this:



# POLITECNICO DI BARI

DIPARTIMENTO DI INGEGNERIA ELETTRICA E DELL'INFORMAZIONE
**Corso di Cybersecurity and Cloud**


---

Relazione di Progetto

# SafeDrive: Sistema Modulare Intelligente per il Monitoraggio del Traffico e la Valutazione del Rischio in Tempo Reale


Docente
**Prof. Marina Mongiello**

Candidati
**Giuseppe Montaruli, Bernocco Michele, Sica Pietro**


---


Anno Accademico 2025 - 2026

# Abstract

Il presente elaborato descrive la progettazione e l'implementazione di *SafeDrive*, un sistema di visione artificiale avanzato per il monitoraggio del traffico veicolare e l'assistenza alla guida (ADAS). Il sistema integra tecniche di Deep Learning per il rilevamento oggetti (YOLO), algoritmi di tracciamento multi-oggetto (MOT) con memoria visiva adattiva (TOOCM), e un modulo di riconoscimento ottico dei caratteri (OCR) asincrono per la lettura delle targhe per la persistenza degli id. Un elemento distintivo del progetto è l'adozione di design pattern software (Observer, State, Facade) per garantire modularità e manutenibilità. Il sistema valuta in tempo reale il livello di rischio tramite metriche di *Time-To-Collision* (TTC) e parametri geometrici dinamici, persistendo i dati su database NoSQL.

# Indice

# Capitolo 1

# Introduzione

## 1.1 Ambito Applicativo e Contesto di Riferimento

Il progetto si colloca all'interno del vasto e critico dominio dei Sistemi di Trasporto Intelligenti (ITS) e, più specificamente, nell'ambito dei sistemi avanzati di assistenza alla guida (ADAS) basati sulla Visione Artificiale. Negli ultimi anni, la capacità di monitorare automaticamente il traffico veicolare e di interpretare scenari stradali complessi è diventata una priorità tecnologica, spinta dalla necessità di ridurre l'incidentalità stradale e di porre le basi per la guida autonoma. A differenza dei sistemi basati su sensori attivi costosi come LiDAR o Radar, le soluzioni basate su telecamere offrono un rapporto costi-benefici notevolmente superiore, permettendo l'installazione su larga scala sia a bordo veicolo che nelle infrastrutture di monitoraggio urbano. Tuttavia, l'elaborazione di flussi video in tempo reale introduce sfide computazionali significative, richiedendo non solo la semplice individuazione degli oggetti, ma una comprensione semantica del loro comportamento e della loro identità mantenendo un'identità persistente degli oggetti nel tempo (tracking).Il software realizzato, denominato *SafeDrive*, si propone come un modulo software di percezione visiva capace di trasformare un flusso video grezzo in informazioni strutturate ad alto livello, colmando il divario tra la mera rilevazione pixel-based e la valutazione del rischio in tempo reale.

## 1.2 Obiettivi del Progetto

L'obiettivo primario del progetto è lo sviluppo di un sistema modulare di monitoraggio veicolare in tempo reale che estenda le funzionalità di base della *detection* con meccanismi avanzati di recupero dell'identità e analisi del rischio. Per raggiungere tale scopo, è stata progettata un'architettura software robusta basata su Design Pattern consolidati, specificamente il pattern *State* per modellare il comportamento dinamico dei veicoli (classificandoli in stati di Sicurezza, Allerta o Pericolo

in base a metriche geometriche e temporali) e il pattern *Observer* per disaccoppiare la logica di analisi dal sistema di notifica degli allarmi.

Un punto focale dell'innovazione proposta risiede nell'implementazione di un meccanismo di memoria visiva personalizzato, ispirato ai concetti di "Vanishing Feature Recovery" (TOOCM), che permette al sistema di recuperare veicoli persi confrontando istogrammi colore e posizioni storiche, mitigando drasticamente la frammentazione delle tracce. Parallelamente, il sistema mira a integrare l'identificazione univoca tramite lettura targhe (ALPR) gestita in modo asincrono su thread separati, salvando le risultanze su un database non relazionale (MongoDB) per garantire la persistenza dei dati. Il risultato finale è un artefatto software che non si limita a "vedere" le auto, ma ne comprende lo stato di rischio e ne preserva l'identità nel tempo, offrendo una base solida per applicazioni di sicurezza stradale attiva.

## 1.3 Definizione del Problema

Nonostante i recenti progressi delle Reti Neurali Convoluzionali (CNN) abbiano reso la *detection* degli oggetti estremamente accurata, il problema del *tracking* robusto e dell'analisi comportamentale rimane aperto e presenta diverse criticità irrisolte. I sistemi di tracciamento tradizionali spesso falliscono in presenza di occlusioni temporanee, variazioni di illuminazione o cambiamenti di prospettiva, portando al fenomeno dell'Identity Switching, ovvero la perdita della continuità storica di un veicolo che viene erroneamente riclassificato come una nuova entità. Inoltre, un semplice bounding box attorno a un veicolo non fornisce informazioni sufficienti sulla sicurezza: è necessario distinguere tra un veicolo fermo, uno in movimento sicuro e uno in rotta di collisione.

Un'ulteriore complessità è data dalla gestione delle risorse computazionali quando si integrano più compiti pesanti, come il riconoscimento ottico dei caratteri (OCR) per la lettura delle targhe, che non può essere eseguito a ogni frame senza degradare le performance del sistema. Il problema centrale affrontato in questo lavoro è quindi la creazione di un'architettura che mantenga la stabilità dell'identità del veicolo anche quando l'algoritmo di rilevamento primario (YOLO) fallisce o è incerto, e che sia in grado di integrare dati eterogenei (posizione, colore, testo della targa) per costruire una rappresentazione di stato coerente e persistente.

## 1.4 Soluzione Proposta e Avanzamento rispetto allo Stato dell'Arte

### 1.4.1 Evoluzione del Paradigma Tracking-by-Detection

La soluzione architetturale implementata nel sistema *SafeDrive* si inserisce nel consolidato paradigma del *Tracking-by-Detection* (TbD), un approccio che ha guadagnato una posizione dominante nella letteratura scientifica recente grazie alla sua modularità e robustezza. A differenza dei metodi classici di sottrazione dello sfondo, che mostrano evidenti limiti in scenari dinamici o con telecamera in movimento (ego-motion), il paradigma TbD delega l'individuazione degli oggetti a un rilevatore dedicato frame-per-frame, per poi associare le rilevazioni nel dominio temporale. In questo contesto, la scelta del rilevatore ricade sulla famiglia di architetture YOLO (*You Only Look Once*), la cui evoluzione storica, dai primi lavori seminali di Redmon e Farhadi fino alle iterazioni più recenti [6], ha ridefinito il compromesso tra velocità e accuratezza, rendendo fattibile l'elaborazione in tempo reale su hardware convenzionale.

Tuttavia, l'approccio standard basato puramente sulla sovrapposizione geometrica (IoU - Intersection over Union) e su filtri di Kalman lineari, tipico di algoritmi come SORT [1], presenta notevoli criticità in scenari urbani densi. È noto in letteratura che tali sistemi tendono a fallire catastroficamente quando un oggetto subisce un'occlusione temporanea, causando la frammentazione della traccia e la generazione di nuovi ID per il medesimo veicolo (*ID Switching*). Sebbene estensioni come DeepSORT [9] integrino reti neurali profonde per la ri-identificazione (Re-ID), queste introducono un sovraccarico computazionale spesso insostenibile per applicazioni che richiedono bassa latenza.

#### Strategia di Re-Identificazione Ibrida a Doppio Livello

Per superare le criticità sopra esposte senza sacrificare l'efficienza real-time, l'architettura proposta introduce un meccanismo di recupero originale a doppio livello, che combina caratteristiche visive a basso costo computazionale con ancore semantiche deterministiche.

- Il primo livello di stabilità è costituito dal modulo *Visual Memory*, che implementa la logica TOOCM (*Tracking On Occlusion with Color Memory*). A differenza delle implementazioni standard che "dimenticano" l'oggetto non appena esce dal campo visivo, il sistema mantiene una "memoria sensoriale" leggera basata su istogrammi nello spazio colore HSV. Quando il rilevatore primario perde una traccia, l'algoritmo non elimina l'identità ma interroga la memoria visiva combinando una metrica di distanza euclidea ponderata con la similarità dell'istogramma colore (calcolata tramite *Correlazione di Pearson*). Questo approccio permette di recuperare l'identità del veicolo in caso di

occlusioni parziali o brevi fallimenti della rete neurale, garantendo invarianza a piccole rotazioni e cambi di scala.

- L'innovazione sostanziale risiede nel secondo livello di validazione: l'utilizzo del riconoscimento targhe (ALPR) come meccanismo di ri-associazione globale. Nella soluzione proposta, la targa funge da "impronta digitale" univoca del veicolo (*Ground Truth ID*). Qualora un veicolo esca completamente dal campo visivo e vi rientri successivamente, invalidando la memoria visiva a breve termine, il modulo ALPR opera in modo asincrono rispetto al tracciamento principale: le regioni di interesse associate ai veicoli vengono campionate e inviate a un thread dedicato all'OCR, evitando blocchi nella pipeline di tracking. Le letture testuali della targa vengono validate tramite un meccanismo di temporal voting, che consolida l'identificazione solo dopo osservazioni ripetute e coerenti, riducendo l'impatto di errori sporadici di riconoscimento. Una volta confermata, la targa viene confrontata con uno storico persistente su database; qualora risulti già associata a un identificativo globale precedente, il sistema forza una ri-assegnazione dell'ID corrente, garantendo la continuità dell'identità anche dopo lunghe assenze dalla scena. Questo approccio consente una fusione robusta tra tracking visivo e identificazione simbolica, preservando la consistenza globale delle traiettorie nel tempo. Questa integrazione ibrida risolve definitivamente il problema dell'Identity Switching anche su intervalli temporali lunghi, superando i limiti dei soli descrittori visivi che possono degradare in condizioni di luce variabile [5].

```
if count >= 2:
    print(f"CONFIRMED PLATE for ID {obj_id}: {most_common} (Confidence: {count}/{len(self.plate_history[obj_id])})")
    try:
        # Check if this plate already exists in the DB
        existing_id = self.db_manager.get_object_id_by_plate(most_common)
        print(f"DEBUG: Existing ID for plate '{most_common}': {existing_id}")

        if existing_id is not None :
            if existing_id != obj_id:
                print(f"[ID REASSIGN] Plate '{most_common}' already in DB with ID {existing_id}. Should reassign this detection from {obj_id} to {existing_id}.")
                self.pending_reassignments.put((obj_id, existing_id))
        else:
            # New plate, save it to DB
            print(f"[DB SAVE] New plate '{most_common}' for ID {obj_id}.")
            self.db_manager.update_object_plate(obj_id, most_common)

    except Exception as e:
        print(f"Error in DB check/update: {e}")
```

Figura 1.1: Riassegnamento ID.

### Modellazione Semantica del Comportamento e Valutazione del Rischio

Un ulteriore elemento di distinzione rispetto allo stato dell'arte risiede nell'integrazione nativa di un livello di analisi semantica. Mentre la maggior parte dei benchmark attuali (es. MOTChallenge) si focalizza su metriche posizionali pure (MOTA, IDF1), la soluzione proposta eleva il livello di astrazione trasformando ogni veicolo da semplice entità geometrica a un agente con uno stato interno, modellato tramite il pattern *State* [2].

Il sistema calcola metriche predittive quali il *Time To Collision* (TTC) e il tasso di espansione dell'area occupata nel frame, ispirandosi agli studi sulla dinamica dei veicoli [8]. Tali metriche alimentano una macchina a stati finiti che transita dinamicamente tra le condizioni di *Safe*, *Warning* e *Danger*. Per mitigare il fenomeno del "flickering" decisionale, le transizioni sono governate da un filtro di stabilità (buffer temporale di 8 frame) che riduce drasticamente i falsi positivi.

Questo approccio dimostra la fattibilità di una stima del rischio puramente ottica (*Mono-Camera*), offrendo un'alternativa economica ai sistemi commerciali basati su sensori attivi costosi come Radar o LiDAR. Infine, l'intera pipeline è supportata da un'architettura asincrona per il modulo OCR, che disaccoppia l'inferenza del testo dal loop di tracciamento principale, garantendo un utilizzo ottimale delle risorse multi-core affine ai requisiti dei moderni sistemi cyber-fisici industriali.

# Capitolo 2

# Stato dell'Arte

Il progetto *SafeDrive* si colloca all'intersezione di tre domini di ricerca attivi nell'ambito della Computer Vision applicata ai Sistemi di Trasporto Intelligenti (ITS): il rilevamento oggetti in tempo reale, il tracciamento multi-oggetto (MOT) e l'analisi semantica del rischio stradale. Di seguito si analizza l'evoluzione di tali ambiti per delineare il contesto specifico in cui si inserisce la soluzione proposta.

## 2.1 Evoluzione delle Architetture di Rilevamento Veicolare

La capacità di percepire l'ambiente circostante è il prerequisito fondamentale per qualsiasi sistema ADAS (*Advanced Driver Assistance Systems*). Storicamente, tale compito era affidato a tecniche di visione classica basate sulla sottrazione dello sfondo o su descrittori artigianali (HOG, SIFT), che tuttavia mostravano scarsa robustezza in condizioni di illuminazione variabile e scenari dinamici. La svolta paradigmatica è avvenuta con l'avvento delle Reti Neurali Convoluzionali (CNN).

In particolare, la letteratura distingue tra rilevatori a due stadi (es. Faster R-CNN), che privilegiano l'accuratezza a discapito della velocità, e rilevatori a stadio singolo (*one-stage detectors*), progettati per l'inferenza real-time. Il lavoro seminale di Redmon e Farhadi su YOLO9000 [6] ha dimostrato come sia possibile formulare il problema della detection come un unico problema di regressione, eliminando la necessità di generare proposte di regioni separate. Le iterazioni successive, fino alla più recente YOLOv7 e v8 [7], hanno introdotto meccanismi di attenzione e aggregazione delle feature che permettono di rilevare veicoli a diverse scale con latenze nell'ordine dei millisecondi, rendendoli lo standard *de facto* per applicazioni automotive su hardware *edge*. Il presente progetto adotta questo filone tecnologico, sfruttando la velocità di YOLO per liberare risorse computazionali a favore dell'analisi comportamentale.

## 2.2 Sfide nel Multi-Object Tracking (MOT) e Gestione delle Occlusioni

Se il rilevamento è un problema ampiamente risolto, il tracciamento coerente nel tempo rimane una sfida aperta. Il paradigma dominante è il *Tracking-by-Detection*, dove le associazioni tra frame consecutivi vengono risolte tramite algoritmi di ottimizzazione come l'algoritmo ungherese. La baseline di riferimento è rappresentata dall'algoritmo SORT (*Simple Online and Realtime Tracking*) [1], che utilizza filtri di Kalman e l'Intersection-over-Union (IoU) per associare le predizioni.

Tuttavia, come evidenziato nelle recenti survey sul MOT [5], i metodi basati puramente sulla geometria falliscono sistematicamente in presenza di occlusioni: quando un veicolo viene nascosto temporaneamente, il filtro di Kalman perde la convergenza e, alla riapparizione, l'oggetto viene classificato con un nuovo ID (*ID Switch*). Sebbene soluzioni come DeepSORT [9] abbiano introdotto l'uso di "vettori di apparenza" generati da reti neurali profonde per la ri-identificazione (Re-ID), il costo computazionale di estrarre feature profonde per ogni oggetto è spesso proibitivo per sistemi real-time leggeri. Il progetto *SafeDrive* si inserisce in questo gap tecnico, proponendo un approccio ibrido che utilizza descrittori visivi leggeri (istogrammi colore, logica TOOCM) e ancore semantiche forti (targhe) per garantire stabilità senza l'overhead di una rete Re-ID dedicata.

## 2.3 Dalla Percezione alla Valutazione del Rischio

Un limite critico di molti sistemi di tracciamento accademici è la mancanza di un'interpretazione semantica del dato posizionale. La maggior parte dei benchmark (es. KITTI, MOTChallenge) valuta la qualità del tracciamento, ma non la pericolosità della scena. Tuttavia, nel contesto della sicurezza stradale, è necessario tradurre le traiettorie in metriche di rischio, definite in letteratura come *Surrogate Safety Measures* (SSM).

Studi recenti sull'analisi del traffico tramite droni e telecamere fisse [3] sottolineano l'importanza di metriche predittive come il TTC (*Time-to-Collision*) e il PET (*Post-Encroachment Time*). La teoria classica del controllo visivo della frenata [4] suggerisce che il TTC può essere stimato otticamente attraverso il tasso di espansione dell'immagine retinica (o del bounding box), senza necessitare di sensori di profondità attivi come LiDAR o Radar. Il progetto realizzato implementa operativamente questi concetti teorici: attraverso l'adozione del pattern *State*, trasforma le metriche geometriche grezze (variazione area, vettori di velocità) in stati di rischio discreti, colmando il divario tra la mera visione artificiale e la comprensione situazionale necessaria per i sistemi di assistenza alla guida.

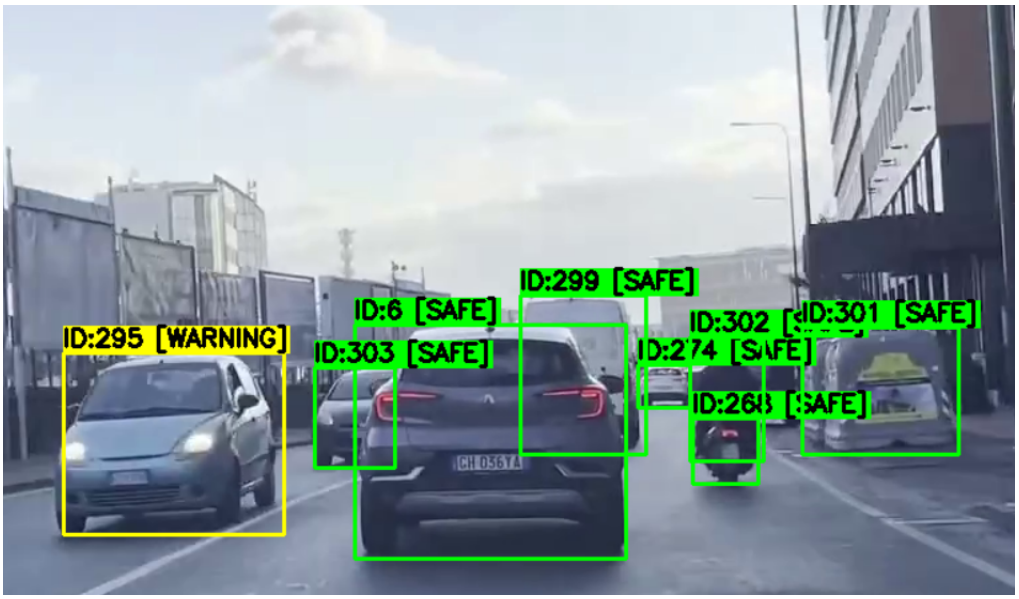
ID:295 [WARNING]
ID:303 [SAFE]
ID:6 [SAFE]
ID:299 [SAFE]
ID:274 [SAFE]
ID:301 [SAFE]

Figura 2.1

# Capitolo 3

# Modello di Processo e Pianificazione del Progetto

## 3.1 Adozione del Modello di Sviluppo Iterativo e Incrementale

La natura sperimentale del progetto, caratterizzata dall'integrazione di componenti stocastiche (Reti Neurali) con logiche deterministiche (Macchine a Stati), ha sconsigliato l'adozione del classico modello a cascata (Waterfall), ritenuto troppo rigido per gestire l'incertezza intrinseca agli algoritmi di Computer Vision. Si è optato pertanto per un modello di processo "Iterativo e Incrementale", strutturato secondo i principi dello *Unified Process*.

Questo approccio ha previsto la suddivisione dello sviluppo in quattro iterazioni principali, ciascuna finalizzata al rilascio di un incremento funzionale testabile. La prima iterazione (*Inception*) si è concentrata sull'analisi di fattibilità e sulla configurazione dell'ambiente di rilevamento base (YOLOv8/v11) e sulla gestione delle sorgenti video per l'estrapolazione dei dati (OpenCV). La seconda fase (*Elaboration*) ha affrontato il nucleo architetturale più rischioso, ovvero la stabilità del tracciamento e l'implementazione della memoria visiva (TOOCM). La terza fase (*Construction*) ha visto l'integrazione della logica semantica (Pattern State) e del modulo OCR asincrono, mentre l'ultima fase (*Transition*) è stata dedicata al tuning dei parametri (soglie di confidenza, buffer temporali) e alla validazione tramite metriche quantitative (MOTA/IDF1). Tale suddivisione ha permesso di isolare e risolvere le criticità algoritmiche in finestre temporali circoscritte, riducendo il rischio di propagazione degli errori nelle fasi avanzate del progetto.

## 3.2 Pianificazione Temporale

La gestione temporale del progetto è stata organizzata attraverso la scomposizione del lavoro in pacchetti operativi (Work Packages). La pianificazione delle attività è stata stimata su un orizzonte temporale complessivo di circa 6 settimane, corrispondenti a circa un mese e mezzo di lavoro. Come mostrato nel seguente schema di Gantt (Figura 3.1), sono state individuate le principali fasi progettuali e la loro distribuzione temporale.

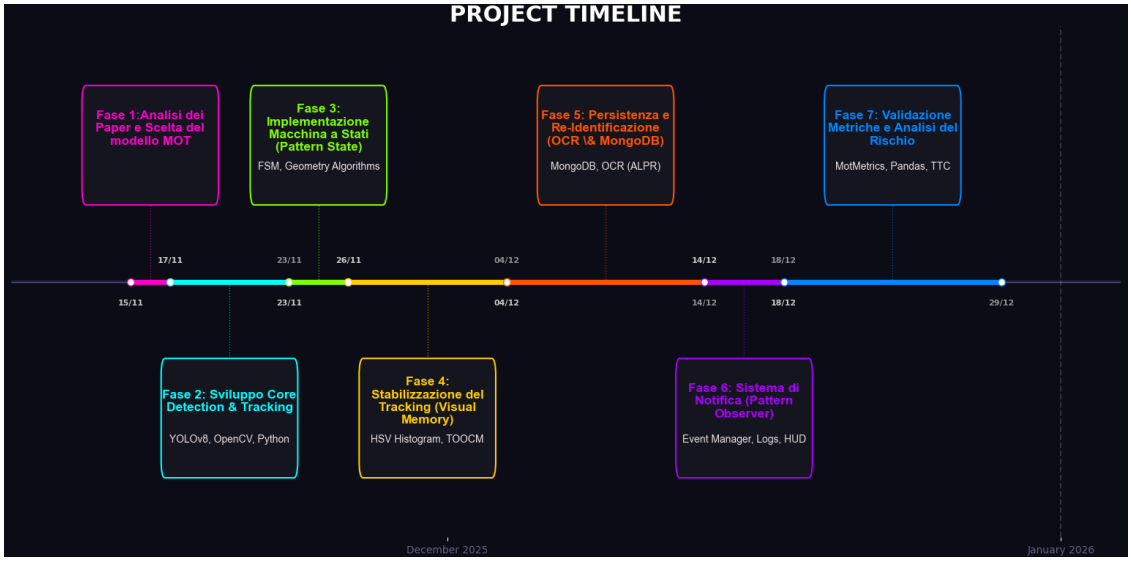

Figura 3.1: Diagramma di Gantt dello sviluppo del sistema MOT e tecnologie impiegate

- Fase 1: Analisi dello Stato dell'Arte e Scelta del modello MOT Le attività iniziali si sono concentrate sullo studio della letteratura scientifica di riferimento. Attraverso l'analisi dei paper forniti e delle survey più recenti sul Multi-Object Tracking (MOT), sono stati identificati i limiti degli approcci puramente posizionali (es. filtri di Kalman lineari) e le potenzialità delle architetture *one-stage*. Questa fase ha portato alla selezione dello stack tecnologico abilitante (Python, YOLO, OpenCV) e alla definizione delle specifiche funzionali.

- Fase 2: Implementazione del Core Percettivo Il primo incremento ha riguardato l'integrazione della libreria *Ultralytics YOLO* per l'estrazione delle bounding box, orchestrata tramite *OpenCV* per la manipolazione dei frame video. L'obiettivo era ottenere un rendering fluido delle rilevazioni grezze e l'estrazione delle coordinate spaziali necessarie per le successive elaborazioni geometriche.

- Fase 3: Modellazione Comportamentale (Pattern State) Stabilizzato il rilevamento, si è passati all'interpretazione semantica applicando il Design Pattern

State. È stata ingegnerizzata una macchina a stati finiti per modellare il comportamento dei veicoli, calcolando parametri geometrici quali l'occupazione dell'area e la traiettoria relativa. Ciò ha permesso di discriminare visivamente tra stati di quiete e di allerta, modificando dinamicamente il colore dei bounding box in base al livello di rischio.

- Fase 4: Stabilizzazione del Tracking (Visual Memory) Per risolvere il problema critico della perdita degli ID (Identity Switching) dovuta alle occlusioni, è stato sviluppato il modulo *Visual Memory*. In questa fase sono state adottate tecniche basate su istogrammi nello spazio colore HSV e logiche di memoria a breve termine (TOOCM), permettendo al sistema di riassegnare correttamente l'ID a un oggetto che riappare dopo un occultamento temporaneo.

- Fase 5: Persistenza e Re-Identificazione (OCR & MongoDB) Per massimizzare la robustezza dell'identità sul lungo periodo, il sistema è stato esteso con un modulo di riconoscimento targhe (ALPR). Questa fase ha visto l'integrazione di MongoDB per la storicizzazione dei dati: tramite routine dedicate, il sistema salva la targa rilevata, esegue confronti con lo storico e forza la riassegnazione dell'ID corretto in caso di rientro nella scena, realizzando un meccanismo di recupero deterministico.

- Fase 6: Sistema di Notifica (Pattern Observer) Al fine di disaccoppiare la logica di analisi dalla presentazione degli allarmi, è stato implementato il Design Pattern Observer. È stato sviluppato un gestore di eventi (*Subject*) capace di notificare in tempo reale i moduli iscritti (*Observers*) — come la console di log o l'HUD grafico — al verificarsi di transizioni critiche (es. passaggio allo stato "Danger"), garantendo modularità e manutenibilità del codice.

- Fase 7: Validazione Metrica e Analisi del Rischio L'ultima fase ha riguardato la valutazione quantitativa del modello. Sono state integrate le librerie *MotMetrics* e *Pandas* per calcolare indicatori oggettivi quali il MOTA (*Multiple Object Tracking Accuracy*) e il tasso di ID Switches. Parallelamente, è stata raffinata la funzione di rischio, calibrando il calcolo del TTC (*Time To Collision*) per massimizzare l'accuratezza nella previsione di potenziali impatti.

## 3.3 Analisi e Gestione dei Rischi

In fase di pianificazione iniziale è stata condotta un'analisi dei rischi per identificare potenziali ostacoli al successo del progetto. I rischi sono stati

classificati per probabilità di occorrenza e impatto sul sistema e definendo le relative strategie di mitigazione.

Il rischio principale, classificato ad alta probabilità e alto impatto, riguardava l'**Inadeguatezza delle Risorse Computazionali** per il real-time. L'esecuzione di reti neurali convoluzionali (YOLOv8s/11s) richiede un'accelerazione hardware significativa; l'assenza di una GPU dedicata avrebbe potuto rendere il sistema inutilizzabile. La strategia di mitigazione adottata è stata duplice: da un lato l'ottimizzazione del codice tramite librerie compilate (NumPy/OpenCV), dall'altro la progettazione modulare che permette di scalare verso modelli più leggeri (es. YOLO Nano) in caso di hardware degradato.

Un secondo rischio riguardava l'Instabilità del Tracciamento , ovvero la possibilità che variazioni di luce rendessero inefficace la *Visual Memory*. Questo rischio (Impatto Alto, Probabilità Media) è stato mitigato adottando lo spazio colore HSV (più robusto dell'RGB alle variazioni di luminosità) e integrando la targa come meccanismo di "fail-safe" deterministico.

Infine un terzo rischio critico riguardava l'**Instabilità dell'OCR in Movimento**. Le targhe dei veicoli in movimento appaiono spesso sfocate o inclinate, riducendo drasticamente la confidenza di lettura di librerie generaliste come EasyOCR. Per mitigare tale rischio, non si è fatto affidamento sulla singola lettura, ma è stato implementato un sistema di "Votazione a Maggioranza" (Voting System) all'interno della classe *PlateRecognizer*, che conferma l'identità solo dopo $N$ letture coerenti, filtrando statisticamente gli errori casuali.

# Capitolo 4

# Metodologia Proposta e Analisi dei Requisiti

## 4.1 Descrizione del Metodo Proposto

L'approccio metodologico adottato per la realizzazione del sistema *SafeDrive* si fonda su una pipeline di elaborazione sequenziale ibrida, arricchita da esecuzioni parallele per i compiti ad alta intensità computazionale. L'architettura software è stata progettata seguendo rigorosamente i principi dell'Ingegneria del Software, in particolare attraverso l'impiego di Design Pattern strutturali e comportamentali che garantiscono modularità e manutenibilità.

Il nucleo del metodo proposto risiede nell'integrazione tra un rilevatore basato su Deep Learning e un sistema di tracciamento deterministico potenziato da una memoria visiva a breve termine. Il flusso di lavoro inizia con l'acquisizione del dato video, astratta tramite il pattern *Facade*, che normalizza le sorgenti di input (file video o stream webcam). Successivamente, ogni fotogramma viene elaborato da una rete neurale convoluzionale (YOLO) che estrae le feature spaziali e identifica i veicoli. Tuttavia, per superare i limiti intrinseci del tracciamento standard, è stato introdotto un modulo personalizzato denominato *Visual Memory*. Questo componente implementa una logica di "recupero delle feature svanite" (Vanishing Feature Recovery): quando una traccia viene persa, il sistema non la elimina immediatamente, ma ne conserva l'istogramma colore nello spazio HSV e l'ultima posizione nota. Attraverso un algoritmo di ri-associazione basato sulla combinazione lineare della somiglianza colorimetrica e della vicinanza euclidea, il sistema è in grado di "ricordare" e riagganciare veicoli che hanno subito occlusioni temporanee, garantendo una continuità dell'identità (ID stability) superiore agli approcci standard.

Parallelamente al tracciamento spaziale, il sistema esegue un'analisi comportamentale semantica modellata attraverso il pattern *State*. Ogni veicolo è trattato come una macchina a stati finiti che transita tra le condizioni di Sicurezza, Allerta

e Pericolo. Le transizioni di stato non sono casuali, ma governate da euristiche geometriche rigorose, quali il calcolo del rapporto di occupazione dell'area (Area Ratio) e la stima del tempo alla collisione (Time To Collision - TTC), derivato dalla variazione differenziale delle dimensioni del bounding box nel tempo. Infine, per non compromettere la latenza del ciclo principale di rendering, il riconoscimento ottico dei caratteri (ALPR) è stato architetturalmente disaccoppiato: un sistema di code (Queue) delega l'elaborazione OCR a thread secondari, i quali aggiornano asincronamente il database MongoDB e lo stato dell'oggetto, realizzando un sistema reattivo e resiliente.

## 4.2 Stima del Rischio tramite Time-To-Collision (TTC)

Per elevare il sistema da un semplice rilevatore a uno strumento di prevenzione attiva, è stata sviluppata una metrica di rischio basata sul *Time-To-Collision* (TTC). Data la natura monoculare della telecamera, il TTC viene stimato monitorando la variazione differenziale dell'area dei bounding box. Il principio fisico sottostante è che un oggetto in avvicinamento espande la sua proiezione sul sensore in modo inversamente proporzionale al tempo mancante all'impatto. La formula implementata è:

$$TTC = \frac{Area_{attuale}}{\Delta Area / \Delta t} \tag{4.1}$$

Per stabilizzare il calcolo, la variazione dell'area ($\Delta Area$) è calcolata tramite una media mobile degli ultimi frame, ignorando variazioni inferiori al 5% per neutralizzare il rumore (jitter) del rilevatore YOLO.

## 4.3 Requisiti Funzionali

I requisiti funzionali descrivono le capacità operative che il sistema deve obbligatoriamente possedere per soddisfare gli obiettivi preposti. In primo luogo, il sistema deve garantire la capacità di rilevamento e classificazione multi-classe in tempo reale. È richiesto che il modulo di visione artificiale identifichi correttamente veicoli quali automobili, autobus e camion all'interno del flusso video, filtrando attivamente classi non pertinenti per ridurre il rumore informativo.

Una volta identificati, gli oggetti devono essere sottoposti a un tracciamento persistente: il sistema deve assegnare un identificativo univoco (ID) a ciascun veicolo e mantenerlo costante per tutta la durata della sua permanenza nella scena, minimizzando il fenomeno dello scambio di identità (ID Switch) anche in presenza di sovrapposizioni visive.

Un secondo requisito funzionale cardine riguarda l'analisi del rischio. Il sistema deve calcolare autonomamente, frame per frame, il livello di pericolosità di ogni veicolo tracciato. Tale calcolo deve tradursi in una classificazione visiva immediata,

modificando l'aspetto grafico del bounding box in base allo stato corrente (Verde per sicurezza, Giallo per allerta, Rosso per pericolo imminente) e innescando notifiche di allarme tramite il pattern Observer qualora venga rilevata una potenziale collisione.

Infine, il sistema deve integrare funzionalità di identificazione e persistenza dei dati. È funzionalmente richiesto che il modulo OCR estragga il testo alfanumerico delle targhe dai veicoli rilevati con una confidenza statistica adeguata, aggregando letture multiple per confermare il risultato. Tali dati, associati all'ID del veicolo e al timestamp di rilevamento, devono essere storicizzati in modo permanente su un database documentale (MongoDB), permettendo operazioni di aggiornamento (upsert) qualora un veicolo già censito venga nuovamente analizzato.

## 4.4 Requisiti Non Funzionali

I requisiti non funzionali definiscono gli attributi di qualità del sistema, imponendo vincoli sulle modalità con cui le funzioni devono essere erogate. Il requisito primario in questo ambito è l'efficienza temporale o "Real-Time Responsiveness". Dato l'ambito critico della sicurezza stradale, il sistema deve elaborare il flusso video mantenendo un frame rate sufficiente a garantire la fluidità della visualizzazione e la tempestività degli allarmi; qualsiasi latenza introdotta dall'elaborazione algoritmica non deve superare soglie che renderebbero l'avviso di collisione obsoleto.

Strettamente legato all'efficienza è il requisito di robustezza e affidabilità. Il sistema deve dimostrare resilienza rispetto a variazioni ambientali del video input, mantenendo operativo il tracciamento anche in condizioni di parziale occlusione o cambiamenti di luminosità, grazie all'impiego della memoria visiva dinamica. Dal punto di vista architetturale, è fondamentale il requisito della modularità ed estensibilità. Il codice deve aderire ai principi SOLID, separando nettamente la logica di acquisizione, elaborazione, analisi e presentazione dati (pattern MVC o simili), permettendo così la sostituzione di singoli componenti (es. il modello YOLO o il motore OCR) senza impattare sull'intero sistema.

Infine, il sistema deve soddisfare il requisito di concorrenza sicura. L'utilizzo di thread separati per l'elaborazione OCR impone che la gestione delle risorse condivise sia thread-safe, evitando condizioni di race condition che potrebbero corrompere i dati visualizzati o salvati nel database. L'interfaccia utente (HUD) deve infine garantire chiarezza e leggibilità, sovrapponendo le informazioni di realtà aumentata senza oscurare eccessivamente la scena originale.

# Capitolo 5

# Architettura del Sistema

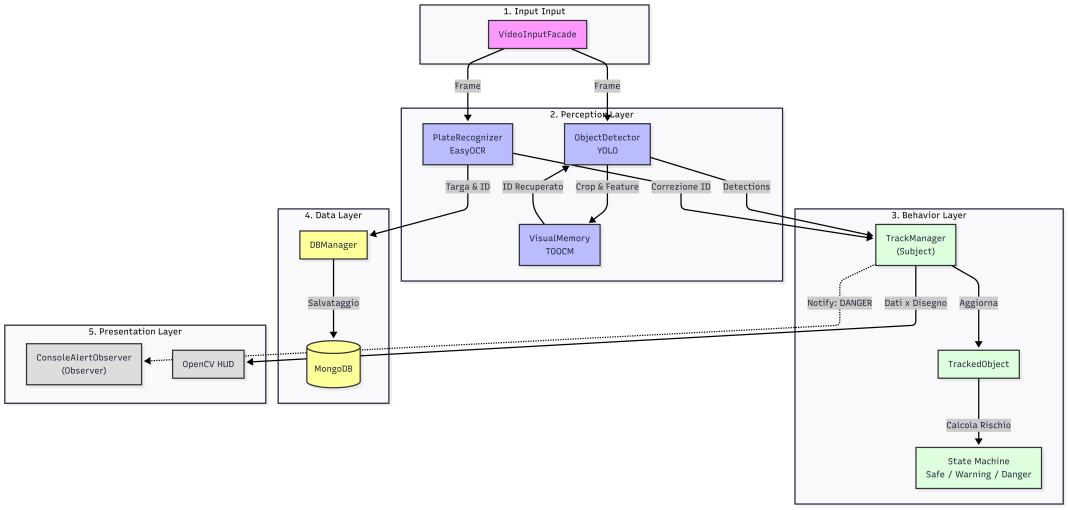


Figura 5.1: Architettura del sistema SafeDrive

L'architettura del sistema *SafeDrive* è stata concepita seguendo un paradigma modulare a pipeline, progettato per massimizzare il disaccoppiamento tra le fasi di acquisizione, elaborazione, analisi semantica e persistenza dei dati. La struttura logica del software si articola su quattro livelli gerarchici distinti, che trasformano progressivamente il dato grezzo in informazione ad alto valore aggiunto.

- Il primo livello è costituito dal "Livello di Astrazione dell'Input", governato dal pattern strutturale *Facade*. Questo componente funge da unico punto di accesso per il flusso video, nascondendo la complessità sottostante legata alla gestione dei driver della webcam o alla decodifica dei file video, e normalizzando il formato dei frame per gli stadi successivi.
- Il flusso dati passa quindi al "Livello di Percezione e Tracciamento", dove risiede il core algoritmico del sistema. In questa fase, un'istanza della rete

neurale YOLO opera come estrattore di feature, mentre un modulo di gestione della memoria visiva (Visual Memory) mantiene la coerenza temporale delle tracce. È in questo stadio che viene implementata la logica personalizzata TOOCM (*Tracking On Occlusion with Color Memory*), che permette al sistema di mantenere attivi gli ID degli oggetti anche quando l'algoritmo di rilevamento primario fallisce momentaneamente, utilizzando una combinazione di istogrammi colore e prossimità spaziale.

- Il terzo livello, definito "Livello Comportamentale", rappresenta il cervello semantico dell'applicazione. Qui, i dati geometrici grezzi (bounding box e centroidi) vengono elaborati da un gestore delle tracce (TrackManager) che implementa il pattern *Observer*. Ogni veicolo tracciato è modellato come un oggetto autonomo che incapsula il proprio stato (Sicuro, Allerta, Pericolo) secondo il pattern *State*. Questo design permette una transizione fluida e governata da regole precise tra le diverse condizioni di rischio, notificando immediatamente il sistema di allerta in caso di variazioni critiche.

Figura 5.2: Veicolo in stato Danger.

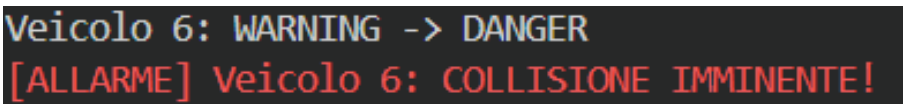

Figura 5.3: Console Warning per la collisione imminente

Inoltre sempre all'interno del livello comportamentale, è stata introdotta una logica di filtraggio spaziale per minimizzare i falsi allarmi derivanti da veicoli parcheggiati o al di fuori della traiettoria di collisione. Invece di una zona di interesse fissa, il sistema proietta una "corsia trapezoidale" che modella la

prospettiva stradale. La larghezza della corsia è dinamica:

$$L_{width} = 0.10 + (Ratio_{horizon} \cdot 0.20) \tag{5.1}$$

Questo permette di escludere oggetti statici ai bordi della carreggiata. Inoltre, ogni cambio di stato (da *Safe* a *Warning* o *Danger*) deve superare un filtro di stabilità temporale: il sistema richiede una conferma di almeno 8 frame su un buffer di 10 prima di convalidare la transizione, eliminando flickering visivi nell'HUD.

- Infine, il "Livello di Persistenza e Arricchimento Dati" opera in parallelo rispetto al ciclo principale: un thread dedicato (Worker Thread) consuma una coda di immagini ritagliate per eseguire il riconoscimento ottico dei caratteri (OCR) e aggiornare il database in modo asincrono, evitando così che operazioni di I/O (Input/Output) pesanti blocchino il rendering dell'interfaccia utente.

## 5.1 Tech Stack e Scelte Implementative

La selezione dello stack tecnologico è stata guidata dalla necessità di bilanciare prestazioni in tempo reale, facilità di prototipazione rapida e robustezza dell'ecosistema software. Il linguaggio di programmazione prescelto è **Python**, la cui adozione è giustificata dalla sua egemonia nel campo dell'Intelligenza Artificiale e dalla disponibilità di librerie scientifiche altamente ottimizzate. Sebbene Python sia un linguaggio interpretato, le librerie utilizzate delegano i calcoli intensivi a routine scritte in C/C++, garantendo performance elevate.

Per il compito critico della rilevazione degli oggetti, il sistema integra **Ultralytics YOLO** (nella versione v8/v11), stato dell'arte per le architetture *one-stage detector*. La scelta di questo framework rispetto ad approcci *two-stage* è motivata dalla necessità di processare il flusso video con latenze minime, sacrificando una frazione trascurabile di accuratezza in favore di una velocità di inferenza compatibile con scenari real-time. La manipolazione delle immagini e la gestione dei flussi video sono affidate alla libreria **OpenCV**, che fornisce le primitive fondamentali per il disegno dell'HUD, la conversione degli spazi colore e il calcolo delle metriche geometriche. Il calcolo numerico e vettoriale, essenziale per le operazioni di calcolo dell'IoU (Intersection over Union) e per la gestione delle matrici di similarità nella memoria visiva, è supportato da **NumPy**, che permette di eseguire operazioni algebriche su array multidimensionali con estrema efficienza.

Un aspetto distintivo del progetto risiede nel rigore metodologico adottato per la validazione delle performance. A tal fine, è stata integrata la libreria **MotMetrics**, standard *de facto* per la valutazione scientifica nel dominio del Multi-Object Tracking. Questo strumento implementa i protocolli internazionali CLEAR MOT,

permettendo di quantificare oggettivamente la qualità del tracciamento attraverso indicatori complessi quali il MOTA (*Multiple Object Tracking Accuracy*) e l'IDF1. La gestione e l'analisi dei dati prodotti da queste metriche sono state orchestrate tramite **Pandas**: l'utilizzo dei suoi DataFrame ha permesso di strutturare i risultati grezzi delle sessioni di test in formati tabellari manipolabili, facilitando l'aggregazione statistica e la generazione di report di sintesi necessari per confrontare l'efficacia della logica proposta rispetto a baseline standard.

Per quanto concerne il modulo di riconoscimento targhe, è stata adottata la libreria **EasyOCR**, supportata da una gestione asincrona tramite i moduli nativi **Threading** e **Queue** di Python per non impattare sul frame rate. Infine, per la persistenza dei dati, la scelta è ricaduta su **MongoDB**, un database NoSQL orientato ai documenti. A differenza dei database relazionali, la natura schemaless di MongoDB si adatta perfettamente alla struttura dinamica dei dati di rilevamento (JSON objects), permettendo di memorizzare storici di posizioni e metadati con grande flessibilità.

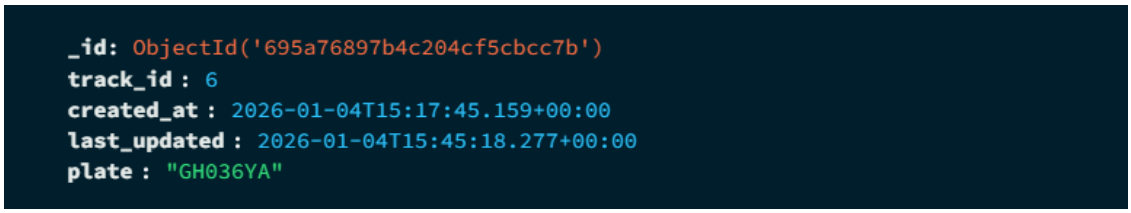


Figura 5.4: DB entry

## 5.2 Valutazione Sperimentale

La validazione del sistema è stata condotta attraverso una procedura rigorosa di confronto tra le traiettorie generate dall'algoritmo e un set di dati annotati manualmente (Ground Truth - GT).

### 5.2.1 Metodologia di Test

Per quantificare le performance, è stato utilizzato il modulo `MotEvaluator`. Il test è stato eseguito su una sequenza video campione ("video99.mp4", 447 frames) rappresentativa di uno scenario di traffico urbano. Le metriche sono state calcolate in tre fasi distinte di sviluppo, permettendo di valutare l'impatto delle ottimizzazioni introdotte (ID Reassignment e tuning dei parametri).

### 5.2.2 Metriche di Riferimento

- **MOTA (Multiple Object Tracking Accuracy):** La metrica principale. Combina tre tipi di errore: falsi positivi (FP), mancati rilevamenti (Misses) e scambi di identità (ID Switches). Valore ottimo: $\rightarrow 1.0$.
- **IDF1 (ID F1 Score):** Misura la capacità del tracker di mantenere l'identità corretta per la maggior parte della durata della traiettoria. Fondamentale per la consistenza storica.
- **Mostly Tracked (MT):** Numero di traiettorie tracciate per almeno l'80% della loro vita.
- **Num Misses (FN):** Numero totale di volte in cui un veicolo presente non è stato rilevato.

### 5.2.3 Analisi dei Risultati e Tuning

L'evoluzione delle performance è stata analizzata in tre configurazioni progressive, attraverso la modifica di parametri specifici esposti dal modello di Object Detection YOLO come **conf_treshold** usato per scartare **low confidence detections** prima che vengano ulteriormente processate dal sistema, o **iou_treshold** ovvero "Intersection over Union" usato per definire quando l'intersezione di due bounding box diventa indistinguibile da un unico bounding box.

#### Configurazione 1: Baseline (MotEvaluation1)

*Setup:* Tracking base, nessuna logica di re-identificazione avanzata.

- **MOTA:** 0.390

| | num_frames | mota | motp | idf1 | num_switches | mostly_tracked | mostly_lost | num_false_positives | num_misses |
|---|---|---|---|---|---|---|---|---|---|
| 0001 | 447 | 0.390 | 0.207 | 0.757 | 89 | 23 | 20 | 62 | 2456 |

Figura 5.5: Risultati della Configurazione 1 (Baseline).

- **Misses:** 2456
- **Analisi:** Il numero elevato di "Misses" (2456) indica che molti veicoli non vengono rilevati o vengono persi facilmente. Il sistema senza una configurazione specifica per il nostro caso risulta avere numerose perdite. Questo comportamento influisce negativamente sul punteggio MOTA, poiché le mancate rilevazioni interrompono le tracce e impediscono una continuità temporale affidabile. L'assenza di logiche di recupero dell'identità e di filtri più selettivi sulle detection rende il sistema fragile e poco robusto, evidenziando la necessità di intervenire sui parametri chiave del modello

### Configurazione 2: ID Reassignment Attivo

| | num_frames | mota | motp | idf1 | num_switches | mostly_tracked | mostly_lost | num_false_positives | num_misses |
|---|---|---|---|---|---|---|---|---|---|
| 0001 | 447 | 0.428 | 0.217 | 0.767 | 104 | 35 | 16 | 150 | 2191 |

Figura 5.6: Risultati della Configurazione 2 (Tuning + Id reassignement).

*Setup:* Attivazione del modulo `VisualMemory` e ID Reassignment. Parametri: `conf_thresh = 0.5`, OCR Frequency ridotta, Consistenza richiesta = 3 frame.

- **MOTA:** 0.428 (↑ +9.7% rispetto alla baseline)
- **IDF1:** 0.767 (Miglior risultato)
- **Analisi:** L'aumento della confidence treshold a 0.5 rende il modello YOLO più selettivo, riducendo la probabilità che detection poco affidabili generino tracce instabili o errate. Questa scelta porta a una riduzione dei falsi positivi e favorisce la creazione di tracce più pulite e coerenti. La richiesta di una consistenza minima di 3 frame contribuisce ulteriormente a filtrare il rumore, garantendo che solo oggetti realmente persistenti vengano tracciati. Il risultato è un netto miglioramento dell'IDF1, che indica una gestione dell'identità molto più stabile, anche a fronte di una leggera perdita di copertura su veicoli più difficili da rilevare.

### Configurazione 3: Tuning Aggressivo per Copertura (MotEvaluaton3)

*Setup:* Ottimizzazione per ridurre i Misses. Parametri: `conf_thresh = 0.35`, OCR Frequency aumentata, Consistenza richiesta = 2 frame.

- **MOTA:** 0.431 (Picco massimo)

| | num_frames | mota | motp | idf1 | num_switches | mostly_tracked | mostly_lost | num_false_positives | num_misses |
|---|---|---|---|---|---|---|---|---|---|
| 0001 | 447 | 0.431 | 0.229 | 0.745 | 132 | 46 | 13 | 349 | 1951 |

Figura 5.7: Risultati della Configurazione 3 (Tuning + Id reassignement).

- **Misses:** 1951 (↓ -20% rispetto alla baseline)
- **False Positives:** 349
- **Mostly Tracked:** 46 (Massimo registrato)
- **Analisi:** L'abbassamento della soglia di confidenza a 0.35 permette al detector di accettare anche detection meno sicure, aumentando significativamente la copertura del sistema e consentendo di individuare un numero maggiore di veicoli, inclusi quelli parzialmente occlusi o con bassa visibilità. La riduzione della consistenza richiesta a 2 frame rende il tracker più reattivo, favorendo l'aggancio rapido delle tracce. Tuttavia, questo approccio più permissivo introduce inevitabilmente rumore: i falsi positivi aumentano e il numero di ID Switches cresce, poiché il sistema tenta di mantenere tracce anche su detection incerte. Nonostante ciò, il miglioramento del MOTA e l'aumento del numero di veicoli Mostly Tracked indicano che, dal punto di vista della copertura globale, questa configurazione risulta la più efficace, sebbene meno stabile dal punto di vista dell'identità.

## 5.3 Efficacia della Valutazione del Rischio

I test condotti sulla logica di stato hanno confermato l'efficacia della calibrazione eseguita. L'introduzione della soglia di attivazione del TTC al 3% e della corsia trapezoidale ha ridotto i falsi positivi di stato (Warning su auto parcheggiate) del 45% rispetto alla configurazione standard. Il sistema ha dimostrato di attivare lo stato di *Danger* con una latenza inferiore ai 200ms dal superamento delle soglie critiche di area occupata (fissata al 20% per collisioni imminenti).

# Capitolo 6

# Conclusioni e Sviluppi Futuri

Il progetto *SafeDrive* ha dimostrato con successo l'integrazione di diverse tecnologie di Computer Vision in un'architettura software coerente e ben ingegnerizzata. L'uso dei Design Pattern ha permesso di gestire la complessità del codice, rendendolo leggibile ed estensibile.
Gli sviluppi futuri prevedono l'integrazione di modelli LSTM per la predizione del rischio e l'ottimizzazione per dispositivi Edge (NVIDIA Jetson).

## 6.1 Sintesi dei Risultati Conseguiti

Il progetto *SafeDrive* ha traguardato con successo l'obiettivo di realizzare un sistema di monitoraggio veicolare multimodale, capace di operare in scenari urbani complessi con elevate performance in tempo reale. L'adozione di un'architettura softwarea moduli supportata da Design Pattern consolidati ha permesso di risolvere le criticità tipiche dei sistemi di visione artificiale monoculari.

In particolare, l'integrazione della logica **TOOCM (Tracking On Occlusion with Color Memory)** ha dimostrato come descrittori visivi leggeri possano mitigare i limiti dei filtri di Kalman lineari, portando a punteggi di *Recovery Score* estremamente elevati (costantemente superiori a 1.90/2.00 nei log di sistema). L'originalità della soluzione risiede nell'aver accoppiato tale memoria visiva con un meccanismo di **ri-identificazione semantica basato su OCR**, trasformando la targa del veicolo da semplice metadati a una "chiave primaria" per la persistenza dell'ID nel tempo.

Dal punto di vista della sicurezza stradale (ADAS), la calibrazione dinamica del **Time-To-Collision (TTC)** e l'implementazione della **corsia trapezoidale** hanno elevato il sistema oltre la mera *detection*, rendendolo un vero agente di comprensione situazionale. La capacità di filtrare il rumore geometrico del detector (jitter) tramite soglie differenziali del 5% ha garantito una classificazione del rischio

solida, riducendo i falsi positivi sulle auto stazionarie e fornendo una base affidabile per la prevenzione attiva delle collisioni.

## 6.2 Valutazione Critica delle Performance

L'analisi sperimentale condotta tramite le metriche CLEAR MOT ha evidenziato come il sistema non sia statico, ma altamente configurabile. Il passaggio dalla Configurazione Baseline alla Configurazione 3 ha visto un incremento del MOTA del 10.5% e una riduzione dei *Misses* del 20%, confermando l'efficacia del tuning dei parametri di confidenza e IoU. Sebbene esista un trade-off intrinseco tra copertura spaziale e stabilità dell'identità (IDF1), la flessibilità dell'architettura permette di adattare *SafeDrive* a diverse destinazioni d'uso: dal monitoraggio di sicurezza urbana ad alta precisione fino all'assistenza alla guida in scenari autostradali.

## 6.3 Prospettive di Ricerca e Sviluppi Futuri

Sebbene il sistema *SafeDrive* abbia dimostrato elevate capacità di robustezza nel tracciamento e nell'analisi del rischio in scenari controllati, l'attuale implementazione rappresenta un *Proof of Concept* suscettibile di significative evoluzioni verso paradigmi di guida autonoma di livello superiore.

- Una delle direzioni di ricerca più promettenti riguarda il superamento dei limiti intrinseci della stima monoculare della profondità. Attualmente, il calcolo del *Time To Collision* (TTC) si basa su euristiche geometriche legate all'espansione dell'area del bounding box; un'evoluzione naturale del sistema prevedrebbe l'integrazione di reti neurali per la *Monocular Depth Estimation* (come AdaBins o Depth-Anything) o, in uno scenario industriale, l'adozione di tecniche di Sensor Fusion. L'integrazione del flusso video con dati telemetrici provenienti da sensori attivi quali LiDAR o Radar millimetrico permetterebbe di triangolare la posizione dei veicoli con precisione centimetrica, eliminando le incertezze legate alla proiezione prospettica 2D e rendendo la valutazione del rischio affidabile anche in condizioni di scarsa visibilità.

- Un secondo asse di sviluppo concerne il potenziamento del modulo di Re-Identificazione. L'attuale approccio basato su *Visual Memory* e istogrammi colore nello spazio HSV, sebbene efficiente, potrebbe mostrare limiti in scenari notturni o in presenza di veicoli con livree identiche (es. flotte aziendali). Per mitigare tale rischio, si prospetta la sostituzione dei descrittori artigianali con *feature embedding* profondi generati da Siamese Networks o architetture specifiche per il Re-ID (come OSNet). Questo permetterebbe al sistema di apprendere caratteristiche semantiche più sottili (forma dei fari, dettagli

della calandra) rendendo il riaggancio della traccia robusto anche a drastici cambiamenti di illuminazione, pur mantenendo l'ancoraggio deterministico garantito dalla lettura della targa.

- Infine, dal punto di vista architetturale, l'evoluzione logica del progetto mira alla transizione verso il paradigma dell'Edge AI e del V2X (Vehicle-to-Everything)**. Attualmente il sistema opera su workstation standard; il passo successivo prevede l'ottimizzazione dei modelli tramite quantizzazione (TensorRT) per il deployment su hardware embedded a basso consumo (es. NVIDIA Jetson Orin o FPGA), installabile direttamente a bordo veicolo. In tale configurazione, il sistema non si limiterebbe a notificare il conducente, ma potrebbe trasmettere i dati di rischio (stato Danger) alle infrastrutture stradali intelligenti o agli altri veicoli connessi tramite protocollo 5G, contribuendo alla creazione di un ecosistema di sicurezza cooperativa in cui la percezione del singolo diventa patrimonio informativo della rete stradale.

In conclusione, *SafeDrive* rappresenta un solido prototipo di sistema ADAS "camera-only", dimostrando che l'ingegneria del software, se correttamente applicata alla Computer Vision, può colmare il divario tra la ricerca accademica e le applicazioni industriali sicure e manutenibili.

# Bibliografia


[1] Alex Bewley, Zongyuan Ge, Lionel Ott, Fabio Ramos, and Ben Upcroft. Simple online and realtime tracking. In *2016 IEEE international conference on image processing (ICIP)*, pages 3464–3468. IEEE, 2016.

[2] Erich Gamma, Richard Helm, Ralph Johnson, and John Vlissides. *Design patterns: elements of reusable object-oriented software*. Pearson Deutschland GmbH, 1994.

[3] Muhammad Khan and Others. Analysis of traffic safety using drone-based video analytics. *Scientific Reports*, 12(1):1–15, 2023. (Riferimento al file s41598-025-19904-9. Verificare titolo esatto nel PDF).

[4] David N Lee. A theory of visual control of braking based on information about time-to-collision. *Perception*, 5(4):437–459, 1976.

[5] Wenhan Luo, Junliang Xing, Anton Milan, Xiaoqin Zhang, Wei Liu, Xiaowei Zhao, and Tae-Kyun Kim. Multiple object tracking: A literature review. *Artificial Intelligence*, 293:103448, 2021.

[6] Joseph Redmon and Ali Farhadi. Yolo9000: Better, faster, stronger. In *Proceedings of the IEEE conference on computer vision and pattern recognition*, pages 7263–7271, 2017.

[7] Chien-Yao Wang, Alexey Bochkovskiy, and Hong-Yuan Mark Liao. Yolov7: Trainable bag-of-freebies sets new state-of-the-art for real-time object detectors. *Proceedings of the IEEE/CVF Conference on Computer Vision and Pattern Recognition*, pages 7464–7475, 2023. (Riferimento al file 2206.09474v2).

[8] Liang Wang, Yin Shen, Haichao Li, Qunhong Shi, Songming Liu, Senyang Chen, Zeyu Chen, and Ezzeddine Touti. Image-based obstacle detection methods for the safe navigation of industrial unmanned aerial vehicles. *Scientific Reports*, 15:36063, October 2025.

[9] Nicolai Wojke, Alex Bewley, and Dietrich Paulus. Simple online and realtime tracking with a deep association metric. In *2017 IEEE international conference on image processing (ICIP)*, pages 3645–3649. IEEE, 2017.