

Object Detection e Object Tracking con input testuale

Leonardo Del Bene

November 2025

Contents

1	Introduzione	3
2	Object Detection guidata da input testuali	3
2.1	Calcolo della similità	4
2.1.1	Rete di similarità	5
2.1.2	Loss	5
2.1.3	Dataset	5
2.1.4	Training	7
2.1.5	Soglia di similarità	7
2.2	Risultati ottenuti	7
2.2.1	Errori frequenti	9
3	Object Tracking guidato da input testuali	12
3.1	Deep SORT: Deep Simple Online and Realtime Tracking	12
3.1.1	Architettura del sistema	12
3.1.2	Vantaggi del modello	13
3.1.3	Sintesi	13
3.2	Risultati ottenuti	13
3.2.1	Analisi dell'impatto degli embeddings di similarità su DeepSORT	16
3.2.2	esempi di tracking	16
3.3	Tracking real-time	17
4	Conclusioni	17

1 Introduzione

Negli ultimi anni, l'integrazione tra **visione artificiale** e **comprendione del linguaggio naturale** ha aperto nuove possibilità nell'analisi automatica dei contenuti visivi. Questo progetto propone un sistema di *object detection* e *tracking* multimodale, capace di individuare e seguire nel tempo oggetti specificati attraverso una **descrizione testuale** fornita dall'utente. Questa sfida è stata affrontata in [6], a differenza di loro che usano un architettura basata su *Grounding Dino*, in questo progetto viene usata una più simile a [2].

L'architettura sviluppata si basa sulla combinazione di due modelli di riferimento:

- **YOLO** (*You Only Look Once*), utilizzato per la rilevazione rapida e accurata degli oggetti presenti in ciascun frame del video [8];
- **CLIP** (*Contrastive Language–Image Pretraining*), impiegato per mettere in relazione le feature visive estratte dagli oggetti con un embedding testuale che rappresenta la query dell'utente [7].

In pratica, l'utente può specificare un input come ad esempio "*a brown dog*" oppure "*a person wearing a red jacket*", e il sistema rileva soltanto gli oggetti coerenti con tale descrizione, ignorando gli altri. Successivamente, viene applicato l'algoritmo **DeepSORT** [9], un metodo di *multi-object tracking* che associa le rilevazioni tra frame consecutivi, mantenendo un'identità coerente per ciascun oggetto tracciato nel tempo.

Il risultato finale è un **video annotato**, in cui vengono evidenziati e seguiti soltanto gli oggetti che corrispondono alla query testuale. Oltre alla componente visiva, il sistema calcola anche diverse **metriche di tracking** per valutare la qualità del tracciamento.

Questo approccio unisce la potenza della *computer vision* tradizionale (YOLO + DeepSORT) con la comprensione semantica multimodale di CLIP, permettendo di filtrare dinamicamente gli oggetti d'interesse sulla base del loro significato e non solo della loro forma.

Gli obiettivi principali del sistema sviluppato sono i seguenti:

1. Integrare modelli di visione e linguaggio in un'unica pipeline coerente.
2. Permettere il tracking di oggetti selezionati tramite descrizioni testuali naturali.
3. Fornire un sistema modulare e adattabile a diversi scenari video.
4. Valutare le prestazioni mediante metriche standard di detection e tracking.

In conclusione, il progetto rappresenta un passo verso sistemi di analisi video più flessibili e semantici, in grado di rispondere a query linguistiche complesse e di adattarsi dinamicamente al contenuto visivo.

2 Object Detection guidata da input testuali

La prima parte del progetto si concentra sull'integrazione dei modelli **YOLO** e **CLIP** per la realizzazione di un sistema di **object detection** multimodale, in grado di individuare soltanto gli oggetti semanticamente coerenti con una descrizione testuale fornita dall'utente.

In particolare, il modello **YOLO** (*You Only Look Once*) viene utilizzato per l'individuazione delle *bounding box* che delimitano gli oggetti presenti all'interno delle immagini. Per ciascuna di queste regioni rilevate, il modello **CLIP** (*Contrastive Language–Image Pretraining*) viene impiegato per estrarre un *embedding visuale*, ossia una rappresentazione vettoriale del contenuto visivo dell'oggetto.

Parallelamente, CLIP riceve in input anche la *query testuale* specificata dall'utente, dalla quale viene calcolato un *embedding testuale* corrispondente. In questo modo, per ogni immagine si ottengono due insiemi di rappresentazioni: uno per gli oggetti rilevati da YOLO e uno per la descrizione linguistica fornita.

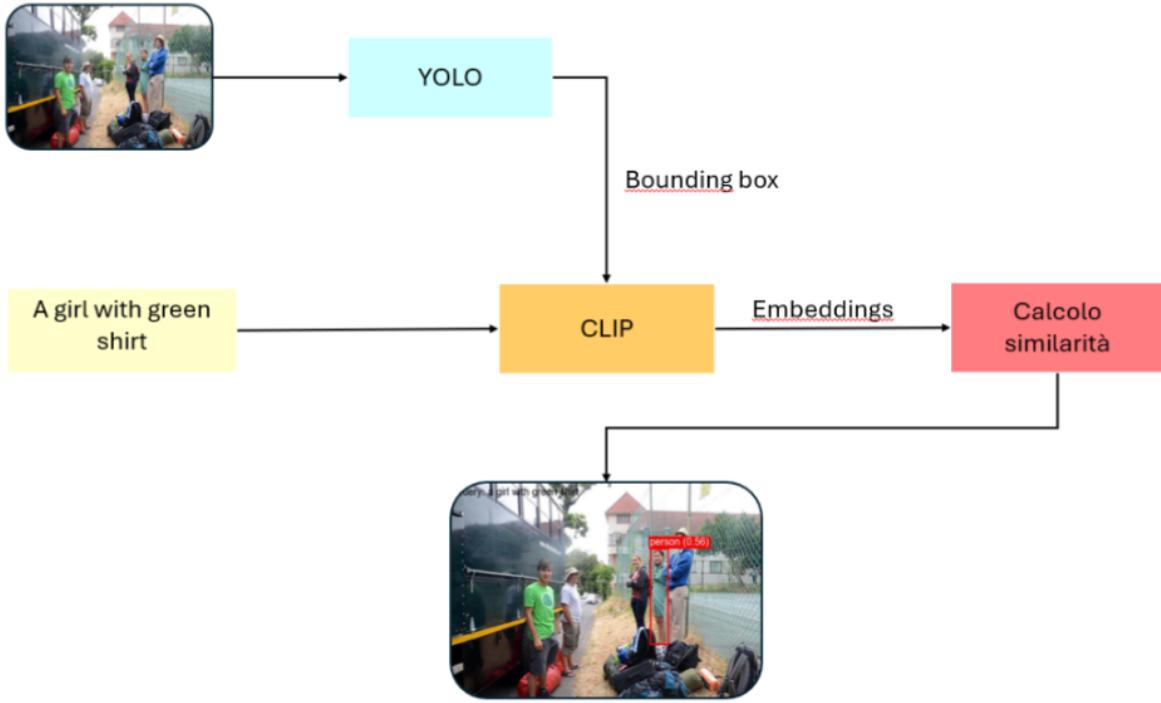


Figure 1: Pipelime Object detection con input testuale

Successivamente, viene calcolata la **similarità semantica** tra l'embedding testuale e ciascun embedding visivo. Solo gli oggetti che presentano un'elevata similarità con la query vengono considerati coerenti e mantenuti tra i risultati. Tale confronto consente di filtrare automaticamente le rilevazioni, eliminando gli elementi visivi non pertinenti al significato espresso dall'input linguistico.

In output, il sistema restituisce l'immagine annotata con le *bounding box* relative esclusivamente agli oggetti che risultano semanticamente affini alla descrizione testuale fornita. Questo approccio permette di combinare l'efficienza della rilevazione basata su YOLO con la capacità di comprensione semantica multimodale offerta da CLIP, fornendo un metodo di object detection guidato dal linguaggio naturale.

2.1 Calcolo della similiarità

Il primo approccio adottato per il calcolo della similiarità tra gli *embedding* visivi e l'*embedding* testuale ottenuti dal modello CLIP è stato basato sulla **similarità coseno**. Tale metrica misura il grado di allineamento tra due vettori nello spazio delle rappresentazioni, considerando unicamente l'angolo tra essi e ignorandone il modulo.

Tuttavia, l'utilizzo della sola similarità coseno ha mostrato alcune limitazioni sperimentali rilevanti:

- la similarità difficilmente supera valori di soglia pari a 0.3, anche per corrispondenze semanticamente corrette;
- la maggior parte delle bounding box ottiene punteggi di similarità concentrati in un intervallo ristretto (tipicamente tra 0.1 e 0.3), rendendo complessa la distinzione tra oggetti rilevanti e irrilevanti rispetto alla query testuale.

La soluzione proposta consiste nello sviluppo di una rete neurale in grado di proiettare gli *embeddings* generati da CLIP in un nuovo spazio di rappresentazione, in modo che risultino più facilmente separabili mediante la similarità coseno. Tale approccio è ispirato al lavoro di [1], che evidenzia come una proiezione apposita degli *embeddings* di CLIP possa migliorare la capacità del modello di catturare relazioni più fini tra testo e immagine. Infatti, gli *embeddings* originali di CLIP, pur essendo altamente informativi, tendono a

risultare poco discriminativi nel distinguere oggetti appartenenti alla stessa classe ma differenti per dettagli sottili, come il colore, la forma o la posa. L'introduzione di uno spazio proiettivo dedicato consente quindi di affinare la rappresentazione semantica, rendendo più efficace il processo di associazione tra descrizioni testuali e contenuti visivi specifici.

2.1.1 Rete di similarità

L'idea proposta da [1] consiste nell'utilizzare una semplice rete *Multi-Layer Perceptron (MLP)* per proiettare gli *embeddings* generati da CLIP in un nuovo spazio di rappresentazione. Nel presente lavoro, tale approccio è stato implementato attraverso un MLP composto da **tre layer lineari**, intervallati da funzioni di attivazione **ReLU** e operazioni di **normalizzazione**. A metà rete è inoltre inserito un layer di *dropout*, con l'obiettivo di ridurre l'overfitting e migliorare la generalizzazione del modello. La stessa architettura viene impiegata sia per gli *embeddings* visivi che per quelli testuali, consentendo così di effettuare la proiezione in un contesto di *shared parameters*, e favorendo l'allineamento semantico tra le due modalità.

2.1.2 Loss

La funzione di perdita utilizzata per addestrare la rete è stata ispirata dal lavoro di [5], a cui è stato aggiunto un termine di regolarizzazione per controllare la deviazione rispetto alla similarità originale di CLIP. La loss totale è definita come:

$$\mathcal{L}_{\text{total}} = \mathcal{L}_{\text{contrastive}} + \lambda_{\text{reg}} \mathcal{L}_{\text{reg}} \quad (1)$$

Il termine contrastivo mira a separare correttamente le coppie positive e negative di embeddings proiettati ed è formulato come:

$$\mathcal{L}_{\text{contrastive}} = \frac{1}{N_+} \sum_{i \in \text{pos}} (1 - \cos(\mathbf{v}'_i, \mathbf{t}'_i)) + \frac{1}{N_-} \sum_{j \in \text{neg}} \max(0, \cos(\mathbf{v}'_j, \mathbf{t}'_j)) \quad (2)$$

Il termine di regolarizzazione, invece, penalizza deviazioni eccessive tra la similarità calcolata sugli embeddings proiettati e quella originaria di CLIP:

$$\mathcal{L}_{\text{reg}} = \frac{1}{N} \sum_{i=1}^N \left[\max \left(0, |\cos(\mathbf{v}'_i, \mathbf{t}'_i) - \cos(\mathbf{v}_i, \mathbf{t}_i)| - m \right) \right]^2 \quad (3)$$

Dove:

- \mathbf{v}'_i e \mathbf{t}'_i indicano gli *embeddings* proiettati tramite il modello MLP,
- \mathbf{v}_i e \mathbf{t}_i rappresentano gli *embeddings* originali di CLIP,
- $\cos(\cdot, \cdot)$ è la similarità coseno,
- m è il *margin* che definisce la tolleranza alla deviazione,
- λ_{reg} è il peso del termine di regolarizzazione.

2.1.3 Dataset

Per l'addestramento della rete, il dataset è stato costruito sviluppando l'idea di [3]:

- Inizialmente sono state utilizzate le immagini originali del dataset COCO.
- Ogni immagine è stata elaborata tramite il modello YOLO, dal quale sono state estratte le *bounding box* degli oggetti presenti.
- Per ciascuna *bounding box*, è stato utilizzato il modello BLIP, come proposto in [4], per generare una *caption* descrittiva dell'oggetto contenuto.

- Gli esempi positivi per l'addestramento sono stati costituiti dalla coppia *bounding box – caption corretta*, mentre gli esempi negativi sono stati creati accoppiando una *bounding box* con una *caption* selezionata casualmente all'interno dello stesso batch.

In figura 2 è possibile osservare 3 esempi di elementi del dataset, sono riportate solo coppie tra *bounding box* e *caption* coerenti.



(a) *A woman standing on the beach*



(b) *A bear is sitting on a tree branch*



(c) *A red apple*

Figure 2: Esempi di elementi del dataset

2.1.4 Training

La rete è stata addestrata per circa 60 epoche, adottando una strategia di *early stopping* per prevenire l'overfitting. Per motivi di efficienza computazionale, non è stato utilizzato l'intero dataset COCO, ma solamente un sottoinsieme di 10000 coppie *bounding box – caption*.

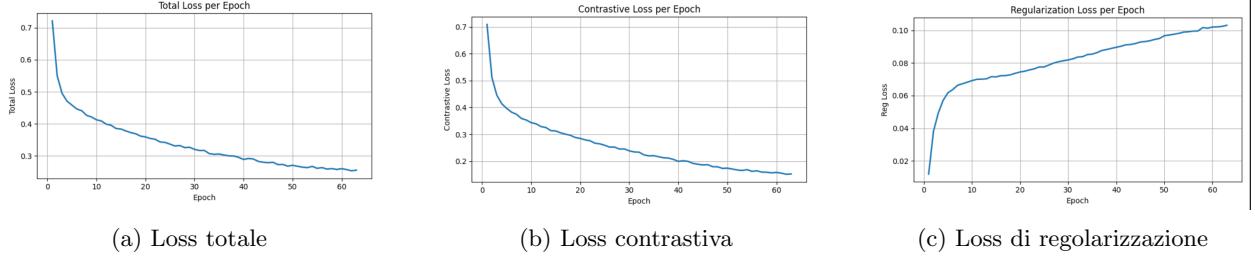


Figure 3: Grafico dell'andamento dell'addestramento

Dalla figura 3 è possibile osservare l'andamento della *Loss* durante l'addestramento. Al termine dell'addestramento, la *loss totale* si è stabilizzata intorno a 0.255, indicando una buona convergenza del modello. In particolare, la *loss contrastiva* ha raggiunto 0.152, mentre il *termine di regolarizzazione* si è assestato su 0.103, evidenziando un equilibrio efficace tra separazione delle coppie positive/negative e mantenimento della coerenza con le similarità originali di CLIP.

2.1.5 Soglia di similarità

Per distinguere tra coppie di *embeddings* ad alta e bassa similarità, è stata implementata una **soglia di similarità**. Tutto ciò che supera questa soglia viene considerato semanticamente coerente con l'input testuale, permettendo così di rilevare più oggetti nell'immagine che corrispondono alla descrizione fornita. Tuttavia, la scelta del valore della soglia risulta complessa, in quanto varia significativamente a seconda dell'immagine e del tipo di oggetto ricercato, e può essere influenzata anche da piccole modifiche nella formulazione dell'input testuale.

Un approccio alternativo consiste nel non utilizzare alcuna soglia, ma selezionare esclusivamente la coppia di *embeddings* con la massima similarità. Questa strategia, pur limitando il numero di oggetti rilevati, risulta più robusta e riduce notevolmente il rischio di identificare oggetti semanticamente incoerenti. Tutte queste considerazioni vengono discusse nel dettaglio nel paragrafo 2.2.

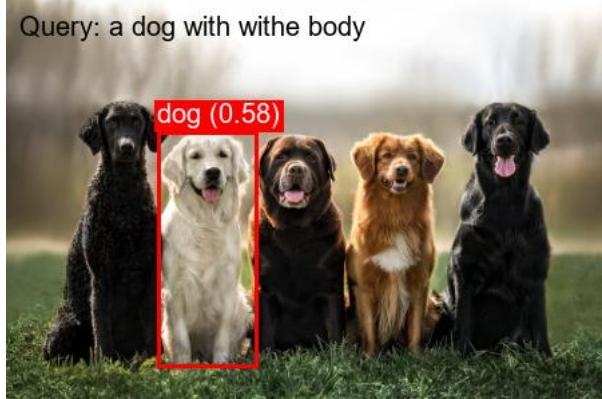
2.2 Risultati ottenuti

In Figura 5 viene mostrato il confronto tra il calcolo della similarità utilizzando la distanza coseno sugli *embeddings* originali di CLIP e la distanza coseno sulle proiezioni generate dalla rete di similarità. Nella Tabella 4 sono riportati i valori di similarità rispetto all'input testuale per ciascuna rilevazione effettuata da YOLO.

È possibile osservare come l'impiego della rete di similarità permetta una separazione più netta tra le detection, facilitando l'individuazione dell'oggetto coerente con la descrizione testuale. Al contrario, utilizzando solo la distanza coseno sugli *embeddings* originali di CLIP, le misure di similarità risultano molto simili tra loro, evidenziando la difficoltà degli *embeddings* originali nel catturare dettagli fini dell'immagine [1], come in questo caso il colore del pelo.

In figura 6 è possibile osservare il confronto dei risultati ottenuti variando la soglia di similarità. Come anticipato, la scelta della soglia corretta risulta essere molto delicata: per una soglia di 0.3 il modello non riesce a riconoscere come semanticamente coerente alcun oggetto. Abbassando la soglia a 0.25, viene correttamente identificata una sola bottiglia; riducendo ulteriormente la soglia a 0.2, il modello considera semanticamente coerenti anche le persone, nonostante l'input testuale fosse “*a bottle of wine*”. I risultati riportati sono stati ottenuti calcolando la similarità tramite la distanza coseno sugli *embeddings* di CLIP.

In figura 7 sono riportati i risultati ottenuti al variare della soglia nel caso in cui venga utilizzata la rete di



(a) Text Input: *a dog with white body*



(b) Text Input: *a dog with white body*

Figure 4: Confronto delle similarità usando solo il coseno o la rete

Detection	Sim coseno	Sim rete
dog	0.220	0.471
dog	0.222	0.580
dog	0.224	0.278
dog	0.220	0.306
dog	0.223	0.557

Figure 5: Confronto tra l'uso della rete di similarità e del solo coseno. Le immagini (a) e (b) mostrano esempi di rilevamento con la rete e con il coseno, rispettivamente. La tabella mostra i valori di similarità corrispondenti.

similarità per proiettare gli *embeddings* di CLIP in uno spazio maggiormente separabile. Si può osservare che, in questo scenario, il valore ottimale della soglia si colloca tra 0.6 e 0.7, mentre nel caso in cui si utilizzi la sola distanza coseno la soglia corretta risulta essere intorno a 0.27. L'impiego della rete di similarità non solo consente di ottenere valori di similarità mediamente migliori, ma introduce anche una maggiore tolleranza nella scelta della soglia, rendendo il sistema più stabile e robusto rispetto alle variazioni del parametro di decisione.

In figura 8 sono presenti 9 esempi di detection avvenute usando la rete di similarità e nessuna soglia fissa, ma considerando la bounding box a più alta similarità.



Figure 6: Confronto delle diverse soglie per una stessa immagine e input *a bottle of wine* (similarità coseno)

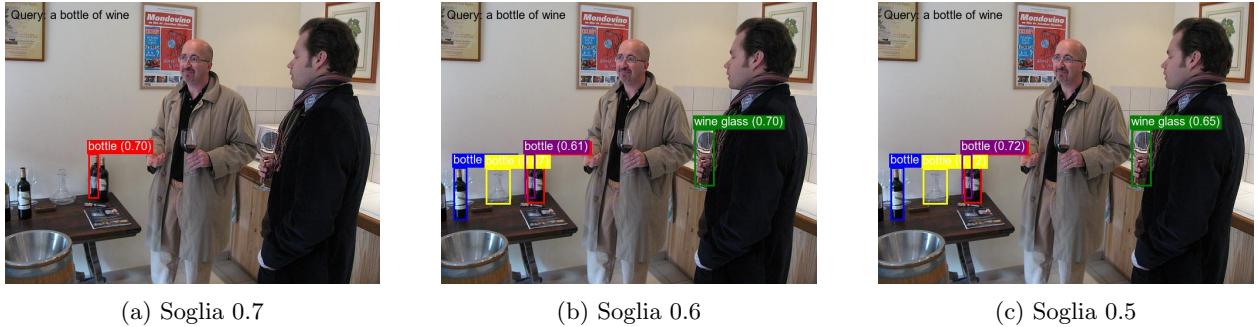


Figure 7: Confronto delle diverse soglie per una stessa immagine e input *a bottle of wine* (rete similarità)

2.2.1 Errori frequenti

Gli errori più frequenti riscontrati durante le fasi di rilevamento riguardano diversi aspetti del processo di *object detection*. Tra i principali si osservano:

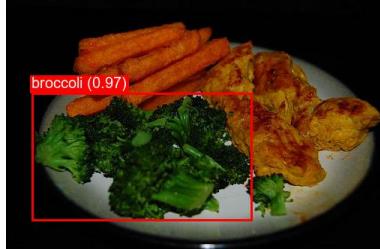
- la ricerca di oggetti che non appartengono a nessuna delle classi previste da YOLO;
- la mancata distinzione tra *sottoclassi*, come ad esempio la differenza tra una *persona* e un *bambino*, oppure tra un *giocatore* e uno *spettatore*.
- spesso viene richiesto una fase di *prompt engeneering*;

In figura 9 vengono mostrati esempi di errore dovuti all'assenza delle categorie *tomato*, *sea turtle* e *shark*. Tuttavia, come evidenziato in figura 9b, in alcuni casi il sistema riesce comunque a fornire una detection coerente: ad esempio, una *sea turtle* può essere classificata come *bird* e, grazie al confronto semantico con CLIP, risulta correttamente associata all'input testuale.

Il principale problema nel tentativo di rilevare oggetti appartenenti a classi non incluse in YOLO è che non viene generata alcuna *bounding box* da passare a CLIP. In rari casi, tuttavia, l'oggetto viene erroneamente assegnato a una classe esistente, consentendo comunque una detection semanticamente corretta.

In figura 10 possiamo osservare come anche la formulazione del *prompt* influenzi in modo significativo il risultato della detection: piccoli cambiamenti nella descrizione testuale possono determinare variazioni sostanziali negli oggetti rilevati.

In figura 11 è possibile osservare 2 esempio di detection sbagliate nel riconoscere delle sottoclassi di *person*.



(a) Text Input: *a dish of vegetables*



(b) Text Input: *a balck car seen from behind*



(c) Text Input: *a tennis ball*



(d) Text Input: *a white car*



(e) Text Input: *baseball referee*



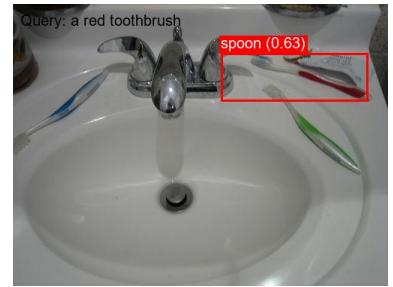
(f) Text Input: *a yellow ski*



(g) Text Input: *a person with brown jacket and red backpack*



(h) Text Input: *a black sheep*



(i) Text Input: *a red toothbrush*

Figure 8: Esempi di object detection guidato da input testuale



(a) input: *a tomato*



(b) input: *a sea turtle*



(c) input: *a shark*

Figure 9: Esempi di errori dovuti a YOLO



Figure 10: Esempi di *prompt engineering*: l'influenza della formulazione testuale sulla coerenza semantica dei risultati.

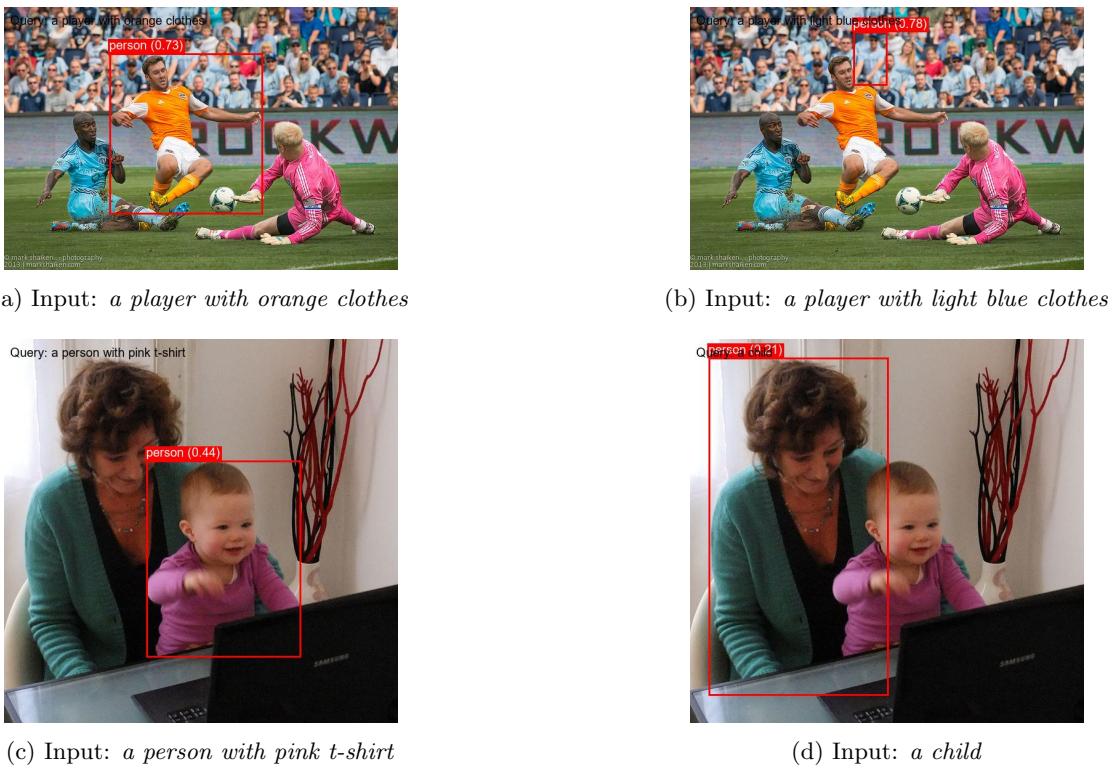


Figure 11: Esempi di errori nel riconoscere le sottoclassi *player* e *child*.

3 Object Tracking guidato da input testuali

Il passo successivo all'*object detection* è il **tracking**, ovvero la capacità di tracciare nel tempo gli oggetti che risultano semanticamente coerenti con la query testuale fornita in input. Per realizzare questo obiettivo, è stata utilizzata la stessa pipeline impiegata per l'*object detection* — composta da **YOLO**, **CLIP** e dal calcolo della similarità — integrandola con l'algoritmo di tracciamento **DeepSORT**, introdotto da [9].

DeepSORT estende l'algoritmo **SORT** (Simple Online and Realtime Tracking) introducendo una rete neurale dedicata all'estrazione di *feature* di aspetto. Mentre SORT associa le *detections* tra fotogrammi consecutivi basandosi unicamente su modelli di movimento (tramite filtri di Kalman) e sulla distanza spaziale, DeepSORT aggiunge una rappresentazione visiva appresa, che consente di distinguere meglio tra oggetti simili e di mantenere le identità coerenti nel tempo anche in presenza di occlusioni o variazioni di posa. In questo modo, il tracciamento risulta significativamente più robusto e accurato. In figura 12 è possibile osservare la pipeline completa per il tracking con input testuali.

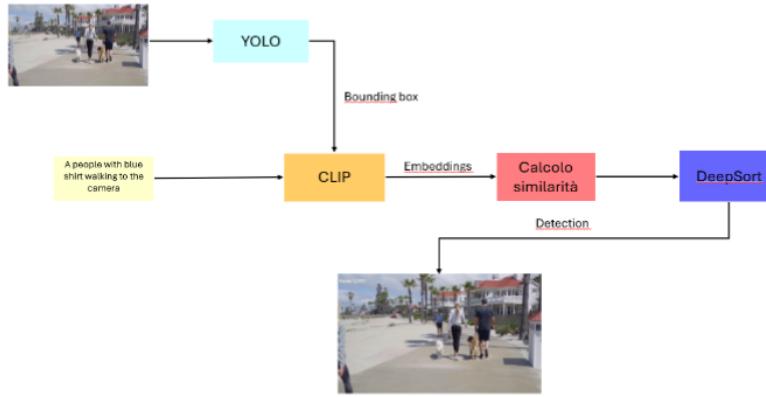


Figure 12: Pipeline Object tracking guidato da input testuale

3.1 Deep SORT: Deep Simple Online and Realtime Tracking

L'algoritmo **Deep SORT** (*Simple Online and Realtime Tracking with a Deep Association Metric*), proposto da [9], rappresenta un'estensione dell'algoritmo **SORT** (Simple Online and Realtime Tracking). L'obiettivo principale di Deep SORT è migliorare la robustezza del tracciamento multi-oggetto (*Multi-Object Tracking*, MOT) in scenari complessi, introducendo una rappresentazione visiva appresa tramite una rete neurale convoluzionale (CNN) per ridurre gli *identity switches* e mantenere le tracce anche in presenza di occlusioni.

3.1.1 Architettura del sistema

Deep SORT segue un paradigma di tipo **tracking-by-detection**: ad ogni frame, le *bounding box* generate da un modello di *object detection* vengono associate alle tracce esistenti mediante un processo di associazione basato su movimento e apparenza.

1. **Modello di stato e filtro di Kalman.** Ogni oggetto è rappresentato da uno stato definito come:

$$\mathbf{x} = (u, v, \gamma, h, \dot{u}, \dot{v}, \dot{\gamma}, \dot{h})$$

dove (u, v) rappresenta il centro della bounding box, γ il rapporto d'aspetto e h l'altezza. Il filtro di Kalman viene utilizzato per predire la posizione della bounding box nel frame successivo, garantendo una stima coerente del movimento nel tempo.

2. **Associazione tra tracce e rilevamenti.** Deep SORT combina due misure di distanza:

- la *distanza di Mahalanobis* $d^{(1)}(i, j)$, che misura la coerenza tra la posizione predetta dal filtro di Kalman e la detection osservata;
- la *distanza coseno* $d^{(2)}(i, j)$ tra i descrittori di apparenza estratti da una rete, che misura la somiglianza visiva.

La distanza complessiva utilizzata per l'associazione è data da:

$$c_{i,j} = \lambda d^{(1)}(i, j) + (1 - \lambda) d^{(2)}(i, j)$$

dove $\lambda \in [0, 1]$ bilancia l'importanza tra le due componenti. L'assegnazione ottimale tra tracce e rilevamenti è risolta tramite l'*Hungarian algorithm*.

3. Matching Cascade. Le tracce più recenti hanno priorità più alta durante l'associazione, riducendo la frammentazione e migliorando la coerenza delle identità.

4. Descrittore di apparenza. Nella sua formulazione classica, DeepSORT utilizza una rete neurale convoluzionale (CNN) pre-addestrata su un dataset di *person re-identification*, in particolare il dataset MARS. Tale rete produce un descrittore di apparenza costituito da un vettore di 128 dimensioni, normalizzato sulla sfera unitaria, che consente di confrontare efficacemente le apparenze degli oggetti attraverso la similarità coseno.

Tuttavia, l'impiego di una CNN addestrata su un dataset di *re-identification* specifico per la classe *person* risulta limitante nel contesto di un sistema *open-vocabulary*. Per questo motivo, è stato scelto di sostituire il descrittore di apparenza tradizionale con gli *embeddings* generati dalla rete di similarità proposta, utilizzandoli come feature di apparenza all'interno del framework DeepSORT.

I risultati ottenuti usando *DeepSORT* con l'approccio basato sulla rete di similarità sono riportati nella Sezione 3.2.1.

3.1.2 Vantaggi del modello

Grazie all'integrazione delle informazioni di movimento e di apparenza, Deep SORT:

- mantiene identità consistenti anche in presenza di occlusioni o sovrapposizioni;
- riduce gli *identity switches* fino al 45% rispetto a SORT;;
- può essere facilmente integrato con diversi modelli di object detection.

3.1.3 Sintesi

In sintesi, Deep SORT combina un modello di movimento (filtro di Kalman) con un modello di apparenza profonda (CNN) per migliorare la robustezza del tracciamento multi-oggetto. Questo approccio consente di ottenere risultati più accurati e stabili nel tempo rispetto ai metodi di tracking basati unicamente su informazioni cinematiche.

3.2 Risultati ottenuti

Per valutare il sistema proposto per il tracking guidato da input testuale, è stato utilizzato il dataset introdotto in [6], denominato *LaMOT*. Il dataset *LaMOT* raccoglie sequenze video provenienti da diversi benchmark consolidati, tra cui *MOT*, *LaSOT* e *VisDrone*, arricchite con annotazioni testuali che consentono di valutare scenari di tracking guidato dal linguaggio naturale.

Le prestazioni del sistema sono state misurate principalmente tramite le seguenti metriche:

- **HOTA** (*Higher Order Tracking Accuracy*): misura complessiva la qualità del tracking combinando la capacità di rilevare correttamente gli oggetti (**DetA**, Detection Accuracy) e di mantenere coerenti le identità nel tempo (**AssA**, Association Accuracy).
- **IDF1**: valuta la coerenza delle identità tracciate confrontando gli ID predetti con quelli reali, combinando precisione e recall sugli ID in un singolo punteggio armonico.

Il dataset *LaMOT* fornisce annotazioni specifiche per il tracking guidato da testo. In questo lavoro, il sistema è stato valutato su un totale di 8 video, così suddivisi:

- 4 sequenze provenienti da **LaSOT**, dataset focalizzato sul *single object tracking* guidato da descrizioni testuali;
- 2 sequenze provenienti da **MOT**, dedicate al *multiple object tracking*;
- 2 sequenze provenienti da **SportsMOT**.

I risultati quantitativi ottenuti sono riportati in Tabella 1, dove sono presentate le metriche HOTA e IDF1 per ciascuna sequenza analizzata.

Sequence	DetA	AssA	HOTA	IDTP	IDFP	IDFN	IDP	IDR	IDF1
car-4	0.899	0.964	0.931	1476	110	55	0.931	0.964	0.947
car-5	0.792	0.450	0.597	4060	418	650	0.907	0.862	0.884
airplane-3	0.331	0.319	0.325	722	142	1318	0.836	0.354	0.497
airplane-10	0.987	0.990	0.989	1553	6	15	0.996	0.990	0.993
mot-13	0.163	0.000	0.000	691	1938	1789	0.263	0.279	0.271
mot-2	0.030	0.000	0.000	228	3478	3949	0.062	0.055	0.058
sportsMot-1	0.563	0.017	0.098	2649	815	1534	0.765	0.633	0.693
sportsMot-2	0.565	0.038	0.146	1807	750	856	0.707	0.679	0.692

Table 1: HOTA e ID per diverse sequenze

Dall’analisi dei risultati emerge che, nei video di *single object tracking*, il sistema sviluppato ottiene prestazioni molto elevate, sia in termini di HOTA che di IDF1, dimostrando una buona capacità di localizzazione e di mantenimento dell’identità dell’oggetto target.

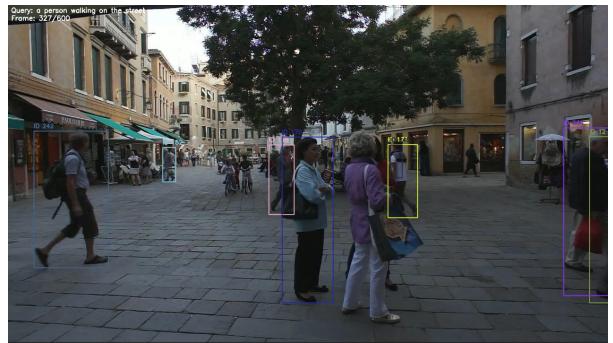
Al contrario, nei video di *multi-object tracking*, le prestazioni risultano significativamente inferiori, in particolare per quanto riguarda la metrica HOTA. Per le sequenze di *SportsMOT*, si osservano valori di **DetA** relativamente soddisfacenti, a fronte di una marcata riduzione di **AssA** e **IDF1**, indicando difficoltà nel mantenere una corretta associazione delle identità tra ground truth e predizioni del modello.

Per quanto riguarda le sequenze provenienti dal dataset *MOT*, le prestazioni risultano prossime allo zero per tutte le metriche considerate. Questo comportamento è principalmente attribuibile a una forte discrepanza tra le bounding box annotate nella ground truth e quelle prodotte dal modello. Tale discrepanza penalizza fortemente il calcolo delle metriche HOTA e IDF1.

Tuttavia, un’analisi qualitativa degli output del sistema suggerisce che queste metriche non riflettano pienamente la qualità visiva del tracking. In Figura 13 è mostrato un frame rappresentativo della sequenza *mot-2*: in Figura 13a è riportata la versione annotata con le ground truth, mentre in Figura 13b è mostrato l’output prodotto dal sistema a partire dall’input testuale “*a person walking on the street*”. È possibile osservare come le bounding box tracciate dal modello differiscano sensibilmente da quelle di ground truth. Questa discrepanza comporta valori molto bassi di HOTA e IDF1, pur in presenza di un tracking visivamente coerente e interpretabile. L’output del sistema, sebbene non perfetto, non risulta qualitativamente scadente come suggerito dalle metriche quantitative. Un comportamento analogo è stato riscontrato anche nella sequenza *mot-13* mostrata in figura 14.



(a) video mot-2 annotato con le ground truth



(b) output del modello

Figure 13: Confronto tra le ground truth e l'output del sistema, con input: *a person walking on the street*



(a) video mot-13 annotato con le ground truth



(b) output del modello

Figure 14: Confronto tra le ground truth e l'output del sistema, con input: *a person walking on the sidewalk*

3.2.1 Analisi dell'impatto degli embeddings di similarità su DeepSORT

Sequence	DetA	AssA	HOTA	IDTP	IDFP	IDFN	IDP	IDR	IDF1
airplane-3	0.318	0.342	0.330	698	152	1342	0.821	0.342	0.483
airplane-10	0.987	0.990	0.989	1553	6	15	0.996	0.990	0.993
car-4	0.941	0.963	0.952	1475	37	56	0.976	0.963	0.969
car-5	0.817	0.449	0.606	4025	215	685	0.949	0.855	0.899
mot-2	0.031	0.000	0.000	209	2524	3966	0.076	0.050	0.061
mot-13	0.166	0.000	0.000	716	1979	1767	0.266	0.288	0.277
sportsMOT-1	0.538	0.007	0.062	2358	1083	1619	0.685	0.593	0.636
sportsMOT-2	0.497	0.002	0.033	1544	804	1081	0.658	0.588	0.621

Table 2: Risultati di tracking in termini di HOTA e metriche di identità (ID) sulle sequenze del dataset *LaMOT* con embeddings delle reti di similarità come descrittori di apparenza.

Nella Tabella 2 sono riportati i risultati del tracciamento in termini di **HOTA** e **IDF1** ottenuti utilizzando DeepSORT con gli *embeddings* prodotti dalla rete di similarità come descrittori di apparenza.

I risultati mostrano un leggero miglioramento sulle sequenze provenienti dal dataset *LaSOT*, sia in termini di HOTA sia di IDF1, mentre si osserva un lieve peggioramento delle prestazioni sulle sequenze di *SportsMOT*.

Tale comportamento risulta coerente con le aspettative: l'uso di descrittori di apparenza non specificamente ottimizzati per la *person re-identification* penalizza il tracciamento in scenari caratterizzati dalla presenza di individui visivamente simili tra loro, mentre favorisce le sequenze in cui l'oggetto da tracciare non appartiene alla classe *person*.

3.2.2 esempi di tracking

In figura 15, 16, 17 e 18 sono riportati 3 distinti frame di alcuni esempi di video annotati con *YOLO-CLIP* + DeepSORT.



Figure 15: Esempio di tracking con input: *a person with red t-shirt run away*.



Figure 16: Esempio di tracking con input: *a people with blue shirt walking to the camera*.

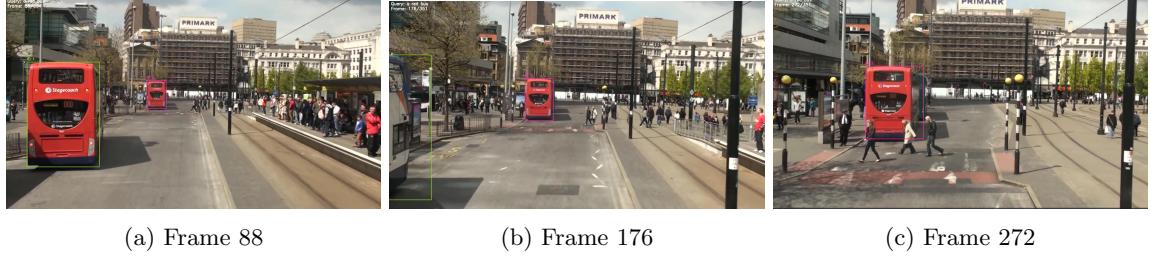


Figure 17: Esempio di tracking con input: *a red bus*.



Figure 18: Esempio di tracking con input: *man palying tennis with brown t-shirt*.

3.3 Tracking real-time

E' possibile usare il sistema *YOLO-CLIP + DeepSORT* per fare tracking real-time. In figura 19 è possibile osservare 3 frame distinti di un video real-time.



Figure 19: Esempio di tracking real-time con input: *a person with yellow hoodie*.

4 Conclusioni

In questo lavoro è stato sviluppato un sistema completo per la *object detection* e il *multi-object tracking* guidati da input testuali, combinando in modo efficace modelli di visione artificiale e comprensione del linguaggio naturale. L'integrazione tra YOLO, CLIP e DeepSORT ha permesso di costruire una pipeline multimodale capace non solo di individuare gli oggetti presenti in un'immagine o in un video, ma soprattutto di selezionare e tracciare nel tempo soltanto quelli semanticamente coerenti con una descrizione fornita dall'utente.

L'analisi ha evidenziato diversi aspetti rilevanti. Da un lato, gli *embeddings* originali di CLIP, pur essendo generali, mostrano limiti nel distinguere differenze fini tra oggetti appartenenti alla stessa classe. Per affrontare questo problema è stata progettata una rete di similarità che proietta tali rappresentazioni in uno spazio più discriminativo, migliorando la separabilità e consentendo di ottenere risultati più affidabili

nel confronto tra testo e immagine. I risultati sperimentali confermano l'efficacia di questo approccio, mostrando una maggiore robustezza nella scelta della soglia e una maggiore coerenza semantica nel filtraggio degli oggetti.

Parallelamente, l'integrazione con DeepSORT ha permesso di estendere il sistema al tracciamento, mantenendo identità coerenti nel tempo. L'utilizzo congiunto del modello di movimento basato su filtri di Kalman e delle feature di apparenza ha garantito un tracciamento stabile e accurato.

Nonostante i risultati positivi, il sistema presenta alcune limitazioni intrinseche. La performance dipende fortemente dalle classi disponibili in YOLO e dalla qualità delle descrizioni testuali, rendendo talvolta necessario un certo *prompt engineering*. Inoltre, il riconoscimento di oggetti non appartenenti alle classi del detector rimane un problema aperto. Anche le sottoclassi non esplicitamente presenti (come "giocatore", "bambino", "tifoso") possono causare errori di rilevamento o ambiguità semantiche.

La capacità di tracciamento del sistema è stata valutata quantitativamente su un sottoinsieme di sequenze del dataset *LaMOT*. I risultati ottenuti mostrano come il sistema sviluppato raggiunga prestazioni molto elevate nel tracciamento di singoli oggetti, mentre evidensi prestazioni inferiori nei contesti di *multi-object tracking*.

L'utilizzo degli embeddings della rete di similarità come descrittori di apparenza ha portato un miglioramento nel tracking nelle sequenze dove l'oggetto non appartiene alla classe *person*; al contrario nelle sequenze di *SportsMOT* si è registrato un peggioramento nelle performance in termini di HOTA e IDF1.

Nonostante tali limitazioni, è possibile affermare che il sistema presenti complessivamente buone capacità di tracciamento. Le prestazioni potrebbero essere ulteriormente migliorate estendendo il dataset di addestramento della rete di similarità, includendo dati più diversificati e non limitati esclusivamente alle immagini del dataset *COCO*.

Nel complesso, il sistema proposto è capace di comprendere, filtrare e seguire oggetti sulla base del loro significato e non solo della loro apparenza visiva.

References

- [1] Messina Falchi Bianchi, Carrara. Is clip the main roadblock for fine-grained open-world perception? *arXiv preprint arXiv:2401.01234*, 2024.
- [2] Zheng Chen. Object detection through simulated annealing for clip-yolo integration. *arXiv preprint arXiv:2305.08467*, 2023.
- [3] Manoj Dhanawade, Dhanashree Bhandigare, Rasika Bhor, Kirti Randhe, and Milind Ankleshwar. Integrating object detection with context-aware caption generation. *International Journal of Creative Research Thoughts (IJCRT)*, 13(5):o31–o38, 2025. IJCRT Paper ID: IJCRT25A5618.
- [4] Hoi Dongxu Li, Junnan Li. Blip: Bootstrapping language-image pre-training for unified vision-language understanding and generation. *arXiv preprint arXiv:2201.12086*, 2022.
- [5] LeCun Hadsell, Copra. Dimensionality reduction by learning an invariant mapping. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2006.
- [6] Xiaoqiong Liu Fan Zhang Li, Luke Liu. Lamot: Language-guided multi-object tracking. In *Proceedings of the IEEE/CVF International Conference on Computer Vision (ICCV)*, 2023.
- [7] Hallacy Ramesh Goh Agarwal Radford, Kim. Learning transferable visual models from natural language supervision. *arXiv preprint arXiv:2103.00020*, 2021.
- [8] Joseph Redmon and Farhadi Divvala, Girshick. You only look once: Unified, real-time object detection. In *Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition (CVPR)*, 2016.
- [9] Paulus Wojke, Bewley. Simple online and realtime tracking with a deep association metric. In *Proceedings of the IEEE International Conference on Image Processing (ICIP)*, 2017.