POLITECNICO DI TORINO

Laurea Magistrale in Ingegneria Informatica - Grafica e Multimedia

Tesi Magistrale

"Progettazione di un sistema di Photofinish automatizzato per l'atletica leggera tramite tecniche di Intelligenza Artificiale"

Relatore	Candidato
Prof. Bartolomeo MONTRUCCHIO	Enrico PIZZUTI DE SARAKA

Luglio 2026

Abstract

La determinazione del riscontro cronometrico di una competizione sportiva non è affidata a macchine fotografiche tradizionali, bensì a sistemi di acquisizione *line-scan*. Tali dispositivi, operando con obiettivo fisso sulla linea del traguardo, campionano decine di migliaia di immagini sottilissime (con risoluzione orizzontale di 1 pixel) al secondo. L'immagine finale, conosciuta come photofinish, risulta essere una ricostruzione spazio-temporale del passaggio degli atleti sul traguardo.

Nonostante l'elevata frequenza di campionamento dei moderni hardware, la fase di refertazione del photofinish rimane vincolata all'intervento umano: il posizionamento manuale della linea del traguardo sul torso dell'atleta (riferimento per le competizioni di atletica leggera) porta a latenze operative e possibili errori che influiscono sulla precisione del dato finale.

Per assistere l'attività umana, il presente lavoro di tesi propone lo sviluppo e l'addestramento di un modello di DSS, basato sull'Intelligenza Artificiale, capace di identificare oggettivamente, a partire dall'acquisizione di immagini reali, il punto di contatto ottimale tra l'anatomia dell'atleta e la linea del traguardo.

Poiché le immagini acquisite dal sistema sono spesso caratterizzate da proporzioni estreme e distorsioni dovute all'acquisizione temporale, per isolare i singoli atleti è stata implementata una pipeline basata sull'architettura *YOLO*. Il sistema integra moduli di *Object Detection* e *Pose Estimation*, ottimizzati mediante strategie di *Sliding Window* e *NMS* classica per gestire l'elevata densità di pixel e non sovrapporre i concorrenti più vicini. La robustezza del modello è stata rafforzata dall'integrazione di filtri euristici per abbattere i falsi positivi.

L'innovazione del progetto risiede nella localizzazione automatica del punto esatto di attraversamento del traguardo da parte del torso degli atleti, elaborato come un problema di regressione di *heatmap* monodimensionale. Sono state progettate e messe a confronto due architetture: una rete convoluzionale dedicata, chiamata "*FinishNet*", ottimizzata con moduli di attenzione, in grado di simulare l'occhio umano, e una *ResNet*, addestrata tramite *fine-tuning*, capace di "abbattere" l'errore umano.

L'addestramento dei modelli *FinishNet* e *ResNet* è stato effettuato mediante

l'utilizzo di una funzione di costo basata sulla discrepanza tra la distribuzione di probabilità originale e la distribuzione di probabilità proposta dalla rete. Il risultato ottenuto dalla funzione di costo ha permesso la realizzazione di un sistema prototipale efficiente.

Nella fase finale del progetto, l'algoritmo di inferenza è stato integrato in un sistema hardware completo e testato fisicamente sul campo. Il prototipo complessivo gestisce l'acquisizione delle immagini in tempo reale da una fotocamera lineare, vincolando i dati ai *timestamp* hardware nativi dal sensore. Parallelamente, un microcontrollore esterno ha agito da sorgente di *trigger* software per comandare e sincronizzare le fasi di salvataggio.

Nonostante le attuali limitazioni dovute al numero esiguo di immagini di addestramento e al necessario intervento di un giudice di gara per la validazione e l'eventuale correzione finale del posizionamento della linea sul torso degli atleti, il progetto sviluppato si dimostra essere un robusto strumento di predizione in grado di gettare basi concrete per un'automazione volta a ottimizzare e superare i limiti di un'analisi esclusivamente umana.

Indice

Elenco delle tabelle **VII**

Elenco delle figure **VIII**

Acronimi **XIV**

1 Introduzione **1**

1.1 Il cronometraggio ufficiale nell'atletica leggera 1

1.1.1 Il sistema FAT e la tecnologia Line-Scan 1

1.1.2 Il regolamento World Athletics: il torso come riferimento . . 5

1.1.3 Il collo di bottiglia operativo: la refertazione manuale 8

1.2 Stato dell'arte 8

1.2.1 L'automazione del photofinish tramite computer vision classica 10

1.2.2 Object Detection su immagini ad alta risoluzione 11

1.2.3 Localizzazione e Heatmap Regression 11

1.3 Scopo del progetto 12

2 Tecnologie impiegate nel sistema **14**

2.1 Sistema hardware di acquisizione 14

2.1.1 Architettura generale del sistema 14

2.1.2 La fotocamera Line-Scan 16

2.1.3 Il sistema di trigger 20

2.1.4 L'unità di elaborazione (PC) 22

2.2 Protocolli e comunicazione 23

2.2.1 Il protocollo GigE Vision 23

2.2.2 Lo standard GenICam e la libreria Harvesters 25

2.2.3 Comunicazione seriale Arduino-PC 26

3 Preparazione e pre-processing del dataset **29**
3.1 Raccolta del dataset ed estrazione delle linee di arrivo 30
3.2 Pre-processing e gestione del formato Line-Scan 37
3.3 Data Augmentation 40
3.3.1 Trasformazioni geometriche e spaziali 41
3.3.2 Alterazioni fotometriche 43
3.4 Suddivisione del dataset 44

4 Architettura del sistema proposto **46**
4.1 Primo stadio: localizzazione e ritaglio degli atleti 47
4.1.1 Analisi delle prestazioni dell'approccio Ensemble (Detection e Pose Estimation) 52
4.2 Secondo stadio: regressione della linea di traguardo 54
4.2.1 Analisi dimensionale e preservazione della risoluzione temporale 54
4.2.2 Il target matematico: Heatmap Regression 56
4.2.3 Architettura proposta (FinishNet) 58
4.2.4 Modello di confronto: ResNet-34 con Transfer Learning . . . 63

5 Risultati sperimentali e validazione **66**
5.1 Setup sperimentale e metriche di errore 66
5.2 Analisi della convergenza (le curve di Loss) 68
5.3 Confronto prestazionale 71

6 Realizzazione del prototipo sperimentale e interfaccia utente **80**
6.1 Integrazione elettrica ed elettronica 81
6.2 Progettazione meccanica e assemblaggio hardware 87
6.3 Sviluppo dell'interfaccia grafica e usabilità 92

7 Conclusioni e Sviluppi Futuri **100**
7.1 Sintesi dei Risultati 100
7.2 Limiti del Sistema Proposto 101
7.3 Sviluppi Futuri 103

Bibliografia **105**

Elenco delle tabelle

2.1 Specifiche tecniche della telecamera Teledyne Dalsa Spyder3 SG-11-01k40-00-R [17]. 19
2.2 Specifiche tecniche della scheda Arduino Uno r3 [19]. 21

Elenco delle figure

1.1 Interconnessione e posizionamento dei componenti del sistema di photofinish. 2
1.2 Le telecamere *line-scan* acquisiscono sempre la stessa porzione di spazio ad intervalli di tempo costanti 3
1.3 L'immagine risultante è la composizione sequenziale delle scansioni effettuate ad intervalli di tempo costanti. 4
1.4 (a) Scan 'O' Vision ULTIMATE [5]; (b) EtherLynx Vision PRO X [6]. 5
1.5 Limite superiore della zona definita "torso" [7]. 6
1.6 Rappresentazione visiva dell'area anatomica corrispondente al "torso" [7]. 7
1.7 Dettaglio sul funzionamento degli algoritmi di *Computer Vision* applicati all'atletica leggera: analisi dinamica del moto nella disciplina di salto con l'asta (Scan 'O' Vision ULTIMATE di Omega). 9

2.1 I tre blocchi principali costituenti il progetto: l'elaboratore centrale comunica con Arduino Uno per sincronizzare le operazioni di acquisizione e con la telecamera per aggiornare i parametri e salvare i *frame* catturati. 15
2.2 High Sensitivity Mode: la camera acquisisce la stessa immagine due volte con le due linee separate, portando ad un risultato più luminoso. 17
2.3 Modello della telecamera Teledyne Dalsa Spyder3 SG-11-01k40-00-R [17]. 19
2.4 Modello della scheda Arduino Uno r3 [19]. 21
2.5 Architettura della pila protocollare *GigE Vision*. I protocolli applicativi dedicati al controllo (*GVCP*) e allo streaming video (*GVSP*) si appoggiano sugli standard di rete commerciali *UDP* e IP. 24

2.6 Architettura a strati del flusso dati per la gestione della telecamera. Il livello applicativo interagisce con il dispositivo tramite lo standard *GenICam*, demandando al *Producer GenTL* (Baumer GAPI) il compito di astrarre i protocolli di rete *GigE Vision*. La comunicazione è separata in un canale bidirezionale di controllo (*GVCP*) e un canale unidirezionale ad alte prestazioni (*GVSP*) per lo streaming del *payload* basato su *UDP*. 26
2.7 Lo schema riassume il processo di invio dei dati tramite la configurazione predefinita 8N1, con la spiegazione dei singoli bit. 28

3.1 Photofinish ufficiale di una gara sui 100m ai Campionati Mondiali di Londra del 2017. Le linee verdi impresse nell'immagine, inserite dai giudici di gara, indicano il tempo di arrivo degli atleti. 31
3.2 Confronto tra l'immagine originale (a) e quella elaborata tramite algoritmi di *inpainting* (b). In prossimità dei torsi dei corridori è presente una linea verticale sfocata. 32
3.3 L'immagine superiore è la versione grezza acquisita dalla telecamera (versione "pulita"); l'immagine sottostante è la versione etichettata per determinare il risultato finale. 33
3.4 Esempi di acquisizioni di photofinish estratte dal *dataset*, rappresentative della variabilità delle condizioni non ideali. Dall'alto verso il basso: 1. acquisizione caratterizzata da forte sovraesposizione, in cui i dettagli cromatici si perdono e le sagome risultano in forte controluce; 2. condizione di bassa luminosità e scarso contrasto con un'evidente dominante cromatica alterata; 3. acquisizione affetta da evidenti artefatti digitali, aberrazioni cromatiche e distorsioni sulle sagome. . 34
3.5 L'immagine superiore è la versione "pulita", quella centrale è la versione etichettata dai giudici. Nell'immagine situata più in basso, rappresentante la differenza tra le due immagini precedenti, è possibile notare la posizione delle linee di photofinish. 36
3.6 Dettaglio dell'interfaccia grafica del programma di etichettatura. Le linee rosse, estratte dalle immagini, possono essere spostate in caso di refusi. 37

3.7 Esempio di funzionamento della tecnica *Sliding Window*: viene ritagliata una sotto porzione dell'immagine che presenta altezza invariata e larghezza impostata via codice (es. 1800 pixel). 39
3.8 Visualizzazione del programma per estrarre i *crop* degli atleti, legandoli alla posizione relativa della linea di arrivo. 40
3.9 Distribuzione delle posizioni delle linee degli arrivi su tutto il *dataset*. Si notano due picchi maggiori dovuti alle caratteristiche intrinseche del dominio di lavoro. 42
3.10 Applicazione di alcune tecniche di *Data Augmentation* (Random Shift + Intensity Jitter) a quattro campioni estratti casualmente dal *dataset* di partenza, con il ricalcolo della coordinata da predire, considerata come *heatmap*. 43

4.1 Schema che mostra il funzionamento del sistema complessivo di refertazione dell'immagine basato sull'AI, dall'immagine in input fino alla predizione della *heatmap* corrispondente. 47
4.2 Visualizzazione del processo di unione al termine della prima fase. In blu sono evidenziati i *bounding box* generati dal modello di *Object Detection*; in verde quelli ricostruiti a partire dai *keypoint* anatomici della *Pose Estimation*. In rosso è evidenziato il macro-box risultante dall'algoritmo di blending, comprensivo del *padding* di sicurezza spaziale. 50
4.3 Visualizzazione del risultato finale al termine anche della seconda fase. 51
4.4 Visualizzazione della distribuzione della dimensione dei *crop* ottenuti isolando i singoli atleti dalle immagini originali di gara. 55
4.5 Nella finestra (a) sono rappresentate le gaussiane a partire dal valore del *Ground Truth* nel *dataset* (linea tratteggiata verde) e quella predetta dalla rete neurale (linea rossa). Nella finestra (b) è rappresentata la gaussiana predetta dall'AI mostrata sotto forma di *heatmap* con la sfumatura dal blu al rosso (blu: probabilità molto bassa, rosso: valore corretto, probabilità molto alta). 58
4.6 Schema ad alto livello della rete "*FinishNet*" nel quale si vedono i macro blocchi principali. 59
4.7 Schema raffigurante il livello *stem* nel dettaglio con le operazioni impiegate e il dettaglio sulle dimensionalità. 60

4.8 Schema raffigurante il blocco residuale con l'aggiunta del blocco SE nel corpo della rete con le operazioni impiegate e il dettaglio sulle dimensionalità. 61
4.9 Schema raffigurante il livello *Head* nel dettaglio con le operazioni impiegate e il dettaglio sulle dimensionalità. La curva gaussiana rappresenta il significato dei dati calcolati in uscita della rete. 62
4.10 Schema raffigurante l'architettura della *ResNet-34*. 65

5.1 Grafico che mostra l'andamento dell'apprendimento durante la fase di *fine-tuning* della *ResNet*. 69
5.2 Grafico che mostra l'andamento dell'apprendimento durante la fase di addestramento della *FinishNet* senza i blocchi SE. 70
5.3 Grafico che mostra l'andamento dell'apprendimento durante la fase di addestramento della *FinishNet* con l'aggiunta dei blocchi SE. 71
5.4 Grafico che mostra il confronto, dei due diversi modelli, dell'andamento della stabilità dell'errore assoluto e della varianza, in relazione alla qualità del lavoro svolto dai giudici di gara. 72
5.5 Grafico che mostra il confronto, dei due diversi modelli, la *ResNet* mostra un'accuratezza molto più elevata fin da subito. 73
5.6 Immagine molto sfocata. 74
5.7 Si nota chiaramente il torso dell'atleta, ma subito dopo è presente molto rumore che probabilmente ha confuso la rete. 75
5.8 Probabilmente la rete è stata ingannata dagli arti: braccio dell'atleta in primo piano o dal piede sullo sfondo. 75
5.9 Immagine molto sfocata e con la presenza di più atleti nello stesso bounding box hanno ingannato la rete. 76
5.10 Immagine molto rumorosa con un basso contrasto rendendo quasi impossibile la rilevazione del torso. 76
5.11 Si nota un netto distacco tra la fine del torso e lo sfondo. 77
5.12 Sebbene il torso dell'atleta possa confondersi con lo sfondo, la rete neurale è riuscita a svolgere un ottimo lavoro. 77
5.13 Si nota un netto distacco tra la fine del torso e lo sfondo. 78
5.14 Sebbene l'atleta avesse il braccio proteso in avanti, la rete ha correttamente ignorato l'arto. 78
5.15 Si nota un netto distacco tra la fine del torso e lo sfondo. 79

6.1 Schema elettrico dell'ingresso per l'alimentazione della telecamera Dalsa. 81
6.2 Progetto 3D della base del *case* per l'alloggiamento della millefori e del circuito di alimentazione. 82
6.3 Progetto 3D del coperchio a incastro per la chiusura del modulo di alimentazione. 83
6.4 Immagine raffigurante il cavo di alimentazione completo e assemblato per il funzionamento della telecamera Dalsa. 83
6.5 Schema elettrico raffigurante il circuito di controllo del pulsante di *trigger*. 85
6.6 Progetto 3D del supporto ergonomico per il pulsante di *trigger*. . . . 86
6.7 Immagine raffigurante il sistema di acquisizione completo per la gestione del *trigger* tramite la scheda Arduino Uno. 87
6.8 Schema dell'interfaccia meccanica della telecamera Dalsa Spyder 3. . 89
6.9 Progetto 3D della staffa di adattamento per il montaggio della telecamera su treppiede. 90
6.10 Setup sperimentale del sistema installato a bordo pista presso il centro sportivo Natta di Rivoli. 91
6.11 Schermata iniziale (Standby) visualizzata a seguito del controllo positivo della connessione hardware (telecamera e *trigger*). 92
6.12 Interfaccia in modalità LIVE. L'operatore utilizza il flusso di immagini continuo per il puntamento meccanico e la calibrazione ottica. 93
6.13 Interfaccia in modalità Gara (Standby). Il flusso video è oscurato in attesa dello start. 94
6.14 Schermata Gara Avviata: il cronometro è attivo e il sistema attende il segnale del *trigger* hardware. 95
6.15 Schermata Acquisizione (REC): alla pressione del *trigger*, i *frame* vengono registrati in memoria e visualizzati a schermo. 95
6.16 Ambiente di Revisione: panoramica dell'interfaccia con la proposta di classifica generata in automatico dall'Intelligenza Artificiale. 97
6.17 Visualizzazione dei *layer* di analisi dell'AI. In giallo i bounding box (*YOLO*), in rosso le linee di rilevamento del torso, corrispondenti alle linee di photofinish (*ResNet*), in verde l'estrazione delle corsie e in viola le linee di contatto piede-pista per l'assegnazione della corsia. . 98

6.18 Analisi di una corsa di mezzofondo. Si evince la gestione discontinua della memoria tramite *trigger*: il gap temporale tra due acquisizioni è reso esplicito graficamente sulla *timeline* senza alterare la sincronizzazione del cronometro tramite linea tratteggiata blu. . . . 99

Acronimi

ACM	Automatic Capture Mode.
ADAM	Adaptive Moment Estimation.
API	Application Programming Interface.
CNN	Convolutional Neural Network.
DSS	Decision Support System.
FAT	Fully Automatic Timing.
FICr	Federazione Italiana Cronometristi.
GenICam	Generic Interface for Cameras.
GenTL	Generic Transport Layer.
GPIO	General Purpose Input/Output.
GUI	Graphical User Interface.
GVCP	GigE Vision Control Protocol.
GVSP	GigE Vision Streaming Protocol.
IoU	Intersection over Union.
IP	Internet Protocol.
KLDivLoss	Divergenza di Kullback-Leibler.
MSE	Mean Squared Error.
NMS	Non-Maximum Suppression.

RGB	Red Green Blue.
SE	Squeeze-and-Excitation.
SNR	Signal-to-Noise Ratio.
TCP	Transmission Control Protocol.
UDP	User Datagram Protocol.
YOLO	You Only Look Once.

Capitolo 1

Introduzione

1.1 Il cronometraggio ufficiale nell'atletica leggera

Nell'atletica leggera esistono tre modalità principali per la determinazione della prestazione cronometrica degli atleti:

- il cronometraggio manuale, in cui il cronometro viene azionato e fermato manualmente dal giudice di gara, rappresenta il metodo più fallace poiché influenzato dal tempo di reazione umano;
- il cronometraggio a transponder, utilizzato principalmente per le competizioni di massa (es. maratona), il cui margine di errore è legato alla rilevazione e alla posizione del chip allegato al pettorale dell'atleta;
- il cronometraggio completamente automatizzato o FAT, che consente l'acquisizione dei tempi d'arrivo senza qualsivoglia intervento umano a seguito dell'inizio dello scorrere del tempo.

1.1.1 Il sistema FAT e la tecnologia Line-Scan

Per convalidare e ufficializzare le prestazioni registrate durante le competizioni di atletica leggera è strettamente necessario che il cronometraggio sia completamente automatizzato (FAT). Alla partenza, il sistema viene avviato in automatico dallo sparo della pistola dello starter, la quale è collegata, via cavo o tramite trasmettitore

radio, all'elaboratore centrale. Sul traguardo è invece posizionata una telecamera *line-scan*, allineata perfettamente alla linea di arrivo, in grado di registrare il passaggio degli atleti, scattando fotogrammi ad una velocità pari a migliaia di linee al secondo.

Secondo la norma TR19.13.1 del Regolamento Tecnico della *World Athletics*, un riscontro cronometrico è ritenuto valido per l'omologazione se la sua misurazione è stata effettuata mediante sistema FAT e il ritardo complessivo tra la detonazione della pistola dello starter e l'avvio del cronometraggio è inferiore al millesimo di secondo [1]. Il sistema di photofinish produce un'immagine composita continua, caratterizzata da almeno 1000 linee verticali al secondo, sincronizzata mediante una scala temporale graduata uniformemente in 0.01 secondi [1].

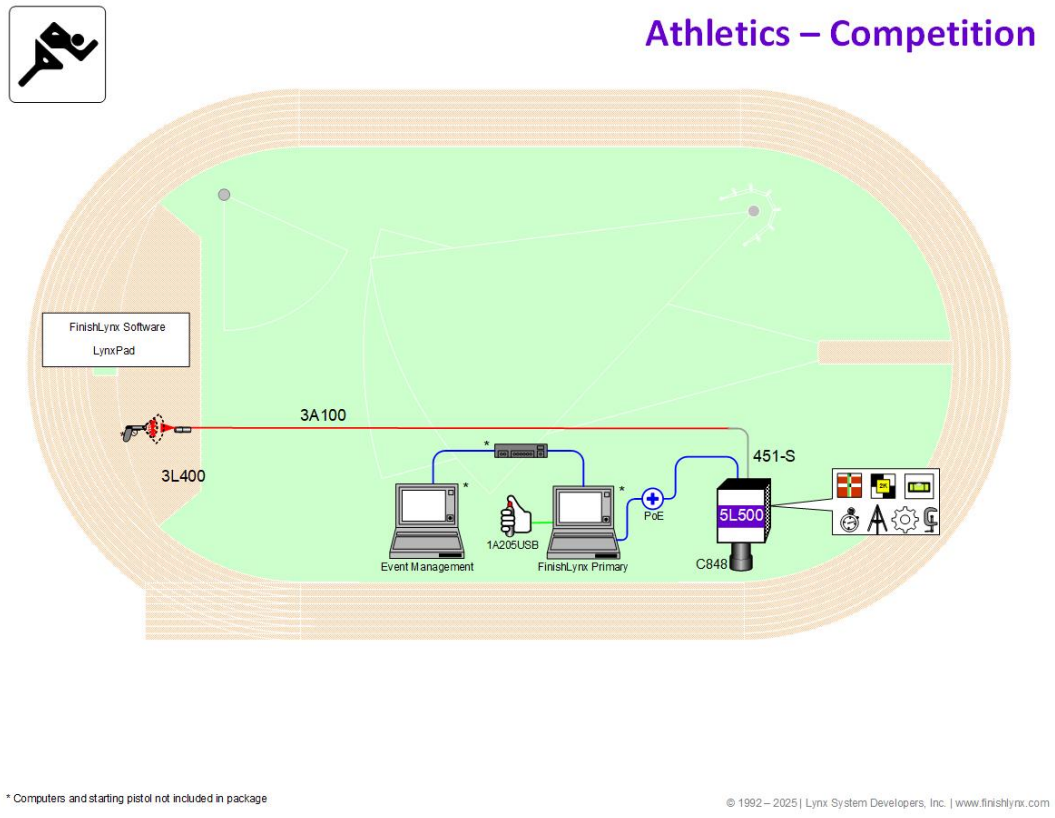


Figura 1.1: Interconnessione e posizionamento dei componenti del sistema di photofinish. Fonte: [2].

Il sistema di photofinish è costituito da telecamere *line-scan*, dispositivi capaci di acquisire immagini verticali sottilissime (tipicamente con risoluzione orizzontale

di 1 pixel), il cui obiettivo è puntato esattamente sulla linea del traguardo. Poiché l'obiettivo è fisso, la telecamera scansiona a ripetizione lo stesso punto migliaia di volte al secondo. Allineando e sequenziando i campioni acquisiti, si ottiene un'immagine in cui l'asse orizzontale non rappresenta lo spazio bensì lo scorrere del tempo: la posizione del busto degli atleti indica con precisione millimetrica l'istante esatto dell'arrivo.

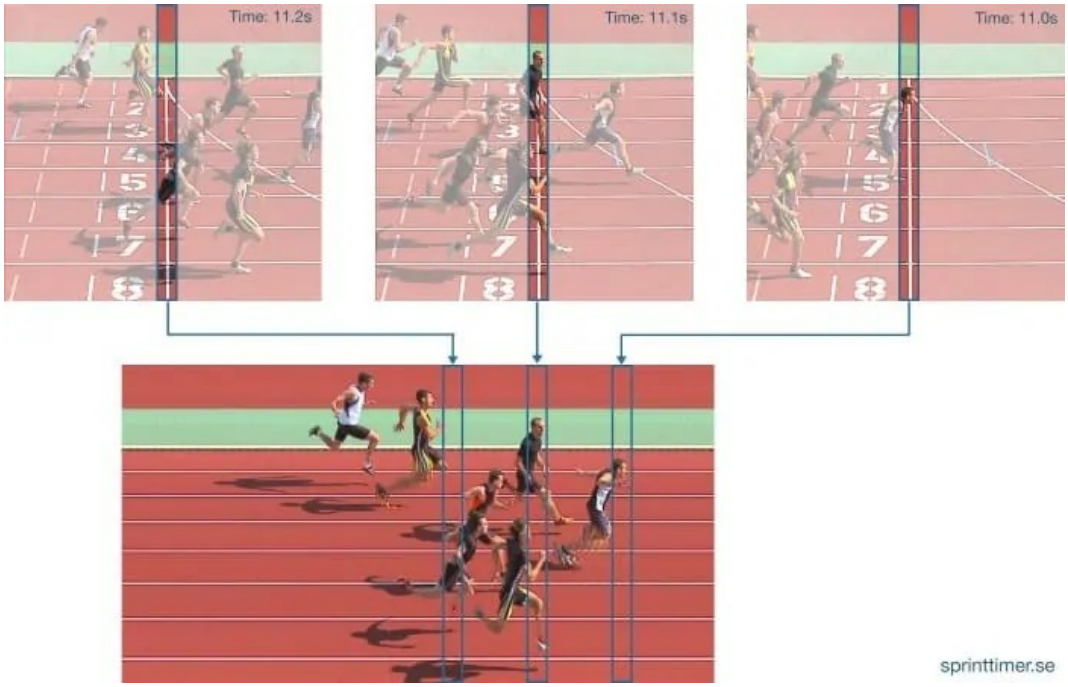


Figura 1.2: Le telecamere *line-scan* acquisiscono sempre la stessa porzione di spazio ad intervalli di tempo costanti. Fonte: [3].

Nel sistema di photofinish, la forma dei corpi dipende dal rapporto tra la loro velocità e la frequenza di campionamento della telecamera: se un atleta si muove lentamente, la sua immagine apparirà dilatata, mentre se si muove velocemente, l'immagine risulterà compressa. È per questo che gli arti degli atleti, muovendosi a velocità differenti rispetto al busto, appaiono spesso deformati o innaturalmente allungati. Lo sfondo bianco visibile dietro ai corridori rappresenta la porzione statica del traguardo inquadrata dalla telecamera.

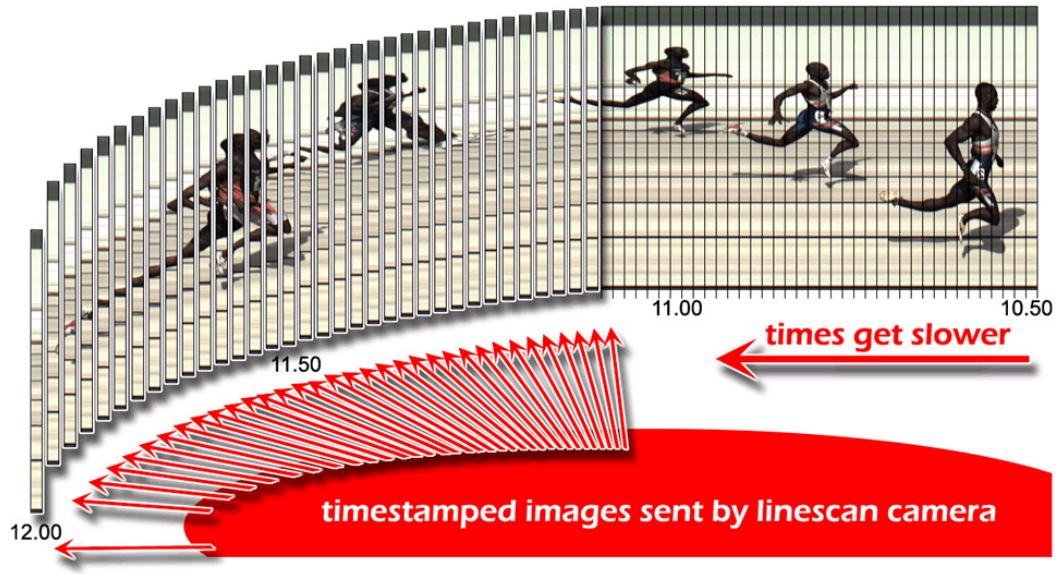


Figura 1.3: L'immagine risultante è la composizione sequenziale delle scansioni effettuate ad intervalli di tempo costanti. Fonte: [4].

Nel panorama attuale dei dispositivi per il photofinish, le tecnologie più avanzate sono rappresentate dai sistemi Scan 'O' Vision ULTIMATE di Omega [5] e EtherLynx Vision PRO X di Lynx System Developers [6]. Le suddette attrezzature sono progettate per catturare fino a 40.000 linee verticali al secondo con una risoluzione temporale infinitesimale, ideale per gli sport in cui la velocità è il fattore maggiormente discriminante (es. atletica leggera, ciclismo, sport motoristici) [6]. Le linee verticali dei sensori impiegati per il photofinish possiedono risoluzioni elevatissime (fino a 5120 pixel) che conferiscono l'acquisizione di dettagli estremamente nitidi, riducendo drasticamente il margine di errore durante l'allineamento manuale del cursore del tempo. Grazie all'elevata velocità di acquisizione dei dispositivi multimediali, il tempo di esposizione per singolo *frame* è quasi nullo.

Una volta generata l'immagine di photofinish, il giudice di gara analizza il grafico spazio-temporale posizionando un cursore sul torso dell'atleta, determinando il tempo di arrivo degli atleti con una precisione al millesimo di secondo. L'integrazione dell'Intelligenza Artificiale mira ad automatizzare il riconoscimento anatomico e il posizionamento della linea, riducendo l'intervento umano nel processo di analisi.

Per massimizzare la qualità luminosa delle immagini acquisite, in particolar modo durante le sessioni serali o notturne, gli impianti sportivi di atletica leggera vengono

dotati di riflettori, posizionati per garantire un'illuminazione ottimale all'altezza del petto degli atleti. Alcuni sensori, come quello di EtherLynx Vision PRO X, integrano tecnologie di amplificazione della luce capaci di restituire immagini nitide anche in condizioni di illuminazione artificiale.

(a) **(b)**

Figura 1.4: (a) Scan 'O' Vision ULTIMATE. Fonte: [5]; (b) EtherLynx Vision PRO X. Fonte: [6].

1.1.2 Il regolamento World Athletics: il torso come riferimento

Individuare l'istante esatto del passaggio sul traguardo può apparire intuitivo, ma nel cronometraggio ad alta precisione la questione assume una complessità tecnica notevole. Il primo passo consiste nello stabilire, in modo inequivocabile, quale parte del corpo sia determinante per la validazione del tempo finale.

Nell'atletica leggera, il piazzamento non è determinato dal primo arto che oltrepassa la linea (come un piede o una mano), né dalla testa. Le norme tecniche sanciscono che il cronometro si arresti solo quando il torso dell'atleta raggiunge il piano verticale del traguardo [7]. Questa convenzione evita che l'esito della gara venga alterato da movimenti accessori, come una falcata allungata o un tuffo in avanti, riflettendo invece unicamente il reale avanzamento del baricentro dell'atleta.

Tuttavia, tradurre il concetto di "torso" in una definizione matematica applicabile alle immagini distorte del photofinish rappresenta una sfida complessa, specialmente per i sistemi di analisi automatizzata. La variabilità dello sviluppo muscolare tra i singoli atleti rende infatti difficile identificare un confine visivo omogeneo e universale.

Per garantire uniformità nei giudizi a livello globale, le linee guida ufficiali della *World Athletics* forniscono una delimitazione anatomica estremamente rigorosa:

- il limite superiore del torso corrisponde all'estremità esterna dell'articolazione della clavicola ed è collocato solitamente al confine tra il centro della spalla e il suo apice esterno, a circa un terzo della distanza tra il collo e la punta della spalla [7];
- il limite inferiore del torso è fissato sulla sezione trasversale che passa per la linea dell'anca che cinge la parte più ampia dei fianchi, situata tra l'addome e il cavallo [7].

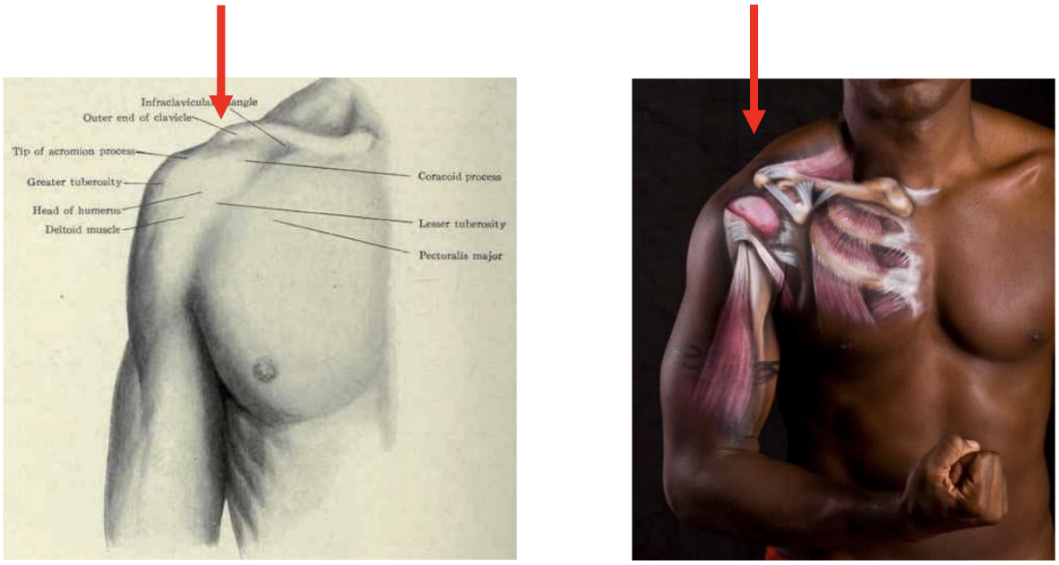


Figura 1.5: Limite superiore della zona definita "torso". Fonte: [7].

Figura 1.6: Rappresentazione visiva dell'area anatomica corrispondente al "torso". Fonte: [7].

La rigorosa specificità dei confini anatomici del torso delinea la complessità del compito computazionale. Istruire un sistema di Intelligenza Artificiale a discriminare con precisione chirurgica tali punti, isolando il torso ed escludendo sistematicamente arti e testa, rappresenta una sfida tecnologica non banale. L'obiettivo dell'elaborato è, dunque, la creazione di un modello di *Computer Vision* capace di operare su immagini intrinsecamente complesse, caratterizzate da numerose sovrapposizioni tra soggetti e forti distorsioni anatomiche, garantendo un'accuratezza nel posizionamento della linea di arrivo pari o superiore a quella del giudizio umano.

1.1.3 Il collo di bottiglia operativo: la refertazione manuale

Nonostante l'evoluzione dell'hardware di acquisizione, l'analisi del photofinish resta ancora attualmente subordinata all'interpretazione umana. Gli strumenti di assistenza esistenti presentano notevoli limiti operativi dovuti a un approccio basato sulla *Computer Vision* classica, privo di comprensione semantica del contenuto.

L'ACM, presente in sistemi come FinishLynx, è progettato per ottimizzare la memoria dell'hardware evitando registrazioni continue attivando il salvataggio dei dati solo quando, tramite algoritmi di *Motion Detection* (nello specifico, *Line Differencing*) [8], rileva una variazione di contrasto dei pixel. Il sistema così strutturato è vulnerabile a numerosi falsi positivi: ombre, riflessi o agenti atmosferici innescano l'acquisizione senza distinguere l'atleta dall'elemento di disturbo.

Durante la fase di analisi dell'immagine risultante, gli algoritmi di *Edge Detection* tentano di individuare il punto corretto identificando la maggiore discontinuità cromatica lungo l'asse temporale. Questo metodo risulta spesso fallace poiché la linea viene tracciata sul primo pixel utile, che solitamente appartiene a un arto o alla testa, e non al torso come richiesto dal regolamento *World Athletics* [7].

A causa di tali lacune, *World Athletics* non riconosce la validità ufficiale dei riscontri cronometrici senza l'intervento di un giudice di gara specializzato, il quale deve identificare manualmente il torso, distinguendolo da elementi fuorvianti come i pettorali non sempre aderenti [7].

Questo processo umano rappresenta un critico "collo di bottiglia" operativo, soggetto a stanchezza e potenziali errori percettivi.

1.2 Stato dell'arte

L'espressione "stato dell'arte" indica il livello più avanzato di sviluppo, conoscenze o prestazioni raggiunto da una determinata tecnologia in un dato momento.

L'analisi delle performance sportive ha vissuto un'evoluzione esponenziale grazie al *Deep Learning*. L'efficacia delle architetture basate su CNN, in particolare l'algoritmo *YOLO* [9], combinata con algoritmi di *Pose Estimation*, ha permesso di analizzare metriche rilevanti, in tempo reale, senza l'ausilio di sensori fisici sugli atleti [10].

L'analisi biomeccanica del movimento ha lo scopo di fornire dati statistici avanzati ad allenatori e spettatori e di esaminare e correggere il gesto tecnico mediante il calcolo di angoli articolari e forze di carico.

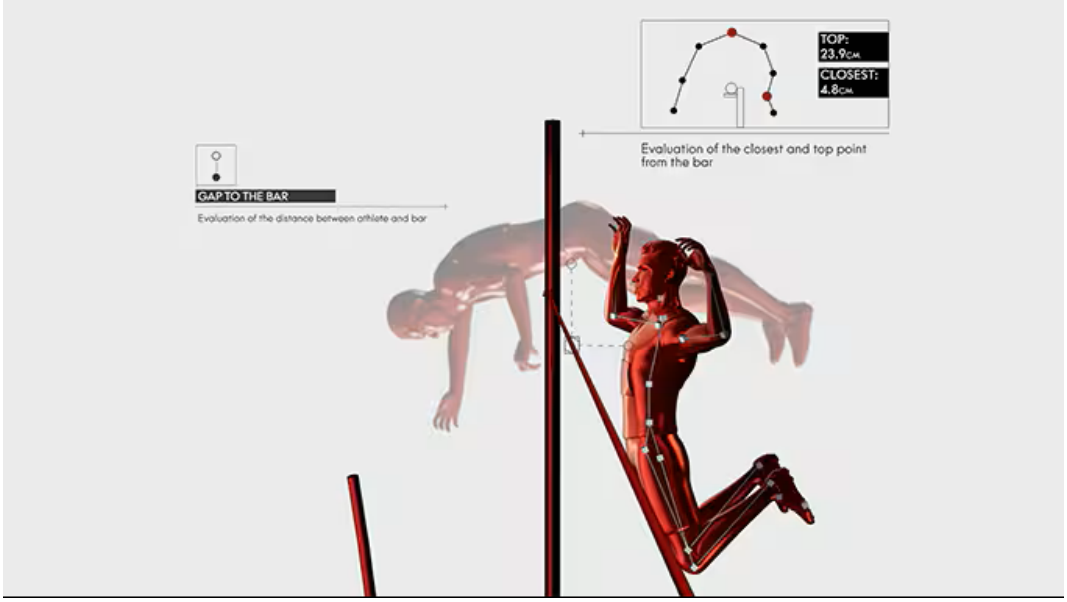


Figura 1.7: Dettaglio sul funzionamento degli algoritmi di *Computer Vision* applicati all'atletica leggera: analisi dinamica del moto nella disciplina di salto con l'asta (Scan 'O' Vision ULTIMATE di Omega). Fonte: [11].

La diffusione di *YOLO* è avvenuta grazie alla sua capacità di elaborare immagini in tempo reale eseguendo la *detection* in un singolo passaggio computazionale.

Tuttavia, l'applicazione di tali modelli al sistema del photofinish introduce il problema del *Domain Shift*: essendo addestrate al riconoscimento di immagini prospettiche standard, le strutture faticano ad operare su acquisizioni *line-scan* deformate.

Nonostante l'elevatissimo livello tecnologico di cui ormai l'ambiente sportivo dispone, il posizionamento della linea di arrivo sul torso degli atleti nel photofinish rimane un processo esclusivamente manuale.

1.2.1 L'automazione del photofinish tramite computer vision classica

Tra i lavori fondamentali che hanno esplorato l'automazione della refertazione del photofinish, prima dell'affermazione del *Deep Learning*, spicca lo studio proposto da Zhao et al. (2013) per il rilevamento automatico della linea di traguardo. Gli autori hanno sviluppato una serie di passi basati su regole euristiche ed elaborazione geometrica dell'immagine [12].

Il sistema proposto in tale studio si articola in quattro fasi principali:

1. all'inizio viene impiegato un algoritmo che calcola una colonna di *background*, guardando la statistica dei pixel, per eliminare il rumore tipico delle immagini *line-scan*, come strisce orizzontali [12];

2. il sistema successivamente assume proporzioni corporee fisse, scartando matematicamente i due quinti inferiori della *silhouette* (corrispondenti alle gambe) e calcola la distanza euclidea per individuare il centro di massa dell'atleta e isolare solo la sua parte frontale [12];

3. in seguito, per evitare che le braccia occludano il petto, viene applicato un filtro geometrico fisso così, se un gruppo di pixel rispetta le specifiche proporzioni spaziali di questo *template*, viene etichettato come "braccio" e cancellato dall'immagine [12];

4. in fine, il bordo frontale dell'atleta viene trattato come una funzione matematica. Attraverso il calcolo della derivata prima e seconda di tale curva, il sistema definisce una Estimative Target Region per localizzare il punto sporgente che rappresenta il petto [12].

Il metodo è stato validato su un *dataset* di immagini contenenti ciascuna un singolo atleta. I risultati hanno riportato un'accuratezza dell'86.3% all'interno di una tolleranza temporale di 4 millisecondi.

Tuttavia, lo studio fa emergere chiaramente le criticità degli approcci deterministici: gli algoritmi subiscono drastici cali di prestazione in condizioni di illuminazione non ottimali o in presenza di rumore visivo. Inoltre, il sistema fallisce frequentemente e produce falsi rilevamenti quando gli atleti presentano pose atipiche, come l'inclinazione della spalla in avanti in prossimità del traguardo [12].

L'uso di regole geometriche rigide (come *template* predefiniti o tagli proporzionali fissi) si rivela insufficiente per gestire la variabilità morfologica, le occlusioni multiple e le deformazioni estreme tipiche dell'atletica leggera.

Tali sfide rendono necessario il passaggio ad architetture guidate dai dati, come le reti *YOLO* e le CNN implementate nel presente lavoro di tesi, capaci di estrarre e astrarre le *feature* del torso in maniera del tutto indipendente da rigidi vincoli matematici.

1.2.2 Object Detection su immagini ad alta risoluzione

Data la discrepanza di risoluzione, l'immissione di immagini acquisite mediante dispositivi line-scan (es. 10.000×1.000 pixel) all'interno degli input standard di *YOLO* (640×640 pixel) comporterebbe un'ingente perdita di informazioni spaziali e morfologiche. Per far fronte a questo limite si adotta una strategia basata su una *Sliding Window* parzialmente sovrapposta [13].

L'immagine originale viene sezionata in segmenti che preservano la risoluzione nativa, permettendo al modello di operare nel suo dominio ottimale. Le predizioni ridondanti generate nelle aree di sovrapposizione sono gestite tramite *NMS* basata su IoU.

Nel prototipo finale, discusso in questo lavoro di tesi, è stata implementata una strategia di validazione *custom* che si affida a filtri euristici per agglomerare i *bounding box* generati, evitando la soppressione di istanze valide e discriminando con precisione i falsi positivi, mantenendo elevate le prestazioni di *detection* di *YOLO*.

1.2.3 Localizzazione e Heatmap Regression

Per l'individuazione di coordinate esatte di un determinato punto d'interesse, come ad esempio l'individuazione dei giunti umani per la posa, la letteratura scientifica sconsiglia l'utilizzo della regressione diretta di valori singoli (coordinate puntuali x) a causa della scarsa stabilità e della difficoltà di convergenza della funzione di perdita (*loss function*) in presenza di errori millimetrici. In questo caso si può scegliere la *Heatmap Regression* per l'estrazione della coordinata desiderata [14].

La rete non predice un singolo numero, ma genera una distribuzione di probabilità spaziale modellata secondo una curva Gaussiana, centrata sul punto di interesse.

Questo approccio offre una *soft supervision* capace di gestire l'incertezza spaziale e le inevitabili tolleranze derivanti dall'etichettatura manuale del *dataset*.

La coordinata definitiva viene infine estratta individuando il valore massimo della distribuzione prodotta, garantendo la precisione voluta.

1.3 Scopo del progetto

L'obiettivo principale di questo lavoro è la progettazione e lo sviluppo di un DSS *end-to-end* automatizzato basato su Intelligenza Artificiale, concepito per assistere e supportare i giudici di gara nella complessa fase di refertazione delle immagini di photofinish.

Il sistema proposto mira a colmare l'attuale divario tecnologico creando un ponte tra l'hardware di acquisizione e i modelli avanzati di *Computer Vision*, pur riconoscendo che la validazione ufficiale e definitiva di un risultato agonistico richiede l'imprescindibile supervisione umana. Lo scopo della tesi è, dunque, automatizzare il posizionamento della linea di photofinish dotando il sistema della sensibilità necessaria per riconoscere il corpo dell'atleta, isolarne le appendici e identificare univocamente il tronco, automatizzando l'estrazione di un risultato ufficiale e riducendo drasticamente i tempi di refertazione

Nello specifico, i contributi pratici e metodologici introdotti si sviluppano in un processo che abbraccia sia lo sviluppo hardware che quello software, gestendo l'intero flusso di lavoro. Il sistema amministra il processo dall'acquisizione fisica del segnale di partenza (per far partire il cronometraggio della competizione), al salvataggio dei *frame* provenienti dalla fotocamera, tramite l'integrazione di un microcontrollore (Arduino Uno), fino alla successiva elaborazione software dell'immagine composita risultante.

Per analizzare tali acquisizioni, il programma implementa una sequenza algoritmica specializzata per le immagini *line-scan*, progettata per gestire e superare le criticità del *Domain Shift*, unendo l'estrazione iterativa dei soggetti, tramite tecniche di *Sliding Window* e filtri geometrici, con modelli di *Heatmap Regression*, garantendo così una stima altamente precisa della coordinata spaziale. L'unione di queste tecnologie conduce ad una ottimizzazione del carico cognitivo e dei tempi di analisi.

Il sistema sviluppato agisce come un supporto per il giudice, fornendo in pochi secondi un'ipotesi di refertazione, permettendo di ridurre i tempi di attesa necessari per l'ispezione visiva manuale e mitigando, al tempo stesso, il rischio di errori derivanti dalla stanchezza dell'operatore.

In sintesi, il lavoro dimostra come le architetture di *Deep Learning* possano essere efficacemente adattate a domini visivi "anomali", come quello del photofinish, offrendo un solido strumento di stima predittiva in grado di modernizzare e velocizzare le operazioni di cronometraggio nell'atletica leggera.

Capitolo 2

Tecnologie impiegate nel sistema

In questo capitolo vengono descritti gli strumenti hardware, l'infrastruttura di rete e gli standard di comunicazione utilizzati per la realizzazione del sistema complessivo di photofinish. Verrà inizialmente presentato lo schema a blocchi ad alto livello dell'architettura risultante. Successivamente, saranno analizzati nel dettaglio i tre nodi fisici che compongono il prototipo: il sensore *line-scan* ad alta frequenza preposto alla cattura ottica, il sistema di *trigger* hardware, basato su microcontrollore (Arduino Uno) per la gestione in tempo reale dell'evento di "start", e infine l'unità di elaborazione centrale (PC), incaricata di gestire il massiccio flusso di rete, l'archiviazione e l'analisi delle immagini.

In seguito, verrà analizzato lo stack dei protocolli (standard *GigE Vision* e *GenICam*) e la comunicazione seriale, necessari per il trasporto affidabile dei dati.

2.1 Sistema hardware di acquisizione

2.1.1 Architettura generale del sistema

Per la realizzazione dell'intera infrastruttura che supporta il processo di refertazione del photofinish è stato progettato un sistema *end-to-end*. L'apparato è in grado di acquisire le immagini in tempo reale, associandole agli istanti temporali di campionamento, e prepararle per l'inferenza della rete neurale. L'architettura del sistema di cattura si compone di tre macro-blocchi principali:

- un'unità di *trigger* intercetta il segnale fisico di inizio del salvataggio dei *frame*, proveniente dalla pressione del pulsante di acquisizione, e invia il comando di

avvio al computer centrale, tramite la scheda Arduino Uno r3, mediante la comunicazione seriale;

- un'unità di elaborazione centrale (PC) riceve il segnale di *trigger* da Arduino Uno, acquisisce il flusso di immagini dalla telecamera tramite protocollo *GigE Vision* e processa le immagini risultanti;
- un'unità di acquisizione ottica costituita da una telecamera industriale ad altissima frequenza allineata con il traguardo.

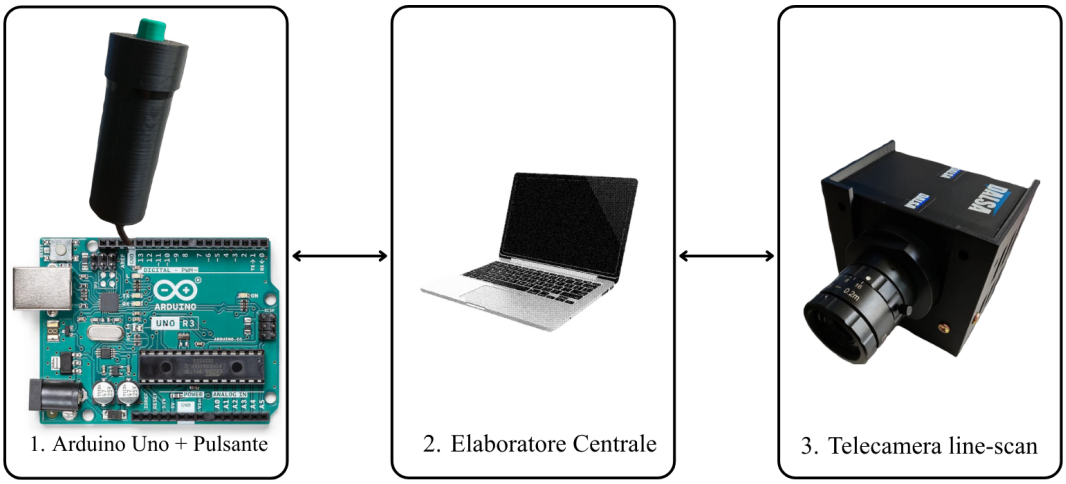


Figura 2.1: I tre blocchi principali costituenti il progetto: l'elaboratore centrale comunica con Arduino Uno per sincronizzare le operazioni di acquisizione e con la telecamera per aggiornare i parametri e salvare i *frame* catturati. Fonte PC: [15], Fonte Arduino: [16].

Una volta che la telecamera è stata avviata, essa acquisisce ininterrottamente e invia i *frame* ottenuti, e i relativi *timestamp*, all'elaboratore. La pressione fisica del pulsante induce il software di elaborazione sul PC ad iniziare a memorizzare in un *buffer* le linee di pixel catturate. Al rilascio del pulsante, l'immagine spazio-temporale smette di essere composta e, al termine della gara, viene fornita in input al modello di analisi.

L'output ottenuto è una stima della posizione della linea dell'arrivo degli atleti, con i relativi tempi associati e le rispettive corsie in cui hanno corso.

2.1.2 La fotocamera Line-Scan

Per fotografare l'arrivo, l'architettura hardware impiega una telecamera industriale Teledyne DALSA appartenente alla famiglia *Spyder3 GigE Vision*, in particolare il modello SG-11-01k40-00-R [17]. A differenza delle tradizionali telecamere *area-scan*, che acquisiscono un fotogramma in un singolo istante, sottoforma di matrice bidimensionale, il dispositivo adotta una tecnologia *line-scan*, cioè a scansione di una singola linea [17].

L'impiego di un sensore industriale *line-scan*, in sostituzione di una tradizionale telecamera *area-scan*, si rende strettamente necessario a causa delle stringenti limitazioni fisiche imposte dall'elevata frequenza di campionamento. In un tipico dispositivo commerciale operante a 60 fotogrammi al secondo (fps), il sensore ha a disposizione circa 16 ms per accumulare fotoni e generare l'immagine. Al contrario, operando a una frequenza di migliaia di linee al secondo (es. 5000 linee/s), il tempo massimo di esposizione concesso per ogni singola acquisizione scende a soli 200 µs. A tali velocità, un sensore standard produrrebbe un segnale fortemente sottoesposto e affetto da rumore, rendendo impossibile la distinzione degli atleti sulla linea di arrivo.

Per ovviare a queste limitazioni fisiche, il sensore CCD impiegato è un dispositivo elettronico all'avanguardia, costituito da una matrice unidimensionale di 1024 pixel continui, caratterizzati da una dimensione fisica quadrata di $14\mu m \times 14\mu m$ e un *fill factor* del 100% [17].

A differenza dei sensori per smartphone, i cui pixel misurano tipicamente pochi micrometri quadrati, quelli sul modello sopracitato presentano un'area nettamente superiore. La superficie maggiorata opera da buca di potenziale più capiente. L'ammontare massimo di carica che un singolo pixel può tenere prima di saturarsi è superiore, così si avrà una *full well capacity* maggiore, permettendo di raccogliere una quantità significativamente superiore di fotoni nel brevissimo tempo di esposizione disponibile.

Il *fill factor* massimizzato garantisce che l'intera superficie della linea sensibile sia esposta alla luce, azzerando gli spazi vuoti tra i pixel adiacenti normalmente occupati dalla circuiteria di controllo. Il parametro assicura un'elevata sensibilità fotometrica all'interno del campo visivo, acquisendo il numero massimo di fotoni incidenti.

Il dispositivo è inoltre dotato di una tecnologia *dual line scan*, definita ad alta sensibilità (*High Sensitivity Mode*), che consente alla telecamera di operare in una specifica modalità sfruttando due linee di sensori parallele contemporaneamente per raddoppiare la luce campionata [17]. La medesima porzione spaziale della linea

del traguardo viene acquisita da entrambe le righe del sensore, le cariche elettriche accumulate vengono poi sommate analogicamente. Quindi, nel medesimo intervallo temporale di esposizione, usando due acquisizioni della stessa porzione, la sensibilità del sistema è fisicamente raddoppiata. Il risultato è un incremento netto del rapporto segnale/rumore (SNR), che garantisce immagini nitide, luminose e temporalmente coerenti anche in condizioni di illuminazione ambientale non ottimali.

Sebbene si tratti di un dispositivo non di ultima generazione, esso rappresenta uno strumento ancora altamente performante, con un'unica criticità rilevante ai fini del progetto: l'acquisizione monocromatica. L'assenza dei canali colore (RGB) priva il modello di *machine learning* di dettagli fondamentali per la discriminazione dei pixel. Di conseguenza, il contrasto tra gli atleti, la pista e gli eventuali artefatti visivi risultano attenuati, rendendo il task di segmentazione e di individuazione del torso più complesso, con il rischio di degradare la precisione complessiva del sistema in fase di inferenza.

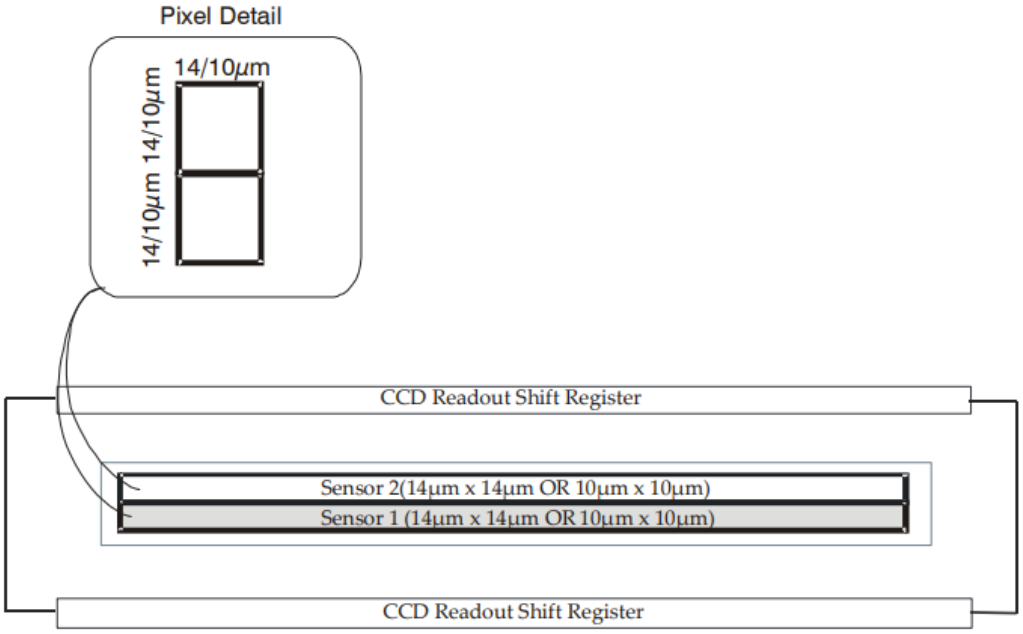


Figura 2.2: High Sensitivity Mode: la camera acquisisce la stessa immagine due volte con le due linee separate, portando ad un risultato più luminoso. Fonte: [17]

Dal punto di vista prestazionale, le telecamere lineari non vengono valutate tramite il classico indice degli fps, bensì mediante la frequenza di riga (*line rate*). A livello hardware, la serie Spyder3 è progettata per raggiungere velocità operative

estreme, arrivando fino a 36 kHz (ovvero 36.000 linee acquisite per ogni secondo) [17]. Nel progetto sviluppato, per garantire un compromesso ottimale tra l'altissima risoluzione temporale richiesta dal passaggio degli atleti e l'esigenza di incamerare sufficiente luce ambientale, la frequenza di acquisizione del sensore (parametro: *AcquisitionLineRateAbs* [18]) è stata vincolata via software all'impostazione di 5000 Hz per giornate molto soleggiate, altrimenti a 1000 Hz per ottenere immagini più luminose.

Per la sincronizzazione delle acquisizioni, il dispositivo è stato mantenuto nella modalità operativa predefinita di fabbrica, denominata Mode 7 (*Internally Programmable Line Rate, Maximum Exposure Time*) [17]. In questa configurazione, la frequenza operativa (*line rate*) non varia dinamicamente, ma è scandita in modo rigoroso e costante da un timer interno alla telecamera, programmato in fase di inizializzazione tramite il protocollo *GenICam* (es. 5000 Hz). Contemporaneamente, il firmware interno del dispositivo calcola e massimizza automaticamente il tempo di esposizione (*Exposure Time*) per ogni singola linea, massimizzandolo per adattarsi al periodo massimo concesso da quella specifica frequenza fissa [17].

Imager Format	Dual line scan
Resolution	1024 pixel $\times$ 1 pixel
Pixel Fill Factor	100%
Pixel Size	$14\mu m \times 14\mu m$
Sensitivity Mode	High, Low, Tall pixel
Gain Range	-10dB to $+10$dB
Back Focal Distance	$6.56 \pm 0.25mm$
Camera Size	72 (h) $\times$ 60 (l) $\times$ 50 (w)mm
Mass	$< 300g$
Power Connector	6 pin male Hirose
GigE connector	RJ45
Input Voltage	$+12$V to $+15$V
Power Dissipation	$< 9W$

Minimum Line Rate	$300Hz$
Maximum Line Rate	$36kHz$

Tabella 2.1: Specifiche tecniche della telecamera Teledyne Dalsa Spyder3 SG-11-01k40-00-R [17].

Infine, per quanto concerne l'interfaccia ottica, il corpo macchina è stato equipaggiato con un innesto standard M42x1, avente una distanza focale di 6,56 mm [17]. Questa interfaccia meccanica ha permesso l'accoppiamento con il gruppo ottico (obiettivo) preposto alla messa a fuoco della linea del traguardo [17].

Figura 2.3: Modello della telecamera Teledyne Dalsa Spyder3 SG-11-01k40-00-R [17].

L'obiettivo industriale, a focale fissa, prodotto da Edmund Optics (Serie C, Part Number 58-000), risponde in modo mirato alle due maggiori criticità, fotometriche e geometriche, imposte dall'applicazione del photofinish.

Dal punto di vista fotometrico, operando a una frequenza alta, il tempo di esposizione del sensore risulta limitato. Per sopperire a questa restrizione temporale, l'impiego di un obiettivo dotato di una elevata apertura di diaframma, pari a f/1.3, è stato necessario: tale caratteristica ottica garantisce la massima luminosità, massimizzando la quantità di fotoni convogliata sul sensore CCD. Operando insieme

alla tecnologia *Dual Line Scan* della telecamera, si ottengono immagini ad alto contrasto anche in condizioni di illuminazione ambientale non controllata.

Geometricamente, la lunghezza focale ridotta di 8,5 mm conferisce alla lente un ampio angolo di campo (*Field of View*). Questa scelta si rivela strategica in fase di installazione fisica a bordo pista. Permette di inquadrare l'intera larghezza della linea del traguardo, coprendo tutte le corsie, pur mantenendo la telecamera a una distanza operativa ravvicinata.

2.1.3 Il sistema di trigger

Il rilevamento dell'evento di inizio della memorizzazione dei *frame*, comandato dai cronometristi tramite la pressione di un pulsante, rappresenta una criticità per l'intero sistema di photofinish.

Invece di delegare la lettura di un segnale elettrico grezzo direttamente all'elaboratore centrale, si è optato per l'impiego di una scheda elettronica Arduino Uno r3, basata sul microcontrollore ATmega328P [19]. Questa scelta architetturale è motivata dalla necessità di alleggerire i compiti del computer principale, rendendo il sistema complessivamente più reattivo.

Il controllo dello stato del pulsante da parte di un'entità esterna, senza le latenze imprevedibili introdotte dai processi di un sistema operativo *general-purpose* come Windows, garantisce un tempo di reazione alla pressione nell'ordine dei microsecondi.

Delegare il *polling* continuo del pulsante a questa scheda permette di liberare il processore del PC, lasciando interamente libere le risorse computazionali a disposizione per l'elaborazione ad altissima frequenza del flusso di immagini provenienti dalla telecamera.

Infine, Arduino offre una piattaforma industriale standardizzata, estremamente affidabile e a costi contenuti, mette a disposizione dei pin GPIO facilmente accessibili e integra nativamente un convertitore hardware USB-Seriale. Questo ha permesso di abbattere la complessità circuitale, fornendo un canale di comunicazione immediato con l'applicativo Python grazie anche alla prontezza garantita dal clock interno a 16 MHz [19].

Microcontroller	ATmega328P
USB connector	USB-B
Digital I/O pins	14
Analog input pins	6
PWM pins	6
Communication	UART
Operating Voltage I/O	5V
Input Voltage (limit)	+6V to + 20V
DC Current I/O pins	$20mA$
Clock speed	$16MHz$
SRAM	$2kB$
FLASH	$32kB$
Dimensions	53.4 (w) × 68.6 (l)mm
Mass	$25g$

Tabella 2.2: Specifiche tecniche della scheda Arduino Uno r3 [19].

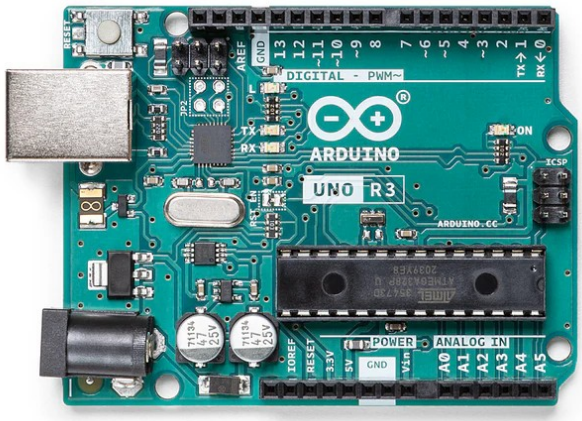


Figura 2.4: Modello della scheda Arduino Uno r3 [19]. Fonte: [16].

2.1.4 L'unità di elaborazione (PC)

La gestione dell'intero sistema di immagazzinamento del flusso dati ad alta frequenza e di analisi algoritmica è affidata all'elaboratore centrale (PC). A causa dell'elevata mole di informazioni generate dalla telecamera e delle stringenti richieste computazionali dei modelli di Intelligenza Artificiale, l'hardware utilizzato deve garantire prestazioni di rilievo per evitare colli di bottiglia durante l'acquisizione. Il calcolatore impiegato nel presente progetto è il modello ASUS ROG STRIX G16, le cui specifiche hardware sono state accuratamente valutate per far fronte a tali criticità.

La base su cui si erige tutto il prototipo, senza la quale la telecamera non funzionerebbe, è costituita dall'interfaccia di rete dedicata. Il sistema è equipaggiato con una scheda *Gigabit Ethernet* (Realtek PCIe GbE Family Controller) [20], capace di arrivare ad 1 Gbit/s, minimo requisito di velocità per il funzionamento della camera. Dall'analisi della scheda tecnica, si evidenzia che un'infrastruttura di rete robusta e interamente dedicata alla comunicazione con la telecamera rappresenta il requisito fondamentale per l'affidabilità dello standard *GVSP*, permettendo di sostenere il massiccio *data rate* unidirezionale in ingresso e di prevenire la perdita dei pacchetti *UDP* (*packet loss*).

L'esecuzione dell'applicativo Python è invece affidata a un processore 13th Gen Intel Core i9-13980HX, affiancato da 16 GB di memoria RAM [20]. L'utilizzo di una CPU multicore si rivela essenziale per la corretta gestione dei task: mentre il processo principale di Python si occupa del salvataggio e della visualizzazione delle immagini tramite la libreria OpenCV, i processi in *background*, gestiti dall'interfaccia *harvesters* [21], lavorano in concorrenza per prelevare ininterrottamente i nuovi *frame* dalla telecamera, allocandoli temporaneamente nei *buffer* della RAM.

Parallelamente, per l'esecuzione ottimizzata delle pesanti routine di elaborazione delle immagini e per l'addestramento dei modelli di Intelligenza Artificiale, il sistema si avvale della potenza di calcolo di una GPU dedicata NVIDIA GeForce RTX 4070 con 8 GB di VRAM [20]. Lo sfruttamento delle architetture di calcolo parallelo delle schede grafiche consente infatti di svolgere in simultanea più operazioni, abbattendo drasticamente i tempi di latenza nell'analisi dei dati e nel *training* delle reti neurali.

2.2 Protocolli e comunicazione

Per gestire il cospicuo flusso di informazioni tra l'elaboratore centrale, la telecamera e il microcontrollore sono necessari protocolli strutturati in modo tale da garantire una comunicazione efficiente senza saturare le risorse a disposizione.

2.2.1 Il protocollo GigE Vision

I sistemi di acquisizione video convenzionali operano tipicamente con frequenze di campionamento comprese tra i 30 e i 60 fps, spingendosi fino a circa 240 fps in applicazioni specifiche. Tali velocità risultano, tuttavia, inadeguate se rapportate alle esigenze delle telecamere industriali con architettura *line-scan*, poiché possiedono frequenze operative superiori di diversi ordini di grandezza, spaziando dalle migliaia fino ad arrivare alle 40.000 acquisizioni lineari al secondo per i sensori attualmente in commercio.

A fronte di un simile volume di dati, le interfacce e i protocolli di comunicazione tradizionali non sono in grado di garantire un flusso continuo. Il mezzo trasmissivo diventa pertanto il principale collo di bottiglia del sistema, portando inevitabilmente a un rallentamento generale o, nel caso peggiore, alla perdita irreversibile di dati.

Per superare questo limite strutturale, l'architettura del sistema si affida allo standard industriale *GigE Vision* [22]. Tale standard è progettato per interfacciarsi con l'hardware *Ethernet* tradizionale, permettendo l'utilizzo di cavi di rete a basso costo per sostenere trasferimenti di dati nell'ordine dei Gigabit per secondo [22]. Il punto di forza del protocollo risiede nella separazione logica della comunicazione in due canali distinti: il controllo del dispositivo è affidato al protocollo *GVCP*, mentre la trasmissione ad alta velocità delle immagini è gestita dal protocollo *GVSP* [22].

Il protocollo *GVSP*, nucleo delle prestazioni del sistema, opera al di sopra del protocollo di trasporto *UDP* [22]. La scelta dell'*UDP* rispetto al comune TCP è cruciale: vengono eliminati i meccanismi di validazione e le conferme di ricezione dei pacchetti di rete (*handshake continuo*), minimizzando drasticamente l'*overhead* di rete. In questo modo, la telecamera può generare un flusso continuo e unidirezionale di informazioni grafiche verso il computer, riducendo al minimo lo sforzo computazionale richiesto al processore principale per gestire i pacchetti in ingresso ed evitando l'interruzione del flusso nel caso di criticità.

Nello specifico, lo standard *GigE Vision* separa nettamente la fase di configurazione da quella di streaming dei dati. Il protocollo *GVCP* funziona da canale di controllo asincrono [23]. Tuttavia, l'applicativo software non comunica mai direttamente tramite *GVCP*, ma si avvale del livello di astrazione superiore fornito dallo standard *GenICam* [24]. In questo modo, il processo Python può impostare parametri fondamentali, come la frequenza operativa (*AcquisitionLineRateAbs*), utilizzando un'interfaccia ad alto livello. Sarà poi il *layer* sottostante a tradurre queste istruzioni e a trasmetterle fisicamente sul cavo *Ethernet* sotto forma di pacchetti *GVCP* [23].

Il protocollo *GVSP*, al contrario, è un canale unidirezionale dedicato in modo esclusivo al trasferimento del *payload*, ovvero i dati grezzi dei pixel [23]. Sfruttando la natura *connectionless* dell'*UDP*, il *GVSP* sacrifica la garanzia di consegna totale per massimizzare il *throughput* e azzerare l'*overhead* [23]. Per un sistema di cronometraggio, come nel caso del photofinish, questa è una scelta architetturale obbligata: eventuali ritrasmissioni di pacchetti (tipiche del protocollo TCP) introdurrebbero ritardi non deterministici, bloccando la pipeline di acquisizione in un momento critico della gara. Risulta pertanto accettabile la remota perdita di un pacchetto (corrispondente a frazioni di millisecondo), piuttosto che l'arresto dell'intero flusso temporale in attesa del suo recupero [23].

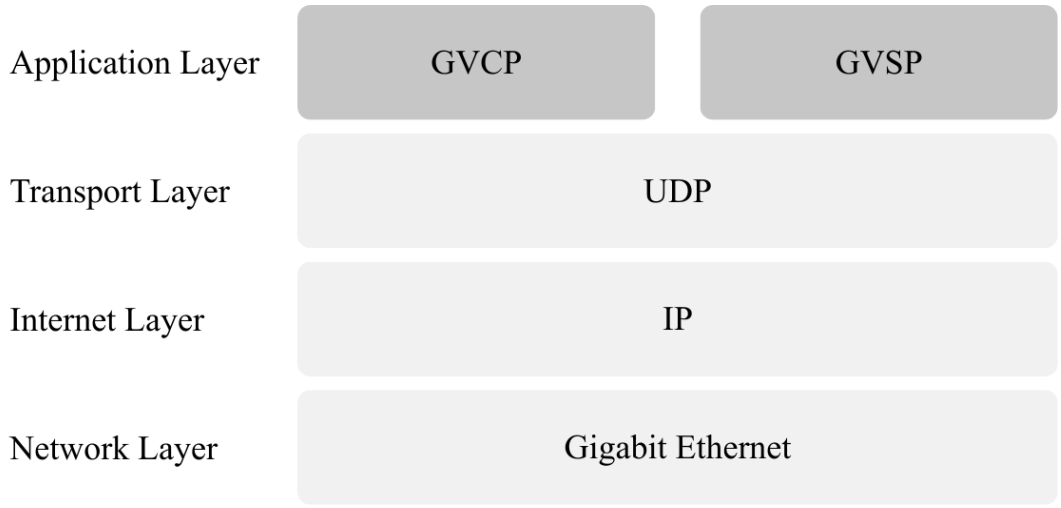


Figura 2.5: Architettura della pila protocollare *GigE Vision*. I protocolli applicativi dedicati al controllo (*GVCP*) e allo streaming video (*GVSP*) si appoggiano sugli standard di rete commerciali *UDP* e IP.

2.2.2 Lo standard GenICam e la libreria Harvesters

La gestione del dispositivo di acquisizione industriale, a livello software, è demandata allo standard *GenICam*, gestito dalla European Machine Vision Association (EMVA) [24]. L'obiettivo primario è quello di fornire un'interfaccia di programmazione (API) universale, slegando il codice applicativo dalle differenti specifiche hardware e dai protocolli proprietari dei singoli produttori [24].

L'architettura software del progetto implementa il modulo *GenTL* dello standard, strutturato secondo il paradigma *Producer-Consumer* [24]. Nel sistema sviluppato, il ruolo di *Consumer* è ricoperto dalla libreria Python *harvesters* [21], che agisce come interfaccia di alto livello per la gestione della telecamera e l'estrazione dei *frame*. Il *Producer*, responsabile della reale traduzione delle istruzioni in segnali di rete *GigE Vision* da inviare alla scheda di rete, è costituito dalla libreria dinamica (CTI) fornita dal Baumer GAPI SDK (bgapi2_gige.cti) [25].

A dimostrazione dell'elevata interoperabilità garantita dallo standard *GenICam*, il sistema in oggetto utilizza il *Producer* sviluppato da Baumer [25] per interfacciarsi e pilotare l'hardware di acquisizione prodotto da Teledyne DALSA. Tale flessibilità svincola di fatto l'infrastruttura dall'obbligo di utilizzare il programma proprietario originale del costruttore. Ciò consente lo sviluppo di applicativi software customizzati e indipendenti, capaci di soddisfare le numerose esigenze degli utilizzatori di queste tecnologie.

A livello operativo, la comunicazione con la telecamera avviene tramite l'interrogazione di un file XML standardizzato, definito *node map*, memorizzato all'interno del dispositivo stesso [24]. Tale mappa espone tutti i registri hardware della telecamera, sotto forma di nodi logici accessibili gerarchicamente [24]. Per garantire la totale interoperabilità semantica tra marchi diversi, la nomenclatura di questi nodi è rigorosamente dettata dal modulo *SFNC* (Standard Features Naming Convention) dello standard *GenICam* [24]. Grazie a questo dizionario universale, è stato possibile configurare dinamicamente i parametri operativi critici utilizzando identificatori testuali standardizzati, indipendenti dal produttore. Ne sono un esempio l'impostazione dell'altezza del *frame* (*Height*), la definizione della frequenza operativa in linea per secondo (*AcquisitionLineRateAbs*) e l'interrogazione del contatore temporale di rete (*GevTimestampTickFrequency*). Questa architettura ha permesso di evitare l'impiego di istruzioni binarie a basso livello, demandando al *Producer GenTL* l'onere di tradurre le richieste del *Consumer* nei corrispondenti pacchetti *GVCP* instradati sulla rete.

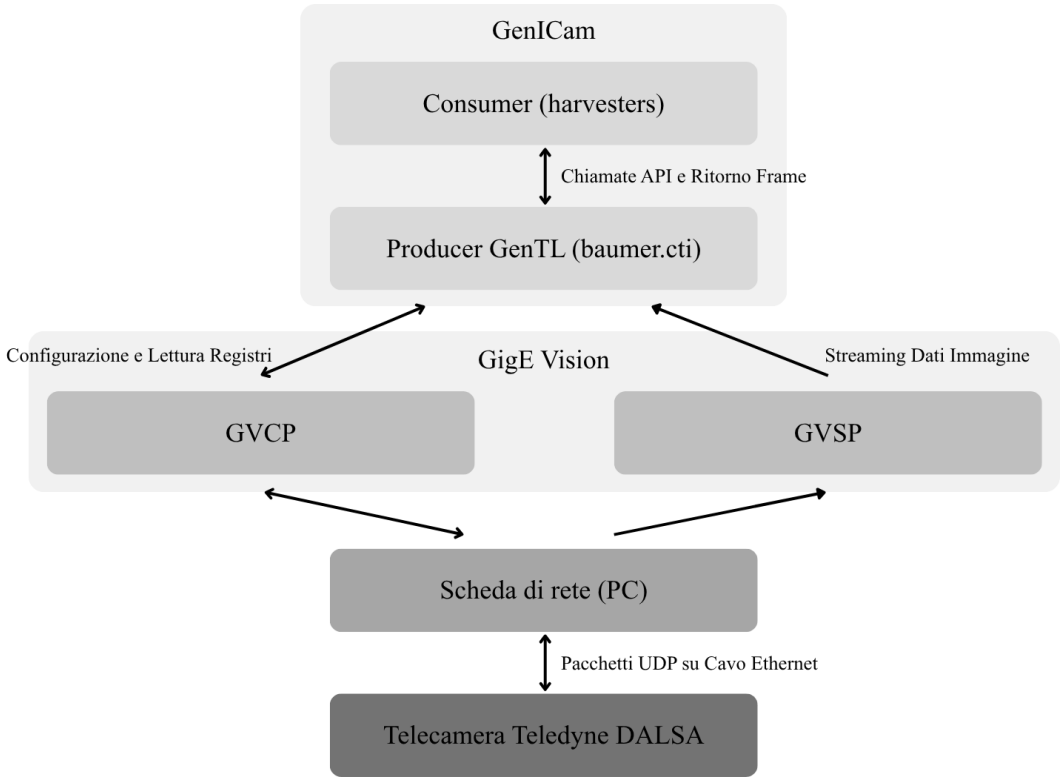


Figura 2.6: Architettura a strati del flusso dati per la gestione della telecamera. Il livello applicativo interagisce con il dispositivo tramite lo standard *GenICam*, demandando al *Producer GenTL* (Baumer GAPI) il compito di astrarre i protocolli di rete *GigE Vision*. La comunicazione è separata in un canale bidirezionale di controllo (*GVCP*) e un canale unidirezionale ad alte prestazioni (*GVSP*) per lo streaming del *payload* basato su *UDP*.

2.2.3 Comunicazione seriale Arduino-PC

Il sistema di acquisizione richiede una sincronizzazione precisa dei tempi tra l'avvio della gara, coincidente con la pressione del pulsante di start (pulsante INVIO dell'elaboratore), e l'inizio della memorizzazione software dei *frame* (pulsante collegato ad Arduino).

Per assolvere a questo compito e scaricare il computer da cicli di controllo hardware, l'architettura sfrutta il microcontrollore ATmega328P, presente sulla scheda Arduino

Uno r3, delegandogli la lettura del segnale di *trigger*.

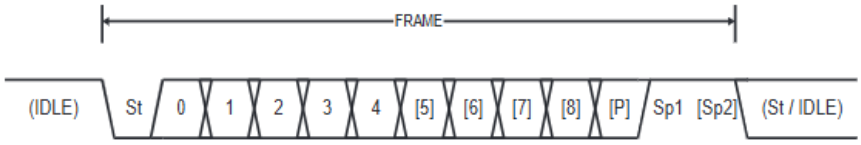
A livello fisico, la comunicazione bidirezionale tra il microcontrollore e il personal computer avviene tramite un'interfaccia seriale asincrona standard (*UART*) instradata tramite connessione USB. Il microcontrollore è dotato di un modulo hardware dedicato, denominato *USART0* (*Universal Synchronous and Asynchronous Receiver and Transmitter*) [26]. Nonostante il modulo supporti trasmissioni sincrone, per questo progetto è stato configurato in modalità puramente asincrona [26]. Essendo le informazioni da scambiare limitate a brevi comandi di controllo e non richiedendo larghezze di banda elevate, ci si è avvalsi della libreria *serial* di Python e della classe *Serial* nativa di Arduino. Entrambe gestiscono nativamente la comunicazione asincrona, eliminando così la necessità di una linea di clock condivisa.

La coerenza dei dati trasmessi è garantita dal *baud rate*, concordato via software a 9600 bit per secondo su entrambi gli endpoint. A questa velocità, l'hardware dell'ATmega328P gestisce automaticamente il data framing, consentendo al ricevitore di individuare correttamente l'inizio e la fine del pacchetto senza un segnale temporale esterno [26].

In fase di inizializzazione (Serial.begin()), il sistema adotta la configurazione predefinita nota come 8N1. Di conseguenza, l'hardware incapsula in automatico ogni carattere utile tra 1 bit di start, 8 bit di dati (inviati dal bit meno significativo al più significativo) e 1 bit di stop a chiusura, mantenendo disabilitato il controllo di parità [26].

A livello applicativo, l'invio dei messaggi è stato implementato tramite l'istruzione Serial.println(), la quale provvede ad appendere in automatico i caratteri di terminazione (Carriage Return e Line Feed), fornendo a Python un marcatore logico essenziale per separare i pacchetti in ingresso.

Figure 19-4. Frame Formats



St	Start bit, always low.
(n)	Data bits (0 to 8).
P	Parity bit. Can be odd or even.
Sp	Stop bit, always high.
IDLE	No transfers on the communication line (RxDn or TxDn). An IDLE line must be high.

Figura 2.7: Lo schema riassume il processo di invio dei dati tramite la configurazione predefinita 8N1, con la spiegazione dei singoli bit. Fonte: [26].

Dal punto di vista logico e applicativo, il protocollo implementato scambia semplici stringhe di testo ASCII, frammentate e trasmesse un carattere alla volta. Il processore interroga il pin digitale associato al pulsante in un loop continuo di *polling*. Per filtrare i disturbi elettromeccanici, ovvero i transitori generati fisicamente dalla chiusura del contatto, il software implementa un algoritmo di *debouncing* temporale: la lettura viene considerata valida esclusivamente se il segnale si mantiene stabile per un intervallo minimo di 50 ms.

A seguito di un riscontro positivo per lo stato logico alto, sul pin di input, il microcontrollore trasmette sulla seriale la stringa testuale "*START_REC*", mentre per lo stato basso trasmette "*STOP_REC*". Lato computer, l'applicativo Python intercetta questi messaggi ascoltando la porta seriale *COM3*. È fondamentale sottolineare che questa lettura avviene in modalità non bloccante (timeout = 0): tale architettura permette al processo principale di ispezionare istantaneamente il canale seriale senza interrompere o introdurre latenze nel ciclo critico di cattura ad altissima frequenza (impostato per raggiungere fino a 5000 linee/secondo) gestito dalla telecamera industriale.

Infine, grazie alle capacità *full-duplex* del modulo *USART0*, il computer può imporre il ripristino della macchina a stati inviando la stringa "*RESET*" verso Arduino, forzando un ricalcolo immediato dello stato del sensore per garantire il perfetto allineamento del sistema prima di avviare una nuova misurazione.

Capitolo 3

Preparazione e pre-processing del dataset

Le prestazioni e la capacità di generalizzazione di un modello di *Deep Learning* sono strettamente vincolate alla qualità e alla rappresentatività dei dati utilizzati durante la fase di addestramento. In questo capitolo viene presentata l'intera pipeline di etichettatura del dato, progettata per la costruzione del *dataset* di addestramento.

Le acquisizioni di photofinish presentano sfide uniche e atipiche nel panorama della *Computer Vision* tradizionale: risoluzioni orizzontali sproporzionate rispetto alla dimensione dell'altezza, notevoli distorsioni anatomiche, variazioni imprevedibili di illuminazione e occlusioni estreme dovute alla concomitanza degli atleti sulla linea di traguardo, fanno sì che non esistano online *dataset* già pronti. Affrontare questo compito di creazione del *dataset*, ha richiesto lo sviluppo di strumenti software personalizzati e flussi di lavoro specifici per indicare alla rete i dati corretti da imparare.

Il capitolo è strutturato in modo sequenziale, seguendo il naturale flusso di vita del dato. Si parte con la descrizione della fase di ricerca e raccolta delle immagini originali, illustrando il passaggio critico dai primi test esplorativi su dati internazionali ottenuti dal sito *World Athletics* [27], fino a giungere all'adozione strategica del *dataset* definitivo fornito dai cronometristi FICr.

Successivamente viene dettagliato il processo di estrazione automatizzato del *Ground Truth* (valore vero, corretto), cioè la linea di arrivo inserita dai giudici, rispetto alle coordinate assolute dell'immagine, basato su tecniche di *Computer Vision* classica e algoritmi di sottrazione matriciale per isolare queste linee di arrivo.

A questo punto entra in gioco il cuore del *pre-processing*, proponendo un'innovativa tecnica di "*Tiling*", in un sistema basato sull'architettura *YOLO*, che permette di evitare la perdita di dettagli importanti durante il ridimensionamento dell'immagine in input alla rete, riuscendo così ad estrarre dinamicamente i singoli atleti e normalizzando le coordinate bersaglio (posizione della linea di arrivo).

Infine, vengono illustrate le strategie per prevenire il fenomeno dell'*overfitting* e garantire la solidità scientifica del progetto, rispettivamente attraverso robuste tecniche di *Data Augmentation*, mirate a simulare le reali condizioni di gara, e una rigorosa suddivisione matematica del *dataset* in sottoinsiemi di *Training*, *Validation* e *Test*.

L'obiettivo finale di questa complessa fase di pre-elaborazione è la creazione di un *dataset* altamente ottimizzato e formattato, prerequisito fondamentale per il successivo addestramento dell'architettura neurale proposta nel Capitolo 4.

3.1 Raccolta del dataset ed estrazione delle linee di arrivo

In una fase preliminare, è stato raccolto un piccolo *dataset* di prova, prelevato dal portale ufficiale *World Athletics* [27], attingendo da un archivio di circa 3.000 immagini di photofinish totali. Tali acquisizioni provenivano da competizioni internazionali di altissimo livello, quali Campionati Mondiali outdoor, indoor e di staffette svolti nel corso degli anni.

Durante le verifiche di idoneità, antecedenti al download massivo del *dataset*, è emerso tuttavia un limite strutturale che lo ha reso inutilizzabile. Per poter addestrare la rete neurale, era necessario estrarre la posizione orizzontale (asse x) delle linee di arrivo e salvarle come *Ground Truth*, ma queste immagini presentavano nativamente la linea del traguardo già sovrimpressa. Il tentativo iniziale di rimuovere questo elemento tramite algoritmi di *inpainting* si è rivelato inefficace. Le tecniche utilizzate introducevano inevitabili artefatti visivi proprio nell'area di maggiore interesse, creando sfocature in corrispondenza del torso degli atleti, nel tentativo vano di ricostruire i pixel esatti mancanti. L'obiettivo era generare un *dataset* "pulito" da fornire in input a una rete neurale composta da due stadi: ricerca dei *bounding box* degli atleti e successiva stima della posizione del torso.

L'immagine del photofinish è una composizione spazio-temporale caratterizzata da repentine variazioni di texture in corrispondenza del passaggio degli atleti. Per tale ragione, gli algoritmi di *inpainting* non sono in grado di stimare correttamente i valori dei pixel mancanti basandosi sulle colonne adiacenti poiché queste presentano una scarsa correlazione statistica. Inoltre, il processo di ricostruzione tende a generare un effetto di sfocatura, eliminando proprio i bordi netti e i dettagli ad alta frequenza che risultano fondamentali per la precisione del modello di rilevamento.

L'impiego di questi dati avrebbe compromesso notevolmente l'addestramento della rete neurale, introducendo un *bias* significativo. Il modello avrebbe infatti imparato ad associare la presenza degli artefatti visivi, quali linee sfocate, alla posizione della linea del traguardo, risultando inutilizzabile in un reale scenario di acquisizione sul campo, in cui le immagini grezze ne sono intrinsecamente prive. Pertanto, questo *dataset* è stato scartato.

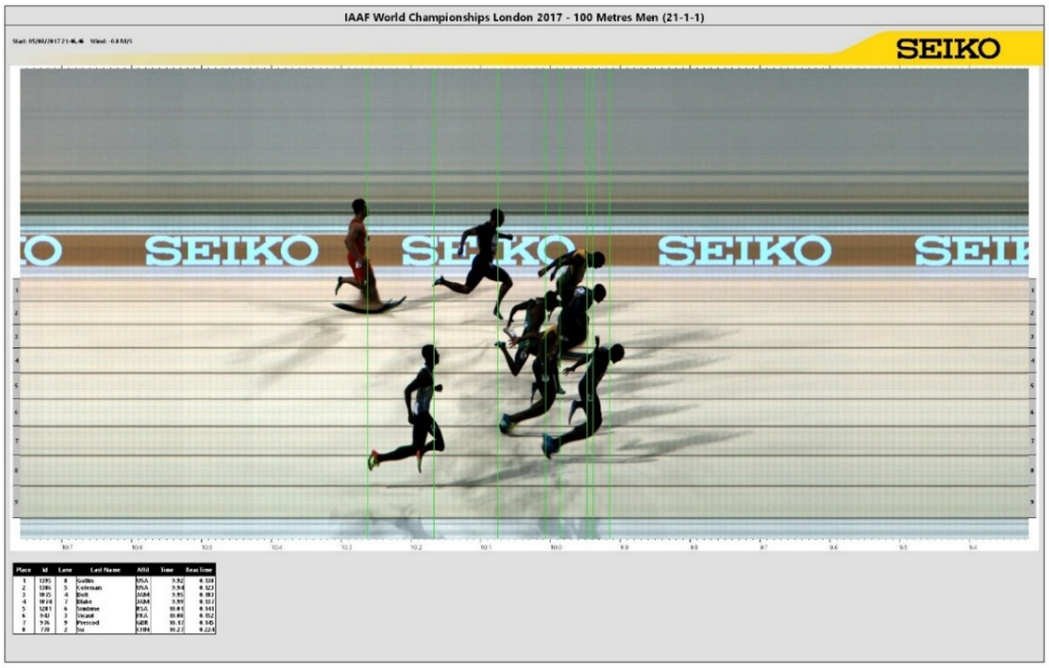
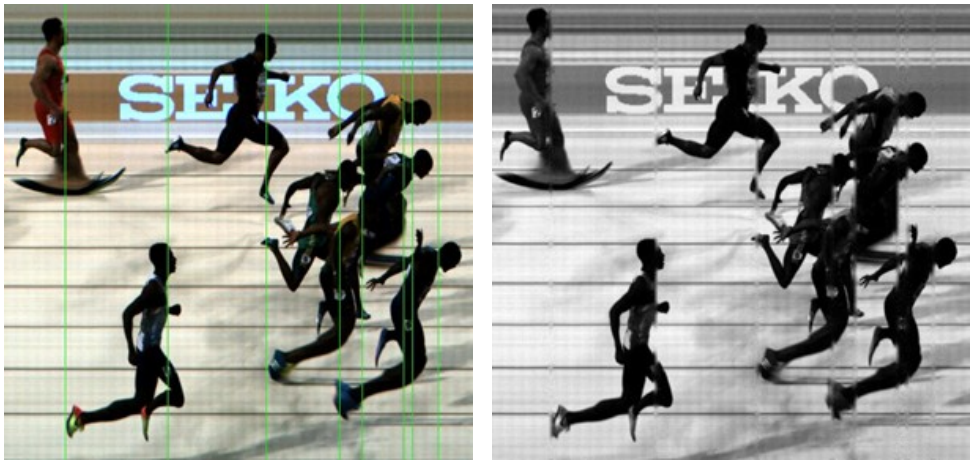


Figura 3.1: Photofinish ufficiale di una gara sui 100m ai Campionati Mondiali di Londra del 2017. Le linee verdi impresse nell'immagine, inserite dai giudici di gara, indicano il tempo di arrivo degli atleti. Fonte: [27].



(a) **(b)**

Figura 3.2: Confronto tra l'immagine originale (a) e quella elaborata tramite algoritmi di *inpainting* (b). In prossimità dei torsi dei corridori è presente una linea verticale sfocata. Fonte: [27].

Per superare questa criticità, è stata avviata una collaborazione diretta con i cronometristi di gara FICr. La suddetta cooperazione ha consentito l'accesso a dati grezzi originali generati, tramite il software ufficiale FinishLynx [28], durante diverse competizioni locali. Dall'archivio sono state selezionate circa 600 acquisizioni. Per ogni gara sono state estratte le stesse foto in due varianti, accoppiate: la versione "pulita", priva di riferimenti grafici, da sottoporre alla rete, e la rispettiva versione con la linea di arrivo posizionata manualmente dai giudici, fondamentale per la successiva fase di annotazione.

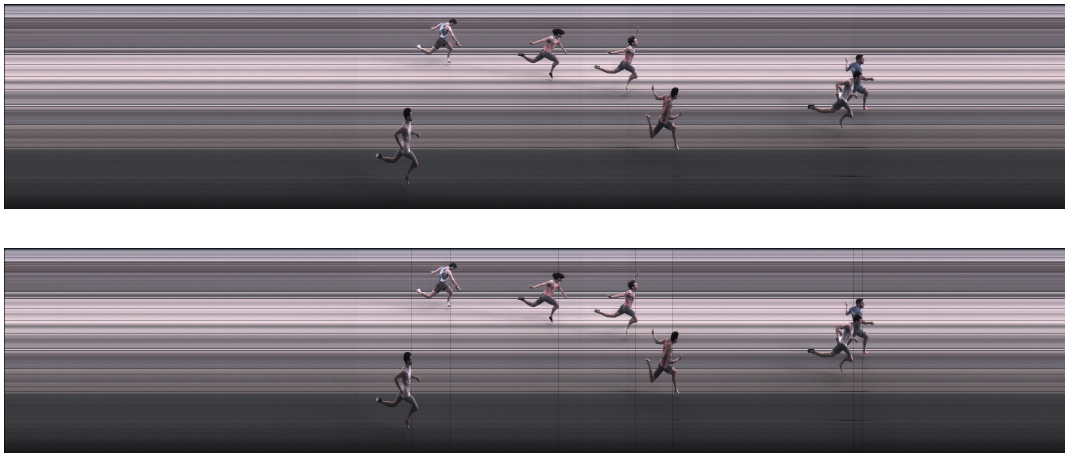


Figura 3.3: L'immagine superiore è la versione grezza acquisita dalla telecamera (versione "pulita"); l'immagine sottostante è la versione etichettata per determinare il risultato finale.

Le immagini fornite dai cronometristi FICr presentano caratteristiche fisiche eterogenee, strettamente dipendenti dal contesto di gara. Dal punto di vista dimensionale, mantengono il formato tipico delle telecamere *line-scan*, risultando intrinsecamente molto sviluppate in larghezza. Tuttavia, la risoluzione orizzontale risulta estremamente variabile a seconda del tipo di competizione e della durata del passaggio degli atleti.

A differenza delle immagini dei circuiti internazionali, acquisite in condizioni ambientali fortemente controllate, questo *dataset* presenta un'ampia variabilità di illuminazione. In diverse acquisizioni si riscontrano condizioni di luce non ottimali, spesso inferiori agli elevati standard richiesti per i sensori ad altissima frequenza che necessitano di illuminamenti molto intensi, nell'ordine delle migliaia di lumen a livello del torso, per venire incontro alla valutazione manuale da parte dei giudici. Tale variabilità si traduce in acquisizioni con basso contrasto, sottoesposizione o con sagome degli atleti scarsamente visibili.

Paradossalmente, una qualità fotografica mediamente inferiore rappresenta un importante valore aggiunto per la ricerca: addestrare la rete neurale su immagini realistiche e affette da disturbi ambientali, favorisce la generalizzazione e aumenta significativamente la robustezza del modello, preparandolo ad affrontare scenari operativi complessi e non idealizzati.

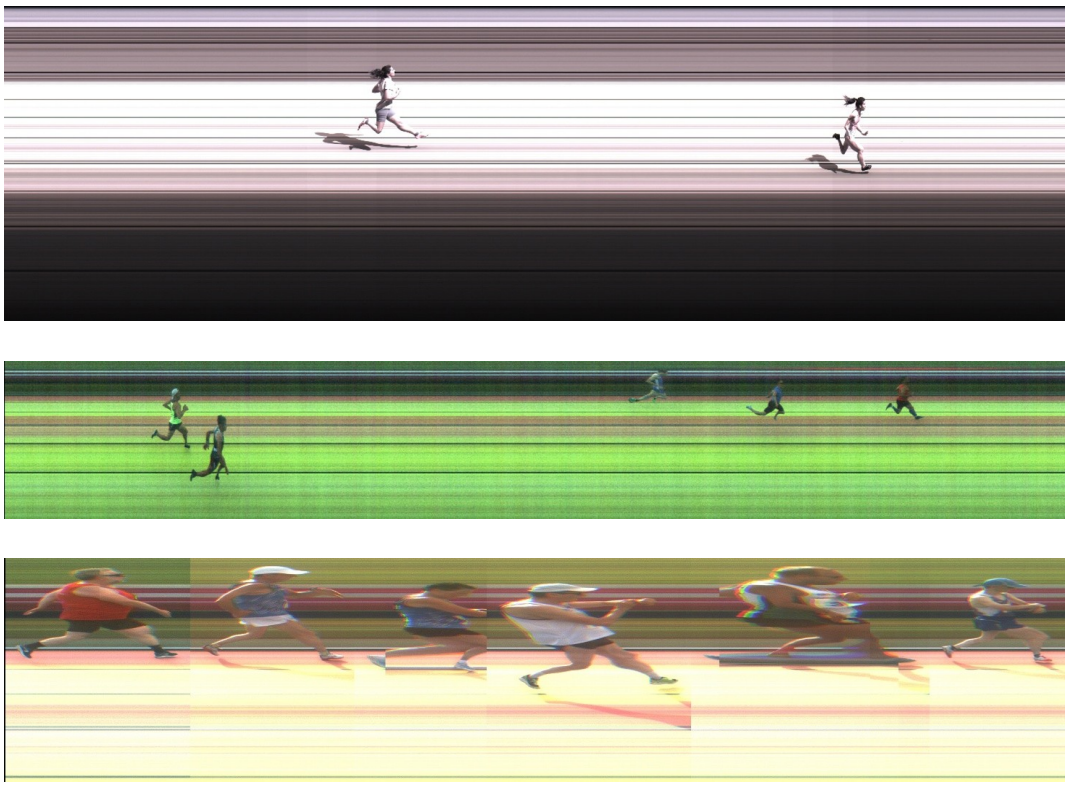


Figura 3.4: Esempi di acquisizioni di photofinish estratte dal *dataset*, rappresentative della variabilità delle condizioni non ideali. Dall'alto verso il basso: 1. acquisizione caratterizzata da forte sovraesposizione, in cui i dettagli cromatici si perdono e le sagome risultano in forte controluce; 2. condizione di bassa luminosità e scarso contrasto con un'evidente dominante cromatica alterata; 3. acquisizione affetta da evidenti artefatti digitali, aberrazioni cromatiche e distorsioni sulle sagome.

Sfruttando la disponibilità di queste coppie perfette, è stato sviluppato uno script Python *custom* per automatizzare e rendere estremamente robusta l'estrazione del *Ground Truth*.

Il flusso di elaborazione si è articolato nei seguenti passaggi:

1. l'algoritmo accede alla directory del *dataset* e appaia le immagini originali con le rispettive controparti etichettate (identificate dal suffisso -e), salvandole in due vettori;

2. il sistema effettua una sottrazione matriciale tra le due immagini RGB. Poiché i due fotogrammi risultano identici, a eccezione delle linee di traguardo, tale operazione consente di isolare i pixel di interesse con elevata precisione.

Occorre tuttavia notare che, trattandosi di immagini salvate in un formato di compressione *lossy* (con perdita di dati, es. JPEG), è possibile riscontrare lievi alterazioni o artefatti visivi nei pixel in prossimità delle linee, generati proprio dall'algoritmo di compressione;

3. a partire dall'immagine differenza, è stata sviluppata una sequenza algoritmica mirata all'estrazione precisa delle linee verticali.

   Inizialmente, è stato impiegato un algoritmo per l'individuazione dei contorni (*Edge Detection*).

   Successivamente, al fine di eliminare il rumore di fondo, si è rivelata cruciale l'applicazione di un filtro morfologico di apertura (*Morphological Open*). Tale filtro, configurato con un *kernel* strettamente verticale di dimensioni 1×40 pixel, è stato progettato per sopprimere gli artefatti isolati e preservare esclusivamente le strutture verticali continue.

   Per localizzare spazialmente le linee, si è proceduto a calcolare il numero di pixel rilevati (pixel bianchi) lungo ogni singola colonna dell'immagine. Fissando una soglia minima di attivazione (es. 300 pixel, pari a circa un terzo dell'altezza complessiva), è stato possibile classificare come candidate "linee di traguardo" solo le colonne che superavano tale densità.

   Infine, una logica di raggruppamento ha provveduto ad accorpare i segmenti adiacenti compresi entro una tolleranza prestabilita, calcolandone la posizione media per restituire un'unica coordinata spaziale x pulita e definitiva;

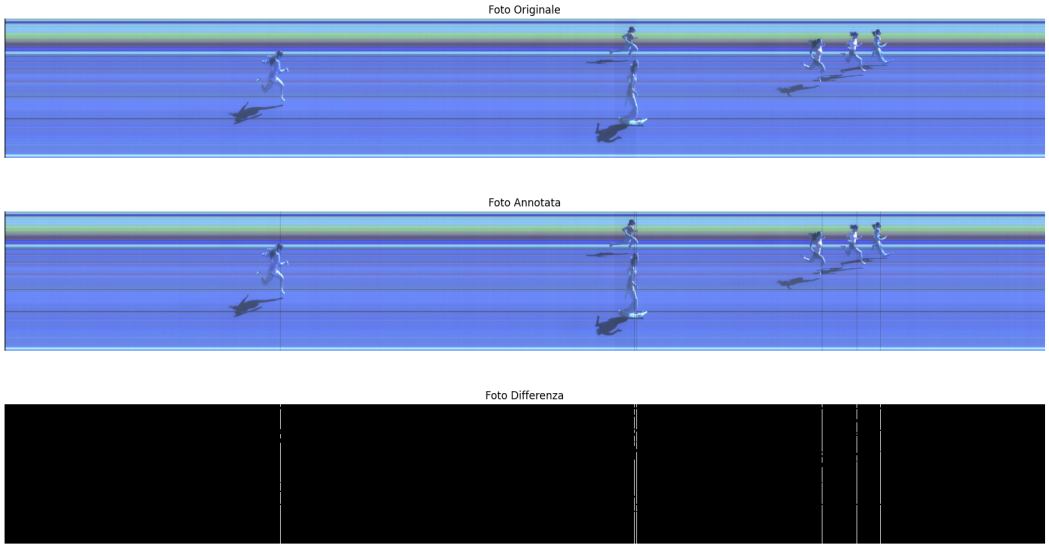


Figura 3.5: L'immagine superiore è la versione "pulita", quella centrale è la versione etichettata dai giudici. Nell'immagine situata più in basso, rappresentante la differenza tra le due immagini precedenti, è possibile notare la posizione delle linee di photofinish.

4. tramite la libreria Matplotlib, è stata generata un'interfaccia grafica (GUI) interattiva che sovrappone le linee rilevate all'immagine originale etichettata. Manualmente si può convalidare i risultati, correggendo eventuali micro-imprecisioni trascinando le linee, eliminandole o aggiungendone di nuove;

5. al termine della validazione, il sistema genera un identificativo univoco (UUID) per rinominare l'immagine e salva le coordinate x finali in un file CSV. Per garantire la sicurezza dei dati durante lunghe sessioni di etichettatura, il codice integra un sistema di log che traccia i file già elaborati, permettendo di interrompere e riprendere il processo in qualsiasi momento senza perdite di dati.

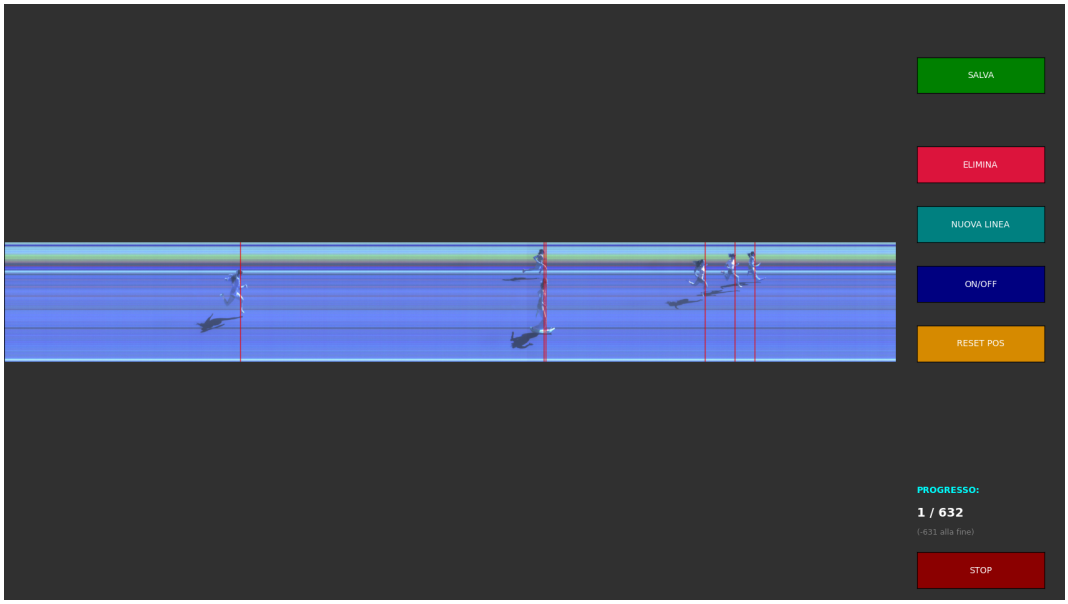


Figura 3.6: Dettaglio dell'interfaccia grafica del programma di etichettatura. Le linee rosse, estratte dalle immagini, possono essere spostate in caso di refusi.

Al termine di questo processo si è ottenuto un database strutturato, in formato CSV, contenente tutte le coordinate assolute delle linee di arrivo associate alle rispettive foto di partenza. Tuttavia, questo formato di dato non era ancora sufficiente per l'addestramento. Affinché il modello potesse imparare a riconoscere il target corretto, era necessario isolare spazialmente i singoli atleti, e traslare le coordinate assolute delle linee in coordinate relative all'ingombro di ogni singolo corridore. Questa problematica è stata demandata a una successiva pipeline tramite l'utilizzo di intelligenza artificiale.

3.2 Pre-processing e gestione del formato Line-Scan

Le immagini di photofinish presentano risoluzioni orizzontali estremamente elevate, raggiungendo spesso svariate migliaia di pixel in larghezza. Fornire un'immagine di tali proporzioni direttamente in ingresso a una rete neurale convoluzionale comporterebbe il rapido esaurimento della memoria della GPU e in caso di ridimensionamento forzato, una notevole distruzione dell'*aspect ratio* che renderebbe le sagome illeggibili.

La tecnica standard per affrontare questo problema è il *Tiling* (o *Slicing*), cioè la suddivisione dell'immagine in "finestre" (patch) più piccole. Un *tiling* geometrico e "cieco" comporterebbe il rischio di tagliare a metà i corpi degli atleti, compromettendo il *dataset*. Per ovviare a questo problema, si è optato per un approccio di "*Tiling Intelligente*", volto a estrarre dinamicamente i *bounding box* dei singoli atleti.

In una prima fase di studio, si è esplorato l'utilizzo di architetture di *Object Detection* standard. L'approccio iniziale prevedeva l'impiego di *YOLOv8x*. I primi risultati sulle immagini ufficiali della *World Athletics* [27] erano discreti, ma il modello soffriva negli arrivi in contemporanea, inglobando atleti ravvicinati in un unico *bounding box* o mancando del tutto alcune rilevazioni. Per tentare di rimediare, si è effettuata un'annotazione manuale correttiva tramite il software *open-source LabelImg* [29], seguita da un *fine-tuning* del modello *YOLO*.

Sebbene i risultati in fase di addestramento fossero eccellenti (Precision 99.1%, Recall 97.9%, mAP50-95 94.2%), le performance in fase di inferenza si sono rivelate essere molto basse. Il modello aveva manifestato un chiaro fenomeno di *overfitting* poichè, invece di generalizzare le caratteristiche del dominio, aveva memorizzato i dati di *training*. La scarsa quantità di dati forniti non era sufficiente per insegnare a una rete così complessa a gestire le sovrapposizioni e le grandi deformazioni anatomiche indotte dalle telecamere *line-scan*. Durante i test, il modello frammentava i corridori, separando erroneamente gli arti dal resto del corpo e rilevandoli come persone distinte.

Alla luce di queste limitazioni, e complice l'adozione del nuovo *dataset* dato dai cronometristi FICr, per associare correttamente le linee del photofinish ai relativi corridori, l'approccio è stato radicalmente riprogettato sviluppando una pipeline ibrida e semi-automatizzata, articolata nei seguenti step:

- per scongiurare la perdita di dettagli causata da ridimensionamenti forzati, l'immagine originale viene ispezionata tramite una tecnica a finestre scorrevoli (*Sliding Window*). Per evitare che un atleta posizionato sui bordi di taglio venga frammentato, la finestra avanza mantenendo un parametro di sovrapposizione (*overlap*), pari al 20% rispetto al ritaglio precedente. Grazie a questo accorgimento geometrico, un atleta che dovesse risultare parzialmente tagliato in una finestra, risulterà perfettamente integro e racchiuso nella finestra successiva. Ogni porzione viene analizzata simultaneamente da due modelli: *YOLOv8x Detection* ricerca le istanze in modo generico, demandando alla logica successiva il filtraggio dei falsi positivi, e *YOLOv8x Pose Estimation*, che si occupa di stimare la posa scheletrica delle figure tramite i *Keypoint* principali, permet-

tendo di dedurre un *bounding box* accurato anche quando il corpo dell'atleta è notevolmente distorto dalle dinamiche di acquisizione *line-scan*;

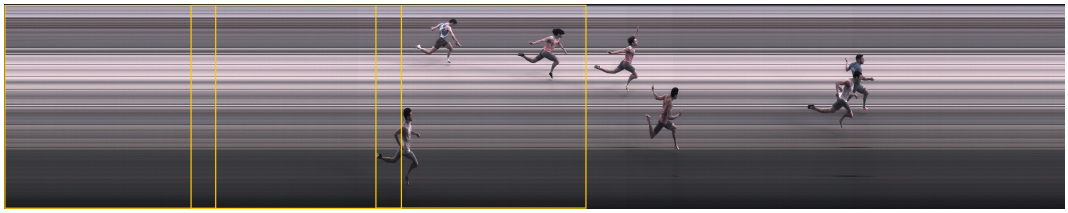


Figura 3.7: Esempio di funzionamento della tecnica *Sliding Window*: viene ritagliata una sotto porzione dell'immagine che presenta altezza invariata e larghezza impostata via codice (es. 1800 pixel).

- i rilevamenti locali ottenuti dalle singole finestre vengono ri-mappati sulle coordinate globali dell'immagine di partenza. A causa della sovrapposizione tra i ritagli, questa operazione genera inevitabilmente rilevamenti multipli e ridondanti per lo stesso atleta. Separare i singoli soggetti nel contesto del photofinish, in cui corridori diversi risultano spesso occlusi o fisicamente adiacenti, è un compito complesso. Per questo motivo, il sistema applica una serie di filtri euristici progettati per valutare e raggruppare i *bounding box*. Questa logica permette di eliminare le ridondanze e mantenere un singolo *crop* accurato per ogni atleta, preservando al contempo le istanze di corridori distinti ma vicini;

- un problema cruciale consiste nell'associare la corretta linea di arrivo (estratta nel Paragrafo 3.1) al *bounding box* appena trovato. Il codice automatizza il processo di associazione calcolando il centro esatto del *crop*, coincidente con il target, e cercando nel database tutte le linee che ricadono in un raggio di 50 pixel da esso. La linea spazialmente più vicina al centro viene proposta come associazione di default;

- le proposte del sistema vengono sottoposte all'operatore tramite un'interfaccia grafica. La linea proposta viene evidenziata e resa interattiva. Se nel ritaglio cadono altre linee, dovute ad atleti vicini, queste vengono tracciate in modo tratteggiato come riferimento visivo. L'operatore può validare la scelta o spostare la linea per correggere eventuali imprecisioni algoritmiche o errori umani pregressi;

- in seguito alla conferma dell'operatore, il sistema ritaglia e salva fisicamente il *crop* dell'atleta. Contestualmente, viene generato un file testuale associato in cui la posizione della linea viene ricalcolata in coordinate locali rispetto al nuovo ritaglio, ottenendo un valore normalizzato continuo tra 0.0 e 1.0. Questa scalatura è un requisito fondamentale per favorire la corretta convergenza matematica della funzione di perdita (*loss function*) durante il successivo addestramento della rete finale.

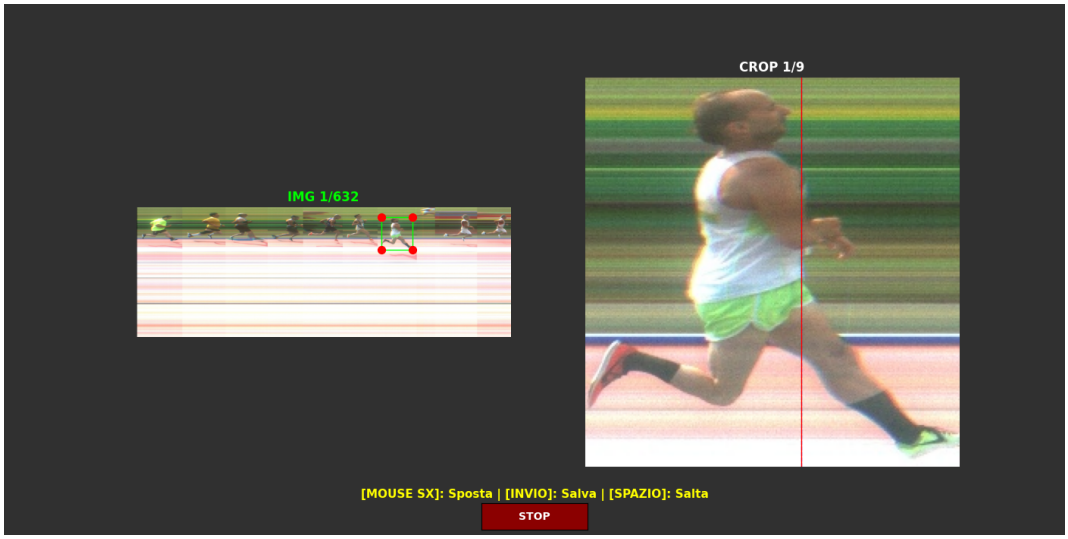


Figura 3.8: Visualizzazione del programma per estrarre i *crop* degli atleti, legandoli alla posizione relativa della linea di arrivo.

3.3 Data Augmentation

Poiché il *dataset* ricavato dalle competizioni locali risulta limitato in termini di numerosità assoluta, l'addestramento diretto di una rete neurale convoluzionale profonda avrebbe comportato un elevato rischio di *overfitting*. Per garantire che il modello imparasse a estrarre e generalizzare le *feature* anatomiche rilevanti, piuttosto che memorizzare i singoli campioni di *training* (*overfitting*), è stata implementata una robusta pipeline di *Data Augmentation* [30].

Le trasformazioni sono state calcolate e applicate in modo dinamico (*on-the-fly*) a ogni epoca di addestramento invece di modificare i dati di partenza o duplicare tutto il *dataset*.

Le tecniche utilizzate sono state accuratamente selezionate per simulare le reali criticità del dominio operativo e si dividono in due categorie principali: *trasformazioni geometriche* e *alterazioni fotometriche*.

3.3.1 Trasformazioni geometriche e spaziali

Queste trasformazioni agiscono sulla struttura spaziale dei *crop* e, a differenza delle classiche pipeline di classificazione, hanno richiesto un aggiornamento matematico rigoroso del *Ground Truth* in questo problema di regressione. Sono state analizzate per rivelare il *bias* del *dataset* studiando la distribuzione delle posizioni della linea di arrivo e cercare di renderla la più piatta possibile. Lo scopo è evitare che la rete impari delle "scorciatoie" ignorando le caratteristiche delle immagini.

Inserendo all'interno di un grafico le distribuzioni di posizione, si evince che esse seguono una distribuzione bimodale, con i picchi agli estremi del grafico e in mezzo una probabilità molto bassa: correndo da destra verso sinistra, o da sinistra verso destra, i busti si trovano quasi sempre agli estremi.

Sono state impiegate due tecniche diverse:

- **Random Shift:** svincola la rete dall'assunzione errata che l'atleta si trovi sempre perfettamente al centro del ritaglio, favorendo l'invarianza spaziale del modello [30]. Dal punto di vista ingegneristico, lo *shift* orizzontale ha richiesto un ricalcolo dinamico della *label* di addestramento. La coordinata x della linea del torso è stata traslata della medesima quantità al fine di mantenere la coerenza spaziale con il torso dell'atleta shiftato. È stato applicato uno spostamento casuale alle immagini, sia sull'asse orizzontale ($\pm 20\%$) sia sull'asse verticale ($\pm 15\%$);

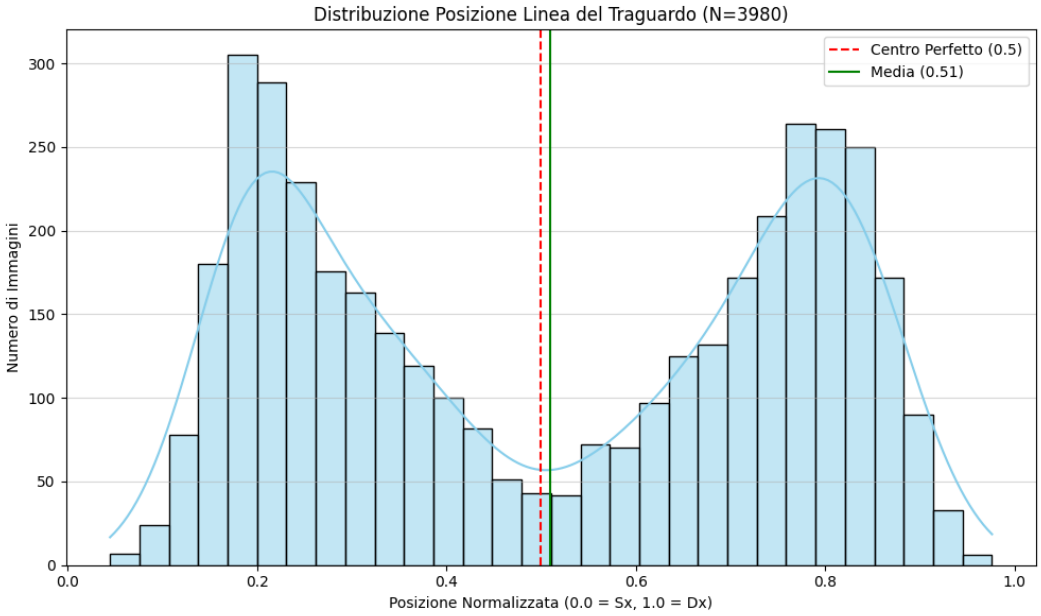


Figura 3.9: Distribuzione delle posizioni delle linee degli arrivi su tutto il *dataset*. Si notano due picchi maggiori dovuti alle caratteristiche intrinseche del dominio di lavoro.

- **Horizontal Flip:** specchia le immagini lungo l'asse verticale con una probabilità del 50%. Nel contesto dell'atletica leggera, questo espediente raddoppia di fatto il *dataset* di addestramento, insegnando al modello a riconoscere la corretta postura del corridore indipendentemente dalla direzione di avanzamento (da sinistra verso destra o viceversa) [31]. Questo ha permesso alla rete di imparare a discriminare correttamente il torso dalla schiena, una problematica che può trarre in inganno producendo dei falsi positivi, con una conseguente riduzione dell'errore finale. Coerentemente con l'applicazione del *flip*, la coordinata bersaglio è stata conseguentemente ricalcolata per specularità, sottraendo il suo valore originario alla larghezza totale del tensore dell'immagine.

3.3.2 Alterazioni fotometriche

Le alterazioni fotometriche e la simulazione del rumore agiscono sui valori di intensità dei pixel per simulare disturbi ambientali e occlusioni fisiche.

- L'approccio **Intensity Jittering** simula l'ampia variabilità di illuminazione riscontrata sul campo durante le acquisizioni reali (es. gare serali con illuminazione artificiale scarsa, zone d'ombra o sovraesposizione solare), rendendo il modello estremamente robusto ai cambiamenti ambientali in fase di acquisizione ottica [31]. La luminosità e il contrasto dell'immagine sono stati variati casualmente in un range del ±30%.

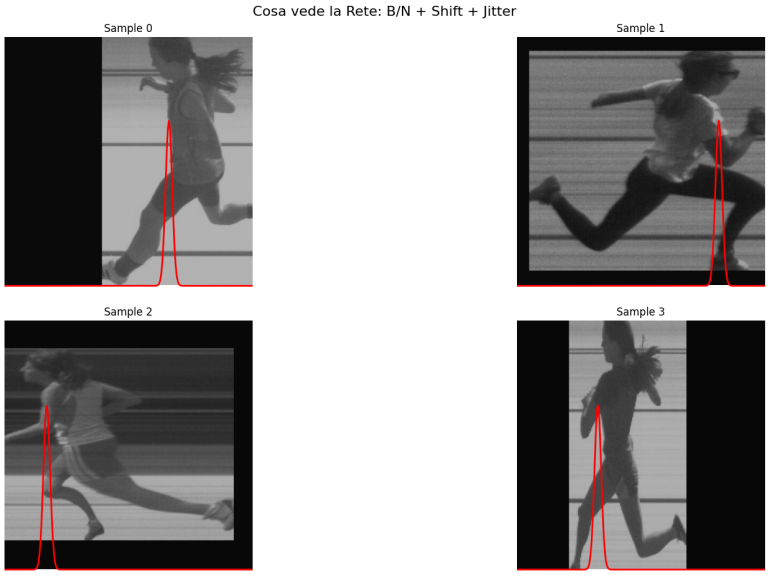


Figura 3.10: Applicazione di alcune tecniche di *Data Augmentation* (Random Shift + Intensity Jitter) a quattro campioni estratti casualmente dal *dataset* di partenza, con il ricalcolo della coordinata da predire, considerata come *heatmap*.

- Nel **Random Erasing** sono state sovrapposte all'immagine, con una probabilità del 50%, delle maschere rettangolari nere di dimensioni variabili (dal 2% al 15% dell'area totale) in posizioni completamente casuali. Questa tecnica si è rivelata di fondamentale importanza per la robustezza del sistema photofinish [32] in quanto permette di evitare le sovrapposizioni.

  Per simulare la parziale occlusione del corpo dovuta agli arti di atleti adiacenti, il *Random Erasing* agisce come un meccanismo di regolarizzazione (*dropout*

spaziale), costringendo la rete neurale a inferire la posizione esatta della linea del torso analizzando il contesto anatomico globale, prevenendo l'eccessiva dipendenza da singole *feature* locali che potrebbero risultare nascoste in un reale scenario di gara [32].

3.4 Suddivisione del dataset

Per valutare oggettivamente le capacità di generalizzazione del modello e prevenire fenomeni di *overfitting*, il *dataset* complessivo è stato suddiviso in tre sottoinsiemi distinti, seguendo le best practice consolidate nella letteratura del *Machine Learning*. Nello specifico, si è optato per una divisione proporzionale pari a 80-10-10:

- **Training Set (80%)**, utilizzato esclusivamente per l'addestramento della rete neurale e per l'ottimizzazione dei pesi tramite l'algoritmo di *backpropagation* dell'errore;
- **Validation Set (10%)**, impiegato al termine di ogni epoca per monitorare l'andamento della funzione di perdita (loss) su dati mai visti durante l'aggiornamento dei pesi. Il *Validation Set* risulta fondamentale per la guida delle scelte architetturali, per il tuning degli iperparametri e per la definizione del momento ottimale in cui arrestare l'addestramento, salvando il modello con le performance migliori;
- **Test Set (10%)**, mantenuto rigorosamente isolato per tutta la fase di sviluppo, viene utilizzato unicamente alla fine del processo per fornire una stima finale e non distorta (*unbiased*) delle prestazioni del modello sul campo.

L'operazione di partizionamento è stata implementata sfruttando la funzione random_split del framework PyTorch. Un aspetto metodologico cruciale implementato in questa fase è stato il blocco del generatore di numeri pseudo-casuali tramite un "*seed*" fisso (manual_seed(42)). Fissare il *seed* randomico è un requisito metodologico essenziale nella ricerca scientifica nell'ambito del *Machine Learning*, in quanto garantisce la totale e perfetta riproducibilità dell'esperimento. Ad ogni esecuzione del codice, infatti, le immagini vengono distribuite esattamente nelle stesse suddivisioni,

permettendo un confronto rigoroso tra diverse architetture o iperparametri senza l'interferenza dovuta alla variazione dei dati di *training*.

Infine, una particolare attenzione è stata posta nella gestione della *Data Augmentation* (descritta nel Paragrafo 3.3). A livello implementativo, la pipeline è stata progettata affinché le trasformazioni geometriche e fotometriche venissero istanziate e applicate dinamicamente esclusivamente al *Training Set*. Il *Validation Set* e il *Test Set*, al contrario, sono stati caricati in formato "grezzo", privi di qualsiasi alterazione artificiale.

Questa rigorosa separazione è fondamentale: valutare il modello su immagini alterate artificialmente introdurrebbe una problematica aggiuntiva, poiché il test avverrebbe su dati non rappresentativi della realtà ottica del sensore e, di conseguenza, le metriche calcolate risulterebbero compromesse, impedendo di stimare con oggettività la reale affidabilità del sistema durante una competizione.

Capitolo 4

Architettura del sistema proposto

La base su cui appoggia tutto il cuore del DSS automatizzato per i cronometristi è basata su un insieme di modelli di Intelligenza Artificiale che cooperano tra di loro per riuscire a stimare il risultato cronometrico corretto. Dopo aver acquisito l'immagine della competizione, questa viene passata in input al programma e come riscontro si ottiene in uscita la lista dei risultati sottoforma di: id dell'atleta, coordinate del *bounding box*, associato all'atleta corrispondente, e la posizione della linea del torso in coordinate assolute (rispetto all'immagine originale).

Per refertare l'immagine in input questo algoritmo fa una serie di passate su diversi livelli di AI partendo da una visione complessiva e andando sempre più nel dettaglio. Questa sequenza di reti è composta in questo modo: due stadi in cascata si occupano di determinare le posizioni delle persone, estrarre i relativi *crop* dei corridori, e l'ultimo stadio finale prende questi ritagli e li analizza per ottenere la coordinata del torso.

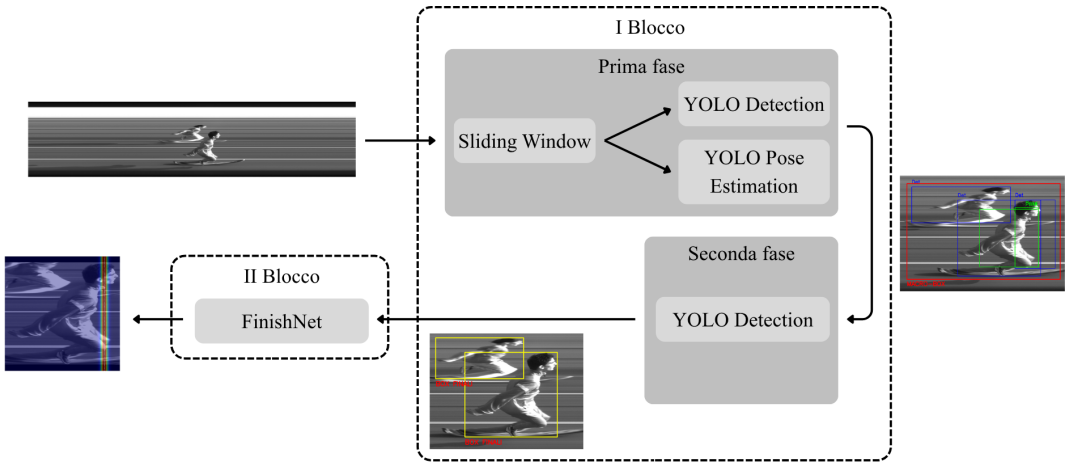


Figura 4.1: Schema che mostra il funzionamento del sistema complessivo di refertazione dell'immagine basato sull'AI, dall'immagine in input fino alla predizione della *heatmap* corrispondente.

4.1 Primo stadio: localizzazione e ritaglio degli atleti

Il primo stadio dell'architettura, la base su cui si appoggia l'intero sistema finale, è dedicato alla localizzazione degli atleti all'interno della foto generata dalla telecamera lineare. Per questo compito è stato scelto il modello *YOLOv8x* [33], la versione "extra-large" della famiglia *YOLOv8*, che offre la massima capacità parametrica e accuratezza tra i modelli disponibili. La scelta di *YOLO* rispetto ad architetture a due stadi (come la R-CNN [34]) è motivata dalla sua natura *single-shot* e da un approccio architetturale basato su *anchor-free* [35]. L'approccio *single-shot* permette al modello di predire simultaneamente le coordinate dei *bounding box* e le probabilità di classe in un'unica passata in avanti (*forward pass*), garantendo tempi di inferenza compatibili con le esigenze di un sistema live. Inoltre, l'onere computazionale della R-CNN [34] risulterebbe aggravato dalla necessità di suddividere l'immagine in sotto-porzioni più piccole: la rete dovrebbe estrarre queste immagini, proporre le "Region of Interest" (RoI) e valutarle una per una, allungando a dismisura i tempi impiegati.

Tuttavia, è la natura *anchor-free* a rendere *YOLO* ideale per questo specifico dominio. Data la natura spazio-temporale di queste foto, i soggetti risultano fortemente

allungati o compressi rispetto alle classiche immagini bidimensionali, a seconda della velocità con cui arrivano davanti all'obiettivo. Eliminando la dipendenza da *anchor box* predefiniti (che imporrebbero proporzioni geometriche rigide non adatte alle *silhouette* distorte del photofinish), *YOLOv8* predice direttamente i centri spaziali degli atleti, calcolando da essi le distanze assolute per formare i vertici del box. Ottenere un *bounding box* affidabile riduce l'incertezza legata alla sovrapposizione dei corpi e permette di isolare l'atleta, potendo così passare il ritaglio direttamente a una rete addestrata *ad-hoc* per analizzarne il torso.

Nonostante queste eccellenti capacità, l'applicazione diretta di *YOLO* alle immagini di photofinish presenta una forte criticità geometrica: queste immagini possiedono un rapporto di forma (*aspect ratio*) estremo, sviluppandosi unicamente in orizzontale (es. una scansione di 2 secondi a 5000 linee/secondo genera una matrice di 10.000×1.000 pixel). Per preservare la risoluzione spaziale anatomica, che andrebbe irrimediabilmente perduta nel ridimensionamento standard della rete, il sistema è stato progettato con un rigoroso approccio a cascata (coarse-to-fine), suddiviso in due fasi operative distinte non solo per le soglie applicate, ma soprattutto per il drastico cambio di risoluzione e per le opposte strategie di gestione dei *bounding box*:

1. **Ricerca globale tramite finestra scorrevole:** Inizialmente, l'immagine grezza non viene processata nella sua interezza. Il sistema applica una tecnica di *sliding window*, ritagliando dinamicamente porzioni di immagine a tutta altezza e di larghezza fissa (1800 pixel) che si sovrappongono parzialmente (20%). Ogni porzione viene analizzata in parallelo da due reti *YOLO* distinte: una specializzata nella *Object Detection* e una nella *Pose Estimation*, al fine di garantire l'individuazione di tutti gli atleti.

   Durante questa fase esplorativa, per evitare che l'algoritmo interno di *YOLO* comprima i ritagli alla risoluzione di fabbrica (640×640 pixel) degradando il dettaglio visivo, la dimensione di inferenza è stata vincolata a coincidere con la larghezza della finestra scorrevole stessa.

   Poiché i modelli elaborano i singoli ritagli come spazi indipendenti, le coordinate predette risultano relative al sistema di riferimento locale del rettangolo. Per ricostruire la posizione globale di ciascun atleta, è stata implementata una mappatura inversa: all'asse orizzontale dei *bounding box* (nello specifico, le coordinate $x_{\min}$ e $x_{\max}$) e ai vertici spaziali dei *keypoints* viene sommato l'offset

di scorrimento (x_{start}) del ritaglio corrispondente, riproiettando così le predizioni sul sistema di coordinate assoluto dell'immagine originale complessiva.

Per massimizzare il recall ed eliminare il rischio di mancare un atleta, le soglie di confidenza in questa fase sono impostate a valori deliberatamente bassi (0.05). Poiché lo sfondo è statico e ripetitivo, l'obiettivo è isolare ogni anomalia visiva in prossimità degli atleti, tollerando l'eventuale cattura di rumore visivo.

Anziché utilizzare i *bounding box* di default estratti dalla *Pose Estimation*, il sistema li scarta e ne ricalcola di nuovi, generando recinti spaziali perfettamente aderenti ai soli *keypoints* anatomici validi. A questo punto, il sistema interviene con una logica puramente agglomerativa: questi box strutturali vengono uniti a quelli visivi generati dalla *Detection*. Tutti i rettangoli risultanti non vengono filtrati, bensì fusi (blending) in "macro-box" unificati. L'algoritmo accorpa tutti i rilevamenti che si sovrappongono o che risultano sufficientemente vicini, applicando un margine spaziale di tolleranza aggiuntivo (30 pixel in orizzontale e 20 in verticale) per inglobare con assoluta sicurezza l'intera figura dell'atleta e i suoi frammenti adiacenti.

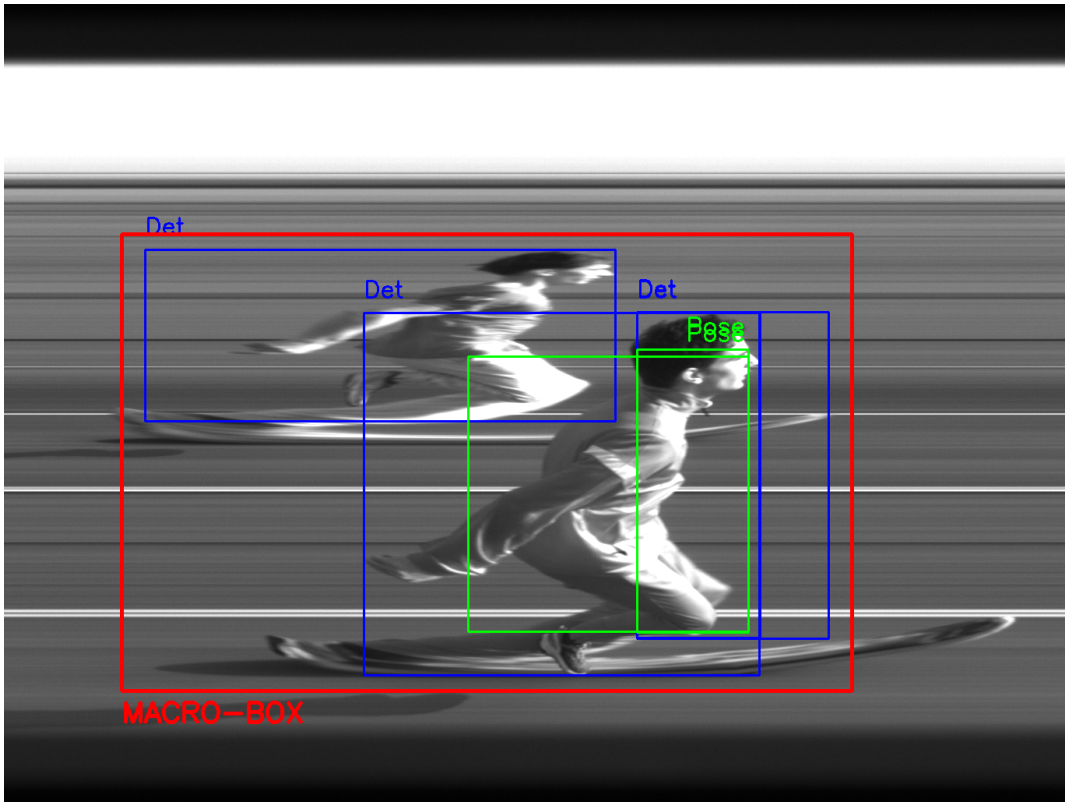


Figura 4.2: Visualizzazione del processo di unione al termine della prima fase. In blu sono evidenziati i *bounding box* generati dal modello di *Object Detection*; in verde quelli ricostruiti a partire dai *keypoint* anatomici della *Pose Estimation*. In rosso è evidenziato il macro-box risultante dall'algoritmo di blending, comprensivo del *padding* di sicurezza spaziale.

2. **Raffinamento locale:** I macro-box generati nella prima fase forniscono una localizzazione approssimativa, che spesso ingloba più atleti in caso di arrivi simultanei o include estese sezioni di pista. Per ottenere coordinate di altissima precisione, questi macro-box vengono utilizzati per estrarre dei ritagli mirati (*crop*) dall'immagine originale.

   In questo secondo stadio, i ritagli vengono analizzati esclusivamente tramite la rete specializzata nella *Object Detection* classica, poiché, come precedentemente discusso, si è rivelata nettamente più robusta sul dominio spazio-temporale rispetto alla stima della posa. Tali ritagli vengono sottoposti a questa seconda inferenza impostando una soglia di confidenza leggermente più restrittiva (0.1).

La vera forza di questa fase risiede nel drastico cambio di contesto visivo: fornendo in input un ritaglio già limitato all'area di interesse, il sistema vincola la dimensione di inferenza a 640×640 pixel, che rappresenta la risoluzione nativa di addestramento dell'architettura *YOLOv8*. Poiché le dimensioni fisiche del singolo ritaglio sono tipicamente inferiori (es. 150×400 pixel), l'algoritmo interno esegue un *upscaling* conservativo, mantenendo intatto l'*aspect ratio* tramite l'inserimento di bande nere di *padding*. L'atleta viene ingrandito fino a riempire l'intero campo visivo della rete. Questo elimina il rumore di fondo e permette ai filtri convoluzionali di concentrarsi unicamente sui micro-dettagli anatomici, riuscendo a calcolare i confini in modo netto e isolando perfettamente il torso di ogni singolo atleta prima di demandare il calcolo della linea di traguardo allo stadio neurale successivo.

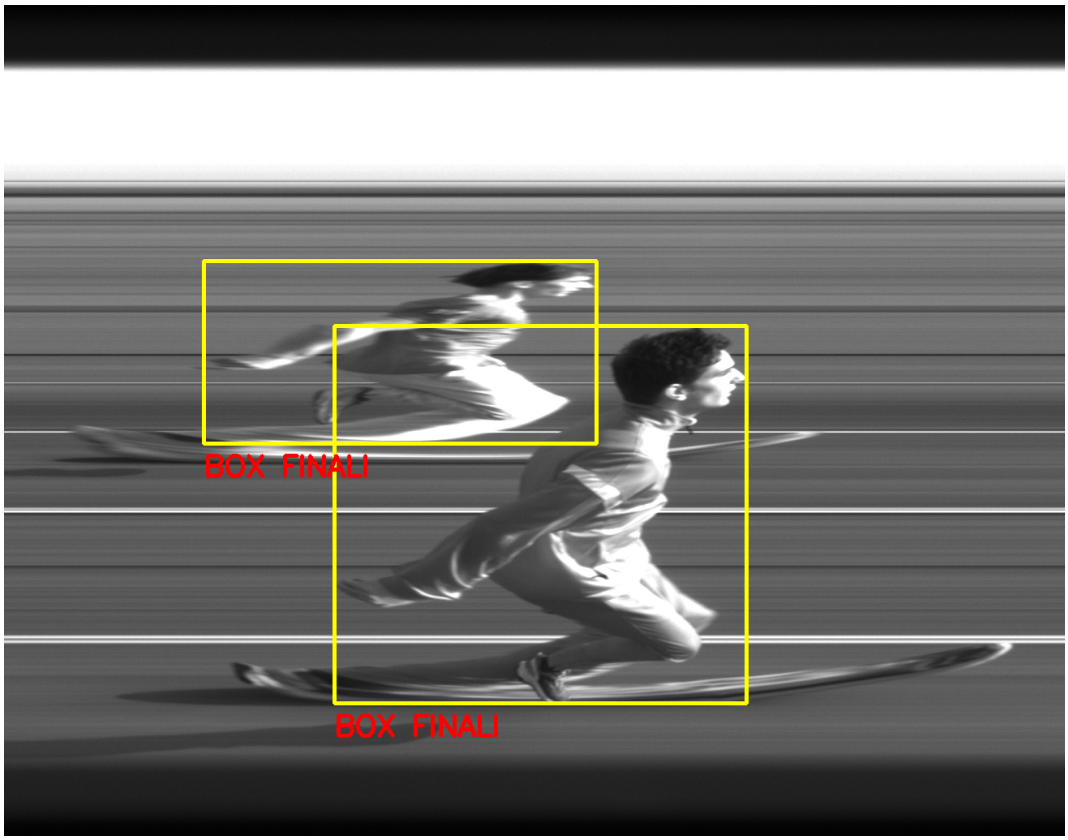


Figura 4.3: Visualizzazione del risultato finale al termine anche della seconda fase.

Poiché la soglia di confidenza impostata è volutamente ancora bassa per evitare

di perdere casi positivi (falsi negativi), la rete può generare del rumore predittivo o rilevamenti multipli sullo stesso soggetto. Per ovviare a questo problema, le coordinate dei *bounding box* raffinate vengono ricondotte al sistema di riferimento globale e sottoposte a una pipeline di pulizia geometrica e deterministica:

- **Simple *NMS*:** In primo luogo, applica una procedura di *Non-Maximum Suppression* per eliminare eventuali predizioni multiple ridondanti formatesi sui bordi dei ritagli, sopprimendo i box che si sovrappongono per un'area (IoU) superiore a una soglia dello 0.4.
- **Rimozione box annidati:** A questa segue la rimozione gerarchica dei box completamente intrappolati all'interno di rilevamenti più estesi, impiegando una severa soglia di contenimento del 95% per cancellare falsi riconoscimenti di singoli arti.
- **Filtri per outlier geometrici:** Infine, per separare in modo definitivo i veri atleti dagli artefatti del *background*, intervengono rigidi vincoli fisici e biomeccanici: il sistema scarta automaticamente i rilevamenti che possiedono un'altezza inferiore al 10% dell'altezza del *frame* totale, quelli che presentano un rapporto di forma eccessivamente sproporzionato, oppure quelli la cui area complessiva (in termini di pixel ricoperti) risulta troppo limitata per essere compatibile con la presenza di un corpo umano.

4.1.1 Analisi delle prestazioni dell'approccio Ensemble (Detection e Pose Estimation)

Per garantire la massima robustezza del sistema e azzerare il tasso di falsi negativi (massimizzando il recall), il primo stadio è stato inizialmente rafforzato seguendo una logica di Blending Ensemble, cioè combinare le previsioni di più modelli indipendenti per creare un modello più preciso e robusto. Nello specifico, al modello di *Object Detection* standard è stato affiancato un modello *YOLOv8x-Pose* per la stima della posa, introducendo una mirata ridondanza architetturale.

L'ipotesi alla base di questa scelta risiedeva nella complementarietà delle due reti: mentre la *Object Detection* si fonda sull'estrazione di *feature* visive globali

e texture (le quali risultano intrinsecamente limitate dal dominio monocromatico derivante dall'acquisizione della telecamera lineare DALSA, in bianco e nero), la *Pose Estimation* vincola la predizione al riconoscimento di *keypoint* anatomici specifici (testa, spalle, ginocchia) e alle relative connessioni scheletriche. Unendo e fondendo i due output tramite un'operazione di blending, il sistema è stato progettato per compensare le incertezze del singolo modello, creando un macro-box sicuro anche in presenza di occlusioni parziali o quasi totali.

Tuttavia, l'evidenza empirica derivata dai test sul dominio specifico del photofinish ha portato a un'importante osservazione sperimentale sulle capacità di generalizzazione delle reti neurali. I modelli di *Pose Estimation*, essendo pre-addestrati su *dataset* convenzionali, assumono intrinsecamente la presenza di proporzioni biomeccaniche standard. Nelle immagini generate dalle telecamere lineari, l'asse orizzontale rappresenta lo scorrere del tempo, questo genera alterazioni strutturali che invalidano i rigidi vincoli spaziali attesi dalla rete Pose. Di conseguenza, si è osservato che, sebbene il modello di *Pose Estimation* restituisca risultati dignitosi su immagini di altissima qualità ottenute in condizioni strettamente controllate (come per il *dataset* derivante dal sito WorldAthletics), le sue prestazioni degradano drasticamente al diminuire della fedeltà visiva della foto.

Questo limite è emerso in modo inequivocabile durante i test sperimentali condotti sul campo. Operando in condizioni ambientali non controllate, caratterizzate da illuminazione naturale variabile e da un setup di acquisizione non professionale (fattori che hanno introdotto un inevitabile rumore visivo e una ridotta risoluzione nei *frame*), le rilevazioni valide sono state prodotte quasi esclusivamente dal modello di *Detection*. Quest'ultimo si è dimostrato ampiamente sufficiente e dominante, adattandosi in modo eccellente alle *silhouette* distorte e rendendo marginale, ai fini pratici, l'apporto della stima della posa. Tale osservazione evidenzia conclusivamente come l'astrazione delle *feature* globali sia nettamente più resiliente ai cambi di dominio spazio-temporali e alle criticità di acquisizione del mondo reale rispetto ai modelli ancorati a mappe di calore anatomiche rigorose.

4.2 Secondo stadio: regressione della linea di traguardo

Fino a questo punto del progetto abbiamo individuato, il più precisamente possibile, i *bounding box* di tutti i corridori presenti nella competizione. Ora manca la parte fondamentale: individuare il torso per determinare il tempo impiegato per concludere la gara. Per questo scopo, vengono estratti dalle immagini acquisite i *crop* degli atleti e, tramite una rete neurale *ad-hoc*, viene stimata la posizione esatta di attraversamento.

4.2.1 Analisi dimensionale e preservazione della risoluzione temporale

La scelta della risoluzione spaziale in ingresso alla rete (512×512 pixel) è stata guidata da un'analisi statistica dei *crop* (estratti dalle immagini del dataset) e da un vincolo fisico insuperabile del dominio del photofinish: la dimensione orizzontale coincide con il tempo. Poiché ogni pixel orizzontale rappresenta una precisa frazione di secondo (es. 1 millisecondo), qualsiasi operazione di rimpicciolimento comporterebbe la fusione di pixel adiacenti, distruggendo irrimediabilmente la risoluzione temporale dell'acquisizione nativa.

Analizzando la distribuzione campionaria delle dimensioni degli atleti, si osserva che la larghezza dei ritagli si concentra prevalentemente nel range tra 150 e 350 pixel, con outlier fisiologici che superano i 500 pixel in caso di posture allungate. L'adozione del formato standard di letteratura (224×224 pixel come hanno molte reti neurali comuni) avrebbe forzato una drastica compressione orizzontale per quasi la maggioranza dei dati del *dataset*, degradando i contorni anatomici critici per la stima del traguardo.

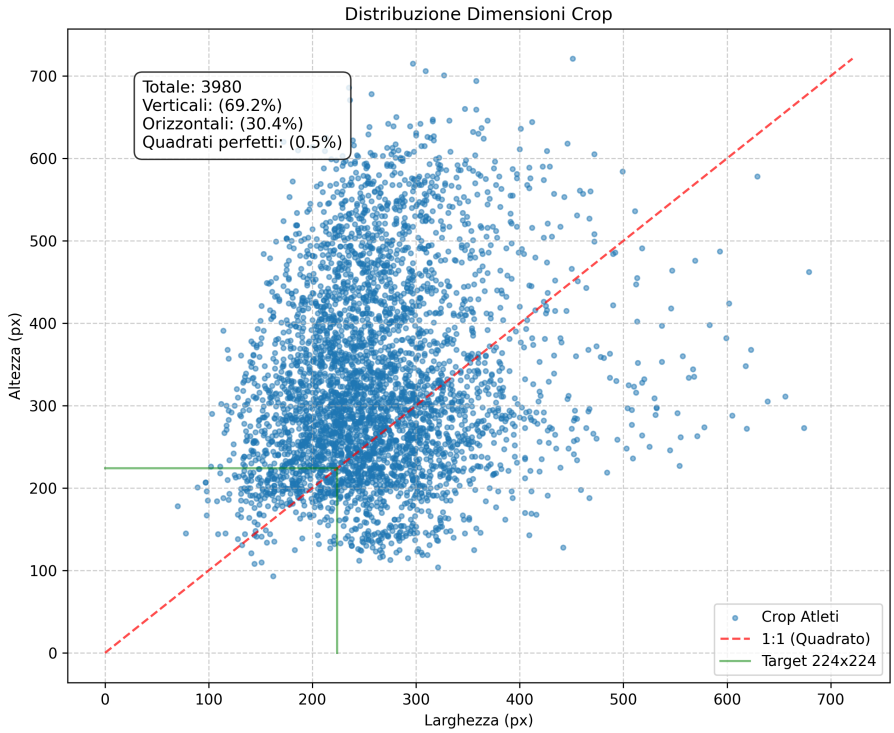


Figura 4.4: Visualizzazione della distribuzione della dimensione dei *crop* ottenuti isolando i singoli atleti dalle immagini originali di gara.

La scelta del formato 512×512 funge da vincolo conservativo, garantendo un margine di sicurezza per coprire in modo ottimale la densità massima della distribuzione statistica. Al fine di normalizzare l'input senza distorcere le proporzioni fisiologiche dell'atleta (fattore critico per la stima corretta del petto), in fase di pre-elaborazione i ritagli non vengono deformati, bensì ridimensionati mantenendo intatto l'*aspect ratio* originale. Lo spazio residuo per raggiungere il formato quadrato 512×512 viene successivamente riempito tramite *padding* simmetrico a valore nullo (bande nere), che la rete impara rapidamente a ignorare grazie alla creazione dei valori di *heatmap*, impostati a zero in questa zona, e alla sua struttura convoluzionale.

Attraverso questa operazione, il sistema esegue un lieve *upscaling* sull'asse orizzontale per la stragrande maggioranza dei campioni. Questa strategia garantisce che il vettore predittivo di uscita, di lunghezza pari a 512 elementi, mantenga una

"granulosità" sufficiente per ospitare la distribuzione Gaussiana senza sacrificare alcun millesimo di secondo, preservando intatta la precisione millimetrica richiesta per la refertazione ufficiale.

Infine, l'adozione del formato 512×512, essendo una potenza di due, garantisce una gestione ottimale dei volumi di attivazione attraverso i ripetuti passaggi di pooling e downsampling delle architetture impiegate, prevenendo la formazione di artefatti matematici da arrotondamento (come l'aliasing).

4.2.2 Il target matematico: Heatmap Regression

Per massimizzare l'affidabilità e la precisione del secondo stadio neurale, si è scelto di abbandonare l'approccio classico della regressione diretta delle coordinate (*Direct Coordinate Regression*), in favore della predizione di una distribuzione di probabilità monodimensionale (*Heatmap Regression*).

La predizione di un singolo valore assoluto X risulta intrinsecamente instabile per le reti neurali convoluzionali. Richiedere al modello un output puntuale equivale a imporre una penalità drastica (assimilabile a una funzione Delta di Dirac) per cui il pixel target possiede una veridicità del 100%, mentre i pixel immediatamente adiacenti (es. $X \pm 1$) vengono considerati completamente errati [36]. Questa rigidità produce gradienti di errore molto severi che ostacolano la convergenza della funzione di costo verso il minimo globale, specialmente in un dominio complesso come il photofinish.

Le acquisizioni delle telecamere lineari presentano infatti altissime varianze dovute all'illuminazione naturale, al rumore di fondo e, soprattutto, a occlusioni biomeccaniche (es. il braccio del corridore che copre parzialmente il torso). Costringere la rete a un'astrazione numerica netta in presenza di input così eterogenei porta inevitabilmente all'instabilità del modello [36].

L'addestramento basato su una distribuzione di probabilità, estesa per tutta la larghezza spaziale del ritaglio (512 pixel), risolve questa criticità fornendo un target "morbido" (*soft label*). Questa scelta progettuale comporta tre vantaggi ingegneristici fondamentali:

1. **Assorbimento del rumore di annotazione:** La determinazione esatta del petto/torso è complessa anche per l'occhio umano. La fase di etichettatura

manuale (creazione del *Ground Truth*) porta con sé un'inevitabile incertezza percettiva. La mappa di probabilità modella matematicamente questa incertezza, insegnando alla rete a concentrarsi su un "intorno" di confidenza piuttosto che su un singolo pixel perfetto [37];

2. **Robustezza alle regole di gara:** Nel cronometraggio dell'atletica leggera, il tempo finale viene sistematicamente arrotondato al centesimo di secondo superiore. Un errore predittivo di pochi pixel cade statisticamente all'interno dello stesso centesimo di secondo, rendendo l'incertezza millimetrica del modello ininfluente ai fini del risultato ufficiale della competizione;

3. **Mappatura spaziale continua:** La rete impara una vera e propria funzione di densità spaziale, assegnando probabilità decrescenti man mano che ci si allontana dal target corretto, favorendo gradienti di apprendimento fluidi e stabili.

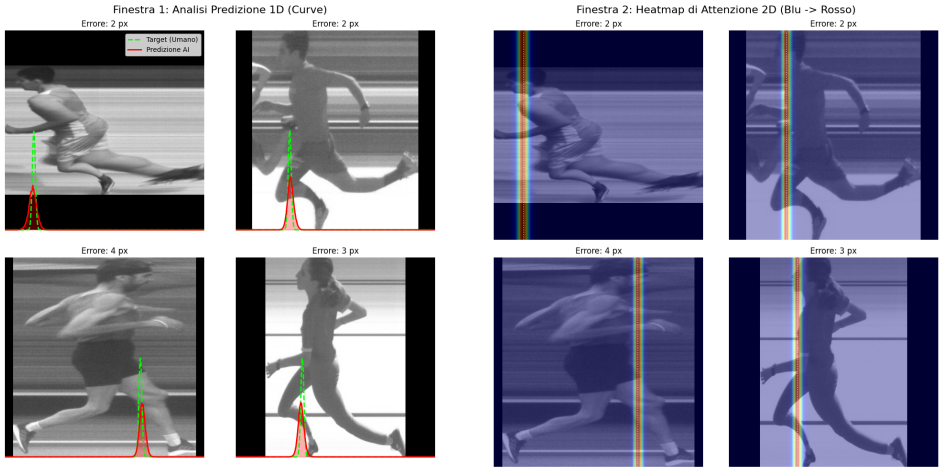
Dal punto di vista matematico, durante la preparazione del *dataset*, la coordinata puntuale annotata dal cronometrista (μ) viene trasformata in una curva Gaussiana monodimensionale. La densità di probabilità target $P(x)$ per ogni pixel x lungo l'asse orizzontale dell'immagine è calcolata secondo la formula della distribuzione normale non normalizzata:

$$P(x) = e^{-\frac{1}{2}\left(\frac{x-\mu}{\sigma}\right)^2} \tag{4.1}$$

Per il dominio specifico del photofinish, la deviazione standard è stata empiricamente impostata a $\sigma = 4$. Questo valore genera una "campana" estremamente stretta e focalizzata in prossimità del valore corretto. Infine, il vettore risultante viene normalizzato dividendolo per la somma totale delle sue probabilità, affinché l'area sottesa dalla curva sia pari a 1.

Un ulteriore accorgimento geometrico è stato implementato per gestire i ritagli degli atleti ridimensionati a 512 pixel tramite *padding*: poiché le bande nere laterali aggiunte per mantenere l'*aspect ratio* originale non contengono informazioni visive, i bordi assoluti del vettore target vengono forzatamente azzerati. Questo garantisce che in quelle porzioni la probabilità sia strettamente nulla e impedisce alla rete di sviluppare *bias* predittivi sui margini artificiali dell'immagine. Il risultato finale di questo processo è un vettore monodimensionale di lunghezza 512, in cui ogni

elemento rappresenta l'esatta probabilità in cui si possa trovare la linea dell'atleta in corrispondenza di quello specifico pixel.



(a) **(b)**

Figura 4.5: Nella finestra (a) sono rappresentate le gaussiane a partire dal valore del *Ground Truth* nel *dataset* (linea tratteggiata verde) e quella predetta dalla rete neurale (linea rossa). Nella finestra (b) è rappresentata la gaussiana predetta dall'AI mostrata sotto forma di *heatmap* con la sfumatura dal blu al rosso (blu: probabilità molto bassa, rosso: valore corretto, probabilità molto alta).

4.2.3 Architettura proposta (FinishNet)

Per la risoluzione del task di *Heatmap Regression* delineato precedentemente, è stata progettata e addestrata un'architettura convoluzionale *custom* (denominata *FinishNet*). Sebbene la rete sia stata ottimizzata specificamente per il dominio del photofinish, la sua topologia macroscopica si ispira fortemente all'architettura delle reti *ResNet*. La scelta di adottare la progressione classica nell'espansione dei canali (64-128-256-512) combinata con il dimezzamento spaziale deriva infatti dalla comprovata efficacia di questo design nell'estrazione gerarchica delle *feature*. Tuttavia, l'obiettivo primario di questo design personalizzato è stato trovare un compromesso ottimale tra capacità espressiva della rete ed efficienza computazionale, evitando l'eccessivo carico di parametri tipico delle architetture generaliste. La rete prende in

input un tensore monocromatico (scala di grigi) di dimensioni 512×512 pixel ed è strutturata in tre macro-moduli sequenziali: lo *stem*, il *body* (costituito da blocchi residuali potenziati con meccanismi di attenzione) e la *head* di regressione:

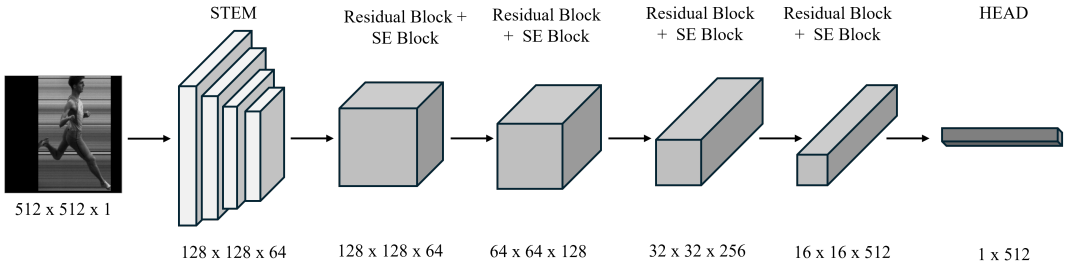


Figura 4.6: Schema ad alto livello della rete "*FinishNet*" nel quale si vedono i macro blocchi principali.

1. **Modulo Iniziale (stem):** Il livello di ingresso della rete, che si interfaccia direttamente con l'immagine originale, ha il compito di estrarre le *feature* a basso livello (bordi, linee, contrasti) e ridurre rapidamente la dimensionalità spaziale per alleggerire il carico computazionale. È composto da una serie di operazioni in sequenza. All'input vengono applicati 64 filtri tramite una convoluzione con un *kernel* ampio 7×7 e passo (*stride*) pari a 2, ideali per cogliere pattern macroscopici come il profilo di un braccio o del petto, dimezzando istantaneamente la risoluzione spaziale (256×256). Il calcolo della dimensionalità dopo aver applicato un livello convoluzionale è calcolato con la seguente formula:

$$\text{dimensione} = \left\lfloor \frac{\mathrm{N} + 2 \times \mathrm{P} - \mathrm{K}}{\mathrm{S}} \right\rfloor + 1 \tag{4.2}$$

Dove:

N : Dimensione dell'immagine in input al livello

P : Padding

K : Dimensione del filtro

S : Stride

Per evitare instabilità durante l'addestramento, questi 64 canali vengono sottoposti a *Batch Normalization*, garantendo che ciascun livello mantenga media

nulla e varianza unitaria, impedendo l'esplosione o l'annullamento dei valori numerici. Segue un livello di attivazione *ReLU*, che introduce non-linearità azzerando i valori negativi, e un livello di *Max Pooling* 3×3 con *stride* di 2. Questa combinazione abbatte ulteriormente la risoluzione del tensore, passando da $512 \times 512 \times 1$ a $128 \times 128 \times 64$, preparando mappe di *feature* dense per l'elaborazione successiva senza saturare la memoria della GPU.

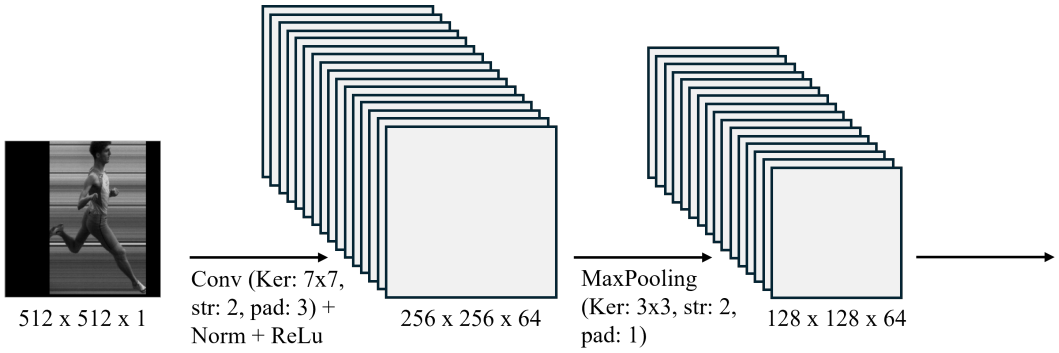


Figura 4.7: Schema raffigurante il livello *stem* nel dettaglio con le operazioni impiegate e il dettaglio sulle dimensionalità.

2. **Corpo della Rete (Residual Blocks e Attenzione SE):** Questo macro-livello elabora le informazioni provenienti dallo *stem* per discriminare i pixel che rappresentano l'anatomia dell'atleta dagli elementi di disturbo. Il nucleo estrattivo è formato da quattro stadi sequenziali basati sull'architettura dei *Residual Block* [38]. Procedendo in profondità, i canali raddoppiano progressivamente (da 64 fino a 512) mentre la risoluzione spaziale viene regolarmente dimezzata tramite convoluzioni con *stride* pari a 2. L'impiego delle *skip connections* permette al gradiente di fluire agilmente all'indietro durante la *backpropagation*, mitigando il noto problema del *Vanishing Gradient*. Invece di forzare l'informazione unicamente attraverso gli strati convoluzionali, che potrebbero degradarla, essa viene bypassata e sommata al risultato finale del blocco. Qualora l'elaborazione alteri le dimensioni spaziali o di canale del tensore, la *shortcut* implementa una proiezione lineare tramite una convoluzione 1×1 priva di funzioni di attivazione, garantendo la compatibilità algebrica e preservando l'informazione di base.

   Per ottimizzare l'apprendimento, al termine di ogni blocco residuale è stato integrato un modulo di SE [39]. Nel dominio del photofinish, l'immagine

contiene un'alta percentuale di informazioni irrilevanti (come il rumore di fondo o le linee di corsia dilatate). Il blocco SE agisce come un sofisticato meccanismo di attenzione visiva bidirezionale:

- **Squeeze:** Un livello di *Adaptive Average Pooling* comprime spazialmente ogni canale in un singolo valore numerico, generando un descrittore globale che ne riassume il contenuto.
- **Excitation:** Due strati completamente connessi (*Fully Connected*), intervallati da una funzione *ReLU* e terminanti con una funzione *Sigmoide*, mappano le relazioni non lineari tra i canali.

L'output di questo processo fornisce un vettore di "pesi" percentuali (compresi tra 0 e 1) che viene moltiplicato per la mappa delle *feature* originale. In questo modo, la rete impara dinamicamente ad "accendere" (esaltare con valori prossimi a 0.99) i filtri concentrati sui dettagli anatomici del torso e a "spegnere" i filtri focalizzati sul rumore di fondo, massimizzando l'estrazione dei dati rilevanti.

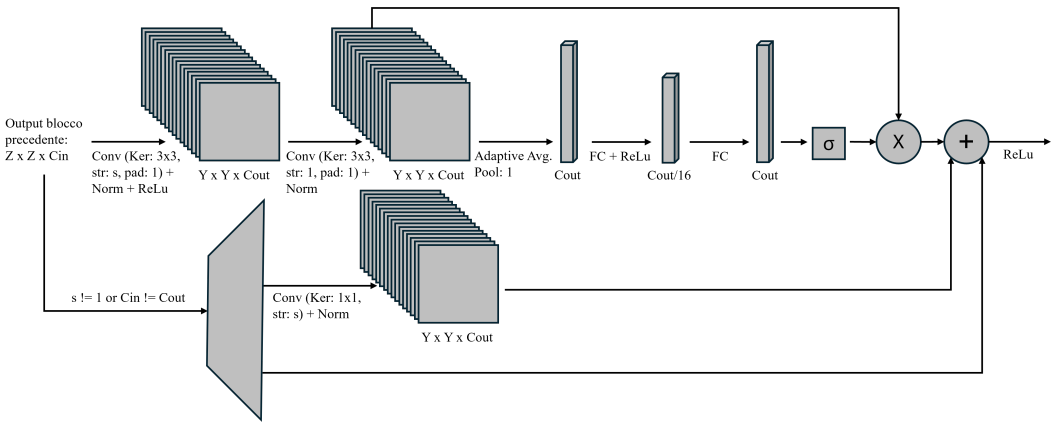


Figura 4.8: Schema raffigurante il blocco residuale con l'aggiunta del blocco SE nel corpo della rete con le operazioni impiegate e il dettaglio sulle dimensionalità.

3. **Testa di Regressione (Head):** Al termine dell'estrazione, la *head* processa questi dati per convertirli nell'output desiderato: la *heatmap* su 512 pixel. Il tensore viene uniformato spazialmente tramite un livello di *Adaptive Average Pooling* dimensionato a 4x4, preservando così un minimo riscontro posizionale all'interno del *frame*. Le caratteristiche estratte vengono appiattite in un vettore monodimensionale (*Flatten*) e processate da due strati lineari profondi

(rispettivamente da 1024 e 512 neuroni), i quali combinano le informazioni per dedurre la posizione esatta della linea target. Per mitigare il rischio di *overfitting*, sono stati introdotti due livelli di *Dropout* con probabilità di azzeramento del 50%. Questo costringe la rete a distribuire la propria conoscenza su tutti i neuroni anziché affidarsi ciecamente a poche *feature* dominanti, rendendo il modello estremamente generalizzabile su dati mai visti. L'architettura si conclude con una funzione di attivazione *LogSoftmax* sull'asse orizzontale, che converte l'uscita grezza in una distribuzione logaritmica di probabilità lunga 512 elementi. Questo rappresenta un requisito matematico stringente: la funzione di costo *KLDivLoss*, così come fornita dal framework PyTorch [40] e utilizzata in fase di addestramento esige logaritmi naturali di probabilità per prevenire errori di *underflow* numerico durante i calcoli in virgola mobile. A differenza dell'errore quadratico medio (MSE), comunemente usato nella regressione, la KL-Divergence è un operatore che misura quanto una distribuzione di probabilità sia "distante" da una distribuzione di riferimento.

Matematicamente, per due distribuzioni discrete P (il target) e Q (la predizione), la divergenza è definita come:

$$D_{KL}(P||Q) = \sum_i P(i) \log \frac{P(i)}{Q(i)} \tag{4.3}$$

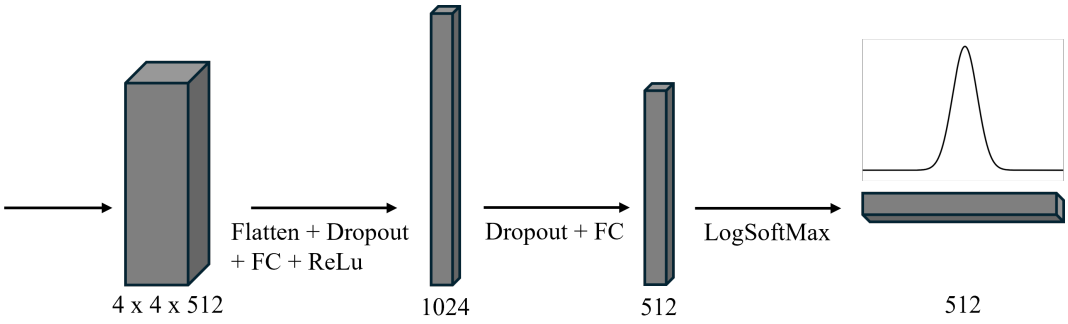


Figura 4.9: Schema raffigurante il livello *Head* nel dettaglio con le operazioni impiegate e il dettaglio sulle dimensionalità. La curva gaussiana rappresenta il significato dei dati calcolati in uscita della rete.

4.2.4 Modello di confronto: ResNet-34 con Transfer Learning

Per validare oggettivamente le prestazioni e l'efficienza computazionale dell'architettura personalizzata *FinishNet*, è stato implementato anche un modello basato su un'architettura standard consolidata in letteratura. La scelta è ricaduta sulla *ResNet-34* [38], una rete neurale profonda 34 *layer* che rappresenta uno standard de facto per l'estrazione di *feature* visive.

Invece di addestrare questo modello da zero si è optato per la tecnica del Transfer Learning sfruttando i pesi pre-addestrati sul vastissimo *dataset* ImageNet. Questa strategia permette alla rete di partire da una conoscenza visiva pregressa (capacità di riconoscere bordi, forme complesse e texture) già presente nei suoi filtri, accelerando la convergenza e riducendo il rischio di *overfitting* sul dominio specifico del photofinish. L'adattamento della *ResNet-34* a questo nuovo task ha richiesto due interventi architetturali:

1. **Adattamento dell'input:** La rete si aspetta in ingresso un tensore a 3 canali, corrispondente a un'immagine a colori nello spazio RGB. Il *dataset* del photofinish è invece composto da immagini monocromatiche, così come le successive acquisizioni, (1 canale in scala di grigi) di dimensioni $512 \times 512 \times 1$. Per risolvere questa incompatibilità senza alterare l'architettura originale, si è scelto di non modificare il primo strato convoluzionale perché avrebbe infatti comportato la perdita e la reinizializzazione casuale dei pesi più importanti per l'estrazione delle *feature* di base. Si è invece operato a replicare tre volte, sull'asse della profondità spaziale tramite un'operazione di ripetizione (x.repeat(1, 3, 1, 1)), l'immagine in input. Questa clonazione genera un tensore "simil-RGB" di dimensioni $512 \times 512 \times 3$, perfettamente compatibile con le aspettative della rete, permettendo ai pesi pre-addestrati di operare in modo ottimale.

2. **Sostituzione della testa di classificazione:** L'architettura *ResNet* originale termina con un livello *Fully Connected* progettato per la classificazione di 1000 categorie. Per riadattare la rete al task di questo progetto di studio, l'intera testa originale è stata rimossa e sostituita con un blocco sequenziale identico a quello impiegato nella *FinishNet*. Questa sostituzione ha inoltre risolto le incompatibilità di ridimensionamento: l'utilizzo di un livello di *Adaptive Average Pooling* come interfaccia tra i filtri convoluzionali e i *layer* lineari ha permesso alla rete di processare agevolmente i tensori derivanti dal formato maggiorato di 512×512 (che producono in uscita mappe 16×16), mappandoli

dinamicamente sulla griglia 4×4 richiesta dal blocco di regressione. Il nuovo modulo prevede l'acquisizione delle *feature* estratte dal blocco residuo finale e il loro passaggio attraverso due strati lineari (da 1024 e 512 neuroni), intervallati da livelli di *Dropout* (p=0.5) per la regolarizzazione. Anche in questo caso, l'uscita è data dalla funzione *LogSoftmax* per generare il vettore logaritmico della *heatmap* a 512 elementi.

Per preservare la conoscenza pregressa dei primi filtri convoluzionali e adattare unicamente gli ultimi *layer* al nuovo dominio visivo, la fase di *fine-tuning* è stata condotta adottando un *Learning Rate* conservativo, ridotto di un ordine di grandezza (10^{-4}) rispetto all'addestramento della *FinishNet*. Scelta fondamentale per evitare passi di ottimizzazione troppo ampi che andrebbero a degradare irreparabilmente i pesi pre-addestrati. La funzione di costo è rimasta inalterata (*KLDivLoss*), garantendo una metrica di penalità identica per la successiva comparazione dei risultati.

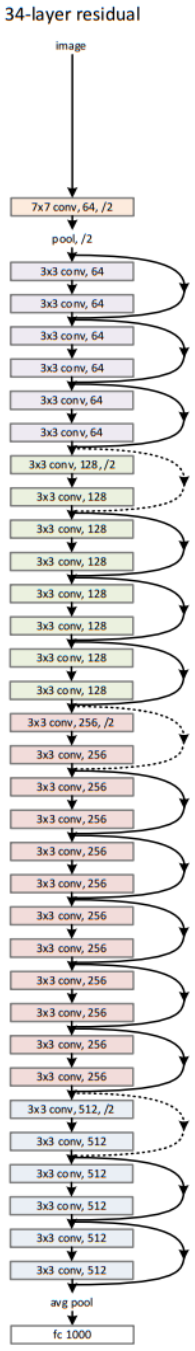


Figura 4.10: Schema raffigurante l'architettura della *ResNet-34*. Fonte: [37].

Capitolo 5

Risultati sperimentali e validazione

5.1 Setup sperimentale e metriche di errore

Al fine di condurre un confronto equo e oggettivo, sia l'architettura proposta (*FinishNet*) sia il modello di *ResNet* sono stati sottoposti al medesimo regime di addestramento. Sfruttando l'infrastruttura hardware descritta nel Capitolo 2, le reti sono state addestrate per 100 epoche con una *batch size* pari a 16. Durante la fase di *training*, le tecniche di *Data Augmentation* (Color Jittering e Random Erasing), precedentemente illustrate, sono state applicate *on-the-fly* per incrementare la varianza del *dataset* in tempo reale.

La scelta di una *batch size* contenuta rappresenta un duplice compromesso, ingegneristico e algoritmico. Sotto il profilo hardware, il valore 16 previene la saturazione della memoria grafica (*Out of Memory*), la quale deve ospitare simultaneamente i pesi del modello, il blocco di immagini corrente, le rispettive etichette e i gradienti necessari per la *backpropagation*. Sotto il profilo algoritmico *batch size* ridotte introducono un rumore stocastico durante l'aggiornamento dei gradienti. Questa stocasticità agisce come un regolarizzatore implicito: ostacola la convergenza verso minimi locali troppo netti in favore di minimi più ampi e piatti, traducendosi in una capacità di generalizzazione più robusta sui dati di *Test*.

L'ottimizzazione dei parametri è stata affidata all'algoritmo ADAM [41], combinato con uno *scheduler* di tipo *Cosine Annealing* [42] per la modulazione dinamica del *Learning Rate*. Rispetto agli ottimizzatori tradizionali, ADAM adatta il tasso di

apprendimento individualmente per ciascun parametro della rete, sfruttando le stime della media e della varianza dei gradienti passati. Questo meccanismo stabilizza l'aggiornamento dei pesi soggetti a gradienti ampi ed accelera quello dei parametri con gradienti modesti, garantendo una convergenza più rapida.

A supporto dell'ottimizzatore, lo *scheduler Cosine Annealing* gestisce il decadimento del *Learning Rate* seguendo un andamento cosinusoidale. Il vantaggio di questa strategia risiede nell'evoluzione della fase di esplorazione: nelle fasi iniziali, un tasso di apprendimento elevato consente al modello di esplorare ampiamente lo spazio delle soluzioni e di eludere eventuali minimi locali sub-ottimali; nelle epoche finali, l'appiattimento asintotico della curva verso lo zero garantisce un *fine-tuning* di precisione, permettendo alla rete di assestarsi esattamente nel minimo globale individuato.

Infine, per garantire l'assoluta riproducibilità statistica degli esperimenti, la suddivisione del *dataset* in *Training* (80%), *Validation* (10%) e *Test* (10%) è stata vincolata attraverso l'impostazione di un *seed* pseudo-casuale fisso.

Per quanto concerne la valutazione quantitativa delle prestazioni, è fondamentale distinguere la funzione obiettivo utilizzata per l'ottimizzazione dalla metrica fisica utilizzata per validare l'applicabilità del modello. Durante le fasi di addestramento e validazione, i modelli minimizzano la *KLDivLoss*, una misura statistica che valuta la discrepanza tra la distribuzione predetta dalla rete e la *Ground Truth* (modellata come una distribuzione gaussiana con $\sigma = 4.0$).

Tuttavia, ai fini pratici dell'automazione del photofinish, la metrica di interesse è l'errore di localizzazione spaziale misurato in pixel. In fase di *Test*, i tensori di output del modello (espressi in log-probabilità) vengono ricondotti a distribuzioni di probabilità standard tramite esponenziazione. Successivamente, la coordinata esatta della linea di rilevamento viene estratta applicando l'operatore matematico *Argmax* lungo l'asse spaziale, il quale individua l'indice del pixel associato al picco di massima confidenza.

Il medesimo operatore viene applicato alla maschera target reale per ottenere la coordinata di riferimento. L'errore di localizzazione per la singola immagine viene dunque calcolato come la distanza assoluta tra i due picchi:

$$\text{Errore} = |y_{\text{pred}} - y_{\text{target}}| \tag{5.1}$$

Le prestazioni finali sull'intero *Test Set* vengono aggregate e presentate sotto forma

di Errore Medio Assoluto (in pixel), affiancato dalla relativa deviazione standard. Questo approccio fornisce una misurazione tangibile dell'accuratezza del modello, traducendo le probabilità della rete neurale in una stima diretta della tolleranza d'errore sul campo di gara. L'applicazione uniforme di tale approccio valutativo a entrambe le architetture investigate fornisce un criterio di confronto rigoroso e oggettivo. Tali metriche consentono, in ultima istanza, di analizzare criticamente i risultati ottenuti e di decretare il modello ottimale da implementare all'interno del prototipo finale.

5.2 Analisi della convergenza (le curve di Loss)

L'analisi delle curve di apprendimento, tracciate monitorando la Kullback-Leibler Divergence durante le 100 epoche di addestramento, permette di evidenziare marcate differenze nelle dinamiche di convergenza tra l'architettura *ResNet* e il modello *FinishNet* proposto.

Osservando il grafico relativo alla *ResNet*, si nota una rapida discesa dell'errore nelle prime 30 epoche. Tale comportamento è imputabile all'elevata profondità della rete e alla presenza delle connessioni residue, le quali garantiscono una considerevole capacità rappresentativa. Tuttavia, a partire dall'epoca 40, emerge una progressiva divergenza tra le due curve: mentre la *Training Loss* continua a decrescere asintoticamente verso lo zero, la *Validation Loss* si stabilizza in un plateau. Questa dinamica è indicativa di un principio di *overfitting*: la complessità parametrica della *ResNet* porta il modello a memorizzare le peculiarità del set di addestramento, saturando la sua capacità di generalizzare su dati non osservati.

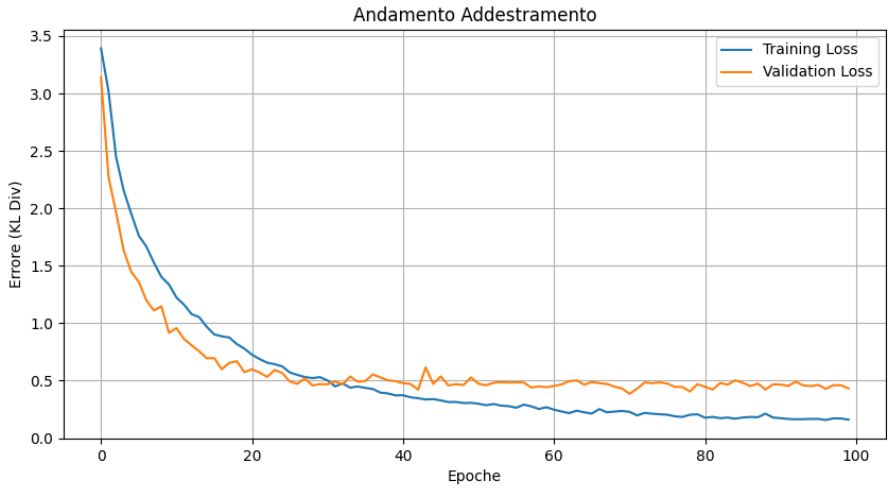


Figura 5.1: Grafico che mostra l'andamento dell'apprendimento durante la fase di *fine-tuning* della *ResNet*.

Di contro, l'andamento dell'architettura *FinishNet* (un primo modello senza i blocchi SE) si presenta significativamente più stabile. La versione base della rete mostra una convergenza più dolce e graduale, proporzionata alla sua natura più leggera. L'assenza di scostamenti critici tra le curve di *Training* e *Validation* conferma la robustezza del modello e l'efficacia delle tecniche di *Data Augmentation* nel prevenire la sovra-parametrizzazione. È opportuno notare che la *Validation Loss* si posiziona costantemente a valori inferiori rispetto alla *Training Loss*; tale fenomeno è giustificato dall'assenza di perturbazioni stocastiche (come Color Jittering e Random Erasing) sul set di validazione.

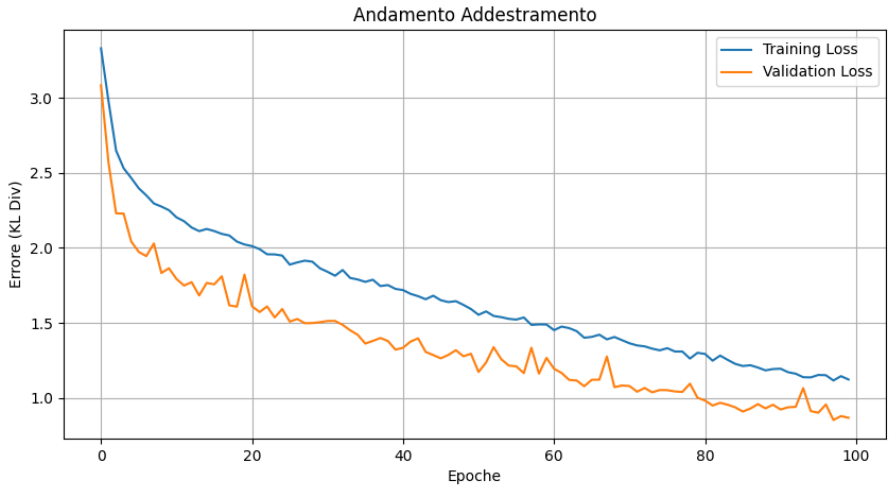


Figura 5.2: Grafico che mostra l'andamento dell'apprendimento durante la fase di addestramento della *FinishNet* senza i blocchi SE.

L'introduzione dei moduli di attenzione Squeeze-and-Excitation (SE) nella *FinishNet* apporta un evidente beneficio in termini di stabilità e capacità di apprendimento. Confrontando le due varianti *custom*, si evince che il valore finale della Validation Loss si assesta su livelli assimilabili; tuttavia, il modello dotato di blocchi SE mostra una curva di validazione nettamente più fluida e priva delle marcate oscillazioni visibili nella controparte di base. Inoltre, l'inclusione del modulo attenzionale abbatte significativamente la Training Loss, risolvendo una lieve tendenza all'*underfitting* sui dati di addestramento. Questo risultato empirico convalida l'ipotesi architettonica: il meccanismo di attenzione adattiva dei canali permette alla rete di filtrare efficacemente il rumore di fondo, garantendo una convergenza più stabile, coerente e robusta, senza sacrificare la leggerezza computazionale del modello focalizzandosi con maggiore precisione sulle feature salienti per la localizzazione spaziale della linea di traguardo.

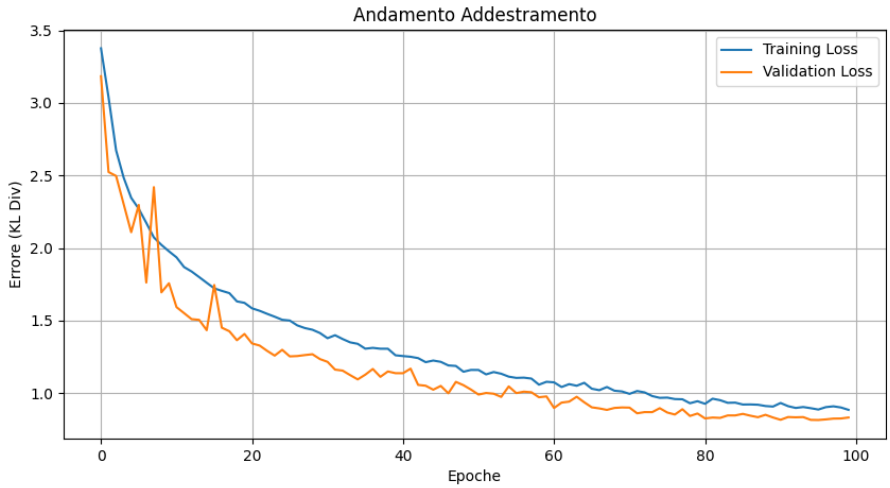


Figura 5.3: Grafico che mostra l'andamento dell'apprendimento durante la fase di addestramento della *FinishNet* con l'aggiunta dei blocchi SE.

L'estensione dell'addestramento oltre le 100 epoche è stata esclusa in quanto l'analisi empirica delle curve ha evidenziato il raggiungimento di una convergenza asintotica per la *FinishNet* e l'insorgenza di *overfitting* per la *ResNet*. Tale limite ha garantito un bilanciamento ottimale tra prestazioni raggiunte e costo computazionale.

5.3 Confronto prestazionale

Per valutare l'efficacia delle architetture neurali proposte, l'errore di localizzazione spaziale è stato confrontato con un riferimento di assoluto rilievo: le misurazioni manuali effettuate dai giudici umani. Questo confronto permette di quantificare non solo le differenze tra i modelli algoritmici, ma soprattutto il reale salto prestazionale offerto dall'automazione rispetto all'attuale standard operativo.

1. **Analisi statistica della varianza:** Il primo indicatore delle prestazioni è fornito dall'analisi della stabilità dell'errore assoluto, illustrata tramite boxplot. Il grafico evidenzia in modo inequivocabile la superiorità della *ResNet*: il modello presenta un rettangolo estremamente compresso verso lo zero, indicando che la stragrande maggioranza delle predizioni si discosta in maniera trascurabile

dal target reale. Al contrario, le misurazioni dei giudici umani rivelano una varianza decisamente più ampia e una mediana dell'errore superiore a quella della *ResNet*. La *FinishNet*, pur mostrando una varianza maggiore rispetto al modello profondo, mantiene un comportamento statistico paragonabile, seppur con un numero superiore di outlier, a quello dei giudici ufficiali.

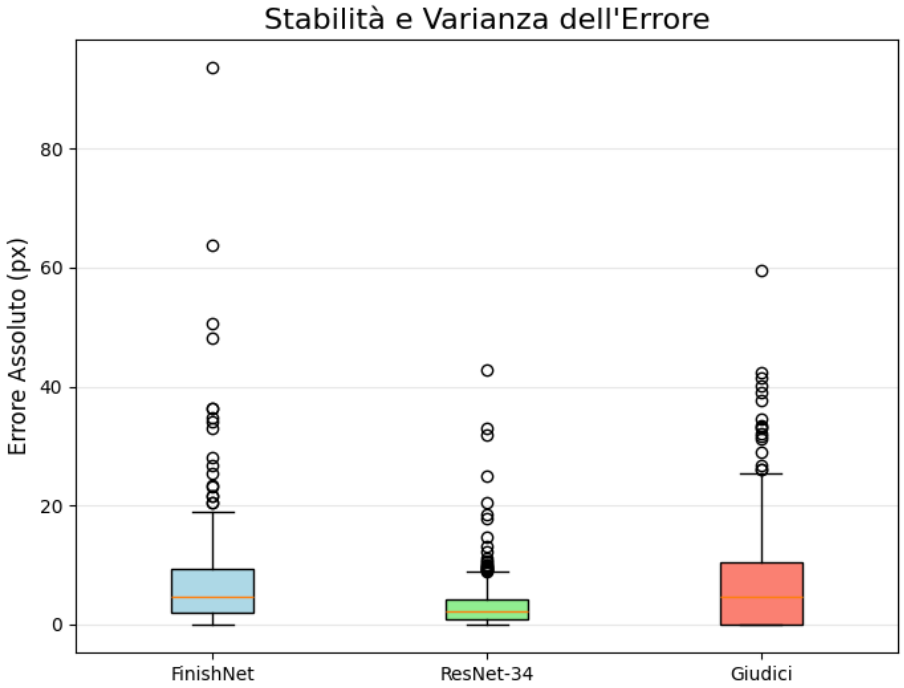


Figura 5.4: Grafico che mostra il confronto, dei due diversi modelli, dell'andamento della stabilità dell'errore assoluto e della varianza, in relazione alla qualità del lavoro svolto dai giudici di gara.

2. **Distribuzione cumulata dell'errore** La validazione quantitativa trova conferma nella curva CED (Cumulative Error Distribution), che traccia la percentuale di successo al variare della tolleranza in pixel. La curva rossa della *ResNet* domina chiaramente lo spazio: la sua impennata ripida dimostra che il modello raggiunge un'accuratezza quasi totale già con margini di errore ristrettissimi, attestandosi su un errore medio di 4.3 pixel. Incredibilmente, l'intelligenza artificiale profonda supera nettamente la precisione della misurazione manuale

umana, la quale si ferma a un errore medio di 7.0 pixel con una curva di convergenza più lenta. La *FinishNet* chiude il confronto con un errore medio di 8.4 pixel, un risultato di tutto rispetto per un'architettura più leggera.

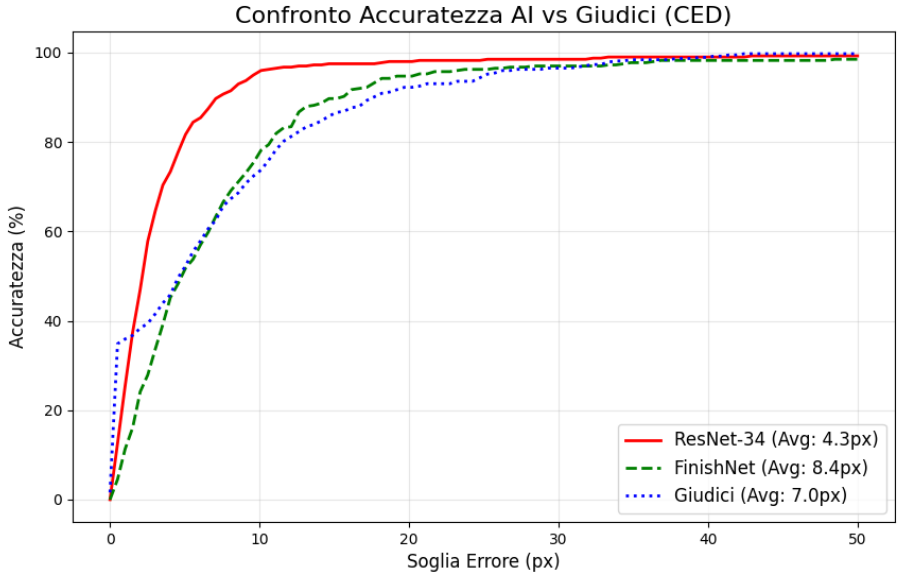


Figura 5.5: Grafico che mostra il confronto, dei due diversi modelli, la *ResNet* mostra un'accuratezza molto più elevata fin da subito.

In questo contesto, la scelta ingegneristica finale appare evidente. Sebbene la *FinishNet* sia leggera e veloce, la *ResNet* si posiziona nel quadrante operativo ideale: offre una velocità di elaborazione inferiore (siccome è caratterizzata da un numero molto più elevato di parametri), ma senza avere vincoli stringenti sulle velocità di analisi, e garantisce il minimo errore assoluto, infrangendo il tetto prestazionale umano, dovuto principalmente a distrazioni.

Nonostante le curve di Validation siano molto simili, le performance della FinishNet e della ResNet sono molto diverse. Durante le fasi di addestramento entrambe le reti imparano a generare curve gaussiane simili a quelle reali del Ground Truth, ad esempio: la FinishNet potrebbe generare una campana un po' più larga, prendendo comunque un buon punteggio di Loss, mentre la ResNet potrebbe generare una campana molto più stretta esattamente sul pixel corretto. Matematicamente la Loss calcolata potrebbe essere la stessa (0.5), ma quando si va ad estrarre il pixel

massimo con l'operazione di ArgMax per tirare la riga del photofinish, la ResNet sbaglia di pochissimi pixel, mentre la FinishNet oscilla di più. Siccome alla fine viene considerata solo la distanza assoluta tra i picchi delle gaussiane, questa metrica diversa da quella impiegata durante l'addestramento, influisce sulle prestazioni finali.

Pertanto, la *ResNet-34* si decreta come il motore inferenziale definitivo per il prototipo di photofinish automatizzato.

Alcuni casi in cui la rete *FinishNet* ha commesso più errori del dovuto:

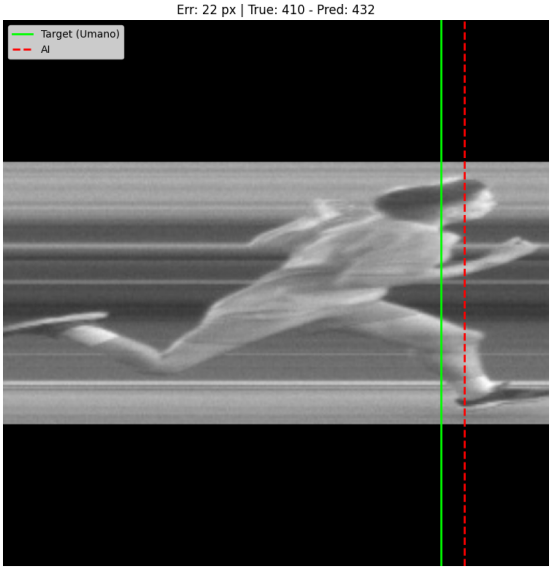


Figura 5.6: Immagine molto sfocata.

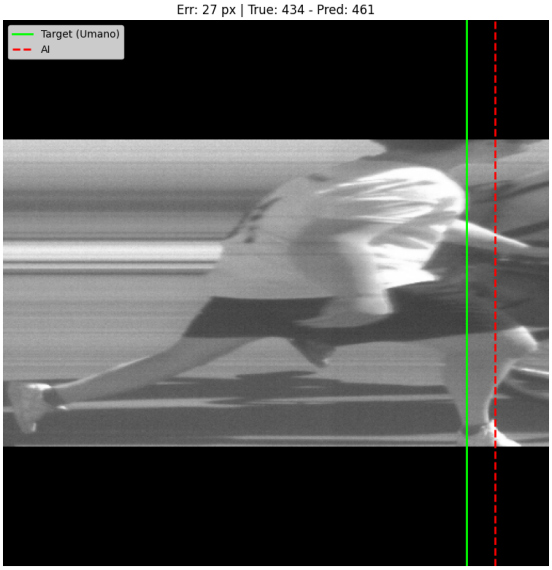


Figura 5.7: Si nota chiaramente il torso dell'atleta, ma subito dopo è presente molto rumore che probabilmente ha confuso la rete.

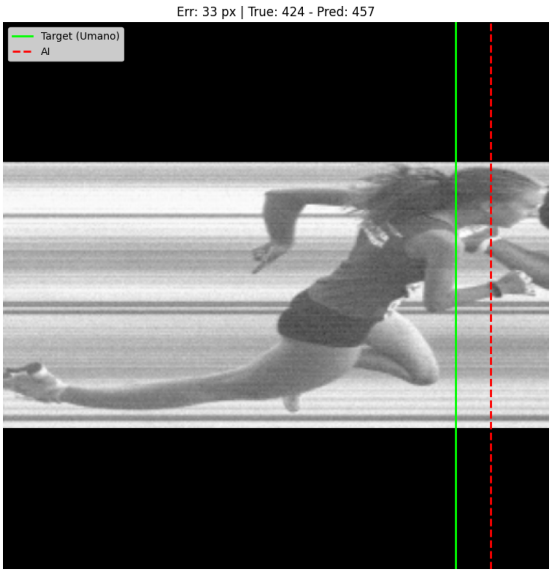


Figura 5.8: Probabilmente la rete è stata ingannata dagli arti: braccio dell'atleta in primo piano o dal piede sullo sfondo.

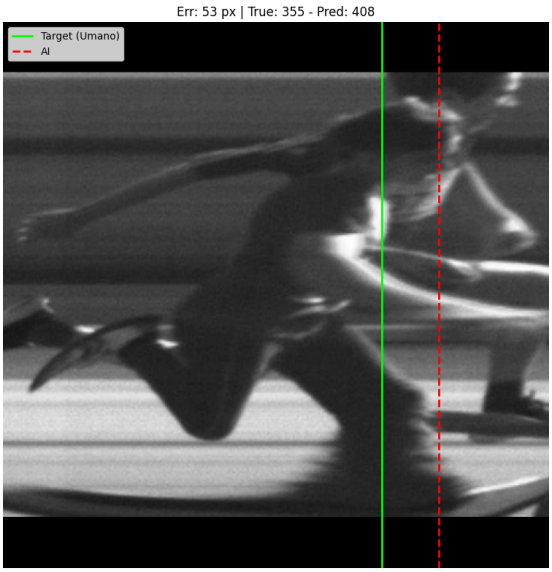


Figura 5.9: Immagine molto sfocata e con la presenza di più atleti nello stesso bounding box hanno ingannato la rete.

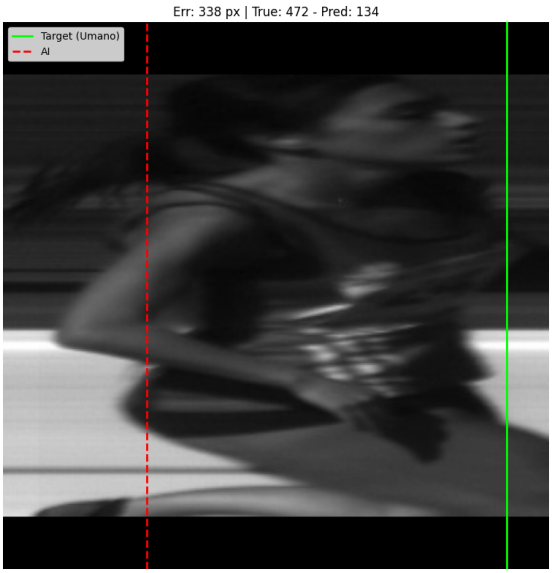


Figura 5.10: Immagine molto rumorosa con un basso contrasto rendendo quasi impossibile la rilevazione del torso.

Alcuni casi in cui la rete *FinishNet* ha performato correttamente:

Figura 5.11: Si nota un netto distacco tra la fine del torso e lo sfondo.

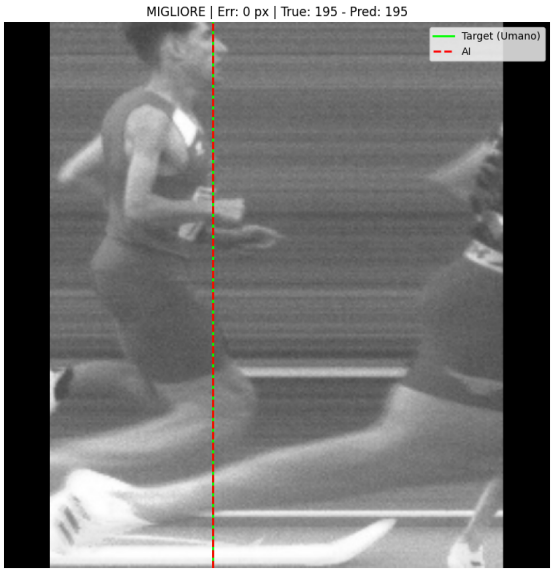


Figura 5.12: Sebbene il torso dell'atleta possa confondersi con lo sfondo, la rete neurale è riuscita a svolgere un ottimo lavoro.

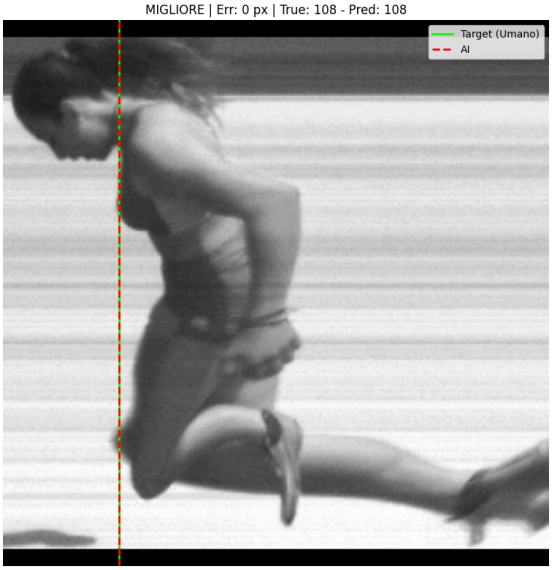


Figura 5.13: Si nota un netto distacco tra la fine del torso e lo sfondo.

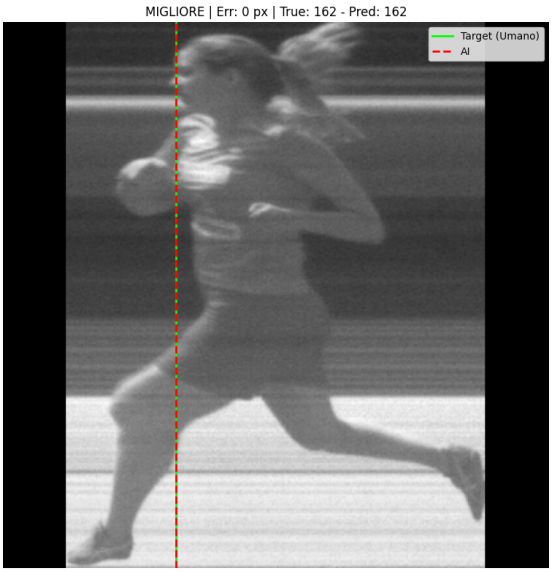


Figura 5.14: Sebbene l'atleta avesse il braccio proteso in avanti, la rete ha correttamente ignorato l'arto.

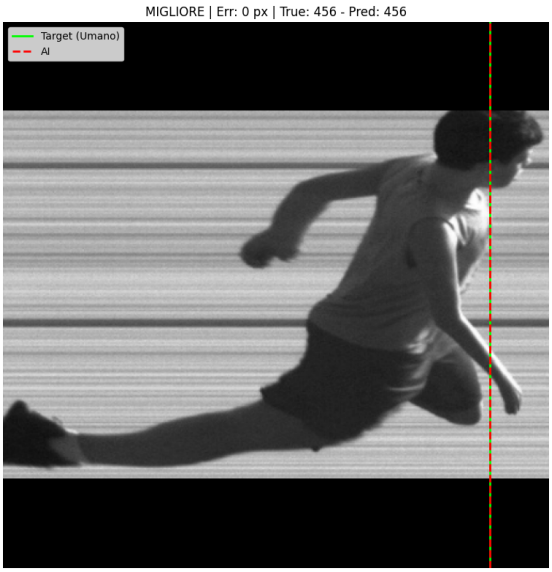


Figura 5.15: Si nota un netto distacco tra la fine del torso e lo sfondo.

Capitolo 6

Realizzazione del prototipo sperimentale e interfaccia utente

A valle della validazione algoritmica discussa nel capitolo precedente, è emersa la netta superiorità dell'architettura *ResNet-34* rispetto alla *FinishNet* in qualità di motore inferenziale. Le buone prestazioni ottenute in termini di precisione ne hanno determinato la selezione come nucleo centrale del prototipo, proponendola come sistema ufficiale di refertazione in unione con l'algoritmo *YOLO*, incaricato del rilevamento dei singoli atleti.

La fase conclusiva del presente lavoro di tesi prevede pertanto l'integrazione e la collaborazione di tutti i sotto moduli trattati in precedenza all'interno di un unico ecosistema. Questo si traduce nello sviluppo di un applicativo software completo, deputato alla gestione dell'intero flusso operativo: dall'inizializzazione della telecamera e relativa acquisizione dei *frame*, fino al salvataggio su disco delle immagini, all'elaborazione algoritmica e alla visualizzazione a schermo dei risultati della competizione.

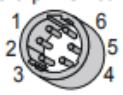
Questo capitolo illustra le fasi di ingegnerizzazione del prototipo finale, descrivendo la progettazione dei supporti meccanici realizzati in stampa 3D, l'integrazione della componentistica elettronica di controllo e, infine, lo sviluppo dell'interfaccia grafica (GUI) concepita per garantire un'interazione intuitiva ed efficiente da parte dei giudici di gara.

6.1 Integrazione elettrica ed elettronica

Il nucleo centrale del sistema di acquisizione è costituito dalla telecamera lineare Teledyne DALSA Spyder3 *GigE Vision* (modello SG-11-01k40-00-R). Come indicato nel manuale utente [17], l'alimentazione di questo dispositivo richiede un cablaggio specifico per interfacciarsi con il connettore industriale Hirose a 6 pin maschio (modello HR10A-7P-6S) [17].

Figure 9: Hirose 6-pin Circular Male—Power Connector

Hirose 6-pin Circular Male



Mating Part: HIROSE HR10A-7P-6S

Table 4: Hirose Pin Description

Pin	Description	Pin	Description
1	Min +12 to Max +15V	4	GND
2	Min +12 to Max +15V	5	GND
3	Min +12 to Max +15V	6	GND

Figura 6.1: Schema elettrico dell'ingresso per l'alimentazione della telecamera Dalsa. Fonte: [17].

Sulla base delle specifiche elettriche della telecamera, che prevedono una tensione di ingresso compresa tra +12V e +15V e un assorbimento di potenza inferiore a 9 W [17], è stato progettato e assemblato un sistema di alimentazione *custom*. Questo circuito accoppia un alimentatore standard (con output di +12V e 2A) a un cavo terminato con un connettore Hirose a 6 pin femmina.

L'interfacciamento tra i due elementi è stato gestito tramite un circuito intermedio realizzato su una scheda millefori. All'ingresso di questa, un connettore riceve l'alimentazione standard e separa le linee di tensione, sulla linea positiva è stato cablato in serie un fusibile di protezione da 1.25A, dimensionato per salvaguardare la telecamera in caso di sovratensioni o malfunzionamenti dell'alimentatore. In uscita, il circuito termina con una morsettiera a cui è stato saldamente fissato il cavo di alimentazione Hirose.

Al fine di isolare i componenti elettrici, scongiurare il rischio di cortocircuiti e proteggere il circuito da polvere e urti meccanici, l'intera basetta millefori è stata inserita in un *case* protettivo su misura. La scatola, comprensiva di un coperchio con chiusura a incastro, è stata interamente progettata al computer, tramite software Blender (versione 3.6) [43], e realizzata tramite stampa 3D.

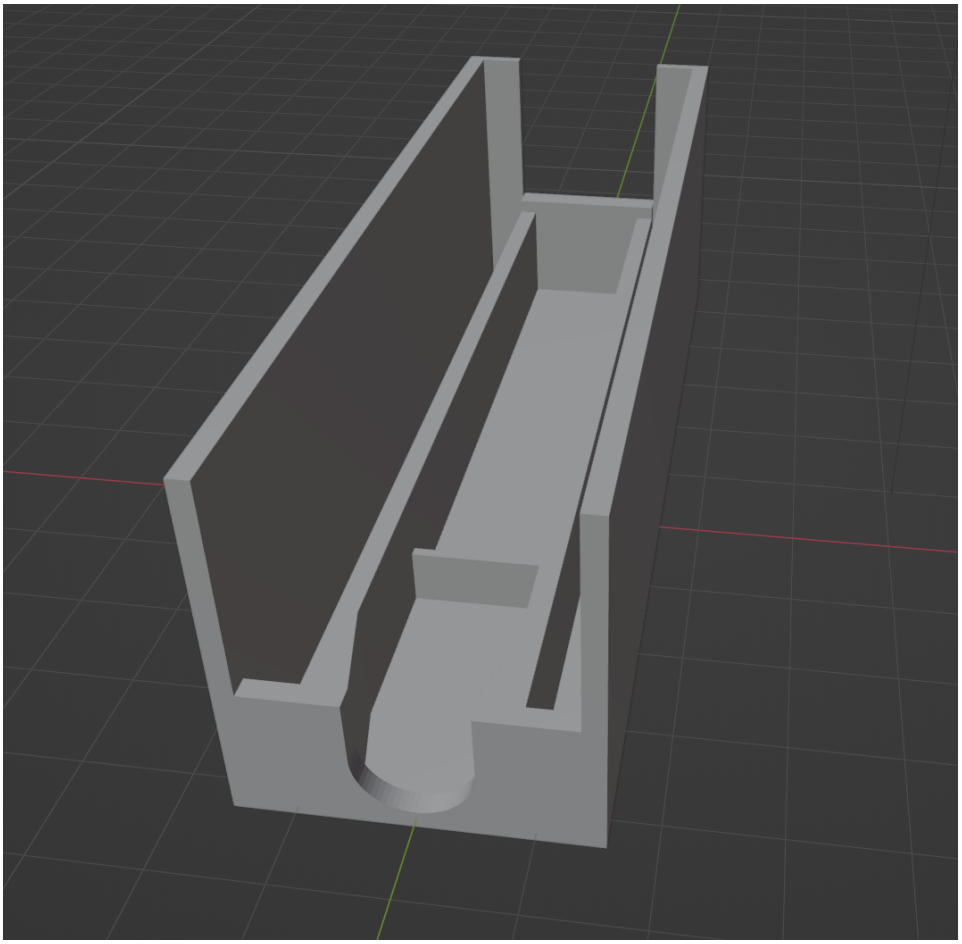


Figura 6.2: Progetto 3D della base del *case* per l'alloggiamento della millefori e del circuito di alimentazione.

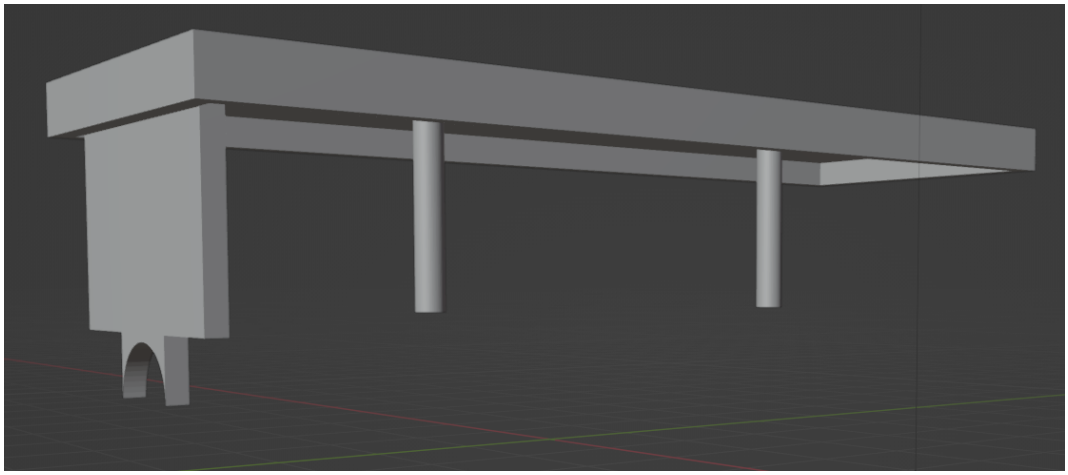


Figura 6.3: Progetto 3D del coperchio a incastro per la chiusura del modulo di alimentazione.

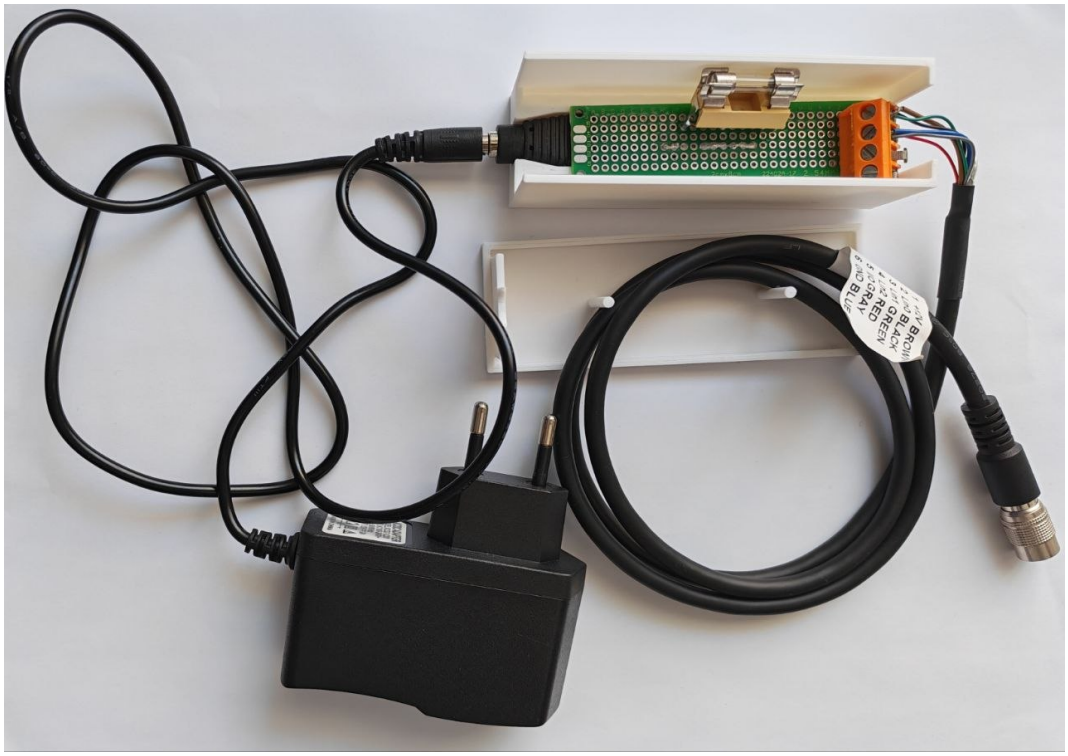


Figura 6.4: Immagine raffigurante il cavo di alimentazione completo e assemblato per il funzionamento della telecamera Dalsa.

Risolto il vincolo fondamentale dell'alimentazione, il secondo aspetto cruciale dell'integrazione hardware riguarda la gestione dei segnali fisici. A livello circuitale, il sistema di *trigger* è basato su un pulsante elettromeccanico, azionato manualmente dall'operatore e interfacciato con una scheda a microcontrollore Arduino Uno.

Il circuito è stato realizzato configurando uno dei pin digitali della scheda come ingresso di lettura. Per garantire una rilevazione logica stabile ed evitare fluttuazioni di tensione indesiderate (effetto floating) quando il pulsante è rilasciato, il contatto è stato cablato avvalendosi di una resistenza di pull-down. In questa configurazione elettrica, il pin digitale viene mantenuto allo stato logico basso (0 V) attraverso la resistenza collegata al potenziale di riferimento (GND). Nel momento in cui l'operatore chiude il circuito premendo il pulsante, la tensione di alimentazione (5 V) viene applicata in modo diretto al pin, portandolo istantaneamente allo stato logico alto. Come descritto in precedenza nella sezione relativa alla comunicazione seriale, le fisiologiche micro-oscillazioni meccaniche dei contatti del pulsante vengono poi filtrate in modo robusto via software, tramite un algoritmo di *debouncing* temporale tarato su una finestra di 50 millisecondi.

L'acquisizione di questo segnale logico alto innesca la logica di controllo di Arduino, che provvede a inviare tramite porta seriale le istruzioni al software principale, determinando l'avvio o l'interruzione del salvataggio dei *frame* provenienti dalla telecamera.

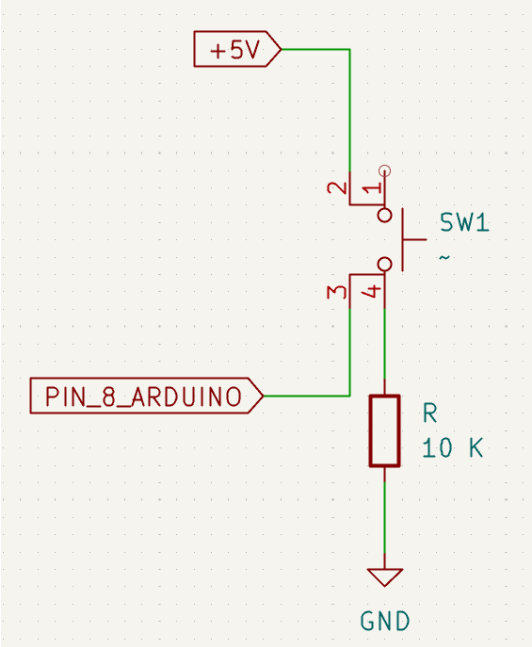


Figura 6.5: Schema elettrico raffigurante il circuito di controllo del pulsante di *trigger*. Fonte: [44].

Considerate le dimensioni ridotte del componente elettromeccanico (circa 6 mm), il suo utilizzo diretto su breadboard durante i test operativi sul campo di atletica sarebbe risultato poco ergonomico e soggetto a disconnessioni accidentali. Per ovviare a questo problema pratico, è stato progettato e stampato in 3D un apposito supporto ergonomico.

Questo *case* permette di alloggiare saldamente il pulsante, agevolandone l'impugnatura e garantendo un azionamento tempestivo e confortevole da parte dell'operatore durante le fasi di cronometraggio.

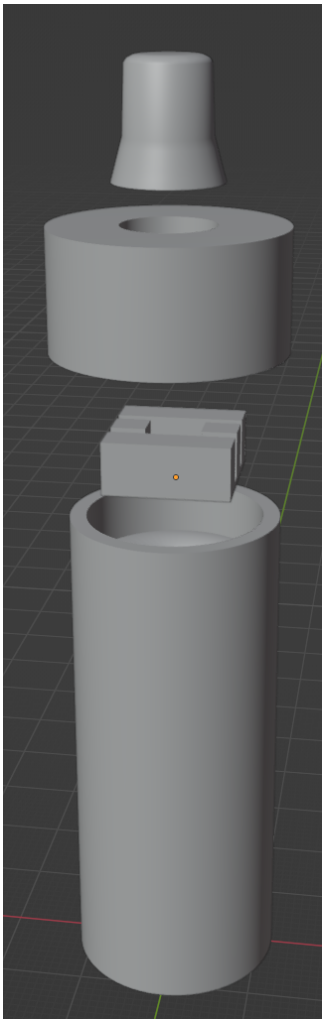


Figura 6.6: Progetto 3D del supporto ergonomico per il pulsante di *trigger*.

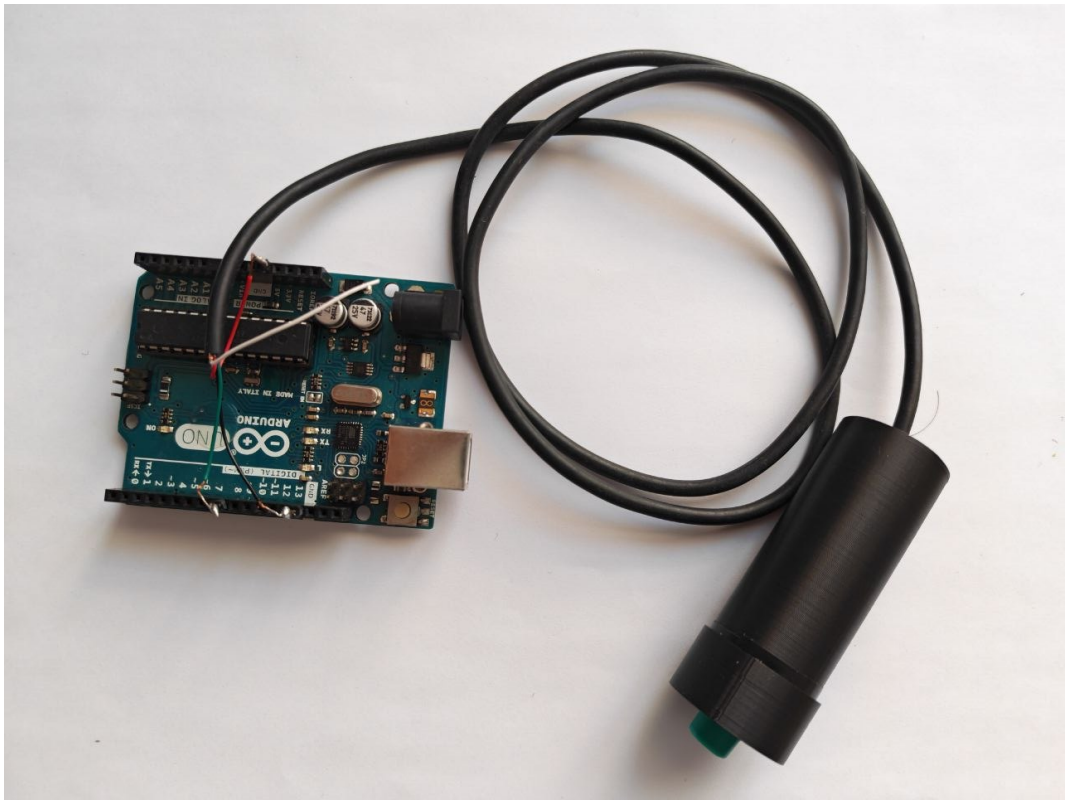


Figura 6.7: Immagine raffigurante il sistema di acquisizione completo per la gestione del *trigger* tramite la scheda Arduino Uno.

6.2 Progettazione meccanica e assemblaggio hardware

Successivamente alla messa in funzione del dispositivo e del sistema di *trigger*, si è resa necessaria un'analisi accurata del posizionamento della telecamera a bordo pista, fondamentale per inquadrare correttamente le corsie e il transito degli atleti. Al fine di mitigare il problema dell'occlusione visiva, ovvero evitare che gli atleti più vicini all'obiettivo coprano quelli nelle corsie esterne, le linee guida internazionali [7] prescrivono un'installazione sopraelevata su supporti esenti da vibrazioni. Tale configurazione deve garantire una visuale libera sul torso e sul pettorale di ciascun corridore. Tuttavia, la normativa non impone parametri metrici assoluti relativi a

distanza, altezza e inclinazione dell'ottica: la posizione di ripresa deve infatti essere calcolata e adattata dinamicamente in funzione degli specifici vincoli architettonici dell'impianto sportivo, variando l'angolo di incidenza in base alla reale distanza del sensore dal traguardo.

Trattandosi di un sensore di visione industriale, la Teledyne DALSA Spyder3 è sprovvista del foro filettato (solitamente da 1/4") comunemente presente nelle fotocamere commerciali per il fissaggio su cavalletto. Per ovviare a questa incompatibilità meccanica, è stata progettata al CAD e successivamente stampata in 3D un'apposita staffa di adattamento. Il design del componente è stato ricavato dalle specifiche dimensionali riportate nel manuale utente [17].

Questo supporto prevede, nella parte inferiore, un inserto compatibile con l'attacco a vite standard dei treppiedi; nella parte superiore, presenta invece una serie di fori passanti allineati con le sedi filettate presenti sullo chassis della telecamera, garantendo così un ancoraggio rigido e sicuro tramite viti.

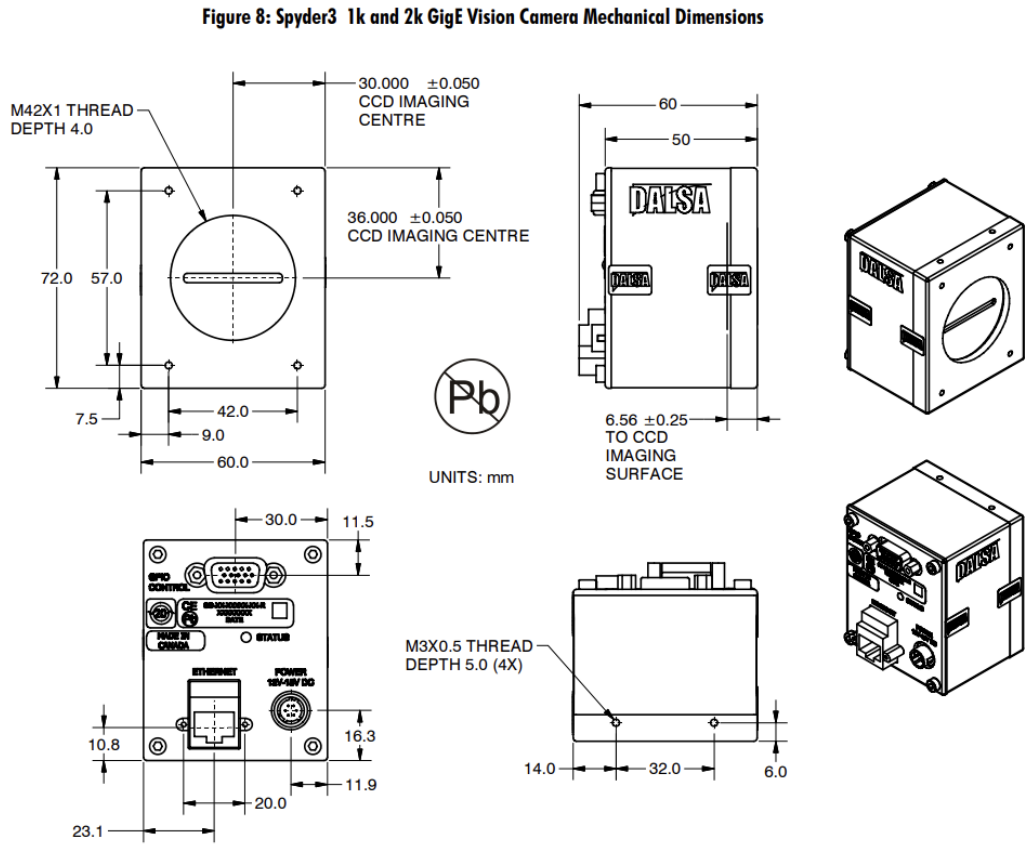


Figura 6.8: Schema dell'interfaccia meccanica della telecamera Dalsa Spyder 3. Fonte: [17].

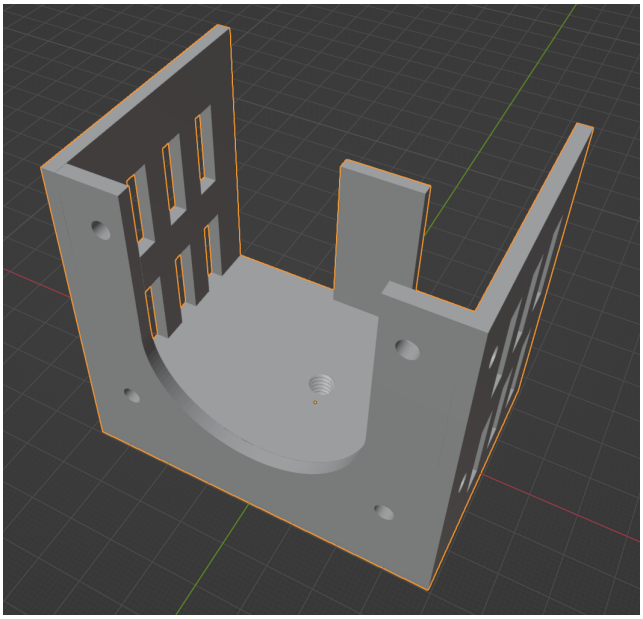


Figura 6.9: Progetto 3D della staffa di adattamento per il montaggio della telecamera su treppiede.

Tramite questo adattatore, il sistema di acquisizione è stato installato su uno stativo a giraffa, grazie al quale si è raggiunta un'altezza operativa di circa 2 metri. Sebbene tale quota non sia abbastanza per soddisfare tutte le richieste per le competizioni ufficiali, si è rivelata un compromesso adeguato e sufficiente per le finalità di test del prototipo.

L'intero apparato è stato infine trasportato, assemblato e validato sul campo presso la pista di atletica del centro sportivo Natta di Rivoli, grazie alla collaborazione della dirigenza della società Atletica Rivoli.

Figura 6.10: Setup sperimentale del sistema installato a bordo pista presso il centro sportivo Natta di Rivoli.

6.3 Sviluppo dell'interfaccia grafica e usabilità

Affinché il giudice di gara possa operare in modo efficace, è essenziale disporre di un sistema software che integri e coordini tutte le componenti hardware descritte nei paragrafi precedenti. A tale scopo è stato sviluppato il software, denominato: PhotoFinish PRO, dotato di un'interfaccia grafica (GUI) concepita per fornire all'operatore uno strumento di controllo intuitivo e a bassa latenza.

Considerata l'elevatissima quantità di dati generati dalla telecamera lineare ad alto framerate, si è scelto di non impiegare framework grafici di alto livello. L'intera interfaccia è stata invece sviluppata sfruttando le primitive grafiche di base della libreria OpenCV [45] (linee, testi, rettangoli), disegnate direttamente sullo schermata. Questa soluzione architetturale garantisce un *overhead* computazionale minimo, evitando di saturare le capacità di elaborazione del PC impiegato.

All'avvio del programma, il sistema esegue un check diagnostico per verificare la corretta connessione della telecamera e del sistema di *trigger*. A seguito di un riscontro positivo, compare una schermata di standby che invita l'operatore ad avviare fisicamente l'acquisizione tramite la pressione del tasto S.

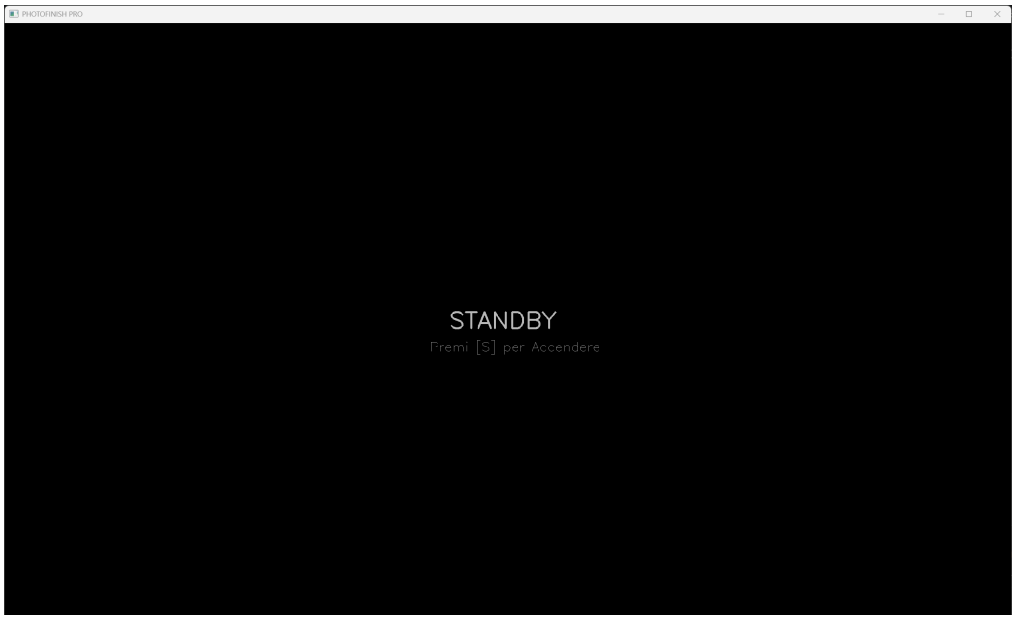


Figura 6.11: Schermata iniziale (Standby) visualizzata a seguito del controllo positivo della connessione hardware (telecamera e *trigger*).

Una volta confermato l'avvio, l'operatore visualizza la *dashboard* di acquisizione, suddivisa in un menu di stato superiore e un'area di visualizzazione inferiore. Per minimizzare il rischio di errori umani durante le competizioni, il sistema è governato da una logica a stati finiti, controllabile interamente tramite scorciatoie da tastiera. Gli stati operativi di acquisizione principali sono due:

1. **Modalità Live (Free Run):** Stato attivato di default all'avvio. La telecamera invia e visualizza a schermo un flusso video continuo in tempo reale senza salvare nessun dato. Questa fase è fondamentale per i giudici: permette di eseguire la calibrazione ottica, regolare la messa a fuoco e, soprattutto, allineare con precisione millimetrica l'obiettivo fotografico con la linea del traguardo fisica posta sulla pista;

Figura 6.12: Interfaccia in modalità LIVE. L'operatore utilizza il flusso di immagini continuo per il puntamento meccanico e la calibrazione ottica.

2. **Modalità Gara:** Terminato il posizionamento, la pressione del tasto TAB fa transitare il sistema nello stato di "Gara Pronta". Il flusso video in tempo reale viene oscurato per liberare risorse di calcolo. Il sistema si pone in ascolto, attendendo l'input di inizio competizione (pressione del tasto INVIO).

Figura 6.13: Interfaccia in modalità Gara (Standby). Il flusso video è oscurato in attesa dello start.

Inviato il comando di inizio gara, il cronometro interno si avvia. La gestione del tempo non è affidata all'orologio del sistema operativo, bensì ai *Tick* hardware (i *timestamp* nativi) generati dal sensore della telecamera ad ogni linea acquisita. Salvando il *timestamp* di partenza, il software calcola per differenza il tempo trascorso. La conversione in secondi reali avviene dividendo tale delta per la frequenza nominale della telecamera impostata in quel momento. Il tempo ricavato, in scorrimento, viene costantemente aggiornato nel menu superiore.

In questa fase, quando gli atleti si approssimano al traguardo, il giudice chiude il contatto del *trigger* fisico (pulsante gestito da Arduino). Fintanto che il pulsante è premuto, il software acquisisce, salva in memoria e mostra a schermo i fotogrammi validi della competizione.

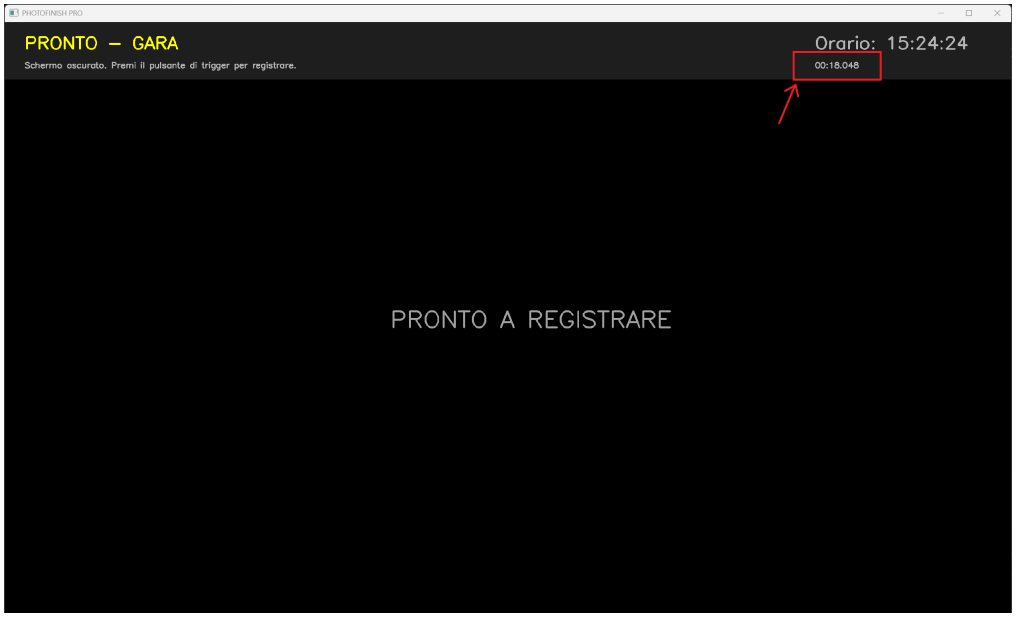


Figura 6.14: Schermata Gara Avviata: il cronometro è attivo e il sistema attende il segnale del *trigger* hardware.

Figura 6.15: Schermata Acquisizione (REC): alla pressione del *trigger*, i *frame* vengono registrati in memoria e visualizzati a schermo.

Conclusa la gara e terminata l'acquisizione (premendo nuovamente INVIO), l'immagine composita finale viene processata in *background* dall'Intelligenza Artificiale e passata automaticamente all'ambiente di revisione.

Questo ambiente è strutturato per massimizzare la leggibilità dei risultati:

- **Navigazione:** Essendo l'immagine finale del photofinish estremamente estesa in orizzontale, la visualizzazione in scala 1:1 è garantita da una barra di scorrimento superiore (*Trackbar*);
- **Filtro corsie:** Sulla sinistra è posizionato un indicatore rosso interattivo trascinabile. Serve a indicare all'algoritmo da quale quota orizzontale in giù deve iniziare a calcolare la presenza delle corsie della pista, filtrando eventuali rumori di fondo (es. tribuna o giudici) dovuti alla natura intrinseca delle telecamere *line-scan* che trasformano ogni punto fermo in linee orizzontali;
- **Timeline temporale:** Nella porzione inferiore, il programma renderizza proceduralmente una *timeline* graduata in centesimi di secondo. L'asse temporale è coerente con la cinematica della gara: supponendo la corsa da sinistra verso destra, i tempi inferiori (primi arrivati) si trovano a destra dell'immagine, aumentando progressivamente scorrendo verso sinistra;
- **Classifica automatica:** Al di sotto della *timeline*, viene stampato a schermo l'ordine di arrivo consigliato dalla rete neurale, completo di tempo finale e assegnazione stimata della corsia.

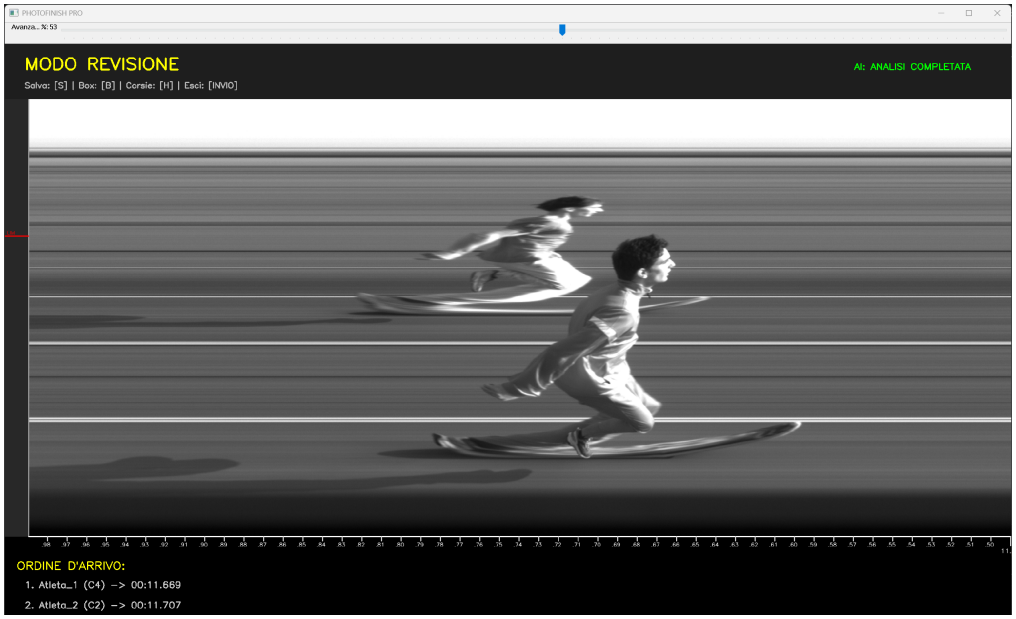


Figura 6.16: Ambiente di Revisione: panoramica dell'interfaccia con la proposta di classifica generata in automatico dall'Intelligenza Artificiale.

Per fornire totale trasparenza al giudice di gara, tramite le scorciatoie B e H è possibile sovrapporre all'immagine le metriche visive che hanno portato l'algoritmo alla decisione finale.

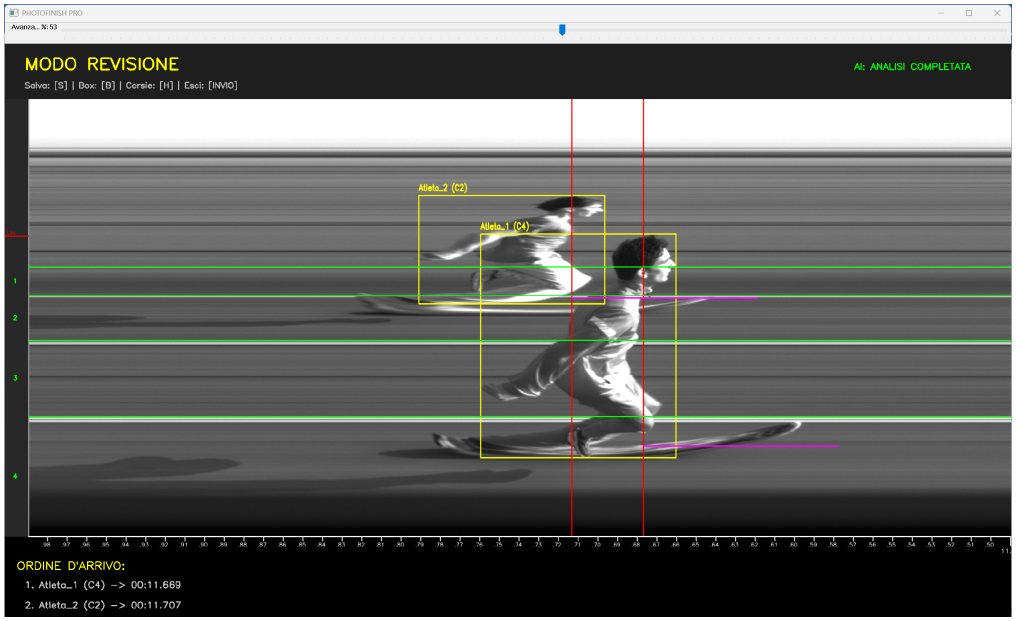


Figura 6.17: Visualizzazione dei *layer* di analisi dell'AI. In giallo i bounding box (*YOLO*), in rosso le linee di rilevamento del torso, corrispondenti alle linee di photofinish (*ResNet*), in verde l'estrazione delle corsie e in viola le linee di contatto piede-pista per l'assegnazione della corsia.

Infine, il software gestisce in modo efficiente le discrepanze temporali tra gli arrivi. Nelle gare prolungate (es. mezzofondo), o quando vi è molto distacco tra gli atleti, il giudice può rilasciare il *trigger* tra un arrivo e l'altro. Questa operazione interrompe la registrazione dei *frame* vuoti, impedendo la saturazione della memoria RAM. Nell'immagine di revisione finale, tali pause vengono rappresentate graficamente tramite apposite linee verticali di interruzione sulla *timeline*, garantendo comunque l'assoluta precisione cronometrica dei blocchi successivi.

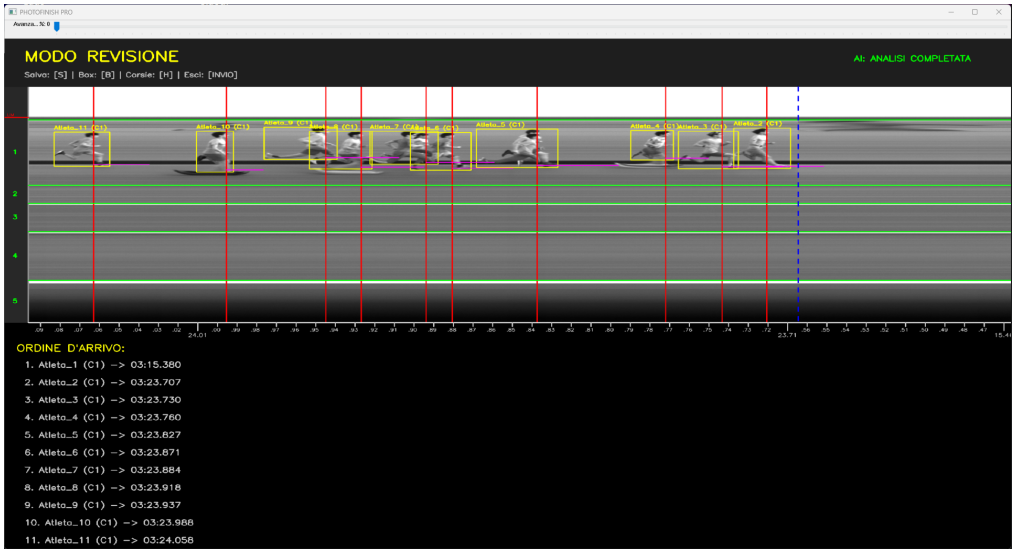


Figura 6.18: Analisi di una corsa di mezzofondo. Si evince la gestione discontinua della memoria tramite *trigger*: il gap temporale tra due acquisizioni è reso esplicito graficamente sulla *timeline* senza alterare la sincronizzazione del cronometro tramite linea tratteggiata blu.

Capitolo 7

Conclusioni e Sviluppi Futuri

In questo lavoro di tesi è stato affrontato il complesso problema dell'automazione della refertazione nel cronometraggio sportivo, proponendo un innovativo sistema di Intelligenza Artificiale basato su architetture neurali profonde per l'analisi di immagini da telecamere lineari (photofinish). L'obiettivo primario di sviluppare un DSS capace non solo di abbattere i tempi di elaborazione, ma di eguagliare e superare la precisione umana, è stato pienamente raggiunto attraverso la realizzazione del prototipo (denominato PhotoFinish PRO) descritto nel capitolo precedente.

7.1 Sintesi dei Risultati

Il presente lavoro di tesi ha dimostrato con successo la fattibilità e l'efficacia del prototipo, validando un sistema automatizzato in grado di superare le criticità geometriche del photofinish attraverso la sinergia tra hardware industriale e algoritmi di intelligenza artificiale. Dal punto di vista algoritmico, la pipeline a due stadi ha impiegato *YOLOv8* con logica a finestra scorrevole per garantire una localizzazione degli atleti nettamente più robusta rispetto alle sole tecniche di *Pose Estimation*.

Tale approccio permette di superare i limiti storici dei metodi di *Computer Vision* tradizionali, i quali si affidavano a euristiche rigide e proporzioni geometriche fisse per l'estrazione del torso o la rimozione delle braccia, risultando spesso inefficaci di fronte a pose anomale o deformazioni estreme [46]. Affiancando a *YOLO* una *Heatmap Regression* (modellata tramite distribuzione Gaussiana monodimensionale),

la rete individua il torso in modo flessibile e stabile. Preservando la risoluzione temporale tramite un *padding* conservativo dei ritagli, la *ResNet-34* si è affermata come l'architettura inferenziale ottima: con un errore medio assoluto di soli 4.3 pixel, il sistema ha infranto il tetto prestazionale dei giudici di gara (7.0 pixel) operando in pochi secondi, mentre lo sviluppo parallelo di una *FinishNet* dotata di moduli di attenzione spaziale (SE Block) ha fornito una solida alternativa *lightweight*.

Tale successo teorico si è concretizzato attraverso l'interfacciamento hardware di una telecamera lineare Teledyne DALSA Spyder 3, supportata da circuiti di alimentazione dedicati, supporti meccanici in stampa 3D e un *trigger* fisico basato su Arduino per l'ottimizzazione della memoria.

Sotto il profilo software, la realizzazione di una GUI proprietaria ad alte prestazioni basata su OpenCV ha permesso di gestire fluidamente l'elevato flusso dati del protocollo *GigE Vision*; parallelamente, la sincronizzazione è stata garantita ancorando il tempo di gara ai *timestamp* nativi dell'hardware anziché all'orologio del sistema operativo.

L'unione dei modelli di *Deep Learning* e algoritmi di estrazione automatica delle corsie ha reso possibile la generazione istantanea degli ordini di arrivo mostrati tramite interfacce visive trasparenti per agevolare il controllo e la validazione finale da parte dell'operatore umano.

7.2 Limiti del Sistema Proposto

Nonostante i promettenti risultati, il sistema presenta alcune limitazioni intrinseche legate sia al dominio dei dati che alle scelte architetturali e infrastrutturali:

- **Occlusioni totali e pose anomale:** Sebbene la tecnica del Random Erasing abbia reso la rete robusta alle occlusioni parziali, il sistema mostra vulnerabilità nei casi di sovrapposizione perfetta tra due atleti sulla linea del traguardo. Inoltre, sebbene il *Deep Learning* abbia mitigato drasticamente i falsi rilevamenti causati da atleti che tagliano il traguardo sporgendo la spalla anziché il petto, una criticità nota e documentata nei sistemi di rilevamento automatico classici [46], la rete fatica ancora a separare le distribuzioni di probabilità in caso di occlusione totale del torso e tendendo a collassare la rilevazione su un'unica predizione nel caso di crop con più persone insieme;

- **Collo di bottiglia del primo stadio:** La scansione tramite *sliding window* su immagini native di dimensioni estreme (es. 10000×1000 pixel) richiede inferenze multiple e operazioni di blending. Pur essendo state ottimizzate, tali procedure rappresentano il limite computazionale principale dell'intera pipeline per un'esecuzione in vero real-time;

- **Dipendenza hardware del secondo stadio:** La scelta della *ResNet-34* come motore inferenziale primario comporta un costo computazionale elevato. Il volume di parametri del modello rende necessaria la presenza di un acceleratore grafico (GPU) dedicato, precludendone l'uso su macchine di fascia bassa. Le architetture più snelle testate (come la *FinishNet*), d'altro canto, mostrano limiti rappresentativi fisici, non riuscendo a eguagliare la precisione millimetrica richiesta;

- **Sincronizzazione manuale del tempo assoluto:** Allo stato attuale, l'avvio del cronometro è vincolato all'interazione umana (pressione del tasto INVIO). Questo meccanismo introduce il tempo di reazione fisiologico dell'operatore rispetto al reale colpo di pistola dello starter, compromettendo la certificabilità del tempo assoluto in contesti di gara ufficiali;

- **Sensibilità alle condizioni ambientali:** Gli algoritmi preposti al rilevamento delle corsie e all'estrazione delle *feature* risentono fortemente delle condizioni di illuminazione. Come evidenziato storicamente anche in letteratura per le tecniche non basate sull'apprendimento automatico [46], il rumore visivo intrinseco delle telecamere *line-scan* si manifesta frequentemente sotto forma di artefatti e strisce orizzontali in caso di illuminazione non ottimale. Di conseguenza, piste con linee sbiadite o assenza di fari compensativi continuano a ridurre sensibilmente la confidenza dei modelli visivi.

- **Acquisizione Monocromatica:** L'attuale architettura hardware si avvale di una telecamera lineare ad alta frequenza che opera esclusivamente in scala di grigi, la totale assenza di informazioni cromatiche limita il potenziale inferenziale della rete neurale. La mancanza dei canali colore (RGB) riduce drasticamente il numero di feature a disposizione dei modelli convoluzionali, rendendo particolarmente complessa la distinzione degli atleti in scenari di sovrapposizione o occlusione parziale, laddove il contrasto cromatico tra le divise di gara avrebbe facilitato la separazione visiva delle figure.

7.3 Sviluppi Futuri

Le limitazioni emerse nel presente studio delineano chiare traiettorie di ricerca per le future iterazioni del sistema, orientate a perfezionare sia l'architettura algoritmica che l'infrastruttura hardware del prototipo:

- **Pre-elaborazione con Background subtraction:** Prendendo spunto dalle tecniche classiche di *Computer Vision*, un'evoluzione del sistema potrebbe integrare algoritmi di *background subtraction* auto-adattivi, specificamente calibrati sulle particolarità delle acquisizioni *line-scan*. Rimuovere il rumore di fondo, che tipicamente si presenta sotto forma di strisce orizzontali, prima del passaggio ai modelli di *Deep Learning* alleggerirebbe il carico dei moduli di attenzione spaziale. Questo approccio migliorerebbe sensibilmente la robustezza della *Heatmap Regression* in condizioni di visibilità non ottimali [46];

- **Evoluzione del dataset e acquisizione RGB:** Per incrementare la capacità di generalizzazione del modello e per ottenere risultati più di alto livello, sarà fondamentale espandere e diversificare il *dataset* di addestramento. In questa ottica, la transizione verso sensori di acquisizione a colori (RGB) garantirebbe alla rete una maggiore ricchezza di *feature* cromatiche e spaziali. Parallelamente, l'adozione di annotazioni a livello di singolo pixel (*Instance Segmentation*) al posto delle attuali bounding box consentirebbe un apprendimento più granulare della morfologia umana, rivelandosi cruciale per la gestione delle occlusioni severe e riuscendo così a refertare anche più atleti insieme piuttosto che mischiare i risultati in un'unica predizione;

- **Miglioramento architetturale:** Per incrementare le prestazioni della Finish-Net, un approccio riguarda la sua evoluzione topologica, espandendo in modo mirato sia la profondità dei layer sia la larghezza dei filtri convoluzionali per aumentarne la capacità rappresentativa. Contestualmente, gli attuali moduli SE potranno essere evoluti integrando meccanismi di attenzione in grado di pesare l'importanza delle feature non solo sui canali ma anche a livello spaziale, aiutando il modello a focalizzarsi con una precisione più elevata sull'anatomia dell'atleta e mitigando ulteriormente l'incidenza degli outliers.

- **Sincronizzazione hardware dello Start e Sensor Fusion:** Per superare l'attuale dipendenza dall'avvio manuale del cronometro, la priorità strutturale

risiede nell'acquisizione automatizzata del segnale di partenza. L'interfacciamento diretto del microcontrollore Arduino con la pistola elettronica dello starter, mediante cablaggio dedicato o protocolli radio a bassissima latenza, permetterà di annullare l'incertezza introdotta dal tempo di reazione umano, assicurando il calcolo di un tempo assoluto rigoroso e idoneo all'omologazione nelle competizioni ufficiali. In un'ottica più ampia, tale integrazione aprirà la strada ad architetture di Sensor Fusion: incrociando le predizioni visive dei modelli di *Deep Learning* con i flussi dati dei tradizionali dispositivi di telemetria (come transponder RFID e sensori di pressione sui blocchi di partenza), sarà possibile realizzare un ecosistema di refertazione e di analisi, altamente ridondante e intrinsecamente robusto contro qualsiasi anomalia di misurazione;

- **Approccio Human-in-the-Loop:** Coerentemente con la natura di un DSS, l'automazione non deve escludere la supervisione del giudice di gara. Le future versioni della GUI dovranno integrare strumenti di *editing* interattivo, fornendo all'operatore la possibilità di modificare, perfezionare o sovrascrivere manualmente i risultati proposti dalla rete neurale (ad esempio effettuando un micro-spostamento della linea di traguardo calcolata), mantenendo così l'autorità decisionale finale sull'ordine di arrivo.

Bibliografia

[1] World Athletics. *Competition and Technical Rule Amendments Approved by World Athletics Council - December 2025*. Principato di Monaco, 2025 (cit. a p. 2).

[2] Lynx System Developers. *Componenti di cronometraggio per l'atletica leggera*. 2026. URL: https://finishlynx.com/it/packages/athletics-timing-systems/ (visitato il giorno 12/06/2026) (cit. a p. 2).

[3] SpeedEndurance.com. *Understanding How a Photo Finish Camera Works*. 2016. URL: https://speedendurance.com/2016/09/18/understanding-how-a-photo-finish-camera-works/ (visitato il giorno 12/06/2026) (cit. a p. 3).

[4] Lynx System Developers. *Fully Automatic Timing and Line-Scan Technology: FAT Slices Explained*. 2026. URL: https://www.finishlynx.com/wp-content/uploads/FAT_slices-768x431-1.webp (visitato il giorno 12/06/2026) (cit. a p. 4).

[5] Swatch Group (OMEGA). *Omega: New Technology for Paris 2024*. 2024. URL: https://www.swatchgroup.com/en/swatch-group/innovation-powerhouse/internet-things/omega-new-technology-paris-2024 (visitato il giorno 12/03/2026) (cit. alle pp. 4, 5).

[6] FinishLynx. *Vision PRO X Camera*. 2025. URL: https://finishlynx.com/it/product/cameras/vision-pro-x-camera/ (visitato il giorno 12/03/2026) (cit. alle pp. 4, 5).

[7] World Athletics. *IAAF Photo Finish Guidelines*. URL: https://worldathletics.org/about-iaaf/documents/technical-information (visitato il giorno 12/03/2026) (cit. alle pp. 5–8, 87).

[8] FinishLynx. *Plugin: Automatic Capture Mode*. URL: https://finishlynx.com/product/finishlynx-plugins/automatic-capture-plugin/ (visitato il giorno 12/03/2026) (cit. a p. 8).

[9] J. Redmon, S. Divvala, R. Girshick e A. Farhadi. «You only look once: Unified, real-time object detection». In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition*. 2016 (cit. a p. 8).

[10] P. Jafarzadeh, L. Zelioli, P. Virjonen et al. «Enhancing hurdles athletes' performance analysis: A comparative study of cnn-based pose estimation frameworks». In: *Multimed Tools Appl* 84 (2025), pp. 34573–34591. DOI: 10.1007/s11042-024-20587-z (cit. a p. 8).

[11] Swatch Group. *Tecnologia di misurazione Omega per il salto con l'asta*. 2026. URL: https://www.swatchgroup.com/sites/default/files/inline-images/680-383-pole-vault.jpg (visitato il giorno 12/06/2026) (cit. a p. 9).

[12] M. Zhao, Z. Xie, Y. Nie, J. Li, F. Shuang e Z. Feng. «An Automatic Timing Method for Photo Finish». In: *8th International Conference on Computer Science & Education (ICCSE 2013)*. Colombo, Sri Lanka, 2013, pp. 902–906 (cit. a p. 10).

[13] F. C. Akyon, S. O. Altinuc e A. Temizel. «Slicing aided hyper inference and fine-tuning for small object detection». In: *2022 IEEE International Conference on Image Processing (ICIP)*. 2022. DOI: 10.1109/ICIP46576.2022.9897990 (cit. a p. 11).

[14] S.-T. Kim e H. J. Lee. «Lightweight Stacked Hourglass Network for Human Pose Estimation». In: *Appl. Sci.* 10 (2020), p. 6497. DOI: 10.3390/app10186497 (cit. a p. 11).

[15] Canva Pty Ltd. *Canva: Visual Suite for Everyone*. 2026. URL: https://www.canva.com/ (visitato il giorno 12/06/2026) (cit. a p. 15).

[16] Arduino. *Arduino Uno Rev3 - Scheda microcontrollore*. 2026. URL: https://m.media-amazon.com/images/I/71ok6q+8tEL._AC_SL1000_.jpg (visitato il giorno 12/06/2026) (cit. alle pp. 15, 21).

[17] Teledyne DALSA. *Spyder3 GigE Vision Dual Line Scan Camera User's Manual (SG-11)*. Rev. 03-032-20027-03. 2010 (cit. alle pp. 16–19, 81, 88, 89).

[18] European Machine Vision Association (EMVA). *GenICam Standard Features Naming Convention (SFNC)*. Ver. 1.5.1. 2011 (cit. a p. 18).

[19] Arduino. *Scheda tecnica Arduino Uno rev. 3*. URL: https://docs.arduino.cc/hardware/uno-rev3/#tech-specs (visitato il giorno 24/03/2026) (cit. alle pp. 20, 21).

[20] ASUS. *Scheda tecnica ASUS ROG STRIX G16 (2024)*. URL: https://rog.asus.com/it/laptops/rog-strix/rog-strix-g16-2024/spec/ (visitato il giorno 24/03/2026) (cit. a p. 22).

[21] K. Kudo. *Harvesters: An Image Acquisition Library for GenTL Producers*. Repository GitHub. URL: https://github.com/genicam/harvesters (visitato il giorno 09/04/2026) (cit. alle pp. 22, 25).

[22] Association for Advancing Automation (A3). *Vision Standards: GigE Vision*. URL: https://www.automate.org/vision/vision-standards/vision-standards-gige-vision (visitato il giorno 03/04/2026) (cit. a p. 23).

[23] Basler AG. *The Elements of GigE Vision - White Paper*. URL: https://machinevisionstore.com/content/downloads/basler/elements-of-gige-vision.pdf (visitato il giorno 03/04/2026) (cit. a p. 24).

[24] European Machine Vision Association (EMVA). *GenICam Standard - Generic Interface for Cameras*. URL: https://www.emva.org/standards-technology/genicam/ (visitato il giorno 09/04/2026) (cit. alle pp. 24, 25).

[25] Baumer Optronic GmbH. *Baumer GAPI SDK - Software per l'acquisizione di immagini*. URL: https://www.baumer.com/it/it/panoramica-dei-prodotti/telecamere-industriali-elaborazione-delle-immagini/software/baumer-gapi-sdk/windows/c/14175 (visitato il giorno 11/04/2026) (cit. a p. 25).

[26] Microchip Technology. *ATmega328P - 8-bit AVR Microcontroller with 32K Bytes In-System Programmable Flash Datasheet*. URL: https://ww1.microchip.com/downloads/en/DeviceDoc/Atmel-7810-Automotive-Microcontrollers-ATmega328P_Datasheet.pdf (visitato il giorno 11/04/2026) (cit. alle pp. 27, 28).

[27] World Athletics. *Competition Calendar and Results*. 2026. URL: https://worldathletics.org/competition/calendar-results?offset=2500 (visitato il giorno 12/06/2026) (cit. alle pp. 29–32, 38).

[28] FinishLynx. *Programma ufficiale FinishLynx*. URL: https://finishlynx.com/it/support/downloads/ (visitato il giorno 11/04/2026) (cit. a p. 32).

[29] Tzutalin. *LabelImg: Graphical image annotation tool and label object bounding boxes in images*. Repository GitHub. 2015. URL: `https://github.com/HumanSignal/labelImg` (visitato il giorno 17/03/2026) (cit. a p. 38).

[30] C. Shorten e T. M. Khoshgoftaar. «A survey on Image Data Augmentation for Deep Learning». In: *Journal of Big Data* 6.1 (2019), p. 60 (cit. alle pp. 40, 41).

[31] A. Krizhevsky, I. Sutskever e G. E. Hinton. «ImageNet classification with deep convolutional neural networks». In: *Communications of the ACM* 60.6 (2017), pp. 84–90 (cit. alle pp. 42, 43).

[32] Z. Zhong, L. Zheng, G. Kang, S. Li e Y. Yang. «Random Erasing Data Augmentation». In: *Proceedings of the AAAI Conference on Artificial Intelligence*. Vol. 34. 07. 2020, pp. 13001–13008 (cit. alle pp. 43, 44).

[33] G. Jocher, A. Chaurasia e J. Qiu. *YOLO by Ultralytics*. Repository GitHub. 2023. URL: `https://github.com/ultralytics/ultralytics` (visitato il giorno 18/03/2026) (cit. a p. 47).

[34] R. Girshick, J. Donahue, T. Darrell e J. Malik. «Rich feature hierarchies for accurate object detection and semantic segmentation». In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2014, pp. 580–587 (cit. a p. 47).

[35] Z. Tian, C. Shen, H. Chen e T. He. «FCOS: Fully Convolutional One-Stage Object Detection». In: *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*. 2019, pp. 9627–9636 (cit. a p. 47).

[36] J. Tompson, R. Goroshin, A. Jain, Y. LeCun e C. Bregler. «Efficient object localization using Convolutional Networks». In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2015, pp. 3320–3329 (cit. a p. 56).

[37] K. He, X. Zhang, S. Ren e J. Sun. «Deep Residual Learning for Image Recognition». In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 770–778 (cit. alle pp. 57, 65).

[38] C. Szegedy, V. Vanhoucke, S. Ioffe, J. Shlens e Z. Wojna. «Rethinking the Inception Architecture for Computer Vision». In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2016, pp. 2818–2826 (cit. alle pp. 60, 63).

[39] J. Hu, L. Shen e G. Sun. «Squeeze-and-Excitation Networks». In: *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*. 2018, pp. 7132–7141 (cit. a p. 60).

[40] A. Paszke et al. «PyTorch: An imperative style, high-performance deep learning library». In: *Advances in Neural Information Processing Systems 32 (NeurIPS)*. 2019, pp. 8024–8035 (cit. a p. 62).

[41] D. P. Kingma e J. Ba. «Adam: A Method for Stochastic Optimization». In: *Proceedings of the 3rd International Conference on Learning Representations (ICLR)*. San Diego, CA, USA, 2015 (cit. a p. 66).

[42] I. Loshchilov e F. Hutter. «SGDR: Stochastic Gradient Descent with Warm Restarts». In: *Proceedings of the 5th International Conference on Learning Representations (ICLR)*. Toulon, France, 2017 (cit. a p. 66).

[43] Blender Online Community. *Blender - a 3D modelling and rendering package*. Ver. 3.6. Blender Foundation. 2023. URL: `https://www.blender.org` (visitato il giorno 16/04/2026) (cit. a p. 81).

[44] KiCad Developers Team. *KiCad EDA - Schematic Capture & PCB Design Software*. 2026. URL: `https://www.kicad.org/` (visitato il giorno 12/06/2026) (cit. a p. 85).

[45] G. Bradski. «The OpenCV Library». In: *Dr. Dobb's Journal of Software Tools* (2000). URL: `https://opencv.org` (visitato il giorno 18/04/2026) (cit. a p. 92).

[46] M. Zhao, Z. Xie, Y. Nie, J. Li, F. Shuang e Z. Feng. «An Automatic Timing Method for Photo Finish». In: *Proceedings of the 8th International Conference on Computer Science & Education (ICCSE 2013)*. Colombo, Sri Lanka, 2013, pp. 902–906 (cit. alle pp. 100–103).