video.

4.2.1 Diagramma dei Package

Il diagramma riportato in Figura 4.2 rappresenta la struttura generale dei package che compongono l'applicazione. Ogni package è stato progettato per gestire un aspetto specifico dell'analisi, garantendo modularità, chiarezza e facilità di manutenzione del codice.

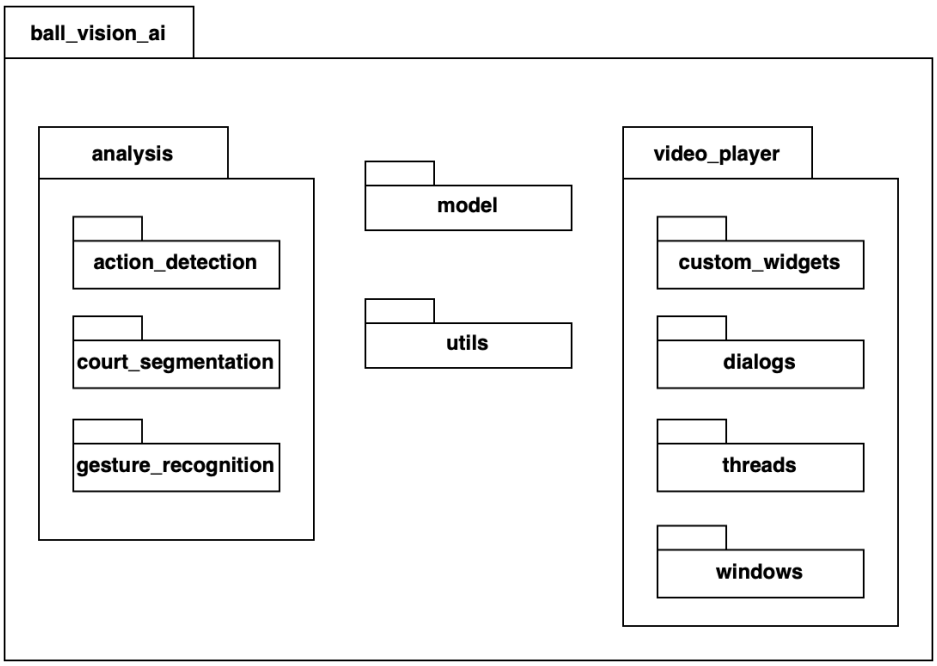


Figura 4.2: Struttura dei package dell'applicazione.

La suddivisione principale si articola come segue:

- **analysis**: Questo package è dedicato all'elaborazione dei dati e comprende i seguenti moduli principali:
  - **action_detection**: Si occupa della rilevazione delle azioni durante la partita, identificando e classificando i gesti tecnici eseguiti dai giocatori.
  - **court_segmentation**: Gestisce la segmentazione del campo da gioco, necessaria per definire i limiti spaziali all'interno dei quali vengono analizzati i gesti e le azioni.

  - **gesture_recognition**: Si concentra sul riconoscimento dei gesti tecnici eseguiti dai giocatori, utilizzando il modello YOLO.

- **video_player**: Questo package è responsabile della gestione dell'interfaccia grafica per la riproduzione dei video e l'interazione con l'utente. Include i seguenti moduli:
  - **custom_widgets**: Contiene widget personalizzati per migliorare l'interazione con l'utente, come slider e pulsanti.
  - **dialogs**: Gestisce finestre di dialogo che permettono, ad esempio, la selezione di file o la configurazione dei parametri.
  - **threads**: Implementa il threading per eseguire operazioni di analisi e riproduzione video in parallelo, garantendo un'esperienza utente fluida.
  - **windows**: Comprende le finestre principali dell'applicazione, come il lettore video e le schermate per la visualizzazione dei risultati.

- **model**: Questo package contiene le classi principali che rappresentano i concetti fondamentali del dominio del beach volley. Tra queste si trovano classi che modellano il campo, i giocatori, le squadre, i gesti tecnici e le partite. Tali classi sono utilizzate da altri package per organizzare i dati e facilitare l'analisi e la presentazione dei risultati.

- **utils**: Fornisce funzioni ausiliarie e strumenti generici utilizzati in più parti dell'applicazione, come la gestione dei file, il calcolo di metriche e operazioni di supporto per l'elaborazione dei dati.

4.2.2 Diagramma delle Classi del Modello

La Figura 4.3 illustra la struttura delle classi del modello e le loro relazioni principali.

- **Ball**: rappresenta la palla durante la partita. Include:

  - `positions`: una lista di posizioni della palla in formato `[x, y, width, height]` per ogni frame.
  - `trajectories`: un dizionario per tracciare le traiettorie della palla durante la partita.

- **Court**: modella il campo di gioco. Include:
  - `corners`: una lista di coordinate `[x, y]` per identificare i quattro angoli del campo.

- **Gesture**: un'enumerazione che rappresenta i gesti tecnici riconosciuti. Ogni gesto è definito da:
  - `class_id`: un identificatore numerico per il gesto.
  - `gesture_name`: il nome del gesto (ad esempio, `"BAGHER"`, `"SPIKE"`).

- **Match**: rappresenta una partita e include:
  - `home_team`: il team di casa, istanza della classe `Team`.
  - `away_team`: il team avversario.
  - `court`: il campo associato alla partita, istanza della classe `Court`.
  - `ball`: la palla utilizzata nella partita, istanza della classe `Ball`.

- **Player**: rappresenta un giocatore e include:
  - `id`: un identificativo unico.
  - `name`: il nome del giocatore.
  - `number`: il numero sulla maglia del giocatore.
  - `positions`: una lista di tuple contenenti il frame e la posizione `[x, y]` del giocatore.
  - `gestures`: una lista di gesti eseguiti dal giocatore durante la partita.
  - `stats`: statistiche del giocatore, istanza della classe `PlayerStats`.

- **PlayerStats**: rappresenta le statistiche individuali di un giocatore e include:
  - `serves`: una lista contenente dati relativi ai servizi eseguiti dal giocatore.
  - `spikes`: una lista con dati relativi alle schiacciate.
- **Team**: rappresenta una squadra e include:
  - `name`: il nome della squadra.
  - `players`: una lista di istanze della classe `Player` che appartengono alla squadra.
  - `player_1_receives_left`: una variabile booleana che indica se il primo giocatore riceve a sinistra.
  - `player_1_serves_first`: una variabile booleana che indica se il primo giocatore serve per primo.

    Queste ultime due variabili sono attualmente implementate come booleani per garantire semplicità ed efficienza nell'accesso ai dati durante il tracciamento delle azioni. Tuttavia, un'alternativa più flessibile potrebbe essere quella di utilizzare variabili come `ID_left_player` e `ID_serve_first`, che memorizzano l'ID numerico del giocatore (ad esempio, `0` o `1`). Questo approccio consentirebbe una gestione più dinamica delle informazioni, facilitando l'espansione del modello per includere statistiche aggiuntive o scenari più complessi, come l'assegnazione di ruoli dinamici ai giocatori.

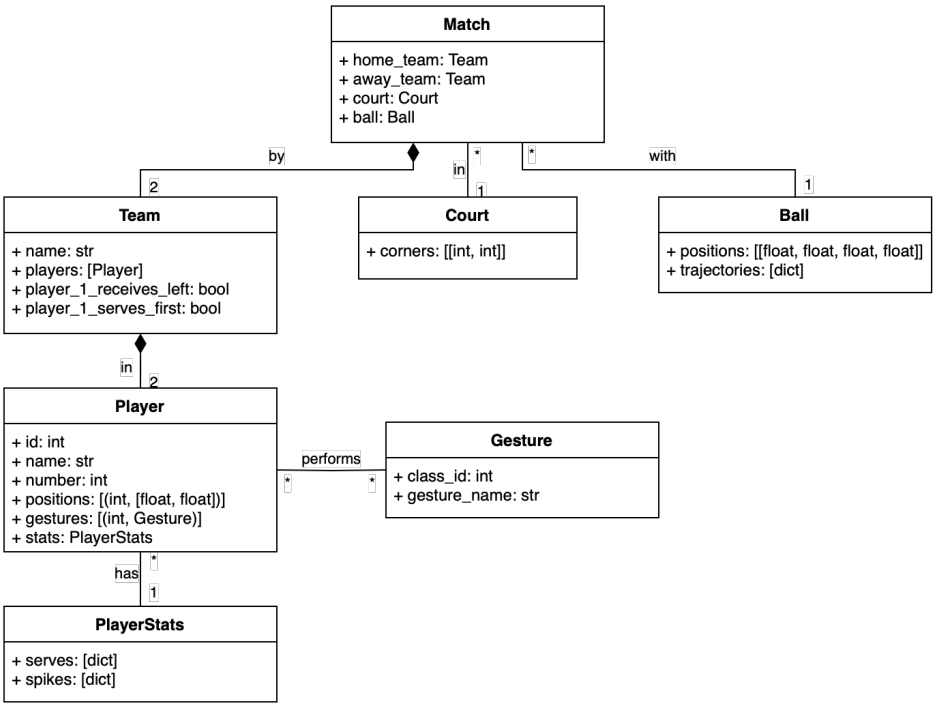


Figura 4.3: Diagramma delle Classi del modello.

Capitolo 5

Video Player

5.1 Flusso Operativo del Video Player

Il video player è progettato per fornire un'esperienza intuitiva all'utente, guidandolo attraverso i passaggi necessari per caricare, analizzare e visualizzare i risultati di un video. Di seguito viene presentato il flusso operativo, illustrato con esempi visivi dei principali step.

5.1.1 Avvio del Video Player

All'apertura dell'applicazione, l'interfaccia principale del video player appare vuota, pronta per accettare un input, come mostrato in Figura 5.1.

5.1.2 Caricamento di un Video

Per caricare un video, l'utente può cliccare su `File > Load Video`, aprendo una finestra di dialogo che permette di selezionare un file in formato `mp4` o `avi`. La Figura 5.2 mostra il dialog di caricamento del video.

Dopo aver selezionato un video, questo viene caricato e visualizzato nel player, come illustrato in Figura 5.3.

Figura 5.1: Interfaccia principale del video player all'avvio.

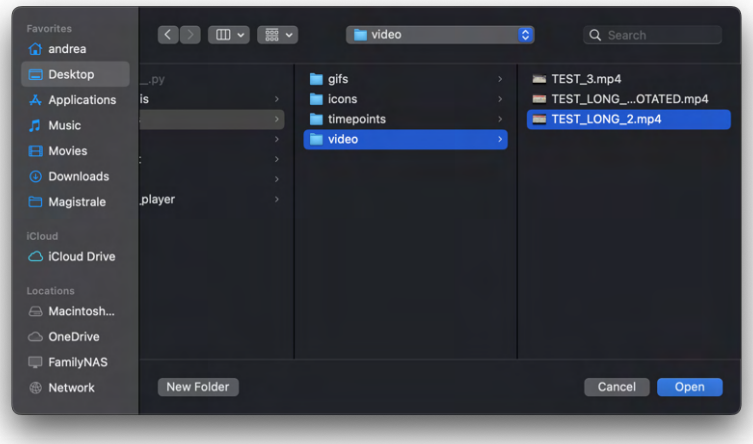


Figura 5.2: Dialog di caricamento del video.

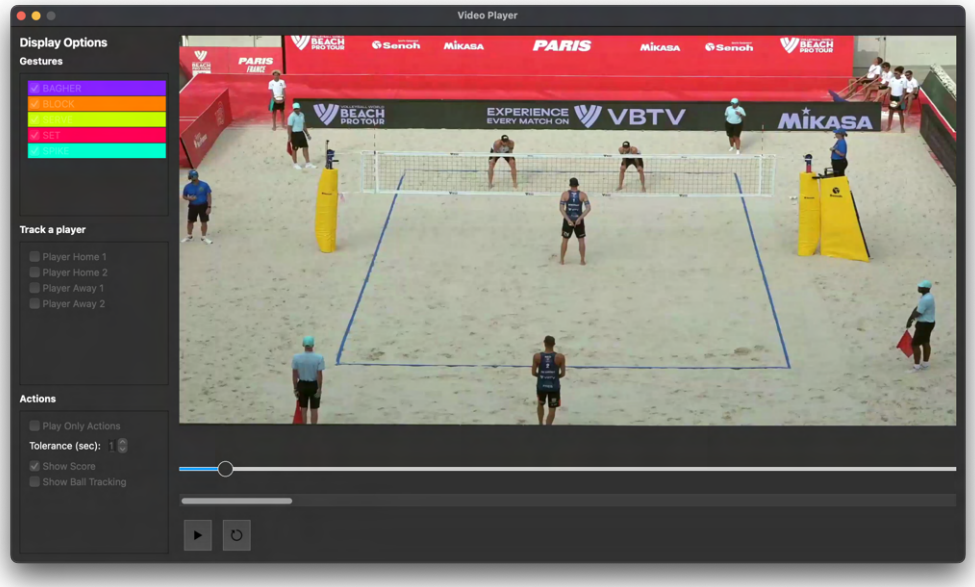


Figura 5.3: Video caricato nel player.

5.1.3 Caricamento di Analisi Preesistenti

Se il video è già stato analizzato in precedenza, l'utente può caricare il file `JSON` contenente i risultati cliccando su `File > Load Timepoints`. Questa opzione consente di risparmiare tempo, evitando di dover eseguire nuovamente l'analisi. La Figura 5.4 mostra il dialog di caricamento del file `JSON`.

5.1.4 Avvio dell'Analisi del Video

Per avviare l'analisi, l'utente può cliccare su `Actions > Analyze Video`, che mostrerà una finestra di dialogo per scegliere la posizione in cui salvare il file `JSON` con i risultati, come illustrato in Figura 5.5.

Il file `JSON` generato rappresenta il risultato dell'analisi dopo l'applicazione delle tecniche di pulizia e riorganizzazione descritte nella Sezione 4.1.3, come evidenziato nel Listing 4.3. Questo formato finale è ottimizzato per essere utilizzato in tutte le funzionalità successive del video player, come il

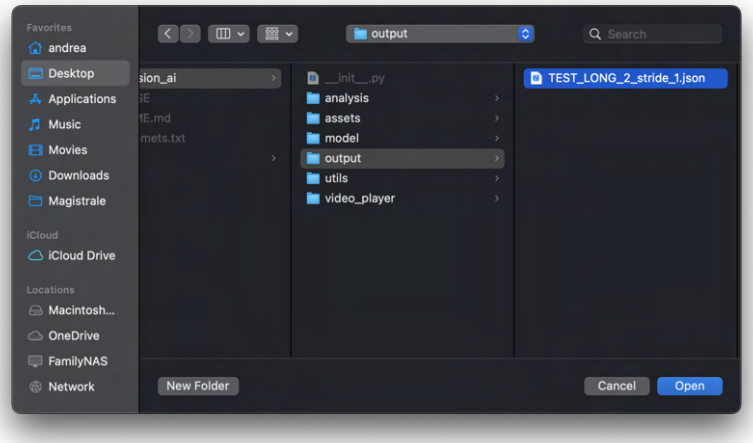


Figura 5.4: Dialog di caricamento del file JSON.

tracciamento dei giocatori, il calcolo delle traiettorie e l'esportazione di video selettivi.

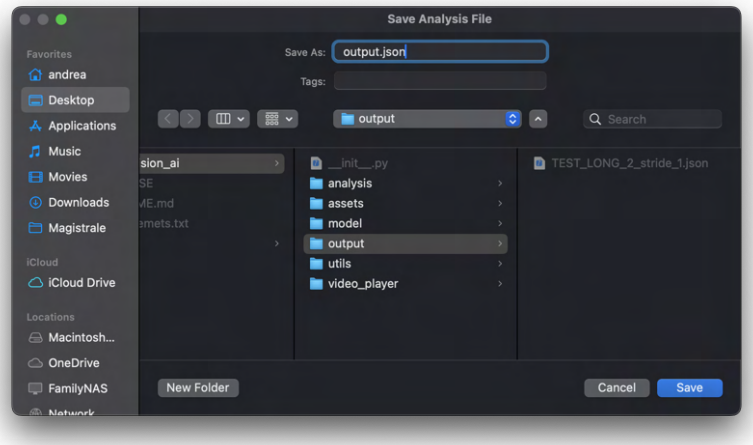


Figura 5.5: Dialog per la selezione del percorso di salvataggio del file JSON.

Successivamente, verrà visualizzata una finestra per impostare i parametri dell'analisi, come la soglia di confidenza o il numero di frame per secondo da analizzare, come mostrato in Figura 5.6.

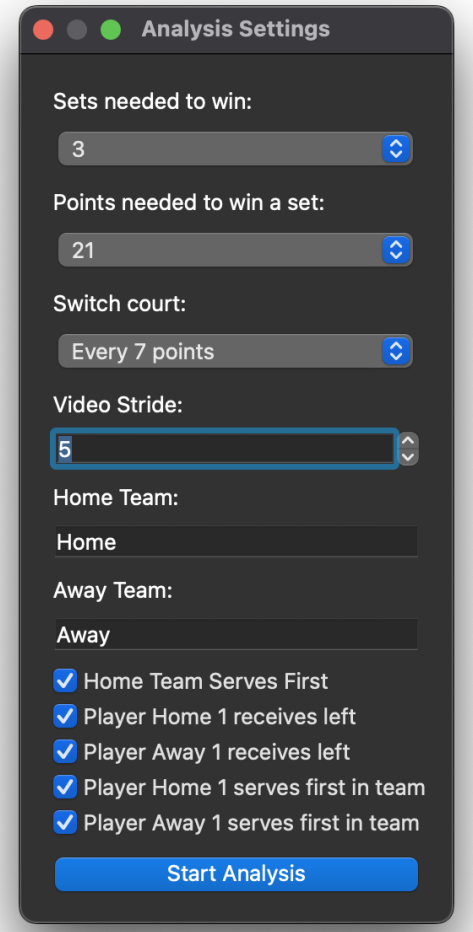


Figura 5.6: Finestra di configurazione dei parametri dell'analisi.

Dopo aver confermato le impostazioni, l'analisi avrà inizio. Una schermata di caricamento indicherà l'avanzamento del processo, come mostrato in Figura 5.7.

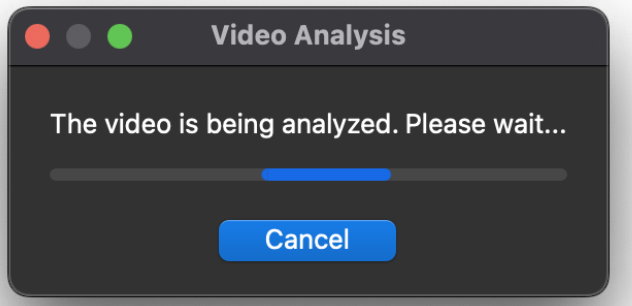


Figura 5.7: Schermata di caricamento dell'analisi del video.

5.1.5 Visualizzazione dei Risultati dell'Analisi

Una volta completata l'analisi, il video player mostrerà i dati estratti sovrapposti al video. L'utente può visualizzare i risultati direttamente oppure interagire con il file `JSON` generato. Questa fase è illustrata in Figura 5.8.

5.1.6 Diagramma degli Stati UI

Per rappresentare in modo chiaro i passaggi tra le diverse schermate del video player, è stato creato un diagramma degli stati che illustra le transizioni logiche tra le interfacce principali. Questo diagramma evidenzia come gli eventi, quali il caricamento di un video o l'avvio dell'analisi, influenzino il flusso dell'applicazione. Ogni stato rappresenta una schermata o un dialogo del video player, mentre le frecce indicano le azioni o gli eventi che causano la transizione verso uno stato successivo.

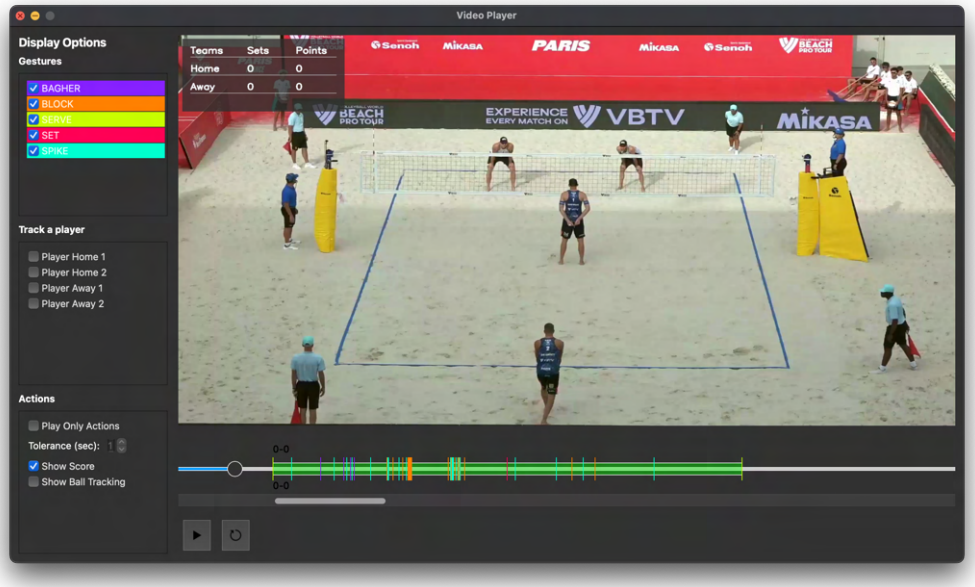


Figura 5.8: Video player con i risultati dell'analisi caricati.

5.2 Funzionalità Principali del Video Player

Il video player integra una serie di funzionalità avanzate per l'analisi e l'interazione con i video di beach volley. Ogni funzionalità è progettata per migliorare l'usabilità dell'applicazione e consentire un'analisi precisa ed efficace.

5.2.1 Filtraggio delle Gesture sulla Timeline

Una delle funzionalità principali del video player è la possibilità di filtrare le gesture tecniche visualizzate sulla timeline. Questa opzione consente di concentrarsi su gesti specifici, facilitando l'analisi dei momenti di gioco rilevanti.

L'utente può selezionare le gesture desiderate tramite un pannello di opzioni che include checkbox associate a ciascun gesto tecnico (Figura 5.10). Una volta selezionate, solo le gesture corrispondenti verranno visualizzate sulla timeline del video (Figura 5.11).

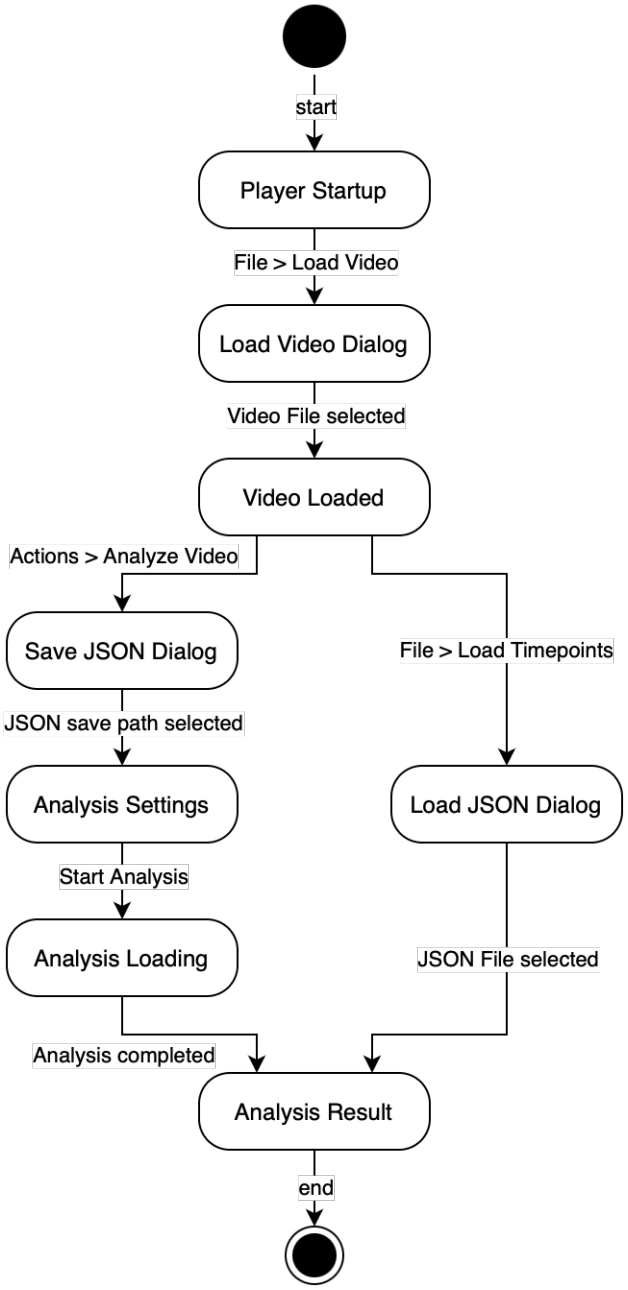


Figura 5.9: Diagramma degli stati delle interfacce utente del video player.

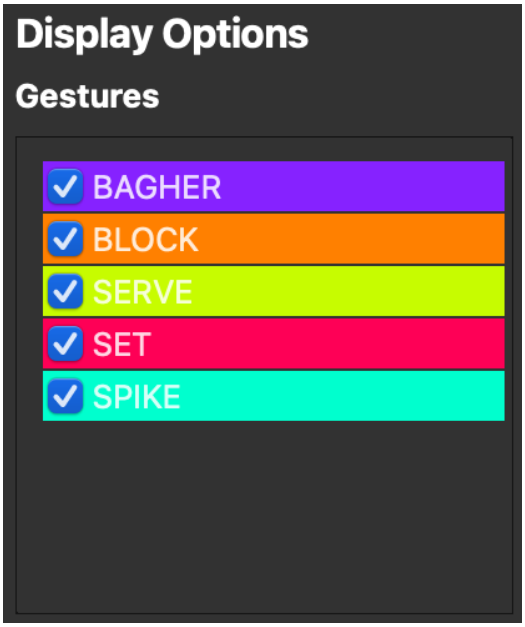


Figura 5.10: Pannello per il filtraggio delle gesture. Le gesture selezionate saranno visualizzate sulla timeline.

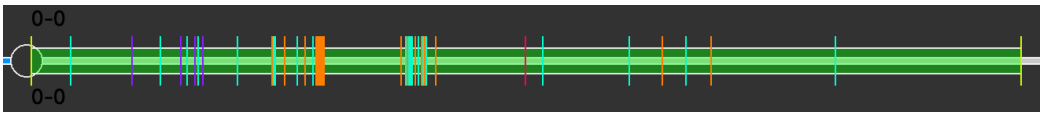


Figura 5.11: Timeline del video con le gesture filtrate e visualizzate.

Questa funzionalità è particolarmente utile per analizzare specifici aspetti del gioco, escludendo momentaneamente le altre gesture che potrebbero non essere di interesse per l'analisi corrente.

5.2.2 Riproduzione dei soli momenti di Azione

Un'altra funzionalità chiave del video player è la possibilità di riprodurre esclusivamente i momenti di azione, saltando automaticamente i momenti morti tra un punto e l'altro. Questa opzione è particolarmente utile per velocizzare l'analisi delle partite, focalizzandosi esclusivamente sui blocchi di gioco rilevanti.

I blocchi di azione sono evidenziati in verde sulla timeline (Figura 5.12), offrendo all'utente una chiara rappresentazione visiva dei momenti di gioco. Attivando la checkbox `Play Only Actions`, il player salterà automaticamente tra i blocchi di azione durante la riproduzione.

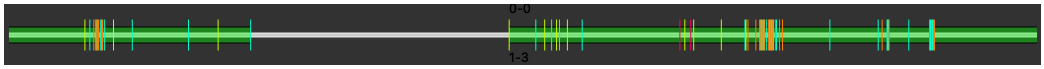


Figura 5.12: Timeline del video con i blocchi di azione evidenziati in verde.

Per garantire una transizione fluida tra i blocchi di azione, l'utente può impostare un valore di tolleranza espresso in secondi, che determina il tempo riprodotto prima e dopo ogni blocco. Questo valore può essere regolato tra 1 e 5 secondi tramite un'apposita interfaccia (Figura 5.13).

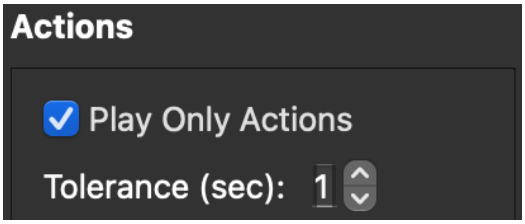


Figura 5.13: Interfaccia per la configurazione della tolleranza temporale nella riproduzione dei momenti di azione.

Questa funzionalità permette di:

- Eliminare i tempi morti e concentrarsi esclusivamente sui momenti di gioco rilevanti.
- Personalizzare la riproduzione, evitando tagli bruschi grazie all'aggiunta di secondi extra prima e dopo ogni azione.
- Risparmiare tempo durante l'analisi delle partite, garantendo al contempo una riproduzione fluida e continua.

5.2.3 Esportazione Video

Il video player offre la possibilità di esportare il video contenente esclusivamente i momenti di azione, eliminando i tempi morti tra un punto e l'altro. Questa funzione consente di ottenere fisicamente un file ottimizzato per la revisione o l'analisi, senza la necessità di dover aprire ogni volta il video player e caricare il file con le analisi per rivederlo.

Per esportare il video, l'utente deve selezionare l'opzione `File > Export Video`. Una finestra di dialogo permetterà di scegliere il percorso e il nome del file di output, che sarà salvato in formato `mp4`.

Dopo aver confermato la posizione di salvataggio, l'applicazione avvierà il processo di esportazione. Una schermata di caricamento indicherà l'avanzamento dell'operazione (Figura 5.14). Al termine del processo, il file esportato sarà disponibile nel percorso selezionato.

5.2.4 Calcolo Automatico del Punteggio

Una delle funzionalità del video player è il calcolo automatico del punteggio durante l'analisi delle partite. Questa funzione identifica i punti segnati dalle squadre in base alla rilevazione delle battute (*serve*) e considera automaticamente i cambi di campo e la fine di ogni set.

Il calcolo viene eseguito in questo modo:

- Quando viene rilevata una battuta, si presume che il punto precedente sia stato segnato dalla squadra che sta per servire.

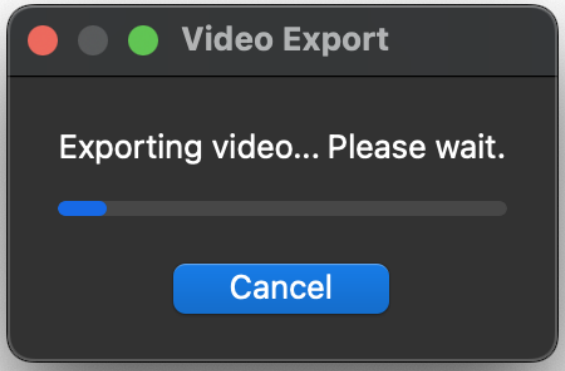


Figura 5.14: Schermata di caricamento durante l'esportazione del video.

- I cambi di campo vengono gestiti automaticamente durante i set, assicurando che i punteggi siano assegnati correttamente alle rispettive squadre.
- Alla fine di ogni set, il punteggio viene azzerato e viene avviato il conteggio per il set successivo.

Visualizzazione del punteggio:

- **Timeline**: Il punteggio è visualizzato sulla timeline, all'inizio di ogni blocco verde che rappresenta un'azione.
- **Frame Video**: Spuntando la checkbox `Show Score`, una tabella con i punteggi delle due squadre verrà visualizzata sul frame video, come mostrato in Figura 5.15.

Questa funzionalità consente di monitorare in modo preciso e immediato l'evoluzione del punteggio durante la partita, offrendo un riferimento temporale chiaro sulla timeline per individuare i momenti chiave del match. La pos-

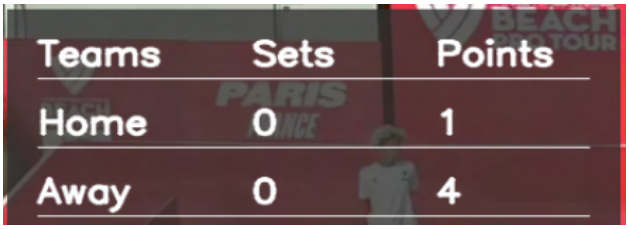


Figura 5.15: Visualizzazione del punteggio sul frame video.

sibilità di associare ogni azione al punteggio corrente aiuta allenatori e analisti a identificare rapidamente le fasi decisive, valutare le dinamiche di gioco e analizzare la correlazione tra le performance dei giocatori e i cambiamenti nel punteggio.

5.2.5 Tracking dei Giocatori e Court 2D Mapper

Una delle funzionalità più avanzate del video player è il tracciamento dei movimenti dei giocatori durante la partita e la visualizzazione delle loro posizioni su una minimappa 2D chiamata *Court 2D Mapper*. Grazie a un sistema di rilevazione automatica basato sui dati generati dal modello e organizzati nei *timepoints*, è possibile seguire i movimenti distinti dei quattro giocatori in ogni frame del video.

Per attivare la minimappa e visualizzarla a schermo, basta cliccare sul menu `Windows > Show Court 2D Mapper`.

Tracciamento dei movimenti Il sistema analizza le posizioni dei giocatori frame per frame, associandole alle azioni rilevate. Durante l'analisi del video:

- Le posizioni iniziali dei giocatori vengono determinate in base all'ordine di servizio e al lato del campo occupato, sfruttando le bounding box rilevate.

- Nei frame successivi, ogni giocatore viene associato alla posizione rilevata più vicina, garantendo un tracciamento fluido anche in presenza di movimenti intensi.
- Le coordinate dei giocatori e le azioni eseguite (es. bagher, schiacciata) vengono salvate per ogni frame, consentendo analisi dettagliate e statistiche avanzate.

Questo approccio garantisce un tracciamento preciso, adattandosi dinamicamente ai cambiamenti durante il gioco e fornendo una base solida per l'analisi tattica e statistica.

Court 2D Mapper Le posizioni rilevate vengono visualizzate in tempo reale su una minimappa 2D del campo da gioco, fornendo una rappresentazione dall'alto del posizionamento dei giocatori, come mostrato in Figura 5.16. Questa funzionalità è utile per:

- Monitorare i movimenti dei giocatori rispetto al campo e alla palla.
- Identificare pattern di gioco, come rotazioni o spostamenti difensivi.
- Analizzare strategie e tattiche basate sulla posizione.

Calcolo delle Posizioni 2D con la Matrice di Omografia Per rappresentare i giocatori nella minimappa 2D, è stata utilizzata una matrice di omografia, calcolata partendo dai quattro angoli del campo rilevati dal modello YOLO. La matrice di omografia consente di trasformare coordinate da uno spazio prospettico (come quello catturato dalla telecamera) a un piano bidimensionale visto dall'alto.

L'**omografia** è una trasformazione geometrica che stabilisce una relazione tra due piani, proiettando un insieme di punti di uno spazio a prospettiva verso un altro spazio con coordinate bidimensionali. Questa tecnica è ampiamente utilizzata in applicazioni di computer vision, come il mapping di oggetti su piani differenti o il tracking di posizioni relative.

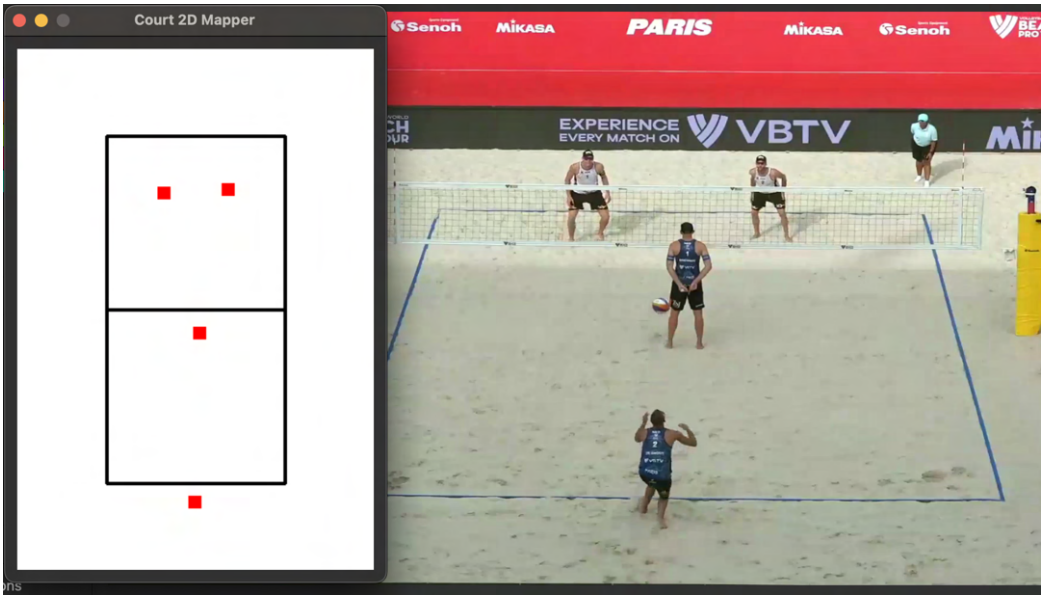


Figura 5.16: Minimappa 2D del campo con le posizioni dei giocatori.

Come funziona l'omografia La matrice di omografia, denotata con H, è una matrice 3×3 che soddisfa la relazione:

$$s \begin{bmatrix} x' \\ y' \\ 1 \end{bmatrix} = H \begin{bmatrix} x \\ y \\ 1 \end{bmatrix}$$

dove:

- (x, y): coordinate originali (spazio prospettico).
- (x', y'): coordinate trasformate (spazio 2D visto dall'alto).
- s: fattore di scala per normalizzare la matrice.
- H: matrice di omografia calcolata usando corrispondenze tra i punti rilevati e i punti desiderati.

Per calcolare H, è necessario avere almeno quattro coppie di punti corrispondenti tra il piano originale (gli angoli del campo rilevati da YOLO) e il piano proiettato (gli angoli di un campo bidimensionale ideale). Una

volta calcolata, la matrice viene applicata alle coordinate dei giocatori per proiettare le loro posizioni sul piano 2D.

Applicazione nell'analisi del beach volley La matrice di omografia permette di:

- Trasformare dinamicamente le posizioni dei giocatori dal video reale al piano della minimappa 2D.
- Ottenere una rappresentazione accurata del posizionamento rispetto al campo, indipendentemente dall'angolazione della telecamera.
- Supportare analisi spaziali avanzate, come la copertura del campo, la densità dei movimenti o l'individuazione di aree di gioco preferite.

Grazie all'uso della matrice di omografia, il *Court 2D Mapper* offre una visione chiara e comprensibile dei movimenti sul campo, consentendo agli utenti di valutare con precisione le dinamiche di gioco e di pianificare strategie più efficaci.

5.2.6 Tracking della Palla e Calcolo delle Traiettorie

Un'altra funzionalità avanzata dell'applicazione è il tracciamento della palla durante tutta la partita. Come per i giocatori, anche la posizione della palla viene calcolata frame per frame, permettendo di determinare le sue traiettorie e velocità, con un focus particolare su battute e attacchi.

Tracciamento della palla Il sistema utilizza le coordinate rilevate dalla palla per ogni frame e filtra eventuali posizioni statiche o incoerenti. Attraverso un algoritmo di tolleranza, il tracciamento mantiene solo le posizioni significative della palla, garantendo precisione e continuità.

Calcolo delle traiettorie Le traiettorie vengono determinate attraverso una regressione polinomiale sui punti rilevati per la palla. Questo consente di

approssimare il movimento della palla con una parabola, utile per analizzare i percorsi delle battute e degli attacchi.

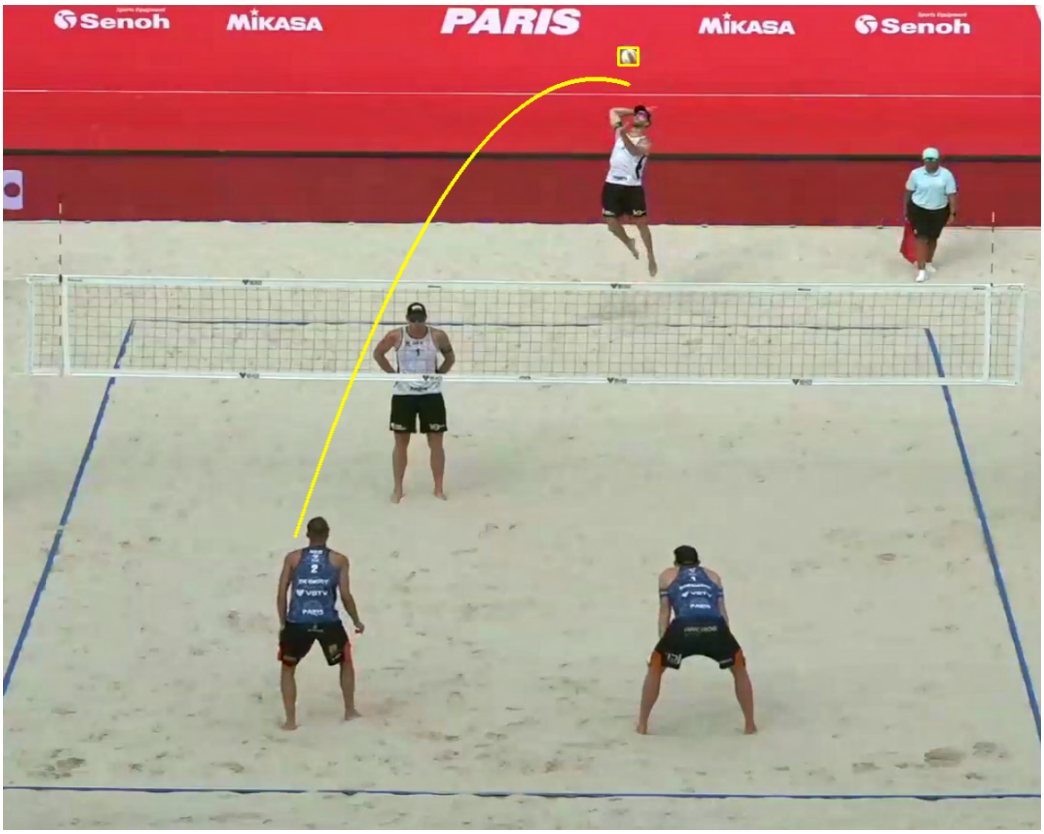


Figura 5.17: Traiettoria della palla calcolata durante una battuta. La parabola è ottenuta tramite regressione polinomiale sui punti rilevati.

Calcolo della velocità Conoscendo le posizioni della palla in sequenza temporale, è possibile stimare la velocità durante la battuta. La velocità viene calcolata considerando la distanza percorsa dalla palla tra frame consecutivi e il frame rate del video. Questa informazione viene mostrata sullo schermo durante la riproduzione.

Il calcolo della velocità avviene prendendo il punto iniziale e il punto finale della traiettoria della palla per ottenere l'"ombra" della traiettoria, ovvero il segmento che congiunge il punto iniziale e finale. Approssimando l'altezza

della palla come costante tra il punto di impatto e il punto di ricezione, è possibile rimuovere il componente altezza, proiettando il segmento a livello del terreno.

È importante notare che questo approccio non è applicabile a tipologie di servizio particolari, come la *skyball*, che segue una traiettoria altamente parabolica e richiederebbe un'analisi più complessa. Tuttavia, tale servizio è utilizzato raramente e non incide in modo significativo sull'analisi complessiva.

Il focus principale è stimare la velocità iniziale della palla, che rappresenta un indicatore importante della potenza del servizio. Considerare l'intera traiettoria parabolica completa richiederebbe calcoli più complessi, senza offrire vantaggi significativi dal punto di vista pratico o tattico. L'approssimazione proposta è quindi adeguata per fornire un'analisi utile ed efficace.

Successivamente, utilizzando la matrice di omografia calcolata in precedenza, la traiettoria viene mappata in una visione 2D dall'alto, risultando in un segmento dritto. Da questa proiezione, è possibile determinare la distanza reale percorsa dalla palla in metri e, con i frame per secondo del video, calcolare una velocità approssimata.

La Figura 5.18 mostra un esempio della proiezione della traiettoria della palla in 2D, all'interno del campo.

Metodo di calcolo La velocità della palla viene stimata attraverso i seguenti passaggi:

- Partendo dalla traiettoria calcolata, vengono identificati il punto iniziale (punto di impatto del giocatore) e il punto finale (punto di ricezione del giocatore avversario).

- La traiettoria viene semplificata in un segmento rettilineo, eliminando il componente relativo all'altezza della palla. Questo segmento rappresenta l'**ombra** della traiettoria sul piano orizzontale.

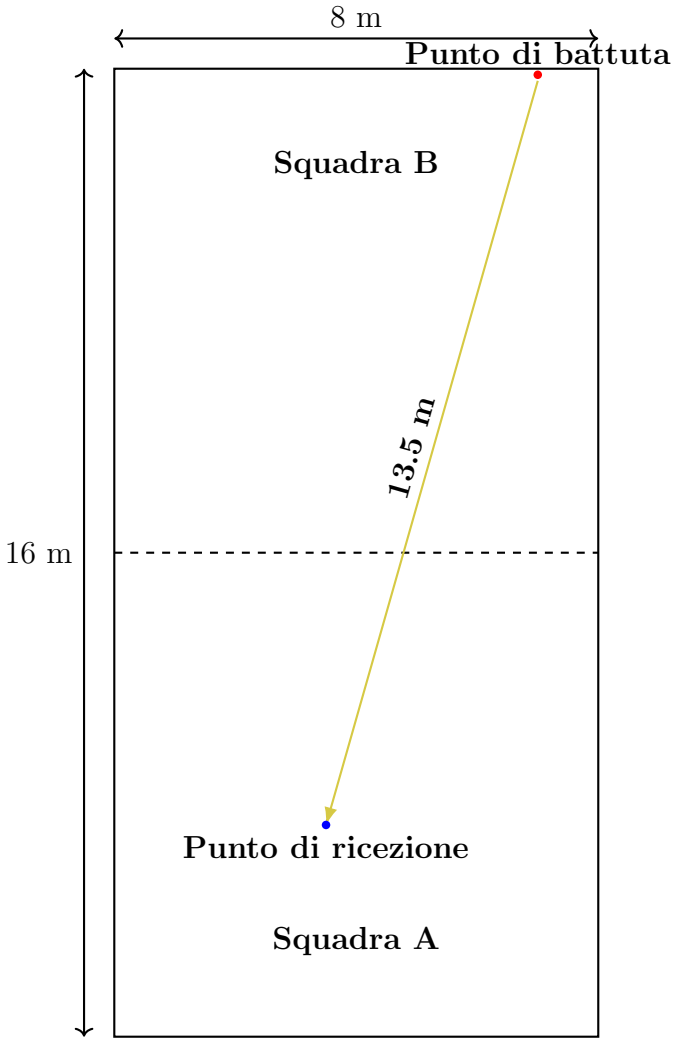


Figura 5.18: Proiezione della traiettoria della battuta in 2D, con la lunghezza espressa in metri.

- Per approssimare questa traiettoria in termini reali, si utilizza la matrice di omografia, già calcolata per la minimappa 2D. La matrice consente di proiettare la traiettoria della palla in uno spazio bidimensionale visto dall'alto, dove le coordinate sono espresse in metri reali.

- Calcolando la lunghezza del segmento proiettato e conoscendo il frame rate del video, è possibile determinare il tempo impiegato dalla palla per percorrere tale distanza. La formula completa per la velocità in metri al secondo ($v_{\mathrm{m/s}}$) è:

$$v_{\mathrm{m/s}} = \frac{d}{t} = \frac{d}{\frac{f_{\mathrm{fine}} - f_{\mathrm{inizio}}}{\mathrm{fps}}} = d \cdot \frac{\mathrm{fps}}{f_{\mathrm{fine}} - f_{\mathrm{inizio}}}$$

  Dove:

  - d: distanza percorsa dalla palla (metri), ottenuta dalla proiezione della traiettoria sul piano 2D;
  - f_{inizio}: numero del frame iniziale, corrispondente al momento in cui la palla parte (es. impatto con il giocatore che effettua la battuta);
  - f_{fine}: numero del frame finale, corrispondente al momento in cui la palla raggiunge il giocatore ricevente;
  - fps: frame rate del video (es. 50 fps), espresso in frame al secondo.

- Per ottenere la velocità in chilometri orari ($v_{\mathrm{km/h}}$), è sufficiente moltiplicare la velocità in metri al secondo per un fattore di conversione pari a 3.6.

Considerazioni sul calcolo Sebbene l'approccio sia approssimato, esso è sufficiente per distinguere qualitativamente i giocatori che eseguono battute più potenti da quelli che effettuano battute meno veloci. Lo scopo di questa funzionalità non è fornire un dato estremamente preciso, ma offrire una misura indicativa della velocità, utile per identificare differenze di stile tra i giocatori.

Figura 5.19: Velocità della palla mostrata durante una battuta.

Visualizzazione delle Traiettorie e Velocità Spuntando la checkbox `Show Ball Tracking`, il video player mostra visivamente:

- Le traiettorie della palla calcolate per ogni battuta e attacco.
- La velocità della battuta, visualizzata in sovrimpressione sotto il punteggio.
- Il bounding box attorno alla palla in ogni frame, per facilitarne il tracciamento visivo.

Le traiettorie e le velocità calcolate non solo arricchiscono l'analisi visiva, ma verranno utilizzate in una funzionalità spiegata successivamente per generare un report dettagliato. Questo report includerà statistiche avanzate, come le velocità medie e delle battute o le direzioni di attacco, fornendo uno strumento prezioso per analizzare le performance e ottimizzare le strategie di gioco.

5.2.7 Filtraggio e Ritaglio del Video in Base ai Giocatori e ai Gesti Tecnici

Un'altra importante funzionalità del video player è la possibilità di filtrare e ritagliare il video in base ai gesti tecnici selezionati e ai giocatori coinvolti. Questo consente di focalizzarsi esclusivamente sugli eventi rilevanti per un'analisi dettagliata.

Selezione dei giocatori e dei gesti tecnici L'utente può scegliere i giocatori da tracciare selezionandoli dalla lista delle checkbox sotto la sezione `Track a player`, come mostrato in Figura 5.20. Allo stesso modo, può selezionare i gesti tecnici da visualizzare spuntando le rispettive checkbox nella

sezione `Gestures`. Questa combinazione di filtri permette di visualizzare solo le azioni di interesse. Ad esempio:

- Se viene selezionato `Player Home 1` e il gesto `Bagher`, il video player mostrerà solo i bagher eseguiti dal giocatore 1 della squadra di casa.
- Se vengono selezionati `Player Home 1`, `Player Home 2` e i gesti `Bagher` e `Set`, il video mostrerà tutti i bagher e i palleggi effettuati dai giocatori della squadra di casa.

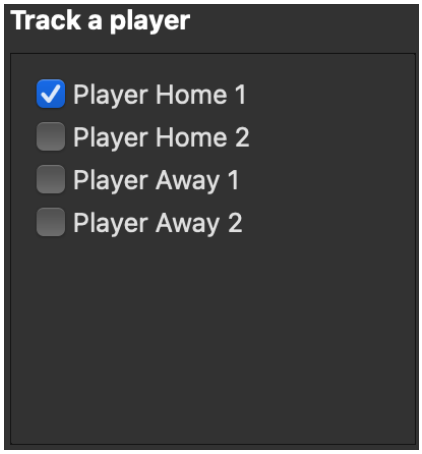


Figura 5.20: Selezione dei giocatori da tracciare.

Riproduzione e ritaglio del video Una volta applicati i filtri, il video player salterà automaticamente ai momenti corrispondenti ai gesti tecnici selezionati, eseguiti dai giocatori scelti. Inoltre, è possibile esportare un video ritagliato contenente esclusivamente queste azioni:

- Cliccando su `File > Export Video`, verrà mostrato un dialog per scegliere la posizione in cui salvare il file video.
- Il file generato includerà solo i gesti eseguiti dai giocatori selezionati.

Applicazioni pratiche La possibilità di combinare filtri su gesti tecnici e giocatori, unita all'opzione di esportare i momenti selezionati, è particolarmente utile per:

- Creare clip video personalizzate per analizzare le prestazioni individuali o di squadra.
- Raccogliere esempi di gesti tecnici per la formazione dei giocatori.
- Evidenziare specifici pattern di gioco o azioni ripetute.

5.2.8 Finestra delle Statistiche dei Giocatori

Una delle funzionalità più avanzate dell'applicazione è la possibilità di analizzare le statistiche individuali di ogni giocatore. Per accedere a questa funzionalità, l'utente può cliccare su `Windows > Show Stats Window`, aprendo una finestra dedicata alle statistiche, suddivisa in due tab principali: *Serve* e *Spike*, come mostrato in Figura 5.21 e Figura 5.22.

Tab Serve La sezione *Serve* raccoglie e visualizza tutte le informazioni relative alle battute eseguite dai giocatori. Per ogni battuta, vengono mostrati:

- La velocità di ciascuna battuta, calcolata utilizzando la tecnica descritta nella Sezione 5.2.6.
- La velocità media di battuta per ogni giocatore.

Tab Spike La sezione *Spike* si concentra sugli attacchi eseguiti dai giocatori, fornendo un'analisi dettagliata del numero di attacchi classificati in:

- **Attacchi in parallelo**: La traiettoria della palla è quasi verticale rispetto al campo, indicando un attacco diretto verso il lato opposto.
- **Attacchi in diagonale**: La traiettoria della palla attraversa il campo in diagonale, direzionandosi verso l'angolo opposto.

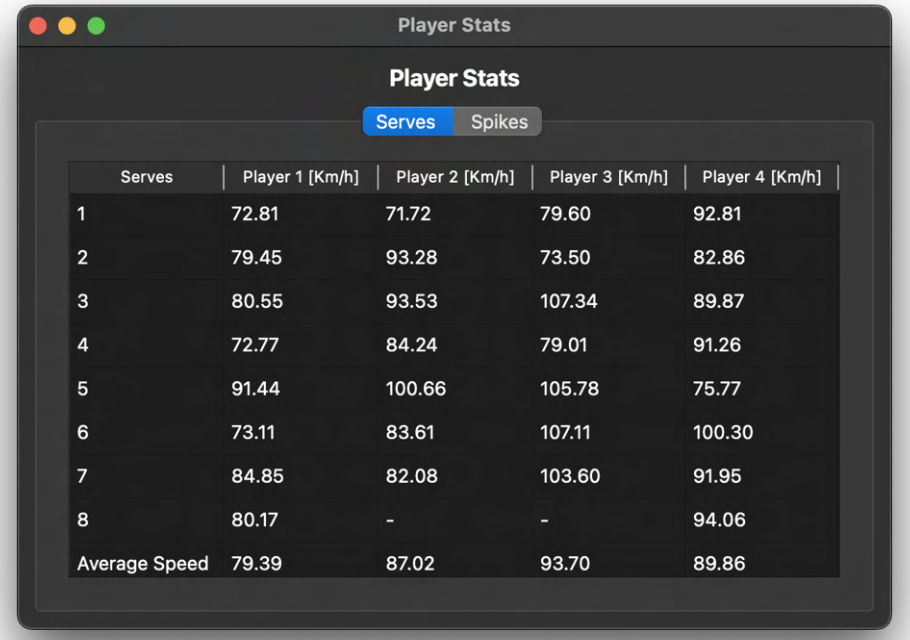


Figura 5.21: Tab *Serve*: Statistiche delle velocità di battuta per ogni giocatore.

Questi dati sono ottenuti analizzando le traiettorie della palla durante gli attacchi, come descritto nella Sezione 5.2.6.

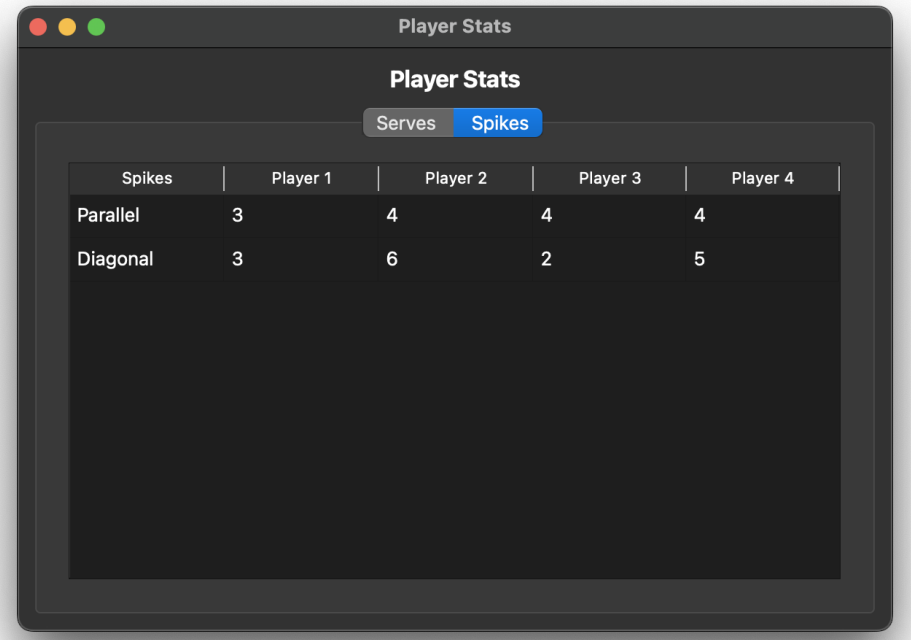


Figura 5.22: Tab *Spike*: Statistiche degli attacchi in parallelo e in diagonale per ogni giocatore.

Vantaggi della Finestra delle Statistiche Questa funzionalità offre un quadro dettagliato delle performance dei giocatori, fornendo dati utili per analisi approfondite e comparazioni tra i membri di ciascuna squadra. Le statistiche visualizzate permettono di:

- Identificare i giocatori con maggiore efficienza nelle battute o negli attacchi.
- Analizzare la tipologia degli attacchi eseguiti per individuare pattern ricorrenti e aree di miglioramento.

- Confrontare le prestazioni tra i giocatori delle due squadre.

Questo strumento si rivela essenziale per allenatori e analisti che vogliono basare le proprie decisioni su dati concreti e immediatamente comprensibili.

Formato di Output per le Statistiche dei Giocatori Per rendere le statistiche dei giocatori compatibili con strumenti di analisi esterni, è stato definito un formato di output standard. Questo può essere esportato in `JSON` o `CSV`, a seconda delle esigenze dell'utente.

Formato JSON Il formato `JSON` è particolarmente adatto per l'integrazione con sistemi informatici e librerie di analisi dati. Un esempio di output è mostrato di seguito:

Listing 5.1: Esempio di output in formato JSON con le statistiche dei giocatori.

```
{
  "match_id": "20231201",
  "players": [
    {
      "name": "Player 1",
      "stats": {
        "serve_speed_avg_kmh": 79.39,
        "spikes_parallel": 3,
        "spikes_diagonal": 3
      }
    },
    {
      "name": "Player 2",
      "stats": {
        "serve_speed_avg_kmh": 87.02,
        "spikes_parallel": 4,
        "spikes_diagonal": 6
      }
```

```
    },
    {
      "name": "Player 3",
      "stats": {
        "serve_speed_avg_kmh": 93.70,
        "spikes_parallel": 4,
        "spikes_diagonal": 2
      }
    },
    {
      "name": "Player 4",
      "stats": {
        "serve_speed_avg_kmh": 89.86,
        "spikes_parallel": 4,
        "spikes_diagonal": 5
      }
    }
  ]
}
```

Formato CSV Il formato `CSV` è utile per l'importazione in fogli di calcolo o software di analisi come Microsoft Excel o Google Sheets. Un esempio di output in `CSV` è illustrato di seguito:

Utilizzo del Formato di Output Il formato `JSON` permette di integrare facilmente le statistiche con applicazioni di analisi avanzata o sistemi di scouting. Il formato `CSV`, invece, è ideale per visualizzazioni rapide e report manuali. Entrambi i formati offrono flessibilità, consentendo di condividere i dati con allenatori, analisti o altre piattaforme software per un'ulteriore elaborazione e utilizzo.

Tabella 5.1: Esempio di output in formato CSV con le statistiche dei giocatori.

Match ID	**Player Name**	**Stat Type**	**Value**
20231201	Player 1	serve_speed_avg (km/h)	79.39
20231201	Player 2	serve_speed_avg (km/h)	87.02
20231201	Player 3	serve_speed_avg (km/h)	93.70
20231201	Player 4	serve_speed_avg (km/h)	89.86
20231201	Player 1	spikes_parallel	3
20231201	Player 1	spikes_diagonal	3
20231201	Player 2	spikes_parallel	4
20231201	Player 2	spikes_diagonal	6
20231201	Player 3	spikes_parallel	4
20231201	Player 3	spikes_diagonal	2
20231201	Player 4	spikes_parallel	4
20231201	Player 4	spikes_diagonal	5

Capitolo 6

Risultati e Valutazione

In questo capitolo vengono presentati i risultati ottenuti dall'applicazione e la loro valutazione. Vengono descritti il dataset utilizzato e il protocollo di testing, gli indicatori di performance adottati, i risultati raggiunti e un'analisi dettagliata degli errori.

6.1 Dataset e Protocollo di Testing

6.1.1 Dataset Utilizzato

Il dataset utilizzato per la valutazione dell'applicazione è lo stesso descritto nel Capitolo 3, Sezione 3.1, dove è stata fornita una panoramica dettagliata delle sue caratteristiche, della composizione e delle operazioni di preprocessing e data augmentation applicate. Di seguito, viene riportato un riepilogo dei dati utilizzati per la fase di testing:

- Dimensione: Il dataset includeva 230 immagini annotate, con un totale di 1142 annotazioni distribuite su 7 classi (*player, ball, serve, bagher, set, spike, block*).
- Annotazioni: Ogni immagine è stata annotata con bounding box per le classi rilevanti, garantendo una rappresentazione completa di ciascun frame.

- Distribuzione delle classi: I dettagli relativi alla distribuzione delle annotazioni sono riportati nel Capitolo 3.

6.1.2 Protocollo di Testing

Per garantire una valutazione accurata e riproducibile, è stato seguito un protocollo rigoroso durante la fase di testing, organizzato come segue:

Suddivisione dei dati Il dataset è stato suddiviso in tre insiemi principali:

- **Training set (70%)**: Utilizzato per addestrare il modello.
- **Validation set (20%)**: Impiegato per monitorare il processo di training e prevenire l'overfitting.
- **Test set (10%)**: Riservato per la valutazione finale delle performance.

Le dimensioni originali del dataset sono state significativamente ampliate grazie all'applicazione di tecniche di *preprocessing* e *data augmentation*, come descritto nella Sezione 3.1. A seguito di queste operazioni, il dataset è stato quasi triplicato, passando da 230 a 552 immagini annotate.

La suddivisione in training, validation e test set è stata effettuata su queste 552 immagini, mantenendo il rapporto di 70%, 20% e 10%, per un totale rispettivamente di 386 immagini per il training set, 110 per il validation set e 56 per il test set. La suddivisione è stata eseguita in maniera casuale, assicurando che la distribuzione delle classi fosse bilanciata nei tre insiemi. Questo garantisce che il modello venga testato su un insieme rappresentativo del dataset.

Modalità di testing Per la valutazione del modello è stato adottato il metodo di *hold-out validation*, che consiste nel mantenere il test set completamente separato durante le fasi di training e validazione. Questo approccio garantisce che il modello venga valutato su dati completamente indipendenti, simulando un'applicazione del modello su dati non visti.

Un'alternativa a questo metodo è la *k-fold cross-validation*, in cui il dataset viene suddiviso in k sottoinsiemi, e il modello viene addestrato e valutato k volte, utilizzando ogni volta uno dei sottoinsiemi come test set e gli altri come training set. Sebbene la *k-fold cross-validation* offra una valutazione più robusta delle performance del modello, è stata scartata in questo caso per via del maggiore costo computazionale che comporta, non giustificato dalla dimensione limitata del dataset.

La scelta del metodo di *hold-out validation* è stata preferita per la semplicità di implementazione e per la necessità di valutare il modello su un test set indipendente, rispecchiando il contesto applicativo reale in cui il modello sarà utilizzato.

6.1.3 Testing dell'Applicazione Finale

Oltre alla valutazione delle performance del modello, è stato condotto un testing completo dell'applicazione per verificare la corretta integrazione del modello, l'efficacia delle funzionalità implementate e l'usabilità complessiva.

Obiettivi del testing applicativo Il testing dell'applicazione finale si è concentrato sui seguenti obiettivi principali:

- **Validità dei risultati**: Assicurare che i risultati prodotti dall'applicazione, come le analisi delle traiettorie o le statistiche dei giocatori, fossero coerenti con i dati elaborati dal modello.

- **Funzionalità delle interfacce utente**: Verificare che ogni funzionalità descritta nel Capitolo 5 fosse completamente operativa e rispondesse alle specifiche richieste.

- **Robustezza e stabilità**: Testare l'applicazione su diversi tipi di video e situazioni, per assicurarsi che non si verificassero crash o comportamenti inaspettati.

- **Esperienza utente**: Valutare la facilità d'uso dell'interfaccia e l'intuitività delle operazioni, come il caricamento dei video, l'avvio dell'analisi e la visualizzazione dei risultati.

Metodologia di testing Per condurre il testing, sono stati utilizzati video di partite reali di beach volley, con annotazioni manuali preesistenti per confrontare i risultati prodotti dall'applicazione con i dati di riferimento. Il processo è stato strutturato in tre fasi principali:

- **Testing funzionale**: Ogni funzionalità dell'applicazione è stata testata individualmente (ad esempio, il caricamento dei video, l'analisi automatica, il tracciamento dei giocatori, il calcolo delle statistiche). Questo include test manuali per verificare il corretto funzionamento delle features grafiche e per effettuare annotazioni manuali nei video di riferimento.

- **Testing di integrazione**: Sono stati testati i flussi operativi completi, verificando la corretta integrazione tra le varie funzionalità, come il passaggio dalla fase di analisi alla visualizzazione delle statistiche.

- **Testing di stress**: L'applicazione è stata testata su video di lunga durata e ad alta risoluzione per valutare la gestione delle risorse e la stabilità del sistema.

Oltre ai test manuali, per tutte le parti automatizzabili del sistema è stato implementato un **package test** specifico, con test automatici eseguiti tramite *GitHub Actions* applicando il principio del *Continuous Testing*. Questo approccio ha permesso di verificare automaticamente il funzionamento del codice a ogni modifica del repository, garantendo un controllo continuo sulla qualità e sull'affidabilità del software.

Eventuali miglioramenti individuati durante il testing sono stati annotati per essere considerati in futuri sviluppi, come discusso nel Capitolo 6.3.2.

6.1.4 Metriche di Valutazione

Per valutare le performance del sistema, sono state utilizzate diverse metriche, ciascuna scelta in base alla sua rilevanza per specifiche funzionalità dell'applicazione. Oltre a *precision*, *recall* e *mean Average Precision (mAP)*, già approfondite nella Sezione 2.1.4, le seguenti metriche sono state adottate per un'analisi più dettagliata:

- **F1-Score**: La media armonica di *precision* e *recall*, utilizzata per valutare il bilanciamento tra la capacità del sistema di identificare correttamente i gesti tecnici (*precision*) e la capacità di catturare tutte le occorrenze dei gesti (*recall*). È particolarmente utile in contesti in cui entrambe le metriche hanno un peso significativo, come il riconoscimento di gesti tecnici. L'F1-Score fornisce un indicatore sintetico delle prestazioni complessive.

  Per il caso multiclasse, l'F1-Score può essere calcolato utilizzando due approcci principali:

  - **Macro-averaged F1-Score**: La media aritmetica degli F1-Score calcolati individualmente per ciascuna classe. È definito come:

    $$F1_{\text{macro}} = \frac{1}{C} \sum_{i=1}^{C} F1_i$$

    dove C è il numero totale di classi e $F1_i$ è l'F1-Score per la classe i.

  - **Weighted-averaged F1-Score**: Una media ponderata degli F1-Score calcolati per ciascuna classe, pesata in base al numero di campioni appartenenti a ciascuna classe. È definito come:

    $$F1_{\text{weighted}} = \sum_{i=1}^{C} \frac{N_i}{N} F1_i$$

    dove N_i è il numero di campioni della classe i, N è il numero totale di campioni, e $F1_i$ è l'F1-Score per la classe i.

Nel presente lavoro, è stato adottato il **Weighted-averaged F1-Score** per valutare le prestazioni del sistema. Questa scelta è motivata dalla distribuzione non perfettamente bilanciata delle classi nel dataset: alcune classi, come *Bagher* e *Serve*, sono rappresentate da un numero maggiore di campioni rispetto a classi meno frequenti, come *Block*. Il Weighted-averaged F1-Score garantisce che il contributo di ciascuna classe alle metriche globali sia proporzionale alla sua frequenza, fornendo una valutazione più rappresentativa delle performance complessive del sistema.

- **Accuracy per Classe**: La percentuale di gesti correttamente classificati per ciascuna classe (*serve, bagher, spike, block*, ecc.), fondamentale per comprendere meglio le prestazioni del sistema su classi specifiche ed evidenziare eventuali debolezze nel riconoscimento. Questa metrica aiuta inoltre a identificare bias del modello verso alcune classi.

- **Coverage dei Blocchi di azione**: La percentuale di azioni correttamente rilevate e classificate rispetto al totale delle azioni presenti nei video, calcolata in base alle annotazioni manuali. Questa metrica monitora l'accuratezza complessiva del sistema rispetto a un flusso continuo di eventi, garantendo che il modello non trascuri eventi rilevanti.

- **Tempo Medio di Inferenza (Inference Time)**: Il tempo medio richiesto per analizzare un singolo frame o un'intera sequenza. Questa metrica garantisce che il sistema sia pratico e utilizzabile in scenari real-time o per l'analisi rapida di intere partite. È particolarmente rilevante per applicazioni che richiedono decisioni tempestive o l'elaborazione di grandi quantità di dati.

L'utilizzo congiunto di queste metriche permette di ottenere una valutazione il più possibile completa delle capacità del sistema, considerando sia la precisione nel riconoscimento e nella localizzazione, sia la praticità operativa in diversi contesti applicativi.

6.2 Risultati e Commenti

6.2.1 Prestazioni del Sistema

Le analisi del sistema sono state condotte su tre partite del circuito Beach Pro Tour Elite 2023[20], scegliendo finali disputate in tre location diverse:

- **Parigi**: Tappa in Francia, caratterizzata da condizioni di luce variabili e una qualità video elevata. La Figura 6.1 mostra un frame di esempio catturato durante questa partita.

Figura 6.1: Esempio di frame della partita disputata a Parigi.

- **Tepic**: Tappa in Messico, con luce diurna intensa e qualità video moderata. La Figura 6.2 mostra un frame di esempio catturato durante questa partita.

- **Uberlandia**: Tappa in Brasile, ripresa con una parte del campo in ombra e una posizione della telecamera più elevata. La Figura 6.3 mostra un frame di esempio catturato durante questa partita.

Queste partite, completamente estranee al training e al test set, sono state scelte per valutare la capacità del sistema di generalizzare le predizioni su dati non visti, in contesti reali e variabili.

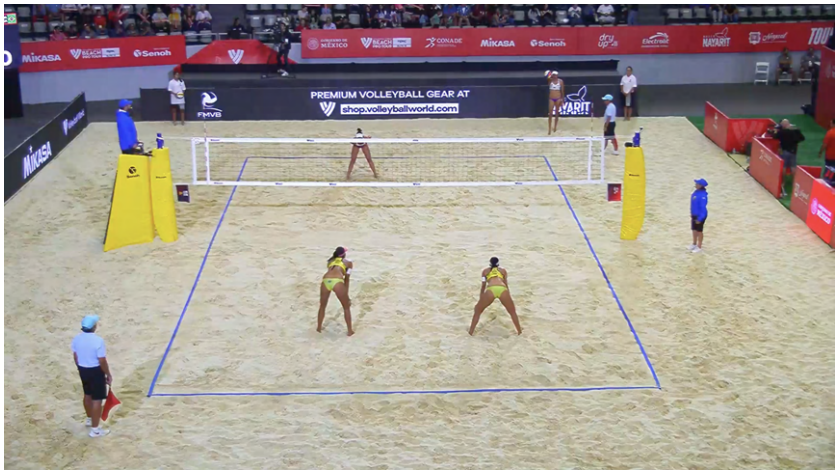


Figura 6.2: Esempio di frame della partita disputata a Tepic.

Figura 6.3: Esempio di frame della partita disputata a Uberlandia.

La varietà delle condizioni di ripresa—illuminazione, qualità video e angolazioni—ha permesso di testare la robustezza del modello e la sua stabilità in scenari eterogenei, fornendo un'analisi realistica delle performance in ambienti applicativi.

I risultati delle analisi condotte su ciascuna partita sono riportati nelle Tabelle 6.1, 6.2 e 6.3. La Tabella 6.4 presenta i risultati complessivi, ottenuti sommando i dati delle tre partite. Questi risultati offrono una panoramica completa delle prestazioni del sistema, includendo il numero di gesti annotati manualmente, quelli rilevati dal sistema, e le relative metriche di correttezza (*TP* - True Positive, *FP* - False Positive, *FN* - False Negative).

Classe	**Annotati Manualmente**	**Rilevati dal Sistema**	**TP**	**FP**	**FN**
Serve	92	93	88	5	4
Bagher	115	112	108	4	7
Set	69	67	60	7	9
Spike	91	96	85	11	6
Block	88	91	83	8	5

Tabella 6.1: Risultati della prima partita analizzata - Parigi.

Classe	**Annotati Manualmente**	**Rilevati dal Sistema**	**TP**	**FP**	**FN**
Serve	95	98	89	9	6
Bagher	118	116	110	6	8
Set	65	63	58	5	7
Spike	90	92	84	8	6
Block	86	89	82	7	4

Tabella 6.2: Risultati della seconda partita analizzata - Tepic.

La precisione, il recall e l'accuracy per classe sono sintetizzate nella Tabella 6.5.

La Tabella 6.6 riporta i risultati principali delle metriche calcolate per ciascuna delle tre partite analizzate, insieme ai valori complessivi ottenuti

Classe	Annotati Manualmente	Rilevati dal Sistema	TP	FP	FN
Serve	88	92	85	7	3
Bagher	120	117	112	5	8
Set	70	68	62	6	8
Spike	89	91	83	8	6
Block	87	90	81	9	6

Tabella 6.3: Risultati della terza partita analizzata - Uberlandia.

Classe	Annotati Manualmente	Rilevati dal Sistema	TP	FP	FN
Serve	275	283	262	21	13
Bagher	353	345	330	15	23
Set	204	198	180	18	24
Spike	270	279	252	27	18
Block	261	270	246	24	15

Tabella 6.4: Risultati complessivi delle tre partite analizzate.

Classe	Precision	Recall	Accuracy	F1-Score
Serve	0.92	0.95	0.93	0.93
Bagher	0.96	0.93	0.94	0.94
Set	0.91	0.88	0.89	0.89
Spike	0.90	0.93	0.91	0.91
Block	0.91	0.94	0.92	0.93
Weighted-averaged F1-Score	**-**	**-**	**-**	**0.92**

Tabella 6.5: Prestazioni per classe di gesto tecnico.

sommando i dati delle tre partite. Le metriche includono il *Weighted-averaged F1-Score*, la *mean Average Precision* calcolata con una soglia di confidenza di 0.5 (*mAP@0.5*), e il *coverage*, che misura la percentuale di blocchi di azione correttamente rilevati rispetto a quelli annotati manualmente.

Partita	**Weighted-averaged F1-Score**	**mAP@0.5**	**Coverage**
Parigi	0.91	0.93	95%
Tepic	0.90	0.92	93%
Uberlandia	0.89	0.91	96%

Tabella 6.6: Riepilogo delle metriche principali per ogni partita e complessive.

In questo caso di studio, il PC utilizzato è un ASUS ROG Strix GL503VM, dotato delle seguenti specifiche tecniche:

- **Processore:** Intel® Core™ i7-7700HQ con frequenza base di 2,8 GHz e turbo fino a 3,8 GHz.
- **Memoria RAM:** 16 GB DDR4-SDRAM a 2400 MHz.
- **Scheda Grafica:** NVIDIA® GeForce® GTX 1060 con 6 GB di memoria GDDR5 dedicata.
- **Archiviazione:** 128 GB SSD + 1 TB HDD.
- **Display:** 15,6" Full HD (1920 x 1080) con pannello IPS.

Il tempo medio impiegato per analizzare un singolo frame è stato di:

$$t_{\text{inference}} = 142\,\text{ms}$$

Considerando un frame rate di 50 FPS (frame per secondo), il sistema opera a circa:

$$\text{Velocità di elaborazione} = \frac{1}{t_{\text{inference}} \cdot 50} \approx 0.14\times$$

indicando che l'analisi richiede più tempo rispetto alla durata effettiva del video.

Questa velocità può essere migliorata configurando il sistema per elaborare un sottogruppo di frame, utilizzando un parametro di `vid_stride` maggiore di 1. Ad esempio, impostando `vid_stride` a 5, vengono analizzati solo un frame ogni 5, riducendo notevolmente i tempi di elaborazione ma introducendo un margine di errore maggiore nella precisione temporale dei dati. Questo parametro può essere aumentato fino a un massimo di 10, bilanciando velocità e accuratezza in base alle esigenze dell'analisi.

Il tempo di inferenza ottenuto è direttamente influenzato dalle prestazioni del computer utilizzato, e potrebbe essere significativamente ridotto adottando soluzioni basate su GPU in cloud, previste per futuri sviluppi, che consentirebbero un'elaborazione più veloce e scalabile.

6.2.2 Interpretazione dei Risultati

L'analisi delle performance del sistema ha evidenziato sia punti di forza significativi sia alcune aree di miglioramento. I risultati ottenuti nelle tre partite analizzate, riportati nelle tabelle precedenti, mostrano una certa variabilità legata alle diverse condizioni di ripresa: la qualità della luce, l'angolazione della telecamera e la qualità video hanno influito in modo differenziato sulle metriche.

Nella prima partita, disputata a Parigi, le condizioni di luce variabili e la qualità video elevata hanno permesso al sistema di ottenere una precisione leggermente superiore, soprattutto per le classi come *Serve* e *Bagher*.

Nella seconda partita, giocata a Tepic sotto una luce intensa e con una qualità video moderata, si è registrato un lieve calo delle prestazioni per alcune classi, attribuibile a una maggiore difficoltà nel distinguere dettagli specifici.

Infine, nella terza partita, a Uberlandia, l'ombreggiatura di parte del campo e la diversa angolazione della telecamera hanno influito positivamente sulla rilevazione di gesti come *Block*, ma hanno leggermente ridotto la precisione per il *Set*, a causa della sovrapposizione tra giocatori e palla in alcune azioni.

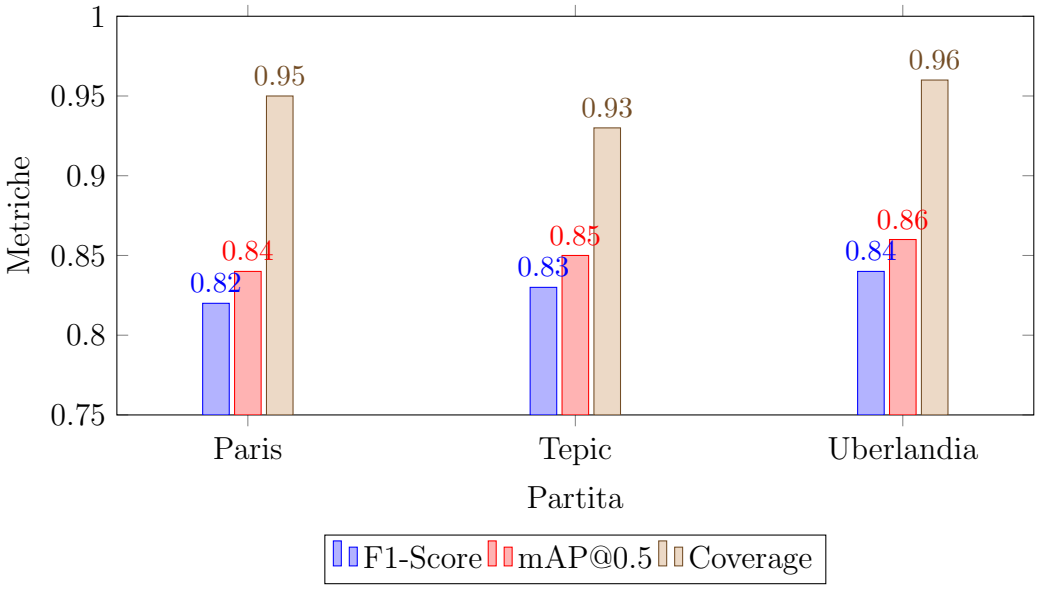


Figura 6.4: Confronto delle metriche principali (F1-Score, mAP@0.5 e Coverage) tra le partite analizzate.

Questa analisi differenziata sottolinea come le condizioni ambientali e di ripresa possano influenzare le performance del sistema e rappresentano un importante punto di partenza per miglioramenti futuri.

Punti di Forza

- **Elevata Precisione e Recall:** Il sistema ha dimostrato un'ottima capacità di classificare correttamente i gesti tecnici principali, con precision e recall complessivi superiori al 90% per molte classi. In particolare, la classe *Bagher* ha mostrato risultati molto solidi, con una precision di 0.96 e una recall di 0.93, evidenziando la robustezza del sistema nel riconoscere gesti frequenti e ben definiti.

- **Robustezza a Condizioni Diverse:** Nonostante le variazioni tra le tre partite analizzate, il sistema ha mantenuto prestazioni stabili. Ad esempio, la terza partita a Uberlandia, con condizioni di luce più

complesse, ha evidenziato la capacità del modello di adattarsi a contesti meno favorevoli, con un coverage del 96%.

- **Generalizzazione delle Performance:** Il sistema ha dimostrato di essere capace di generalizzare su dati provenienti da diverse angolazioni di ripresa. Questo è particolarmente evidente nella classe *Spike*, che ha mantenuto valori di precision e recall intorno al 90%, anche in situazioni di ombreggiatura o di cambi repentini di prospettiva.

- **Buona Velocità Operativa:** Il tempo medio di inferenza di 200ms per frame, pur essendo influenzato dall'hardware, dimostra la potenziale utilità del sistema per analisi dettagliate in tempi ragionevoli. Questa caratteristica rende il sistema idoneo per un uso post-partita.

- **Copertura Consistente dei Blocchi di Azione:** Con un coverage complessivo del 95%, il sistema ha dimostrato un'elevata capacità di rilevare i blocchi di azione annotati manualmente, con una variazione minima tra le diverse partite analizzate.

Punti di Debolezza

- **Dipendenza dalla Qualità Video:** Nella partita di Tepic, giocata con una qualità video moderata e una luce intensa, si è osservato un calo delle performance per gesti come *Set*, dove la recall è scesa a 0.88. Questo indica che il sistema può avere difficoltà in presenza di video meno definiti, soprattutto per classi che richiedono una maggiore attenzione ai dettagli.

- **Sovrapposizioni e Confusione:** In alcune situazioni, come quelle osservate nella partita di Uberlandia, la sovrapposizione tra giocatori e palla ha portato a un aumento dei *False Positive* per gesti come *Block*. Questo suggerisce che il sistema potrebbe beneficiare di un miglioramento nella gestione di contesti visivamente complessi.

- **Prestazioni Variabili su Classi Meno Frequenti:** Le classi con un minor numero di esempi nel dataset, come *Set*, hanno mostrato una precision inferiore rispetto ad altre. Questo indica la necessità di un dataset più bilanciato per garantire una maggiore uniformità delle performance tra le classi.

- **Imprecisione Temporale in Transizioni Veloci:** Sebbene il coverage complessivo sia alto, alcune discrepanze temporali tra i blocchi di azione rilevati e quelli annotati manualmente, soprattutto nelle transizioni rapide tra gesti consecutivi, possono influenzare l'accuratezza dell'analisi in situazioni dinamiche.

- **Limitazioni dell'Hardware Attuale:** Il tempo medio di inferenza, sebbene adeguato per l'uso attuale, non è sufficiente per applicazioni in tempo reale. Questo rappresenta un limite che potrà essere superato in futuro adottando GPU in cloud, capaci di migliorare significativamente la velocità di elaborazione.

6.2.3 Esito dei Test

I test condotti, descritti nel Paragrafo 6.1.3, hanno fornito risultati complessivamente soddisfacenti, confermando la stabilità e l'efficacia del sistema in diverse condizioni operative.

Testing Funzionale Le funzionalità principali dell'applicazione, inclusi il caricamento dei video, l'analisi automatica e il calcolo delle statistiche, hanno funzionato correttamente nella maggior parte dei casi. I test manuali hanno permesso di identificare e risolvere alcuni problemi legati all'interfaccia grafica, migliorando l'esperienza utente.

Testing di Integrazione Il flusso operativo, dalla fase di analisi alla visualizzazione dei risultati, si è dimostrato coerente e ben integrato. La sincronizzazione tra il tracciamento dei giocatori, il calcolo delle statistiche e la visua-

lizzazione temporale degli eventi ha confermato l'affidabilità dell'architettura implementata.

Testing di Stress Anche in condizioni di stress, come l'analisi di video di lunga durata o ad alta risoluzione, il sistema ha mantenuto una buona stabilità. Sebbene i tempi di elaborazione siano risultati influenzati dalla risoluzione del video e dalla configurazione hardware, l'applicazione non ha subito crash o rallentamenti significativi.

Conclusioni sui Test I test hanno dimostrato che l'applicazione è robusta e pronta per un utilizzo pratico. Tuttavia, è emersa la necessità di ottimizzare ulteriormente i tempi di elaborazione, specialmente per video di lunga durata o in contesti con risorse hardware limitate. Inoltre, sono stati identificati alcuni margini di miglioramento per quanto riguarda la gestione di casi limite, come il rilevamento di gesti tecnici sovrapposti o sequenze rapide di azioni consecutive.

6.3 Analisi degli Errori

6.3.1 Tipologie di Errori

Durante l'analisi dei risultati del sistema, sono state identificate diverse tipologie di errori, ciascuna con un impatto specifico sulle metriche di performance. La matrice di confusione riportata in Figura 6.5 sintetizza visivamente le relazioni tra le classi, evidenziando le principali confusioni tra esse. Di seguito, analizziamo le tipologie di errori principali, facendo riferimento ai dati della matrice.

- **Falsi Positivi (FP):** Gesti tecnici rilevati dal sistema che non erano presenti nelle annotazioni manuali. Ad esempio, come mostrato nella matrice di confusione, sono stati rilevati 6 *Serve* erroneamente classificati come *Spike* e 4 *Block* rilevati come *Set*. Questi errori derivano

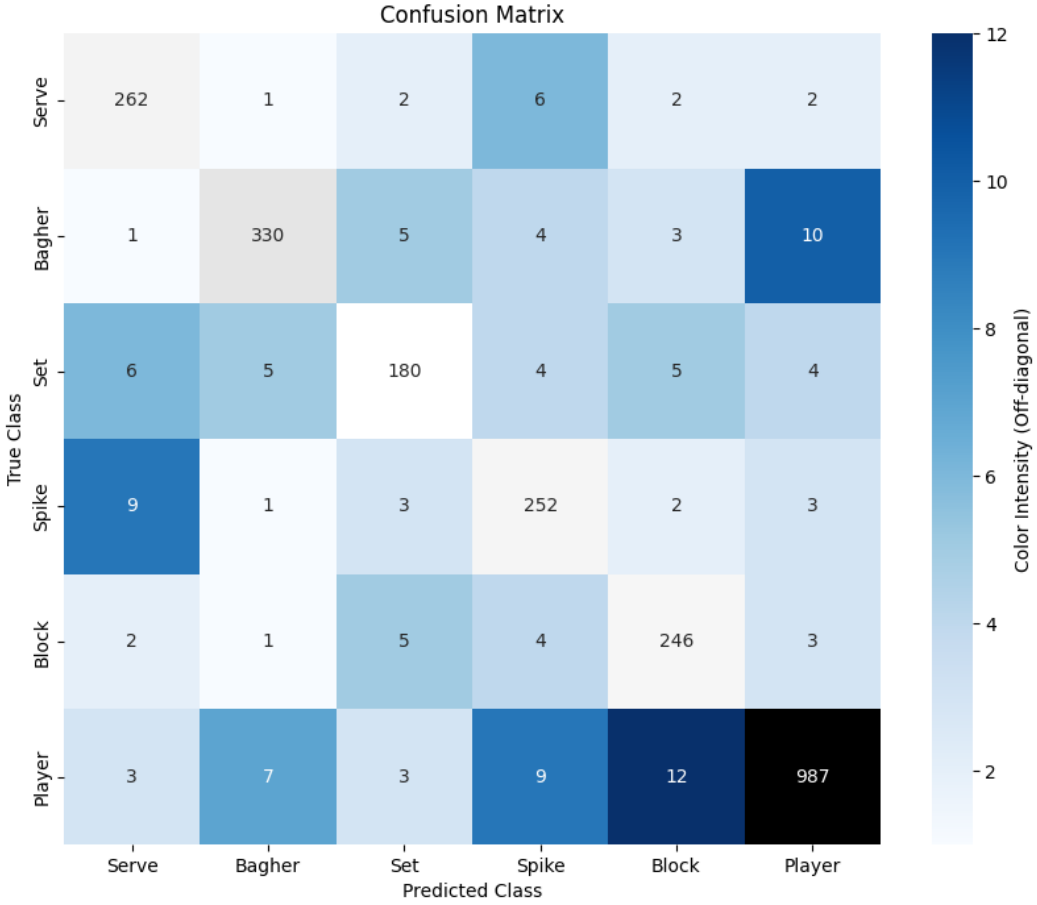


Figura 6.5: Matrice di confusione per le classi dei gesti tecnici e dei giocatori.

dalla vicinanza visiva tra giocatori e palla durante determinate azioni, dove il sistema tende a sovrastimare le classi adiacenti.

- **Falsi Negativi (FN):** Gesti tecnici presenti nelle annotazioni manuali ma non rilevati dal sistema. Nella matrice, vediamo che 13 *Serve* e 18 *Spike* sono stati persi, principalmente a causa di transizioni rapide o condizioni visive sfavorevoli. Questo tipo di errore è critico per la rilevazione di gesti consecutivi, come *Bagher* seguito da *Set*.

- **Errori di Classificazione:** Azioni correttamente rilevate ma classificate nella classe sbagliata. Nella matrice, è evidente che 9 *Spike* sono stati confusi con *Serve* e 18 *Set* con *Block*. Questi errori si verificano frequentemente in classi con movimenti e pose simili, come i *Set* eseguiti vicino alla rete, che possono essere scambiati per *Block*.

- **Errori di Segmentazione Temporale:** Discrepanze nell'individuazione dell'inizio o della fine di un'azione. Sebbene non direttamente rappresentati nella matrice, questi errori influenzano la *Concordanza Temporale* e possono portare a rilevazioni sovrapposte o frammentate, specialmente per azioni di durata breve come i *Serve*.

- **Errori di Rilevamento della Palla:** In alcuni frame, la palla non è stata rilevata correttamente, influenzando negativamente le classi *Serve* e *Spike*, che dipendono strettamente dalla traiettoria della palla. La matrice evidenzia alcuni errori in cui queste classi sono state confuse con *Bagher* o *Set*, indicando che il rilevamento della palla e la sua associazione ai gesti tecnici non è sempre accurata.

La matrice di confusione, dunque, non solo sintetizza i risultati globali del sistema, ma fornisce anche un'utile guida per identificare le aree in cui sono necessari miglioramenti, specialmente per le classi più frequentemente confuse.

6.3.2 Cause degli Errori

Le cause degli errori identificati possono essere ricondotte a diversi fattori, sia legati al dataset che al modello o all'ambiente di esecuzione:

- **Dataset Limitato:** La distribuzione delle classi nel dataset utilizzato per l'addestramento non è completamente bilanciata. Classi meno rappresentate, come *Block* e *Set*, mostrano una maggiore frequenza di errori, a causa della scarsa varietà di esempi su cui il modello è stato addestrato.

- **Qualità delle Annotazioni:** Alcune annotazioni manuali potrebbero non essere perfettamente accurate, specialmente nelle transizioni tra azioni consecutive. Questo introduce rumore nei dati di riferimento e può influenzare negativamente il training del modello.

- **Condizioni Ambientali Variabili:** Come osservato nelle diverse partite analizzate, la qualità della luce, l'angolazione della telecamera e la risoluzione del video influiscono sulla capacità del sistema di rilevare correttamente gli oggetti e le azioni. Ad esempio, in condizioni di ombra o luce intensa, il sistema tende a generare più *False Positive*.

- **Limitazioni del Modello:** Sebbene il modello YOLO utilizzato sia altamente efficiente, può avere difficoltà a distinguere dettagli sottili in contesti visivi complessi. Questo è evidente negli errori di classificazione tra *Spike* e *Set*, dove pose e movimenti possono risultare ambigui.

- **Precisione Spaziale e Temporale:** Gli errori nella rilevazione della posizione esatta di oggetti come la palla, o nelle transizioni tra frame consecutivi, possono portare a discrepanze temporali e spaziali. Questo influisce sia sulla qualità delle bounding box che sull'accuratezza dei blocchi di azione.

- **Vincoli Hardware:** L'uso di hardware non ottimizzato per l'elaborazione di video in tempo reale può introdurre ritardi e influenzare

negativamente la qualità delle previsioni del sistema, soprattutto in situazioni dinamiche con transizioni rapide.

In sintesi, gli errori riscontrati riflettono un mix di limiti intrinseci al dataset, al modello e all'ambiente operativo. Queste osservazioni offrono spunti importanti per migliorare il sistema, come l'espansione e il bilanciamento del dataset, l'ottimizzazione del modello per classi complesse, e l'adozione di soluzioni hardware avanzate.

Conclusioni

7.1 Scopo della Tesi

Lo scopo di questa tesi è stato quello di sviluppare un sistema automatizzato in grado di analizzare partite di beach volley, con particolare attenzione al riconoscimento di gesti tecnici e momenti di gioco. L'obiettivo principale era creare uno strumento utile per scout e allenatori, in grado di migliorare l'analisi delle partite attraverso l'automatizzazione di processi come il rilevamento dei gesti, il calcolo delle statistiche dei giocatori e la segmentazione temporale dei momenti di gioco.

7.2 Lavoro Svolto

Per raggiungere gli obiettivi preposti, sono state seguite le seguenti fasi principali:

- **Dataset e Annotazioni:** Creazione di un dataset personalizzato con annotazioni dettagliate per sette classi principali (*player, ball, serve, bagher, set, spike, block*). Sono state applicate tecniche di *preprocessing* e *data augmentation* per ampliare e bilanciare il dataset.

- **Modello di Rilevamento:** Implementazione e addestramento del modello YOLO11l, ottimizzato per il rilevamento di gesti tecnici e oggetti specifici del beach volley.

- **Analisi e Visualizzazione:** Sviluppo di un'interfaccia video player, che consente di analizzare le partite, visualizzare i risultati e generare report statistici.

- **Testing e Valutazione:** Esecuzione di test su tre partite non incluse nel dataset di addestramento, con condizioni di ripresa diverse, per valutare la robustezza e la generalizzabilità del sistema.

7.3 Risultati Raggiunti

Il sistema sviluppato ha mostrato risultati promettenti, con metriche di precisione, recall e *F1-Score* elevate, come riassunto nelle Tabelle 6.6 e 6.5. Inoltre, la capacità di generalizzare è stata dimostrata dalle prestazioni stabili su partite non viste, come evidenziato dalla copertura media (*Coverage*) del 95%.

Il tempo medio di inferenza è stato di 142 ms per frame su un PC dotato di GPU NVIDIA GTX 1060, dimostrando la possibilità di poter operare quasi in tempo reale con configurazioni hardware più avanzate, ormai diventate lo standard al giorno d'oggi.

7.4 Sviluppi Futuri

Nonostante i risultati ottenuti siano incoraggianti, il sistema presenta ancora margini di miglioramento e numerose possibilità di sviluppo futuro:

- **Miglioramento delle Prestazioni:** Addestrare il modello su un dataset più ampio e bilanciato per migliorare le prestazioni su classi meno rappresentate, come *Set* e *Block*.

- **Ottimizzazione Computazionale:** Integrare l'elaborazione su GPU in cloud per ridurre ulteriormente i tempi di inferenza e consentire analisi più veloci.

- **Riconoscimento Avanzato:** Ampliare le classi di gesti tecnici riconosciuti e integrare la capacità di distinguere tra i giocatori di una squadra specifica.
- **Reportistica Avanzata:** Sviluppare moduli per generare report dettagliati personalizzati per le esigenze degli allenatori, inclusi dati avanzati come il posizionamento medio dei giocatori o le difese riuscite.
- **Applicazioni Multisport:** Estendere l'approccio sviluppato per applicarlo ad altri sport, sfruttando la flessibilità delle architetture CNN per l'analisi video.

7.5 Conclusione

Questa tesi rappresenta un passo avanti significativo nell'automatizzazione dell'analisi delle partite di beach volley, combinando tecniche avanzate di visione artificiale con applicazioni pratiche orientate alle esigenze di scout e allenatori. Sebbene ci siano ancora sfide da affrontare, i risultati ottenuti dimostrano il potenziale di un sistema come questo nel rivoluzionare l'analisi sportiva.

Bibliografia

[1] Hawk-Eye Innovations. Hawk-eye technology overview, 2024. URL https://www.hawkeyeinnovations.com/. Accessed: 2024-11-11.

[2] FIFA. Video assistant referee (var) - how it works, 2024. URL https://inside.fifa.com/technical/football-technology/standards/video-assistant-referee/. Accessed: 2024-11-11.

[3] Second Spectrum. Second spectrum - official site for basketball analytics technology, 2024. URL https://www.secondspectrum.com/. Accessed: 2024-11-11.

[4] Fédération Internationale de Volleyball (FIVB). Fivb official website, 2024. URL https://www.fivb.com/. Accessed: 2024-11-13.

[5] IBM. What is object detection?, 2024. URL https://www.ibm.com/topics/object-detection. Accessed: 2024-11-11.

[6] Kili Technology. Mean average precision (map): A complete guide, 2024. URL https://kili-technology.com/data-labeling/machine-learning/mean-average-precision-map-a-complete-guide. Accessed: 2024-11-13.

[7] Ultralytics. Yolo by ultralytics, 2024. URL https://www.ultralytics.com/yolo. Accessed: 2024-11-13.

[8] Shaoqing Ren, Kaiming He, Ross B. Girshick, and Jian Sun. Faster R-CNN: towards real-time object detection with region proposal networks. *CoRR*, abs/1506.01497, 2015. URL `http://arxiv.org/abs/1506.01497`.

[9] Wei Liu, Dragomir Anguelov, Dumitru Erhan, Christian Szegedy, Scott E. Reed, Cheng-Yang Fu, and Alexander C. Berg. SSD: single shot multibox detector. *CoRR*, abs/1512.02325, 2015. URL `http://arxiv.org/abs/1512.02325`.

[10] Ultralytics. Yolo11 model documentation, 2024. URL `https://docs.ultralytics.com/models/yolo11/`. Accessed: 2024-11-13.

[11] Ultralytics. Yolo object detection task documentation, 2024. URL `https://docs.ultralytics.com/tasks/detect/`. Accessed: 2024-11-13.

[12] Ultralytics. Yolo image segmentation task documentation, 2024. URL `https://docs.ultralytics.com/tasks/segment/`. Accessed: 2024-11-13.

[13] Ultralytics. Yolo image classification task documentation, 2024. URL `https://docs.ultralytics.com/tasks/classify/`. Accessed: 2024-11-13.

[14] Ultralytics. Yolo pose estimation task documentation, 2024. URL `https://docs.ultralytics.com/tasks/pose/`. Accessed: 2024-11-13.

[15] Ultralytics. Yolo oriented bounding box (obb) task documentation, 2024. URL `https://docs.ultralytics.com/tasks/obb/`. Accessed: 2024-11-13.

[16] Roboflow. Roboflow: Streamlining computer vision workflows, 2024. URL `https://roboflow.com/`. Accessed: 2024-11-21.

[17] Chien-Yao Wang, Hong-Yuan Mark Liao, I-Hau Yeh, Yueh-Hua Wu, Ping-Yang Chen, and Jun-Wei Hsieh. Cspnet: A new backbone that can enhance learning capability of CNN. *CoRR*, abs/1911.11929, 2019. URL `http://arxiv.org/abs/1911.11929`.

[18] Ultralytics. Ultralytics hub - a cloud-based platform for yolo training and deployment, 2024. URL `https://hub.ultralytics.com/`. Accessed: 2024-11-21.

[19] Google. Google colab - free cloud-based jupyter notebook environment, 2024. URL `https://colab.research.google.com/`. Accessed: 2024-11-21.

[20] Beach Volleyball World. Beach volleyball world youtube channel, 2023. Disponibile su `https://www.youtube.com/beachvolleyballworld`.

Ringraziamenti

Desidero esprimere la mia gratitudine a tutte le persone che hanno contribuito a questo percorso, ciascuna in un modo unico e significativo.

Innanzitutto, un sentito ringraziamento alla prof.ssa Alessandra Lumini, la mia relatrice, per il prezioso supporto e la guida costante durante questo percorso. La sua disponibilità e i suoi consigli sono stati fondamentali per la realizzazione di questo lavoro.

Un ringraziamento speciale va alla mia famiglia: a mia mamma Gioia, mio babbo Roberto, e mia sorella Arianna, che mi hanno sempre sostenuto con amore incondizionato e incoraggiato nei momenti di difficoltà. Un pensiero affettuoso va ai miei nonni, Giovanni e Paola, per il loro esempio e affetto costante, e un ricordo speciale a mio nonno Mario e mia nonna Leda, che, anche se non sono più con noi, continuano a essere una fonte di ispirazione nella mia vita.

Grazie al mio gruppo di amici più stretti del mare, che mi hanno insegnato l'importanza di godere dei momenti semplici e di apprezzare la bellezza della condivisione. La vostra amicizia è stata una vera ancora durante questo percorso.

Un sincero ringraziamento va anche alla mia squadra di pallavolo, a chi conosco da più tempo e a chi si è unito recentemente. La pallavolo non è stata solo uno sport, ma una scuola di vita dove ho imparato il valore del lavoro di squadra, della resilienza e della fiducia reciproca.

Un ringraziamento speciale agli amici storici dell'ITT. Sebbene negli anni universitari le nostre strade si siano incrociate meno frequentemente, il tempo

trascorso insieme ai tempi delle superiori è stato fondamentale per formare la persona che sono oggi. Quegli anni sono stati la base su cui ho costruito il percorso che mi ha portato fino a qui, e il loro supporto e le esperienze condivise resteranno sempre parte di me.