

UNIVERSITÀ DEGLI STUDI DI PADOVA

DIPARTIMENTO DI MATEMATICA

CORSO DI LAUREA IN INFORMATICA

---

# Riconoscimento e tracciamento di elementi su video ad alta risoluzione

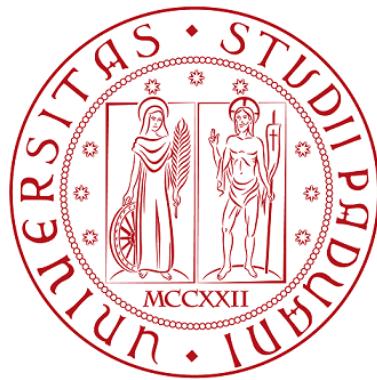
---

*Laureando:*

Davide LIU

*Relatore:*  
Prof. Lamberto BALLAN

*Tutor aziendale:*  
Leonardo DAL ZOVO



Anno Accademico 2018/2019

---

# Indice

<b>1 Introduzione</b>	<b>6</b>
1.1 Note esplicative . . . . .	6
1.2 Struttura del documento . . . . .	6
<b>2 Ambiente Aziendale</b>	<b>8</b>
<b>3 Analisi dei problemi</b>	<b>9</b>
3.1 Tecniche di computer vision . . . . .	9
3.2 Computer vision applicata su immagini ad alta risoluzione . . . . .	11
3.3 Frammentazione dell'immagine . . . . .	12
3.4 Tecniche di object tracking . . . . .	14
3.5 Object tracking con continue object detections . . . . .	15
<b>4 Progettazione</b>	<b>17</b>
4.1 Progettazione algoritmo per riconoscimento di elementi in un'immagine frammentata . . . . .	17
4.1.1 Scomposizione del frame originale in regioni . . . . .	17
4.1.2 Rimozione degli elementi individuati più volte all'interno delle aree di sovrapposizione . . . . .	18
4.1.3 Creazione di raggruppamenti di labels correlate . . . . .	18
4.1.4 Miglioramento: raggruppamenti di labels utilizzati come region proposal	21
4.2 Progettazione algoritmo per tracciamento di elementi . . . . .	22
4.2.1 Filtro di Kalman . . . . .	22
4.2.2 Assegnazione detection-tracker . . . . .	23
4.2.3 Gestione delle detections e dei trackers non assegnati . . . . .	25
4.2.4 Utilizzo di metriche di supporto per l'assegnazione detection-tracker .	26
<b>5 Tecnologie</b>	<b>28</b>
5.1 Python . . . . .	28
5.2 Pycharm . . . . .	28
5.3 Tensorflow . . . . .	29
5.4 OpenCV . . . . .	29
5.5 Numpy . . . . .	29
5.6 Matplotlib . . . . .	30
5.7 Pytest . . . . .	30

---

<b>6 Sviluppo</b>	<b>31</b>
6.1 Sviluppo algoritmo per ricomposizione delle labels in un'immagine frammentata	31
6.1.1 Implementazione . . . . .	31
6.1.2 Test di integrazione . . . . .	33
6.1.3 Pipeline completa . . . . .	33
6.2 Sviluppo sistema di tracking . . . . .	35
<b>7 Risultati ottenuti</b>	<b>37</b>
7.1 Metriche utilizzate per detection su singole immagini . . . . .	37
7.1.1 Precision . . . . .	37
7.1.2 Recall . . . . .	37
7.1.3 F1 . . . . .	38
7.1.4 Intersection over union . . . . .	38
7.1.5 Average Precision . . . . .	38
7.1.6 Mean Average Precision . . . . .	39
7.2 Dataset utilizzato per detection su singole immagini . . . . .	39
7.2.1 Impostazione parametri detection . . . . .	40
7.3 Risultati ottenuti nella detection su singole immagini . . . . .	41
7.3.1 Risultati per categoria . . . . .	41
7.3.2 Risultati per parametri . . . . .	45
7.4 Metriche utilizzate per tracking su video . . . . .	45
7.4.1 Mostly Tracked Trajectories . . . . .	46
7.4.2 Mostly Lost Trajectories . . . . .	46
7.4.3 ID switches . . . . .	46
7.4.4 Fragments . . . . .	46
7.4.5 MOTP . . . . .	46
7.4.6 MOTA . . . . .	47
7.4.7 IDF1 . . . . .	47
7.5 Dataset utilizzato per tracking su video . . . . .	47
7.5.1 Impostazione parametri tracking . . . . .	48
7.6 Risultati ottenuti nel tracking su video . . . . .	49
7.6.1 Risultati per categoria . . . . .	49
7.6.2 Risultati per parametri . . . . .	50
7.6.3 Risultati per parametri con detection . . . . .	52
<b>8 Glossario</b>	<b>54</b>
<b>9 Bibliografia</b>	<b>56</b>



---

## Elenco delle tabelle

1	Lista delle categorie del dataset utilizzato per effettuare i test con relativo AP	43
2	Risultati generali detection con diversi parametri	45
3	Risultati tracking per categoria	49
4	Risultati tracking per parametri	50
5	Risultati tracking per parametri	52

---

## Elenco delle figure

1	Logo di Studiomapp . . . . .	8
2	Esempio di un'immagine con box, categoria e probabilità per ogni elemento riconosciuto in essa . . . . .	10
3	Esempio di object detection in un frame di un video in 4K . . . . .	12
4	Esempio di un'immagine in alta risoluzione suddivisa in regioni senza sovrapposizioni . . . . .	13
5	Frames di un video nei quali viene tracciata un' auto (ordinati da sinistra a destra e dall'alto al basso) . . . . .	15
6	Esempi di labels erroneamente individuate a causa della frammentazione e relativa corretta ricostruzione . . . . .	19
7	Esempi di labels erroneamente individuate a causa della frammentazione e relativa errata ricostruzione . . . . .	21
8	Esempio di assegnazione detection-tracker . . . . .	25
9	Logo di Python . . . . .	28
10	Logo di Tensorflow . . . . .	29
11	Diagramma delle classi del sistema di tracking . . . . .	35
12	Esempio di intersezione e di unione . . . . .	38
13	Immagine satellitare appartenente al dataset utilizzato . . . . .	40
14	Esempi di oggetti molto difficili da individuare e classificare . . . . .	44

---

# 1 Introduzione

Negli utili anni la computer vision è diventata un ambito di ricerca molto importante sia nel mondo accademico, sia per le sue applicazioni nel mondo reale. I due sotto-problemi principali nei quali essa si suddivide sono la detection ed il tracking.

Il primo problema ha come compito quello di insegnare ad una macchina ad interpretare una singola immagine mentre il secondo problema estende lo stesso compito ma nell'ambito dei video, ovvero una sequenza di immagini dette frames. La detection è già ampiamente utilizzata in ambito commerciale per esempio nei sistemi di riconoscimento facciale, riconoscimento e lettura di testi a mano, diagnosi mediche, controllo di prodotti industriali, analisi del territorio etc.

Il tracking è invece un argomento di ricerca più recente rispetto alla detection e trova applicazione nei sistemi di video-sorveglianza, veicoli a guida autonoma, riconoscimento di azioni etc. tuttavia alcuni di questi sistemi come per esempio i veicoli a guida autonoma non sono ancora maturi e sono tuttora oggetto di sperimentazioni.

## 1.1 Scopo del progetto

Lo scopo del progetto di stage è quello di progettare e realizzare un sistema di riconoscimento e tracciamento di specifici elementi all' interno di un video ad alta risoluzione.

Questo progetto comporta sfide e complessità aggiuntive rispetto all' analisi degli elementi presenti in una singola immagine sia per il fatto che un video è composto da una sequenza di frames anziché da una singola immagine, sia per il fatto che i frames trattati sono in alta definizione e quindi elaborare l'intero frame con una sola detection comporterebbe una perdita di qualità significativa.

Riassumendo, i due problemi principali che sono stati affrontati, in ordine sequenziale, sono i seguenti:

- Riconoscimento di specifici elementi in immagini con frammentazione;
- Tracciamento di specifici elementi in un video;

Ognuno dei sotto-problemi viene prima analizzato a fondo, in seguito ne viene discussa una sua possibile soluzione ed infine viene mostrato come essa è stata realizzata ai fini di ottenere un prodotto il più performante possibile. Alla fine, il prodotto finale viene realizzato come combinazione dei due sotto-prodotti.

---

## **1.2 Note esplicative**

Allo scopo di evitare ambiguità a lettori esterni, si specifica che all'interno del documento verranno inseriti dei termini con un carattere 'G' come pedice, questo significa che il significato inteso in quella situazione è stato inserito nel Glossario

## **1.3 Struttura del documento**

Il documento è organizzato nel seguente modo. Nel capitolo 2 viene fornita una panoramica riguardante l'azienda presso la quale è stata svolta l'attività di stage. Il capitolo 3 analizza i problemi da affrontare e le varie tecniche utilizzate nello stato dell'arte per provare a risolverli. Il capitolo 4 esamina approfonditamente gli algoritmi sviluppati per far fronte ai problemi proposti mentre il capitolo 5 spiega quali tecnologie e strumenti sono stati impiegati per la loro realizzazione. Il capitolo 6 riguarda l'implementazione degli algoritmi ed il 7 ne riporta i risultati ottenuti. Infine sono presenti il Glossario e la Bibliografia.

## **1.4 Struttura del lavoro**

Lo stage ha avuto una durata di circa 320 ore produttive, di queste, almeno 40 ore sono state utilizzate per lo studio delle tecnologie utilizzate le quali includono il linguaggio di programmazione Python e alcune delle sue librerie, tra cui Numpy, OpenCV e Tensorflow. Circa 60 ore sono state impiegate per lo studio delle basi di machine learning e computer vision e per lo studio delle soluzioni più comuni ai problemi da affrontare. Altre 40 ore sono state impiegate per ideare e progettare delle soluzioni efficaci per i risolvere i problemi descritti sulla base di soluzioni già esistenti. Lo sviluppo dei prodotti realizzati ha richiesto circa 140 ore di lavoro, compresi i relativi test di validazione. Le restanti 40 ore sono state dedicate alla raccolta ed analisi dei risultati ed alla stesura della documentazione. L'elaborazione dei risultati ha richiesto in totale circa una settimana di tempo ed è stata eseguita su una macchina appositamente dedicata e sempre funzionante in modo da poter svolgere altre attività in parallelo.

---

## 2 Ambiente Aziendale



Figura 1: Logo di Studiomapp

STUDIOMAPP, fondata a fine 2015, è una startup innovativa con sedi a Ravenna e Roma, in Italia. Sviluppa algoritmi di intelligenza artificiale specifici per Geo-calcolo e dati geo-spaziali in modo da fornire soluzioni innovative per smart cities, mobilità, trasporti e logistica, turismo e beni culturali, immobiliare e real estate, agricoltura, territorio e gestione delle risorse naturali, adattamento ai cambiamenti climatici.

E' la prima startup dell'Emilia Romagna selezionata dall'ESA BIC Lazio, l'incubatore dell'Agenzia Spaziale Europea (ESA) ed è supportata da importanti istituzioni, acceleratori e grandi attori. Membro fondatore dal 2016 della rete Copernicus Academy, si impegna a diffondere i benefici dell'utilizzo dei dati di Osservazione della Terra per la qualità della vita dei cittadini e la competitività delle PMI.

Ha acquisito una solida esperienza nella promozione di Copernicus, il programma europeo di osservazione della Terra, nell'ecosistema Startup condividendo conoscenza e formazione. Ad oggi la società ha organizzato più di 40 eventi che hanno raggiunto più di 1000 persone che vanno dal pubblico generale, agli studenti, alle startup, ai funzionari pubblici, alle autorità locali, alle PMI.

---

## 3 Analisi dei problemi

### 3.1 Tecniche di computer vision

La computer vision è un ambito dell'intelligenza artificiale il cui scopo è quello di insegnare alle macchine non solo a vedere un' immagine, ma anche a riconoscere gli elementi che la compongono in modo da poter interpretare il suo contenuto come farebbe il cervello di un qualsiasi essere umano. Nonostante le attuali tecniche di deep learning rendano possibile questo compito, è comunque necessaria una grande quantità di immagini e di tempo per poter allenare una rete neurale a sufficienza in modo da riuscire a riconoscere correttamente degli oggetti in un' immagine non incontrata durante il processo di allenamento.

L'obiettivo è quindi quello di riconoscere e classificare alcuni specifici elementi presenti in un'immagine localizzandone anche la posizione esatta all'interno dell'immagine. Una volta trovata la sua locazione, l'oggetto viene messo in evidenza disegnando un rettangolo attorno ad esso, detto anche bounding box, in modo che lo racchiuda con la maggiore precisione possibile. La classificazione ha invece come scopo quello individuare la categoria di appartenenza di un oggetto e la probabilità che essa sia realmente quella corretta.

Per classificare un singolo elemento in un'immagine viene tipicamente utilizzata una Convolutional Neural Networks (CNN) allenata con grandi quantità di immagini che possono essere tranquillamente reperite in rete già raggruppate in datasets come ad esempio ImageNet o Coco.

La vera sfida salta fuori non appena ci troviamo a dover identificare e classificare nella stessa immagine diversi oggetti appartenenti a categorie diverse, di differenti dimensioni e posizioni e talvolta anche sovrapposti. Questa situazione è molto comune quando ci troviamo ad osservare qualsiasi foto rappresentante il mondo reale. Il risultato ottimale sarebbe quindi di avere una bounding box di dimensioni corrette per ogni oggetto identificato mostrando anche la categoria di appartenenza dell'elemento insieme alla sua probabilità.

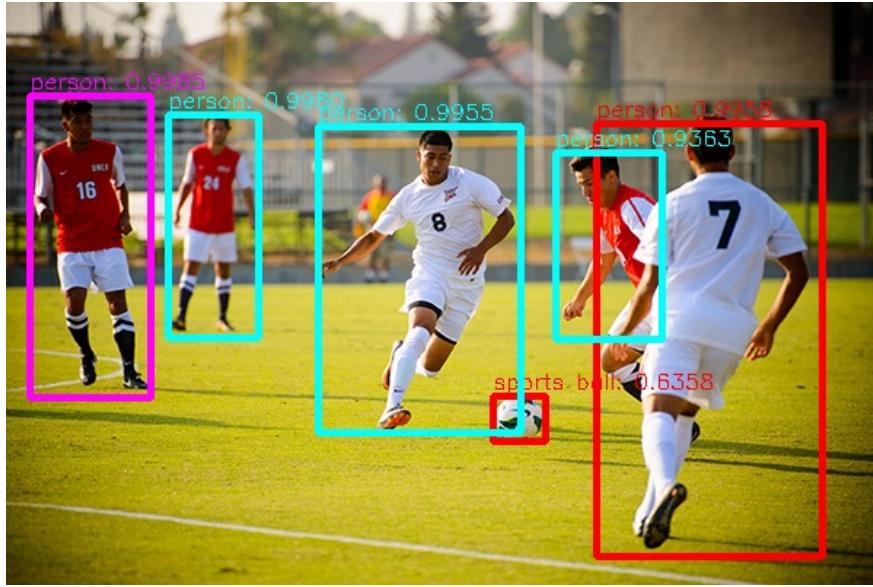


Figura 2: Esempio di un’immagine con box, categoria e probabilità per ogni elemento riconosciuto in essa

Per localizzare gli elementi, la tecnica più semplice consiste nel far scorrere una sliding window di varie dimensioni attraverso tutta l’area dell’immagine effettuando una classificazione per ogni locazione raggiunta. In caso di esito positivo la sliding window corrente corrisponderà al bounding box dell’oggetto classificato. Al termine del processo è possibile usare un algoritmo di non-max suppression per rimuovere eventuali sovrapposizioni. Lo svantaggio di questo metodo è che bisognerebbe effettuare una detection per ogni sliding window comportando quindi un elevato costo computazionale. In particolare, nel caso di immagini in alta definizione si potrebbe addirittura arrivare ad avere migliaia di sliding windows comportando allo stesso tempo migliaia di detections per una sola immagine. Oltre al costo computazionale, questo metodo ha lo svantaggio di non sfruttare appieno la potenza di una rete neurale.

Per effettuare una detection, il metodo più utilizzato allo stato dell’arte è quello di utilizzare una R-CNN (Regional Convolutional Neural Network) la quale tramite un algoritmo greedy chiamato Selective Search estrae dall’immagine delle regioni di interesse nelle quali è probabile che vi sia presente un oggetto. Queste regioni vengono poi date singolarmente in input ad una normale CNN la quale ha il compito di estrarne le caratteristiche principali per permettere ad una Support Vector Machine (SVM) presente nell’ultimo strato della CNN di rilevare la presenza di un oggetto ed eventualmente classificarlo.

Alternativamente, si possono prima utilizzare dei layers convoluzionali per estrarre una map-

---

pa delle caratteristiche più significative dall'immagine completa ed in seguito applicarci sopra un algoritmo di ricerca come Selective Search per trovare le regioni di interesse sulle quali effettuare la classificazione. Quest'ultimo metodo è conosciuto come Fast-CNN in quanto si ha un' efficienza maggiore rispetto a R-CNN per il fatto che l'operazione viene svolta solamente una volta per immagine invece che una volta per ogni regione di interesse.

Tuttavia, questi tipi di soluzioni presentano il difetto di richiedere molto tempo per eseguire l'operazione di ricerca delle regioni di interesse.

La versione più avanzata della Fast-CNN ed attualmente in uso nei sistemi di computer vision attuali è chiamata Faster R-CNN e risolve il bottleneck della sua antecedente sostituendo la Selective Research con una Region Proposal Network (RPN). Essa prende come input un'immagine di qualsiasi dimensione e restituisce come output un insieme di rettangoli associati ad una probabilità che essi contengano un oggetto o meno. Come per Fast-CNN, viene prima effettuata una convoluzione per costruire la mappa delle caratteristiche più significative dell'immagine, in seguito viene utilizzata una sliding window per scorrere la mappa delle caratteristiche e darle in input a due fully-connected layers, dei quali, uno serve per individuare le coordinate del box dell'oggetto<sup>1</sup> mentre l'altro serve per ritornare la probabilità che nel box vi sia effettivamente un oggetto<sup>2</sup>. Infine queste regioni vengono passate ad un'altra rete che avrà come al solito il compito di riconoscere e classificare l'oggetto.

### 3.2 Computer vision applicata su immagini ad alta risoluzione

Sebbene sul web sia abbastanza facile reperire grandi quantità di immagini con cui allenare i propri modelli, tuttavia la maggior parte di questi datasets contiene solamente immagini a bassa risoluzione ed è difficile trovare in rete grandi datasets di immagini o video in 4K. Inoltre, la maggior parte dei modelli sono stati progettati per lavorare su immagini a bassa risoluzione (tra i 200 e i 600 pixels) sia per il fatto che una bassa risoluzione è comunque sufficiente per riconoscere e classificare un elemento, sia perchè è più efficiente lavorare su immagini di bassa qualità che su immagini in con una risoluzione molto alta.

Lo svantaggio che questo comporta è che nelle immagini in bassa risoluzione si perdono molti dei dettagli che invece potrebbero essere catturati da un'immagine o da un video ad alta risoluzione. In aggiunta, i video in 4K o addirittura 8K sono al giorno d'oggi sempre più diffusi perciò anche gli attuali modelli dovranno prima o poi adattarsi per trattare efficacemente immagini di tali risoluzioni. Nella figura sottostante si può vedere un esempio di un frame in alta risoluzione nel quale, diminuendone le dimensioni e perdendo quindi qualità, non sarebbe stato possibile riconoscere alcune delle persone individuate nel frame.

---

<sup>1</sup>Box-regression layer

<sup>2</sup>Box-classification layer



Figura 3: Esempio di object detection in un frame di un video in 4K

### 3.3 Frammentazione dell'immagine

Un'immagine in 4K ha una risoluzione di 3840 x 2160 pixels per un totale di 8294400 di pixels. Una rete neurale in grado di ricevere un input così corposo dovrebbe allenare un numero molto elevato di parametri risultando così in un processo molto lungo e dispendioso tanto che molti dei modelli attuali effettuano un ridimensionamento dell'immagine per adattarla al meglio al loro input. L'idea per aggirare il problema è quella di frammentare l'immagine in diverse sotto-immagini di dimensioni minori e quindi gestibili efficacemente da una singola rete neurale. Questa particolare strategia è chiamata frammentazione dell'immagine e risulta essere molto utile in quanto permette di analizzare un'immagine ad alta risoluzione scomponendola in frammenti di dimensioni minori anziché gestirla nella sua interezza.

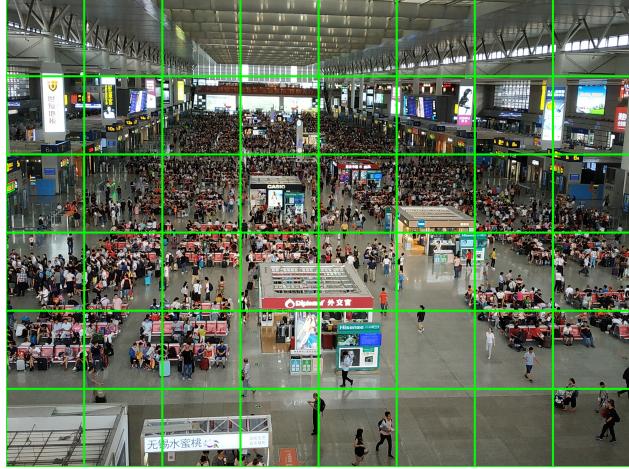


Figura 4: Esempio di un’immagine in alta risoluzione suddivisa in regioni senza sovrapposizioni

Tuttavia questo procedimento non è esente da difficoltà. Nel caso in cui un oggetto dovesse trovarsi su più regioni diverse, esso potrebbe venire identificato più volte o addirittura essere riconosciuto ogni volta come se fosse un oggetto appartenente ad una categoria diversa.

Un primo approccio per risolvere questo problema è quello di porre maggiore attenzione alle labels degli oggetti posizionati in prossimità dei confini delle regioni in quanto con molta probabilità è possibile che l’oggetto continui invece nella regione adiacente piuttosto che essere interamente contenuto nella regione esaminata. Se è quindi presente una label anche in una regione adiacente si passa allora alla verifica che le due labels possano effettivamente appartenere allo stesso elemento. Per fare ciò bisogna assicurarsi che le due labels siano compatibili sia in termini di categoria che di posizione entro una certa soglia ed in caso affermativo fonderle in una sola label contenente l’oggetto intero.

Un’altra soluzione esaminata è quella di suddividere l’immagine intera in regioni con sovrapposizione il cui scopo è quello di generare intersezioni tra labels in prossimità dei confini. In questo modo un elemento contenuto all’interno di un’area di sovrapposizione tra più regioni genererebbe due o più labels quasi completamente sovrapposte. Il problema può essere facilmente gestito con un algoritmo di Non-Maximum Suppression, il quale, per ogni insieme di labels parzialmente sovrapposte e con stessa categoria, tende ad eliminare le labels con probabilità minore tenendo valida solo quella con probabilità massima. Tuttavia il caso più insidioso ed allo stesso tempo più frequente avviene quando due o più labels non solo hanno un’intersezione dentro un’area di sovrapposizione ma la loro box si estende anche al di fuori

---

di esse. E' proprio per far fronte a questo problema che è stato ideato ed implementato un algoritmo il quale verrà descritto nella *sezione 4.1*.

### 3.4 Tecniche di object tracking

Il tracciamento di oggetti in un video è uno di quei problemi presenti nell'ambito dell'informatica che non sono ancora stati risolti ottenendo risultati soddisfacenti. Il tracciamento consiste non solo nel localizzare e tracciare degli oggetti di interesse attraverso una sequenza di frames ma anche nel riconoscere che l'oggetto tracciato è sempre lo stesso.

Il problema non è per niente banale se pensiamo che in un video uno stesso oggetto può spostarsi, nascondersi dietro qualche altro elemento, deformarsi, cambiare le sue dimensioni, la sua illuminazione, la sua velocità ed il tipo di background. Nel caso ancora peggiore un oggetto tracciato potrebbe addirittura scomparire in un frame per poi ripresentarsi solamente dopo che sono trascorsi un numero casuale di frames.

Un algoritmo di tracking ottimale dovrebbe essere in grado di far fronte a tutti questi problemi riconoscendo quindi l'oggetto da esso tracciato tramite per esempio l'assegnazione di un ID univoco.

allo stato dell'arte, questo problema viene affrontando eseguendo inizialmente una normale object detection sul primo frame del video in modo tale da individuarne tutti gli elementi presenti e le rispettive labels. In seguito, ognuno di questi elementi viene assegnato ad un tracker che avrà il compito di tracciare l'elemento nei frames successivi.

Tracciare un elemento è un' operazione meno onerosa rispetto alla sua individuazione in quanto il tracker conosce già alcune informazioni relative all'oggetto tracciato acquisite nei frames precedenti potendo così tenere in memoria uno storico del suo stato, come per esempio, le sue ultime locazioni. Per aumentare la sua precisione, un tracker non tiene conto solamente della locazione dell'elemento osservato ma può anche conservare altre utili informazioni aggiuntive come la sua direzione, una previsione delle locazioni future analizzandone la sua traiettoria oppure può memorizzare un hash<sup>3</sup> dei pixels presenti all'interno del bounding box dell'oggetto per poi confrontarlo con l'hash dello stesso oggetto calcolato nel frame successivo. I trackers più performanti come CSK, MOSSE e GOTURN utlizzano alcune delle informazioni sopra descritte per costruire un filtro di correlazione in modo da localizzare l'oggetto nel frame successivo e migliorare la precisione del filtro con le successive individuazioni, lo scopo del filtro è quello di minimizzare la differenza tra l'output ricostruito e quello originale.

Nonostante tutti questi accorgimenti è però inevitabile che col trascorrere dei frames i trac-

---

<sup>3</sup>Al contrario degli hash usati in crittografia, un hash applicato alle immagini è progettato in modo tale che piccole variazioni dei pixels dell'immagine non risultino in un hash molto diverso da quello originale

---

kers cominceranno ad essere sempre più imprecisi nell'individuare il loro oggetto tracciato. In particolare, questa situazione può accadere molto velocemente in quei video dove avvengono molti spostamenti e sovrapposizioni tra elementi. E' quindi buona norma aggiornare i trackers con le locazioni corrette effettuando una nuova detection ogni fissato numero di frames.

E' qui che si presenta il problema di maggiore rilevanza: una nuova detection effettuata su un nuovo frame non tiene conto delle informazioni acquisite in precedenza in quanto queste con molta probabilità sarebbero errate o imprecise. Il problema è quindi quello di riassegnare a ciascun oggetto individuato lo stesso ID che possedeva in precedenza ed assegnare un nuovo ID ad ogni oggetto comparso nel video per la prima volta.



Figura 5: Frames di un video nei quali viene tracciata un'auto (ordinati da sinistra a destra e dall'alto al basso)

### 3.5 Object tracking con continue object detections

Considerata la scarsa affidabilità degli attuali algoritmi di tracciamento, nella soluzione individuata gli oggetti vengono identificati effettuando una nuova detection per ogni frame del video in modo da assicurarsi di avere sempre una buona accuratezza ed allo stesso tempo migliorare l'efficacia dei trackers migliorando gradualmente la precisione dei loro filtri di predizione. Questo metodo sacrifica l'efficienza del processo per migliorarne l'efficacia. A meno che non venga utilizzato un algoritmo di detection molto veloce come SSD (Single Shot Detector) non è possibile applicare il tracking sui video in tempo reale in quanto il frame rate che ne risulterebbe sarebbe troppo basso. A seguito di una nuova detection, per effettuare una corretta riassegnazione degli ID viene fatto un confronto tra le labels presenti nel frame precedente con quello corrente con lo scopo di trovare una corrispondenza biunivoca tra due

---

labels e capire quando entrambe si riferiscono allo stesso elemento in modo da garantire un corretto trasferimento dell'ID. Per rendere tutto ciò possibile viene utilizzato un filtro in grado di predire le locazioni future di un oggetto tenendo traccia dei suoi bounding boxes e poi correggersi tramite misurazioni successive. Anche per realizzare questa soluzione è stato ideato ed implementato un algoritmo descritto nella *sezione 4.2*.

---

## 4 Progettazione

### 4.1 Progettazione algoritmo per riconoscimento di elementi in un'immagine frammentata

Per quanto riguarda il problema della frammentazione dei frames in 4K è stato individuato un apposito algoritmo in grado di effettuare il riconoscimento degli oggetti nei frames presenti in un video senza doverli ridimensionare ma utilizzando la tecnica della frammentazione dell'immagine.

#### 4.1.1 Scomposizione del frame originale in regioni

Un frame in 4K viene quindi scomposto in una matrice di  $R \times C$  sotto-immagini chiamate  $\text{regione}_G$  in modo tale che ogni regione sia efficacemente analizzabile da un modello come Faster R-CNN. Per facilitare l'operazione di riconoscimento degli elementi da parte della rete, ogni regione si sovrappone leggermente con le sue regioni adiacenti. Per definire la quantità di pixels da coinvolgere nella sovrapposizione viene definito uno  $\text{stride}_G$  che indica quanti pixels della regione tralasciare, sia in verticale che in orizzontale, prima che cominci quella successiva, ovviamente lo stride deve essere minore della larghezza di una regione. Una Faster R-CNN dopo aver elaborato singolarmente ogni regione come se fosse una singola immagine darà in output una lista di  $\text{label}_G$  con le seguenti caratteristiche:

- (**max-x, max-y**): coordinate del vertice in alto a sinistra del rettangolo rappresentante il bounding box dell'oggetto riconosciuto;
- (**min-x, min-y**): coordinate del vertice in basso a destra del rettangolo rappresentante il bounding box dell'oggetto riconosciuto;
- **Categoria<sub>G</sub>** : è un numero naturale che indica la categoria di appartenenza dell'elemento individuato;
- **Score<sub>G</sub>** : rappresenta la misura di probabilità che la classificazione ottenuta sia effettivamente quella corretta.

Successivamente viene aggiustata la posizione delle labels individuate in modo da trasstrarle nella loro posizione corretta all'interno dell'immagine originale non frammentata. Questo viene fatto aggiungendo un adeguato offset alle coordinate dei vertici dei bounding boxes delle labels sulla base della loro regione di appartenenza.

---

#### 4.1.2 Rimozione degli elementi individuati più volte all'interno delle aree di sovrapposizione

A causa della presenza delle aree di sovrapposizione dovute alla struttura delle regioni, gli elementi giacenti in queste particolari zone del frame verranno individuati tante volte quante sono le regioni che si sovrappongono in quella determinata area. Per eliminare le copie duplicate e tenerne solo una viene utilizzato un algoritmo chiamato Average Non-Max Suppression (ANMS) che è una variante del Non-Max Suppression tipicamente utilizzato dai modelli di visione artificiale. Per ogni gruppo di labels sovrapposte, invece che tenere la label con lo score maggiore ed eliminare tutte le altre, il nuovo bounding box viene calcolato come la media dei bouding boxes box di tutte labels e lo score viene calcolato come la media tra tutti gli scores. Questo metodo è fondato sul ragionamento che non bisognerebbe buttare via delle informazioni già possedute ma piuttosto riutilizzarle per scoprire qualcosa di nuovo. Per esempio ad uno stesso elemento visualizzato dentro due sezioni differenti di un' immagine potrebbero venirgli assegnati due score diversi. Mentre NMS conserverebbe solo il valore più alto tra i due, ANMS li utilizzerebbe entrambi per ottenere un valore ancora più affidabile aumentando quindi la veridicità della classificazione.

#### 4.1.3 Creazione di raggruppamenti di labels correlate

A questo punto tutti gli elementi sono stati individuati e classificati ma rimane comunque il problema che, a causa della precedente scomposizione, gli oggetti situati all'interno delle aree di sovrapposizione risulterebbero individuati due o più volte. Come si può vedere in figura 6, questo numero varia in base al numero di regioni sulle quali giace l'oggetto. Il secondo problema è che due elementi *vicini*<sup>4</sup>, anche se classificati nella stessa categoria, non è detto che necessariamente debbano rappresentare lo stesso elemento. Un esempio di questo caso lo si può sempre notare in figura 6. Un caso ancora peggiore lo si ha quando non solo l'elemento è situato su più regioni differenti ma sussiste anche il problema che ogni parte dell'elemento verrebbe classificata in modo diverso a causa della loro ambiguità. Infine, è anche possibile che un oggetto si distribuisca su più regioni adiacenti ed ogni sua label presenti dimensioni diverse per ogni regione. La soluzione individuata consiste nel raggruppare labels correlate tra loro in insiemi di labels dette raggruppamenti<sub>G</sub> per poi racchiuderli in una label che identifica l'elemento rappresentato dal raggruppamento. Due labels sono in correlazione tra di loro se soddisfano una condizione di correlazione sotto riportata.

**Condizione di correlazione:** Per effettuare un corretto raggruppamento delle labels viene anche tenuta in considerazione la categoria a loro associata tramite la classificazione insieme

---

<sup>4</sup>Due labels sono considerate vicine se sono intersecate tra di loro ed intersecano lo stesso confine di regione



Figura 6: Esempi di labels erroneamente individuate a causa della frammentazione e relativa corretta ricostruzione

allo score assegnato. Per definire il risultato della condizione è inoltre necessario stabilire una **soglia di affidabilità** che indica lo score minimo che una label deve possedere per poter considerare la sua classificazione come affidabile o meno. Di seguito vengono riportati i vari casi per decidere se la condizione è vera o falsa.

- **True:** Le due labels hanno la stessa categoria ed entrambe con score uguale o maggiore della soglia;
- **True:** Le due labels hanno la stessa categoria ma almeno una delle due ha score minore della soglia;
- **True:** Le due labels hanno categoria diversa ma almeno una delle due ha score minore della soglia;
- **False:** Le due labels hanno categoria diversa ed entrambe con score uguale o maggiore della soglia;

Prima di cominciare con il raggruppamento, vengono inizialmente individuate tutte le labels che intersecano i confini di regioni causati dalle aree di sovrapposizione o che non distingono più di un fissato numero di pixels, detto  $\text{overlap}_G$ , da esse. L'overlap viene definito in quanto anche se nell'immagine reale un oggetto interseca un confine di regione è possibile che a causa di errori di imprecisione del modello, il box della label risulti leggermente distaccato dalla linea che rappresenta il confine. E' quindi possibile ovviare a questo problema aumentando lo spessore del confine di tanti pixels quanti indicati dall'overlap. Per lo stesso motivo precedente, ai fini di controllare se una label interseca un'altra entità o meno viene anche tenuta

---

in considerazione una tolleranza  $G$  che indica di quanto il bounding box di una label può essere distante da un'entità affinché questa venga comunque considerata come intersecata. Per creare i raggruppamenti di labels è stato ideato il seguente algoritmo:

1. Vengono tenute solo le labels che intersecano almeno un confine di regione e vengono inizializzate come *non controllate* e *non raggruppate*;
2. Viene selezionata una label qualsiasi *non controllata* e la si imposta come *controllata*;
3. Per ogni label *controllata* ma non ancora *raggruppata* controlla se ci sono altre labels *non controllate* che rispettino ognuna delle seguenti condizioni:
  - Devono essere *vicine* o distanti entro la tolleranza fissata;
  - Devono rispettare la *condizione di correlazione*;
  - La loro box non deve intersecare una regione al di fuori delle aree di sovrapposizione che sia già intersecata da una qualsiasi altra label *controllata*<sup>5</sup>;
  - Deve essere rispettata una **soglia di matching**: La posizione delle loro aree deve essere compatibile entro una certa soglia ovvero che, i lati lungo il quale le due labels vengono unite siano tali che la differenza tra quello maggiore e quello minore sia inferiore ad una certa soglia che può essere sia definita come una proporzione rispetto ad uno dei due lati oppure un valore in pixels.
4. Le labels così trovate diventano a loro volta *controllate*;
5. Si ripetono i punti 3 e 4 fino a che non sia più possibile trovare ulteriori labels;
6. Tutte le label *controllate* vengono ora classificate come *raggruppate* e viene assegnato un numero progressivo ad ogni label *raggruppata* in modo da identificarne il gruppo di appartenenza;
7. Si ripetono i punti da 2 a 6 fino a che tutte le labels non vengano raggruppate. L'algoritmo in questo modo termina sempre ed è possibile che un raggruppamento comprenda una sola label.

E' da notare che labels intersecanti ma interamente comprese in una sola regione non sono motivo di interesse in quanto l'algoritmo di Non-Maximum Suppression utilizzato dalla rete

---

<sup>5</sup>Questo perché se due labels sono state individuate come elementi distinti all'interno di una regione allora è probabile che lo siano anche nell'immagine intera in quanto viene supposto che un modello non commetta errori di riconoscimento

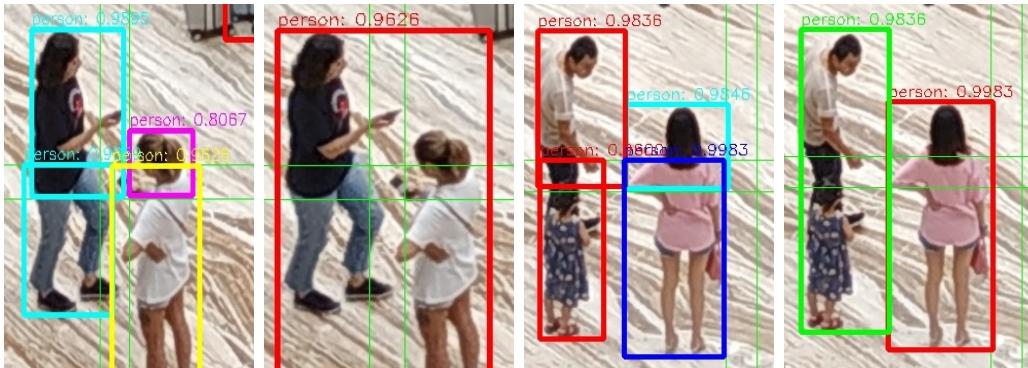


Figura 7: Esempi di labels erroneamente individuate a causa della frammentazione e relativa errata ricostruzione

in fase di post-processing ci assicura che labels intersecanti nella stessa regione individuino elementi diversi. In seguito bisogna trasformare ogni raggruppamento in una nuova label che racchiuda tutte le labels che lo compongono. Per fare questo vengono esaminate le coordinate di ogni vertice di tutte le bounding boxes di un raggruppamento in modo tale da trovare quattro nuovi vertici di un rettangolo che soddisfi i requisiti sopra discussi. La nuova label così creata andrà a sostituire le labels del rispettivo raggruppamento e per deciderne la categoria e lo score viene applicata una **regola di classificazione**: categoria e score assegnati saranno pari alla categoria e allo score posseduti dalla label appartenente al raggruppamento con score maggiore.

A questo punto l'algoritmo può dirsi concluso ed è in grado di riconoscere gli elementi in un'immagine in 4K con un'accuratezza accettabile e buona velocità. Tuttavia in casi particolari come quello mostrato in figura 7 l'algoritmo commetterebbe un errore di classificazione in quanto individuerebbe due elementi distinti con una sola label comune. In questi casi non c'è modo di sapere se le due labels effettivamente appartengono a due oggetti distinti o se rappresentano lo stesso oggetto.

#### 4.1.4 Miglioramento: raggruppamenti di labels utilizzati come region proposal

Un ulteriore miglioramento dell'algoritmo potrebbe essere ottenuto utilizzando le labels ottenute dal procedimento descritto in precedenza come nuove regioni sulle quali applicare nuovamente una detection per identificare gli elementi contenuti nella regione ma con maggiore precisione in quanto questa volta l'area non verrà affetta da problemi di frammentazione dando quindi la possibilità alla rete di esaminare l'area per intero. La regione viene prima

---

inizializzata rimuovendo la sua label e poi ripopolata con le nuove labels identificate dalla rete. Il primo problema che salta fuori è che durante questo procedimento la rete identificherà nuovamente anche quegli elementi che casualmente si trovavano dentro la regione coinvolta ma che erano già stati trovati anche in precedenza. Tuttavia questo problema viene tranquillamente risolto applicando un algoritmo di Average Non-Max Suppression, utilizzato già in precedenza, per eliminare oggetti quasi completamente sovrapposti. Il secondo problema riguarda ancora gli oggetti che stanno a cavallo tra la regione interessata e l'immagine originale, questa volta però, avendoli già individuati nella loro interezza durante la prima fase è quindi solamente necessario unire la nuova label con quella già trovata in precedenza in modo da ottenerne una nuova con precisione maggiore. Un caso particolare lo si ha quando la regione in esame risulti essere così estesa da vanificare i vantaggi ottenuti dalla frammentazione. Per far fronte a questo problema basta ridimensionare l'area coinvolta fino a portarla ad avere dimensioni gestibili da una rete. In questo caso la perdita di risoluzione e quindi di dettagli non comporterebbe un grave problema in quanto gli elementi visibili solo grazie all'alta definizione sarebbero già stati individuati nella fase precedente. Nel caso in cui questi oggetti dovessero venire nuovamente identificati verrebbero gestiti dall'ANMS per ottenerne una migliore approssimazione. Questa funzionalità permette di migliorare l'accuratezza quando si vogliono identificare oggetti che si estendono su due o più regioni o per migliorare il riconoscimento di gruppi di elementi sovrapposti e molto vicini tra loro in prossimità di un confine.

## 4.2 Progettazione algoritmo per tracciamento di elementi

Per affrontare il problema del tracciamento degli elementi, viene utilizzato un algoritmo di tracking supportato da una detection applicata ad ogni frame del video al fine di garantirne una migliore accuratezza. Essendo i video in qualità 4K, per effettuare la detection viene sempre utilizzato l'algoritmo descritto in sezione 4.1 in modo da non ridurre la qualità dei frames.

### 4.2.1 Filtro di Kalman

Lo scopo di un tracker è quello di predire la posizione di un oggetto in un frame a partire dallo storico delle sue locazioni passate per mezzo di un filtro. I trackers implementati utilizzano un filtro di Kalman in quanto essi si rivelano molto efficaci nell'effettuare predizioni anche in sistemi soggetti a continui cambiamenti come lo è per esempio un video. Il secondo vantaggio è quello di garantire una buona resistenza contro i rumori causati da detections imprecise, le quali possono per esempio avere luogo in presenza di oggetti parzialmente occultati o deformati a causa di qualche spostamento. Infine, questa tipologia di filtri sono

---

anche computazionalmente veloci in quanto, una volta implementati, la loro esecuzione si traduce in semplici moltiplicazioni tra matrici. L'applicazione del filtro di Kalman consiste in due fasi distinte: predizione ed aggiornamento. La prima fase ha come scopo quello di usare la locazione precedente per predire quella attuale effettuando anche una piccola correzione per far fronte alle variazioni introdotte da possibili fonti esterne (rumore). In seguito sono riportate le formule relative alla fase di predizione:

$$x_k = F_k x_{k-1} + B_k u_k$$

$$P_k = F_k P_{k-1} F_k^T + Q_k$$

Dove  $x_k$  è la predizione della posizione dell'oggetto  $x$  nel frame  $k$ ,  $F_k$  è il modello di transizione di stato applicato alla posizione  $x_{k-1}$ ,  $B_k$  è una matrice di controllo alla quale viene applicato il vettore di controllo  $u_k$  e rappresentano le variazioni subite da  $x_k$  causate da una fonte esterna, in questo rappresentano movimenti irregolari dell'oggetto rispetto alla sua traiettoria.  $P_k$  è la predizione della covarianza di  $x_k$  mentre  $Q_k$  è la covarianza del processo che genera rumore. Nella seconda fase, quella di aggiornamento, viene usata la misurazione corrente, che in questo caso sarà il bounding box dell'oggetto tracciato individuato dalla nuova detection, per rifinire ulteriormente la sua locazione esatta. In seguito sono riportate le formule relative alla fase di aggiornamento:

$$\begin{aligned} x'_k &= x_k + K(z_k - H_k x_k) \\ P'_k &= P_k - K H_k P_k \\ K &= P_k H_k^T (H_k P_k H_k^T + R_k)^{-1} \end{aligned}$$

Dove  $x'_k$  è la nuova stima della posizione ottenuta dopo aver effettuato l'aggiornamento,  $K$  è una matrice detta anche guadagno di Kalman,  $z_k$  è il valore misurato, in questo caso corrisponderà alla label individuata con la detection effettuata sul frame  $k$ .  $H_k$  è una matrice che serve per scalare  $z_k$  in modo tale da renderlo compatibile con lo stato dell'oggetto tracciato ed infine  $R_k$  è la covarianza di  $z_k$ . Riassumendo, lo scopo del filtro di Kalman è quello di individuare la posizione reale di un oggetto a partire da una sua predizione rispetto alla sua posizione precedente e da una misurazione reale che però può essere soggetta ad imprecisioni.

#### 4.2.2 Assegnazione detection-tracker

Il passo successivo è quello di abbinare ognuno dei bounding box stimati dai filtri dei trackers con le bounding boxes individuate da una detection. Questo risultato viene ottenuto tramite un algoritmo di assegnazione conosciuto anche come algoritmo di Kuhn-Munkres. La metrica utilizzata dall'algoritmo è l'IoU tra i bounding boxes stimati dai trackers e quelli individuati dalla detection, tale metrica viene spiegata in dettaglio nella sezione 7.1.3. L'obiettivo

---

dell'algoritmo è quello di assegnare le detections ai trackers massimizzando la somma dell'IoU delle bounding boxes associate. Questa soluzione si basa sul ragionamento che più due bounding boxes sono sovrapposte, più è probabile che esse rappresentino lo stesso oggetto. Inoltre, perché due bounding boxes possano essere associate, il loro IoU deve anche essere maggiore o uguale di una certa soglia, detta **soglia di assegnazione**, in quanto tra due boxes poco sovrapposte è probabile che non vi sia alcuna correlazione. Il primo passo dell'algoritmo consiste nel creare una matrice di dimensioni  $n \times m$  dove  $n$  è il numero di oggetti individuati dalla detection e  $m$  è il numero di trackers. In ogni cella della matrice viene calcolato il valore dell'IoU tra la box predetta dal tracker  $i$  ed il box  $j$  individuato dalla detection, con  $1 \leq i \leq m$  e  $1 \leq j \leq n$ . Se le colonne sono minori delle righe allora bisogna ruotare la matrice in modo tale che le colonne siano tante almeno quante sono le righe. In seguito vengono ripetuti i seguenti passi:

1. Per ogni riga, sottrarre il valore minimo della riga a tutti gli elementi della stessa riga. In questo modo, in ogni riga sarà presente almeno una cella con valore pari a 0.
2. Per ogni colonna, sottrarre il valore minimo della colonna a tutti gli elementi della stessa colonna. In questo modo, in ogni colonna sarà presente almeno una cella con valore pari a 0.
3. Tracciare delle linee attraverso tutte le righe e le colonne che contengano almeno un elemento pari a 0 in modo tale da tracciare il minor numero di linee possibile.
4. Se sono state tracciate esattamente  $k = \min(n, m)$  allora l'algoritmo termina qui, altrimenti si procederà con il passo 5.
5. Trova la cella minore che non sia tracciata da alcuna linea e poi sottrarre quel valore a tutte le righe non tracciate, sommare poi quel valore ad ogni colonna tracciata. Infine tornare al passo 3.

Al termine dell'algoritmo si selezionano  $k = \min(n, m)$  zeri dalle celle della matrice tali che ogni zero appartenga ad una sola riga e ad una sola colonna. Le coordinate di queste celle corrisponderanno agli assegnamenti detection-tracker ottimali da attuare nella matrice originale.

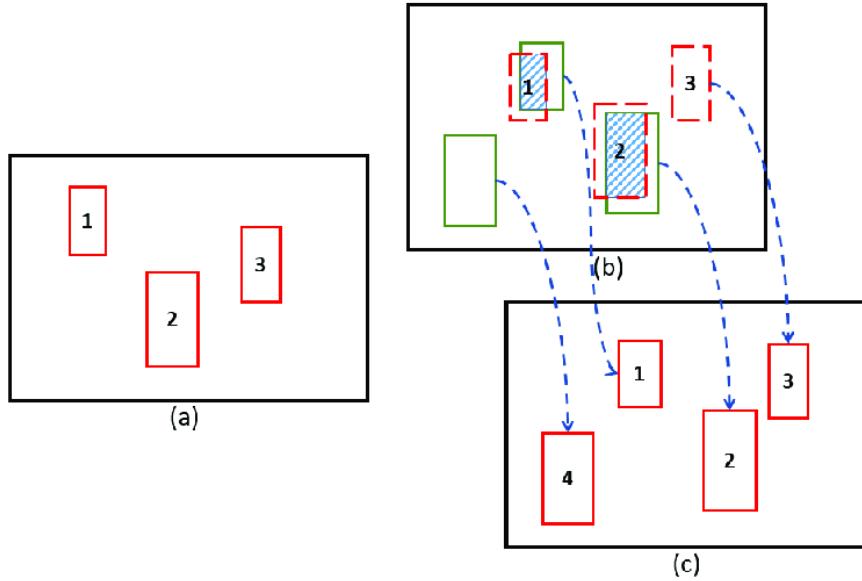


Figura 8: Esempio di assegnazione detection-tracker

Come si può vedere in figura 7 i rettangoli rossi raffigurano le bounding boxes dei trackers mentre quelle verdi rappresentano le bounding boxes individuate dalla detection. I trackers 1 e 2 vengono assegnati regolarmente, l'elemento tracciato dal tracker 3 è sparito dal frame ed è comparso un nuovo elemento che verrà tracciato da un nuovo tracker.

#### 4.2.3 Gestione delle detections e dei trackers non assegnati

Nel caso in cui nella matrice precedente si abbia avuto che  $n$  è diverso da  $m$ , significa che sono presenti dei trackers o delle detections che non sono state assegnate. Inoltre, le coppie tracker-detection associate con successo dall'algoritmo il quale IoU sia però inferiore alla soglia stabilita verranno scartate ed aggiunte nelle rispettive liste di trackers e detections non assegnate. La mancata assegnazione di un tracker al suo elemento potrebbe avere due cause scatenanti: la prima è che l'oggetto tracciato sia momentaneamente sparito dal video a causa di una detection non andata a buon fine. Questo evento può essere per esempio dovuto ad una momentanea perdita di qualità di un frame, da un occultamento o da una sfocatura causata da un movimento. La seconda causa è che l'oggetto sia effettivamente assente in un frame del video a causa di un suo movimento verso l'esterno del frame oppure a causa di un cambio di inquadratura. In questo caso è poco probabile che l'oggetto scomparso ritorni a presentarsi nel frame successivo ma è più ragionevole pensare che esso non si ripresenterà più, almeno nel breve periodo. In quanto un tracker non ha modo di sapere quale delle

---

due cause sia quella che abbia realmente portato alla sua mancata assegnazione, allora in presenza di questo evento viene solamente incrementato un contatore interno al tracker che tiene conto dei frames consecutivi trascorsi senza che il tracker venga associato al suo oggetto tracciato. Non appena questo contatore supera una certa soglia, detta **soglia di cancellazione**, il relativo tracker viene cancellato e l'elemento da esso tracciato viene considerato come perduto. Se un elemento perduto dovesse successivamente ricomparire in un frame esso verrà considerato come un nuovo oggetto venendo quindi tracciato da un nuovo tracker. Un discorso analogo vale anche per quelle labels che sono individuate da una detection ma che non vengono assegnate a nessuno dei trackers già esistenti. Con molta probabilità si tratta di una nuovo oggetto ed è quindi opportuno creare un nuovo tracker per iniziare a tracciarlo. Tuttavia, prima che il legame tracker-detection diventi effettivo non basta che l'assegnazione avvenga una sola volta. Potrebbe infatti accadere che a causa di una detection sbagliata, un oggetto di una determinata categoria compaia erroneamente in un solo frame per poi non ripresentarsi più. In questo caso sfortunato si verrebbe a creare un tracker la cui utilità sarebbe poco significativa consumando un Id per essere associato ad un oggetto individuato per errore. Per risolvere questo problema ogni tracker implementa un altro contatore per tenere traccia dei frames consecutivi nei quali al tracker stesso viene assegnato un oggetto, questo contatore rappresenta quindi la durata in frames del tracciamento. Solamente dopo che questo contatore abbia superato una certa soglia, detta **soglia di validazione**, verrà assegnato un ID al tracker e la sua corrispondente bounding box verrà mostrata nel video. Una volta terminata questa fase, vengono utilizzate le nuove detections per aggiornare i filtri dei trackers e rifinire la stima della locazione degli oggetti tracciati (fase di aggiornamento).

#### 4.2.4 Utilizzo di metriche di supporto per l'assegnazione detection-tracker

Durante il processo di assegnazione detection-tracker viene utilizzata l'IoU come metrica per decidere come abbinare le varie bounding boxes presenti in un frame. Tuttavia in presenza di sovrapposizioni questa metrica potrebbe non risultare abbastanza robusta. Pensiamo per esempio al caso in cui due persone A e B che, camminando in direzioni opposte, incrociano brevemente la loro traiettoria. In un determinato frame è possibile che, essendo le bounding boxes delle due persone molto sovrapposte sovrapposte, venga commesso un errore nell'assegnazione detection-tracker causando quindi uno scambio di ID. Da quel frame in poi la persona A verrà quindi tracciata con l'ID B e viceversa. Per ridurre la probabilità che questo sfortunato evento accada, la metrica utilizzata per determinare il valore di matching tra le bounding boxes di una detection e un tracker non sarà soltanto l'IoU tra le loro aree ma possono venire aggiunte anche altre metriche di supporto come il rapporto tra la dimensione delle loro aree ed il rapporto tra le dimensioni dei loro lati. Questo ragionamento si basa sul fatto che tra due frames consecutivi, queste metriche normalmente non possono subire

---

una grande variazione rispetto allo stesso oggetto e quindi possono essere utili nell'effettuare un'eventuale associazione qualora affidarsi all sola IoU non risulterebbe molto efficace. Ovviamente, possono anche essere definiti dei pesi per rendere ciascuna metrica più importante o meno.

---

## 5 Tecnologie

### 5.1 Python

Il linguaggio di programmazione utilizzato per perseguire gli obiettivi del progetto è stato Python v3.7, si tratta di un linguaggio di programmazione di alto livello il cui obiettivo è quello di facilitare la leggibilità del codice ed adattarsi a diversi paradigmi di programmazione come quello procedurale, ad oggetti e funzionale. La motivazione per la scelta dell'utilizzo di questo linguaggio, oltre alla sua semplicità, è per il suo supporto di numeri frameworks e moduli relativi al deep learning e alla computer vision tra i quali Tensorflow e OpenCV. Qualsiasi modulo aggiuntiva può essere semplicemente installata eseguendo il comando:

```
pip install nome_modulo
```

Per lanciare un programma viene utilizzato il comando:

```
python nome_file.py
```

Altri moduli che sono stati utilizzati comprendono Numpy, Matplotlib e Pytest.



Figura 9: Logo di Python

### 5.2 Pycharm

Pycharm è un IDE per programmare in Python sviluppato da JetBrains. Le sue caratteristiche più importanti includono:

- Un sistema intelligente di completamento automatico del codice;
- Analisi statica del codice eseguita a tempo di esecuzione;
- Individuazione e risoluzione veloce degli errori tramite proposte di correzione;
- Possibilità di lavorare in un ambiente di sviluppo virtuale dove per ogni progetto vengono installate solamente le proprie dipendenze e i propri moduli.

---

### 5.3 Tensorflow

Tensorflow è un framework gratuito e open-source per lo sviluppo e l'allenamento di modelli di machine learning come le reti neurali. Ha la particolarità che i dati vengono gestiti attraverso dei grafi computazionali dove i nodi rappresentano delle operazioni matematiche da eseguire e gli archi rappresentano degli array multidimensionali contenenti i dati sui quali svolgere le operazioni (tensori). La sua architettura permette di svolgere le operazioni sia usando la CPUs che le GPUs in modo da eseguire operazioni con alto livello di parallelismo. Il modello di rete neurale utilizzato per effettuare la detection sulle immagini è stato implementato con Tensorflow.



Figura 10: Logo di Tensorflow

### 5.4 OpenCV

OpenCV è una libreria open-source orientata allo sviluppo di applicazioni di computer vision in tempo reale. E' stata scritta originariamente in C++ ma offre anche il supporto ad altri linguaggi di programmazione come Python. OpenCV è un' ottima libreria quando si ha bisogno di manipolare immagini e video permettendo di compiere operazioni sia di alto livello che di basso livello operando sui singoli pixels. Sono inoltre presenti diversi algoritmi di object detection ed object tracking già implementati permettendo quindi di sperimentarli tutti senza apportare troppe modifiche al codice esistente. La libreria è stata anche utilizzata per scomporre un video nei suoi singoli frames oltre che per disegnare su di essi.

### 5.5 Numpy

Numpy è un modulo di Python che fornisce il supporto per la gestione di matrici e array multidimensionali di grandi dimensioni. Dispone anche di una vasta collezione funzioni

---

matematiche per lavorare ad alto livello su di essi. Viene spesso utilizzato per operare su grandi quantità di dati in modo rapido ed efficiente.

## 5.6 Matplotlib

Matplotlib è una libreria grafica sviluppata per Python impiegata principalmente per disegnare grafici o altre figure con lo scopo di rappresentare dei dati utilizzando il minor numero di righe di codice possibile. In fase di allenamento di una rete neurale viene spesso usato per mostrare l'andamento della variazione dell'errore e dell'accuratezza.

## 5.7 Pytest

Pytest è un framework per Python che permette di scrivere dei test per testare il codice permettendone la loro esecuzione in modo automatico. Per installarlo viene utilizzato il comando:

```
pip install pytest
```

mentre per lanciarne l'esecuzione viene usato il comando:

```
pytest nome_file
```

Per convenzione il file contenente i test relativi ad un solo modulo è stato chiamato "nome\_modulo\_test.py" dove "nome\_modulo.py" è il modulo ad esso associato. I metodi che eseguono un test devono iniziare con "test\_," , in caso contrario non verranno riconosciuti come tali e la loro esecuzione non verrà lanciata. Sono stati implementati dei test di unità per tutte le funzioni più complesse o critiche in modo da verificarne il corretto funzionamento. Sono anche stati sviluppati dei test di integrazione per ogni algoritmo implementato in modo da assicurarsi del loro corretto funzionamento. Per valutare l'esecuzione dell'intero sistema sono invece state utilizzate delle metriche per misurare la bontà del risultato in quanto i test di sistema non sarebbero stati adatti per misurare un risultato non deterministico come il riconoscimento di oggetti.

## 5.8 Excel

Il foglio elettronico Excel si è rivelato molto utile nel visualizzare ed ordinare grandi quantità di dati. In particolare, i risultati ottenuti eseguendo le detections sono stati automaticamente collezionati e raggruppati in un foglio elettronico tramite un script in Python ai fini di agevolare il loro ordinamento e selezione dei risultati migliori.

---

## 6 Sviluppo

### 6.1 Sviluppo algoritmo per ricomposizione delle labels in un'immagine frammentata

Al fine di implementare l'algoritmo è stata creata una libreria contenente numerose funzioni per lavorare con le labels ritornate dal modello che esegue la detection. Ad esempio è possibile ottenere la posizione delle bounding boxes o controllare le loro intersezioni con altre entità e calcolarne l'IoU. Una labels è composta da un array di tipo numpy formato da quattro numeri interi caratterizzanti le coordinate del vertice in alto a sinistra e quello in basso a destra del bounding box, un intero per rappresentare la categoria di appartenenza dell'elemento ed infine un numero decimale compreso tra 1 e 0 per indicarne lo score.

Al fine di rendere l'algoritmo più flessibile viene data la possibilità ad un utente di ridefinire alcune funzioni in modo da adattarle alle proprie esigenze adottando un design pattern di tipo template. In particolare le funzioni ridefinibili sono la **condizione di correlazione**, la **regola di classificazione** più una terza funzione che è in grado di, dato in input un gruppo di bounding boxes, ritornare come output la bounding box risultante dalla loro unione. Tutte e tre le funzioni ridefinite possono essere passate come direttamente tra i parametri dell'algoritmo, altrimenti verranno utilizzate alcune funzioni di default il cui comportamento è quello descritto nella sezione 4.1.

Altri parametri passabili includono: una lista di labels trovate con una detection ed eventualmente pre-processate con NMS, le dimensioni delle regioni, dello stride e della quantità di overlap, sia in verticale che in orizzontale, il valore della tolleranza in pixels ed il valore della soglia di matching in percentuale. Alla fine l'algoritmo ritornerà una lista di tutte le labels presenti nell'immagine dopo l'elaborazione.

Nella seguente sezione ne viene riportata una sua possibile implementazione.

#### 6.1.1 Implementazione

```
1 import numpy as np
2 import modules.labels as lb
3
4 def process_image_labels(boxes, region_size=(300, 300), stride_size=(300, 300),
5     , overlap=(0, 0), tol=0, threshold_match=50, group_condition=None,
6     find_category=None, find_group_label=None):
7
8     # set default functions
9     if group_condition is None:
10         group_condition = lb.group_condition
11     if find_category is None:
```

```

10     find_category = lb.find_category
11 if find_group_label is None:
12     find_group_label = lb.find_group_label
13
14 w, h = lb.get_img_dim_from_boxes(boxes)
15 regions = lb.generate_regions(w, h, region_size, stride_size)
16 # Keeps only the labels intersecting one or more region edges
17 ne_boxes, ne_boxes_indexes = lb.get_intersect_edge_labels(boxes, regions,
18 tol, overlap)
19 # All labels are set as not-checked and not-grouped
20 checked = np.zeros((np.ma.size(ne_boxes, 0)))
21 grouped = np.ones((np.ma.size(ne_boxes, 0))) * (-1)
22 # number of label groups
23 n_group = 0
24 for i in range(len(ne_boxes)):
25     checked_boxes = np.zeros((0, 6))
26     # Select a label and sets it as checked
27     if not checked[i]:
28         checked[i] = True
29         checked_boxes = np.vstack([checked_boxes, ne_boxes[i]])
30         while True:
31             found = 0
32             # For each label checked but not-grouped:
33             for j in range(len(ne_boxes)):
34                 if checked[j] and grouped[j] == -1:
35                     for k in range(len(ne_boxes)):
36                         if not checked[k] and lb.intersection(ne_boxes[j],
37                         ne_boxes[k], tol) and group_condition(ne_boxes[j], ne_boxes[k]) and lb.
38                         intersect_common_edge(ne_boxes[j], ne_boxes[k], regions, tol, overlap) and
39                         lb.matched(ne_boxes[j], ne_boxes[k], threshold_match) and not lb.
40                         belong_same_region_strict_group(ne_boxes[k], checked_boxes, regions, 0):
41                             # The new label is set as checked
42                             checked[k] = True
43                             found += 1
44                             checked_boxes = np.vstack([checked_boxes,
45                             ne_boxes[k]])
46             # The cycle is repeated until no new labels can be found
47             if found == 0:
48                 break
49             # All the checked labels are set as grouped and is them assigned a
50             # number with the purpose to identify their group ID
51             for j in range(len(ne_boxes)):
52                 if checked[j] and grouped[j] == -1:
53                     grouped[j] = n_group
54             n_group += 1

```

```

48
49     # Now that we have grouped the labels , each group is merged into a label
50     # containing all of them
51     new_boxes = np.zeros((0, 6))
52     for i in range(n_group):
53         v = np.zeros((0, 6))
54         for j in range(len(ne_boxes)):
55             if grouped[j] == i:
56                 v = np.vstack((v, ne_boxes[j]))
57         # Find the coordinates of the label containing the whole group
58         (xx1, yy1), (xx2, yy2) = find_group_label(v)
59         # Find the category and the score of the label containing the whole
60         # group
61         categories, scores = find_category(v)
62         for j in range(len(categories)):
63             new_boxes = np.vstack((new_boxes, np.array([xx1, yy1, xx2, yy2,
64             categories[j], scores[j]])))
65     # new_boxes are the labels resulting from grouping algorithm
66
67     # The algorithm is over now thus we can return the results
68     ne_boxes_indexes = ne_boxes_indexes.astype(int)
69     old_boxes = np.delete(boxes, ne_boxes_indexes, 0)
    final_boxes = np.append(new_boxes, old_boxes, 0)

    return final_boxes

```

Listing 1: Python example

### 6.1.2 Test di integrazione

Al fine di verificarne la correttezza sono stati implementati 80 test di integrazione. Essi hanno il compito di testare l'algoritmo su diverse disposizioni e quantità di labels impostando i parametri con differenti valori in modo tale da riconoscerne il maggior numero di errori possibile. Sono state testate in totale cinque disposizioni di labels differenti con due diversi valori di overlap, tolleranza, stride e dimensione delle regioni arrivando ad un totale 80 test passati tutti con esito positivo.

### 6.1.3 Pipeline completa

L'algoritmo appena descritto è solo uno degli step che compongono la pipeline per eseguire la detection completa e poi poterla testare. Il procedimento completo viene descritto nella lista sottostante:

1. Detection;

- 
2. NMS;
  3. Inizializzazione parametri (opzionale);
  4. Ricostruzione labels;
  5. Calcolo metriche (opzionale).

La detection viene effettuata tramite un modello già allenato messo a disposizione dall'azienda ed implementato con Tensorflow. In fase di implementazione sono state invece utilizzate una Faster-RCNN o una SSD (Single Shot MultiBox) allenate sul dataset Coco ed implementate in OpenCV. In particolare, l'elevata efficienza di SSD si è rivelata molto utile in caso servisse una detection molto veloce sacrificando però la qualità.

Successivamente, viene applicato l'algoritmo di NMS con la variante ANMS descritta nella sezione 4.1 ai fini di garantire risultati migliori.

In fase di testing si potrebbe voler testare l'algoritmo di ricostruzione delle labels impostando diverse combinazioni di parametri in modo da selezionare quelli ottimali. Quest'operazione può essere portata a termine lanciando uno script bash in grado di eseguire diverse volte l'algoritmo sulla base di alcuni parametri impostati a piacere utilizzando sempre come input le labels ottenute da NMS. Al termine di ogni esecuzione, i risultati ottenuti vengono salvati sotto forma di file di testo e possono essere passati come input ad uno script in Python in grado di calcolare i valori delle metriche finali. I risultati sono stati salvati nel seguente formato per renderlo compatibile con lo script per calcolare le metriche:

1. **bbox\_left**: la coordinata x del vertice in alto a sinistra della bounding box predetta;
2. **bbox\_top**: la coordinata y del vertice in alto a sinistra della bounding box predetta;
3. **bbox\_right**: la coordinata x del vertice in basso a destra della bounding box predetta;
4. **bbox\_down**: la coordinata y del vertice in basso a destra della bounding box predetta;
5. **score**: affidabilità della predizione;
6. **object\_category**: categoria di appartenenza dell'oggetto predetto.

I risultati ottenuti dalla pipeline completa verranno mostrati più avanti in sezione 7 insieme ad una spiegazione dettagliata delle metriche utilizzate.

## 6.2 Sviluppo sistema di tracking

Lo sviluppo del sistema di tracking su video è più complesso rispetto a quello dell'algoritmo di ricostruzione delle labels perciò ne verrà solamente riportato un diagramma delle classi in UML. E' da sottolineare che il diagramma non è completo ma ne è stata riportata una sua semplificazione.

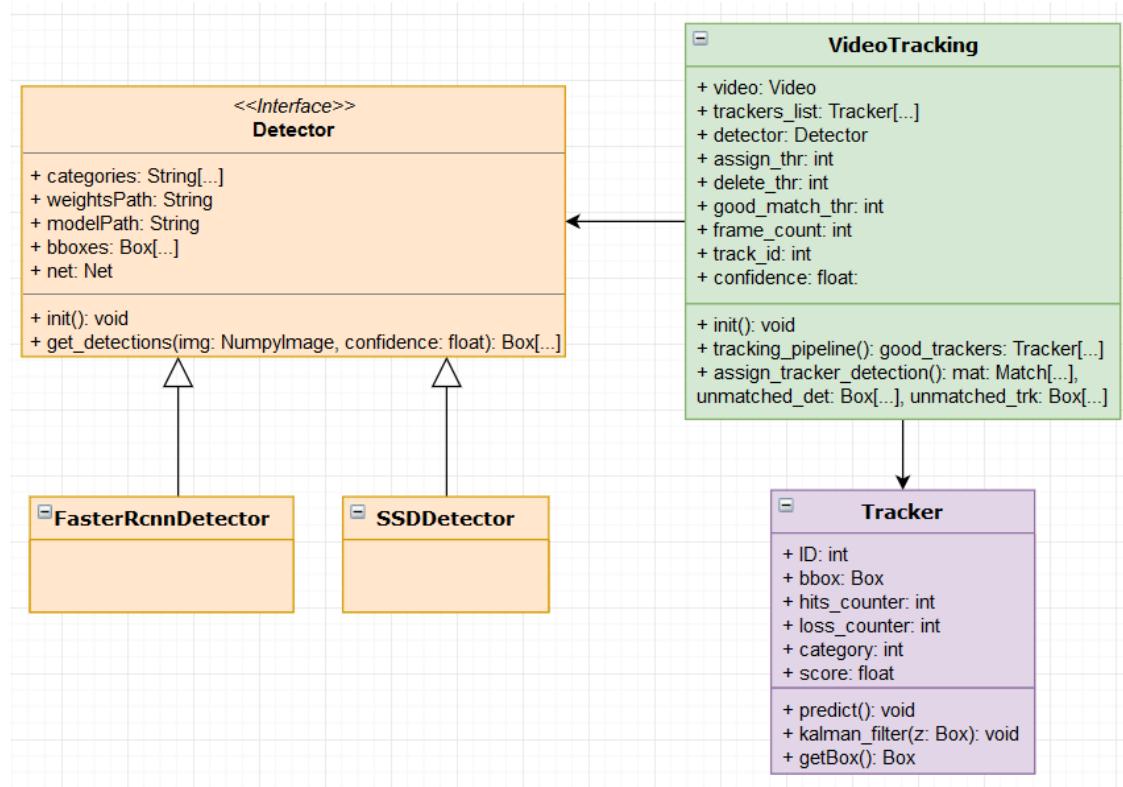


Figura 11: Diagramma delle classi del sistema di tracking

Nel diagramma si può individuare uno strategy pattern tra l'interfaccia *Detector* e le sue due implementazioni *FasterRcnnDetector* ed *SSDDetector*. In questo modo, nel caso in cui servisse una nuova tipologia di rete per effettuare la detection, basta semplicemente implementare l'interfaccia *Detector* con la struttura della nuova rete a patto che quest'ultima sia già stata sufficientemente allenata. Infatti, i campi dati *weightsPath* e *modelPath* indicano i percorsi dover trovare i files da cui leggere i pesi e la struttura del relativo modello. Il metodo che effettua la detection vera e propria è il metodo *get\_detections*, il quale, data in input un'immagine in formato numpy ed una soglia di affidabilità ritorna una lista di labels

---

sotto forma di numpy array corrispondenti agli elementi individuati.

La classe *Tracker* implementa la tipologia di tracker descritta in sezione 4.2. Ogni tracker traccia un solo elemento tenendo quindi traccia del suo ID, la sua bounding box, la sua categoria ed il suo score. I campi dati *hits\_counter* e *loss\_counter* sono i rispettivi contatori di corrette associazioni in frames consecutivi e mancate associazioni sempre in frames consecutivi. I restanti metodi servono per interagire con il filtro di Kalman, in particolare il metodo *kalman\_filter* si occupa della fase di aggiornamento mentre il metodo *predict* svolge la fase di predizione.

La parte principale del sistema è formata dalla classe *VideoTracking* la quale gestisce l'intera pipeline per svolgere il tracking. Ordinatamente, ogni frame del video viene processato dal metodo *tracking\_pipeline*, il quale, ritorna i trackers corrispondenti alle labels da mostrare sul frame corrente. Per prima cosa viene richiamato il metodo *assign\_tracker\_detection* che si occupa di effettuare le associazioni tracker-detection ed ottenere anche tutti i trackers e tutte le detections che per vari motivi non sono state assegnate. In seguito vengono aggiornati tutti i trackers con i relativi filtri e contatori, inoltre vengono creati dei nuovi trackers per tracciare gli eventuali nuovi elementi ed allo stesso tempo vengono cancellati quei trackers che da troppi frames consecutivi non sono stati associati. Infine la classe *VideoTracking* completa l'iterazione mostrando a video le labels degli oggetti identificati e tracciati ed eventualmente salvando il frame appena elaborato in un cartella specifica in modo da poter ricostruire il video con il tracciamento applicato.

Anche in questo caso, come per la detection, è possibile salvare la lista degli oggetti tracciati sotto forma di file di testo. I risultati salvati rispettano il seguente formato:

1. **frame\_index**: indice del frame nel video;
2. **target\_id**: ID dell'oggetto;
3. **bbox\_left**: la coordinata x del vertice in alto a sinistra della bounding box predetta;
4. **bbox\_top**: la coordinata y del vertice in alto a sinistra della bounding box predetta;
5. **bbox\_width**: larghezza in pixels della bounding box predetta;
6. **bbox\_height**: altezza in pixels della bounding box predetta;
7. **score**: affidabilità della predizione;
8. **object\_category**: Categoria di appartenenza dell'oggetto predetto.

Questo formato è lo stesso formato utilizzato dalla sfida VisDrone2019 in quanto lo script in Matlab utilizzato per calcolare le metriche finali e valutare la bontà del tracciamento è stato rilasciato pubblicamente dagli organizzatori stessi.

---

## 7 Risultati ottenuti

### 7.1 Metriche utilizzate per detection su singole immagini

Nell'ambito della computer vision vengono tipicamente utilizzate due tipi di metriche per misurare due diverse proprietà:

- Corretta determinazione della posizione degli oggetti;
- Rilevamento dell'esistenza degli oggetti nell'immagine e la loro corretta classificazione.

Le metriche utilizzate per misurare la bontà dei risultati del progetto sono F1, AP (Average Precision) e mAP (mean Average Precision) le quali misurano la seconda proprietà sopra elencata. Prima di iniziare a descrivere le metriche è necessario definire alcuni concetti che saranno poi coinvolti per il calcolo dei valori delle metriche.

#### 7.1.1 Precision

La precision misura l'accuratezza delle classificazioni ed indica la percentuale di classificazioni corrette sulla base di quelle totali e viene calcolata come:

$$precision = \frac{TP}{TP + FP}$$

Dove TP (True Positives) è il numero di classificazioni corrette e FP (False Positives) è il numero di classificazioni errate. E' da sottolineare che se uno stesso elemento viene individuato più di una volta, solo la prima volta conterà come TP mentre il resto delle volte conterà come FP. Questa metrica fornisce un'idea sulla correttezza dei risultati.

#### 7.1.2 Recall

La recall misura quanti oggetti sono stati individuati e classificati correttamente sulla base degli oggetti totali, viene calcolata come:

$$recall = \frac{TP}{TP + FN}$$

Dove TP (True Positives) è sempre il numero di classificazioni corrette mentre FN (False Negatives) è il numero di oggetti che non sono stati individuati ma che se sarebbe stato corretto individuare. Questa metrica fornisce un'idea sulla completezza dei risultati.

---

### 7.1.3 F1

F1 è una metrica che combina la precision con il recall e può essere interpretata come la media armonica tra la precision e la recall. Il valore massimo ottenibile è 1 raggiungibile nel caso in cui sia la precision che la recall siano pari ad 1 mentre il valore minimo è 0 ottenuto quando entrambe le metriche che lo compongono assumono valore pari a 0. Viene calcolato come:

$$F1 = 2 \times \frac{precision \times recall}{precision + recall}$$

### 7.1.4 Intersection over union

L'intersection over union (IoU) misura il grado di sovrapposizione tra due aree e viene usato per misurare il grado di sovrapposizione tra la box dell'oggetto individuato e la box dell'oggetto reale. Viene calcolata come:

$$IoU = \frac{AI}{AU}$$

Dove AI (Area dell'Intersezione) corrisponde all'area dell'intersezione tra le due bounding boxes e AU (Area dell'Unione) corrisponde all'area dell'unione tra le due bounding boxes. Una label per essere considerata come corretta deve avere un valore di IoU maggiore di una soglia stabilità (per esempio 0.5).

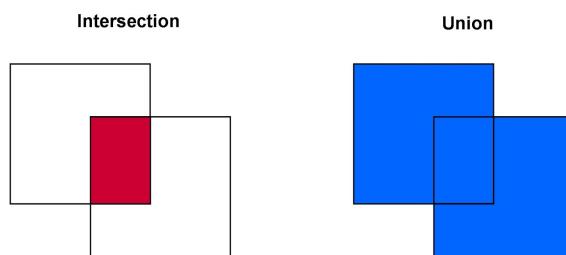


Figura 12: Esempio di intersezione e di unione

### 7.1.5 Average Precision

Un'altra delle metriche utilizzate è l'average precision (AP) e per calcolarla si fa la media delle precisioni di undici diversi valori di recall equamente distribuiti. Questa metrica viene

---

applicata ad una sola categoria di elementi e viene calcolata tramite la seguente formula:

$$AP = \frac{1}{11} \sum_{Recall_i} Precision(Recall_i)$$

Dove  $i=[0, 0.1, 0.2, \dots, 1.0]$  in quanto  $0 <= recall <= 1$ . Inoltre la precisione di uno specifico valore di recall viene calcolata nel seguente modo:

$$Precision(Recall_i) = \max Precision(Recall_j) \quad \text{and} \quad j >= i$$

L'AP è un valore che riassume la forma della curva precision/recall per una data categoria.

#### 7.1.6 Mean Average Precision

La mean average precision (mAP) è la media delle AP di tutte le categorie calcolata su diverse soglie di IoU:

$$mAP_{IoU=x\%} = \frac{1}{n} \sum_{i=0}^n AP_i$$

Dove  $i$  è il numero totale di categorie sulle quali è stata calcolata la propria AP e  $x$  rappresenta diverse soglie di IoU.

## 7.2 Dataset utilizzato per detection su singole immagini

I test sono stati eseguiti su un dataset di immagini satellitari in alta risoluzione con circa 3000 pixels di altezza e 4000 di larghezza. Il modello di rete è stato allenato per riconoscere elementi appartenenti a sessanta categorie, le quali, verranno riportate nella tabella 1 insieme ai loro relativi risultati. Il dataset utilizzato è particolarmente impegnativo da analizzare a causa delle ambiguità tra molte delle sue categorie in quanto molte di esse rappresentano lo stesso elemento ma sotto forma di tipologie diverse. Per esempio, gli oggetti appartenenti alle categorie dalla 8 alla 17 sono tutti dei camion, ma di diversa tipologia, tanto che perfino un essere umano, se non esperto in materia, farebbe fatica a distinguerli. La seconda difficoltà viene aggiunta dal fatto che le immagini sono state prese da satelliti i quali orbitano ad un'altezza di migliaia di chilometri dalla superficie terrestre. Data l'enorme distanza, gli oggetti di dimensioni minori come le automobili sono molto difficili da individuare ed ancora più complesso lo è classificarli nella loro corretta categoria. E' per questo motivo che viene applicata la frammentazione visto che risulta fondamentale non perdere risoluzione durante la detection. In seguito, svolge un ruolo fondamentale l'algoritmo di ricostruzione dell'immagine frammentata in quanto permette di ricomporre le bounding boxes di quegli elementi che sono

---

stati suddivisi su più regioni. In seguito viene mostrata una delle immagini presa dal dataset utilizzato per calcolare le metriche.



Figura 13: Immagine satellitare appartenente al dataset utilizzato

### 7.2.1 Impostazione parametri detection

Il parametro utilizzato per determinare quali labels tenere tra tutte quelle risultanti da una detection è l' **affidabilità**.

Nella seguente lista vengono elencati i parametri che intervengono in fase di ricostruzione delle labels a seguito della detection:

- **Dimensione regioni;**
- **Dimensione stride;**
- **Overlap;**

- 
- **Tolleranza;**
  - **Soglia di matching.**

Una spiegazione più dettagliata riguardo ai parametri la si può trovare in sezione 4.1 o nel *Glossario*. Ogni immagine è stata suddivisa in regioni di dimensioni fisse da 640x640 pixels. A causa della difficoltà nell'individuazione degli elementi, il loro score sarà relativamente basso risultando quindi in una scarsa affidabilità. La **soglia di affidabilità** per la **condizione di correlazione** necessaria per effettuare un match è stata perciò impostata a 0. Di conseguenza, due labels di diversa categoria non potranno mai venire unite. Le altre funzioni ridefinibili sono state lasciate con il loro comportamento di default descritto in sezione 4.1. I restanti parametri sono invece variabili e ne sono state testate diverse combinazioni ai fini di ricercare quelle combinazioni di parametri in grado di fornire i risultati migliori.

La detection è stata effettuata su 100 immagini dove per ciascuna sono state testate 4 soglie diverse di **affidabilità**=[0.01, 0.1, 0.2, 0.3], 5 valori di **overlap**=[0, 1, 2, 3, 4], 5 di **tolleranza**=[0, 1, 2, 4, 4], 4 **soglie di matching**=[20%, 40%, 60%, 80%] per un totale di 625 combinazioni di parametri diverse. Le metriche sono state calcolate come media delle metriche per ogni combinazione di parametri e per ogni singola immagine.

Per quanto riguarda il test, una bounding box individuata tramite detection è considerata come corretta solo se la sua IoU con la bounding box reale è maggiore di una certa soglia di IoU. Questa soglia è la stessa soglia utilizzata per il calcolo di dell'mAP. Per effettuare i test questa soglia è stata fissata al 50%.

## 7.3 Risultati ottenuti nella detection su singole immagini

Nella seguente sezione vengono riportati i risultati ottenuti per misurare la qualità della detection e della frammentazione.

### 7.3.1 Risultati per categoria

Nella tabella sottostante vengono riportate tutte le 60 categorie coinvolte nel test insieme al loro AP. I valori mostrati sono quelli relativi alla migliore combinazione di parametri per ogni specifica categoria. La detection è stata effettuata con **affidabilità** 0.01 in quanto questo è il valore che massimizza i valori di AP e mAP. AP (Original) mostra il valore dell'AP dopo avere effettuato la detection con frammentazione, AP (NMS) è il valore dell'AP dopo aver applicato NMS ed infine AP è il valore dell'AP calcolato dopo aver applicato l'algoritmo di ricostruzione dell'immagine frammentata.

<b>Numero</b>	<b>Categoria</b>	<b>AP (Original)</b>	<b>AP (NMS)</b>	<b>AP</b>
1	Fixed-wing Aircraft	0.224319	0.225709	0.234913
2	Small Aircraft	0.423006	0.421561	0.487984
3	Cargo Plane	0.793246	0.780495	0.809706
4	Helicopter	0.343750	0.343750	0.455165
5	Passenger Vehicle	0.000514	0.000561	0.000616
6	Small Car	0.360364	0.349376	0.354816
7	Bus	0.320155	0.319980	0.322155
8	Pickup Truck	0.004013	0.003817	0.003832
9	Utility Truck	0.022319	0.022180	0.022249
10	Truck	0.093073	0.096008	0.097272
11	Cargo Truck	0.112346	0.110611	0.113069
12	Truck w/Box	0.405223	0.382302	0.389924
13	Truck Tractor	0.011702	0.011205	0.011360
14	Trailer	0.128112	0.118257	0.121111
15	Truck w/Flatbed	0.083729	0.084326	0.086383
16	Truck w/Liquid	0.028560	0.029433	0.029457
17	Crane Truck	0.070424	0.070487	0.103344
18	Railway Vehicle	0.000000	0.000000	0.000000
19	Passenger Car	0.621967	0.634951	0.653586
20	Cargo Car	0.528376	0.535936	0.537790
21	Flat Car	0.408165	0.436869	0.436869
22	Tank Car	0.000501	0.000552	0.000563
23	Locomotive	0.492844	0.498693	0.500454
24	Maritime Vessel	0.132787	0.122395	0.137974
25	Motorboat	0.304536	0.309246	0.317420
26	Sailboat	0.476163	0.492080	0.501056
27	Tugboat	0.375028	0.379075	0.385908
28	Barge	0.229450	0.232367	0.237915
29	Fishing Vessel	0.131085	0.134347	0.145607
30	Ferry	0.451626	0.445131	0.494575
31	Yacht	0.554554	0.561531	0.583826
32	Container Ship	0.562968	0.555579	0.624139
33	Oil Tanker	0.562263	0.615124	0.615124
34	Engineering Vehicle	0.114510	0.114657	0.116365
35	Tower Crane	0.012496	0.007083	0.007517

— Continua nella pagina successiva —

Numero	Categoria	AP (Original)	AP (NMS)	AP
36	Container Crane	0.053083	0.050028	0.060336
37	Reach Stacker	0.314528	0.317667	0.320265
38	Straddle Carrier	0.573863	0.572912	0.574765
39	Mobile Crane	0.131302	0.139258	0.155806
40	Dump Truck	0.166618	0.169499	0.169579
41	Haul Truck	0.868698	0.859369	0.889144
42	Scraper/Tractor	0.028915	0.028613	0.031487
43	Front Loader/Bulldozer	0.129784	0.138587	0.140348
44	Excavator	0.340663	0.335421	0.336678
45	Cement Mixer	0.173915	0.182477	0.182591
46	Ground Grader	0.266859	0.265571	0.265616
47	Hut/Tent	0.025208	0.023967	0.026293
48	Shed	0.009840	0.009854	0.010107
49	Building	0.535869	0.482216	0.490359
50	Aircraft Hangar	0.019961	0.025452	0.027149
51	Damaged Building	0.020142	0.021223	0.022819
52	Facility	0.125133	0.114053	0.139637
53	Construction Site	0.017198	0.016450	0.016892
54	Vehicle Lot	0.087038	0.081471	0.095994
55	Helipad	0.020963	0.022512	0.022866
56	Storage Tank	0.453776	0.452578	0.474606
57	Shipping Container Lot	0.159574	0.140996	0.149238
58	Shipping Container	0.101220	0.102641	0.102708
59	Pylon	0.657612	0.661616	0.661769
60	Tower	0.070426	0.071527	0.071872

Tabella 1: Lista delle categorie del dataset utilizzato per effettuare i test con relativo AP

Osservando i risultati si può subito notare che per ogni categoria si ha che  $AP \geq AP(NMS)$  notando quindi un miglioramento delle metriche a seguito dell'applicazione dell'algoritmo. Per alcune categorie si ha che  $AP = AP(NMS)$ , in questi casi significa che nessuna delle bounding boxes relative a quella categoria è stata coinvolta nella frammentazione. Per alcune categorie può succedere che  $AP(\text{Original})$  risulti maggiore sia di  $AP(NMS)$  che di  $AP$ , questo fenomeno si verifica con più frequenza in presenza di molti oggetti parzialmente sovrapposti

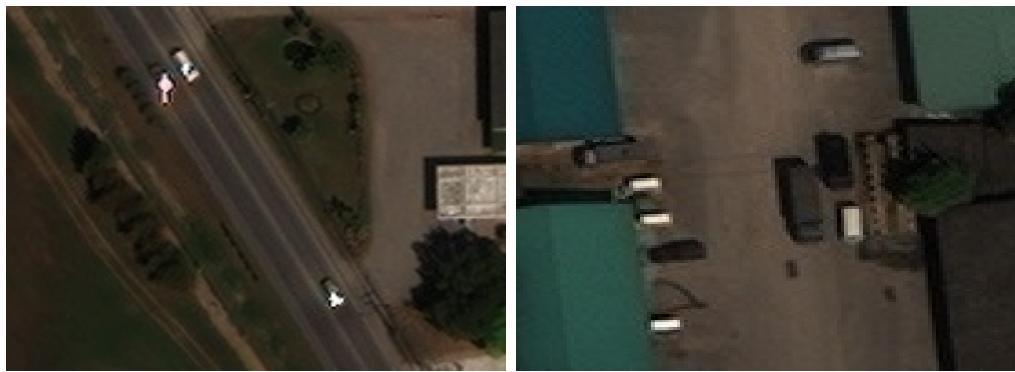


Figura 14: Esempi di oggetti molto difficili da individuare e classificare

tra loro. In questo caso NMS potrebbe causare delle imprecisioni scartando delle bounding boxes che dovrebbero essere in realtà mantenute. Alcune categorie come la numero 5 e la numero 8 presentano un valore di AP particolarmente basso. Si tratta di oggetti che a causa della bassa risoluzione risultano molto piccoli ed ambigui perfino per un occhio umano, per cui, anche il modello spesso commette errori nella loro individuazione e classificazione.

### 7.3.2 Risultati per parametri

La seguente tabella mostra riporta alcuni dei risultati ottenuti con diverse combinazioni di parametri mostrando i valori di F1 e mAP. mAP(NMS) indica il valore dell'mAp calcolato dopo aver effettuato NMS sull'immagine frammentata mentre mAP è il valore dell'mAP calcolato dopo aver applicato l'algoritmo di ricostruzione dell'immagine frammentata riportandone anche l'aumento in percentuale. Affidabilità è il valore della **soglia di affidabilità** sopra la quale non scartare una detection.

Affidabilità	Stride	Overlap	Tolleranza	Match(%)	F1	mAP(NMS)	mAP
0.01	640	0	0	0	0	0	0 (+0.0%)
0.01	640	0	0	0	0	0	0 (+0.0%)
0.01	640	0	0	0	0	0	0 (+0.0%)
0.01	640	0	0	0	0	0	0 (+0.0%)
0.01	640	0	0	0	0	0	0 (+0.0%)
0.01	640	0	0	0	0	0	0 (+0.0%)
0.01	640	0	0	0	0	0	0 (+0.0%)
0.1	640	0	0	0	0	0	0 (+0.0%)
0.1	640	0	0	0	0	0	0 (+0.0%)
0.1	640	0	0	0	0	0	0 (+0.0%)
0.1	640	0	0	0	0	0	0 (+0.0%)
0.1	640	0	0	0	0	0	0 (+0.0%)
0.1	640	0	0	0	0	0	0 (+0.0%)
0.1	640	0	0	0	0	0	0 (+0.0%)
0.2	640	0	0	50	0	0.249199	0.250742(+0.0%)
0.2	640	2	4	60	0	0.249199	0.253616(+0.0%)
0.2	640	4	4	80	0	0.249199	0.252815(+0.0%)
0.2	640	0	0	0	0	0	0 (+0.0%)
0.2	640	0	0	0	0	0	0 (+0.0%)
0.2	640	0	0	0	0	0	0 (+0.0%)

Tabella 2: Risultati generali detection con diversi parametri

## 7.4 Metriche utilizzate per tracking su video

L'obiettivo del tracking è quello di identificare tutti gli oggetti in un video e tracciarli nel tempo. Quest'ultimo compito viene svolto assegnando un ID unico a ciascun oggetto assicurandosi che esso rimanga lo stesso per tutta la durata del video. Di conseguenza nel tracking sono due gli aspetti fondamentali di cui tenere conto:

- 
- Precisione con cui viene determinata la locazione di un oggetto;
  - Correttezza del tracciamento (anche in presenza di occultamento, perdita di qualità, etc...)

In seguito vengono descritte le metriche utilizzate per valutare la qualità del tracciamento.

#### 7.4.1 Mostly Tracked Trajectories

Mostly Tracked Trajectories (MT), misura la percentuale di elementi, rispetto a quelli totali, che sono stati tracciati correttamente per almeno l'80% della loro traiettoria. Indica quindi la correttezza del tracciamento, valori più alti sono considerati migliori.

#### 7.4.2 Mostly Lost Trajectories

Mostly Lost Trajectories (ML), misura la percentuale di elementi, rispetto a quelli totali, che sono stati tracciati correttamente per non più del 20% della loro traiettoria. Al contrario di MT, valori più bassi sono considerati migliori.

#### 7.4.3 ID switches

ID switches (IDs), viene definito come il numero di volte che un oggetto tracciato cambia il suo ID assegnato. Questo può accadere per esempio nel momento in cui due elementi simili incrociano le loro traiettorie ed i trackers non sono in grado di distinguerli. Anche in questo caso, valori minori esprimono risultati migliori.

#### 7.4.4 Fragments

Fragments (FM) conta il numero di volte che una traiettoria viene interrotta durante il tracciamento. Può accadere per esempio a seguito di un occultamento di un elemento, in questo caso un oggetto verrebbe momentaneamente perso e di conseguenza la sua traiettoria risulterebbe frammentata. Valori minori sono considerati migliori.

#### 7.4.5 MOTP

MOTP (Multiple Object Tracking Precision) rappresenta la capacità del sistema di tracking nello stimare con precisione la posizione degli elementi del video attraverso tutti i suoi frames. Viene calcolata nel seguente modo:

$$MOTP = \frac{\sum_t \sum_i d_t^i}{\sum_t c_t}$$

---

Dove  $t=[1,\dots,n]$  è l'indice dei frames del video ed  $n$  è il numero di frames totali,  $i=[1,\dots,c_t]$  è l'indice delle associazioni detection-tracker effettuate nel frame  $t$ ,  $d_t^i$  è la distanza tra la posizione reale dell'oggetto  $i$  e la sua corrispondente posizione stimata nel frame  $t$ ,  $c_t$  è il numero di associazioni detection-tracker effettuate nel frame  $t$ .

#### 7.4.6 MOTA

MOTA (Multiple Object Tracking Accuracy) fornisce una misura delle prestazioni del sistema di tracking nel riconoscere gli oggetti presenti nel video e costruire le loro traiettorie. Viene calcolata nel seguente modo:

$$MOTA = 1 - \frac{\sum_t (fn_t + fp_t + mme_t)}{\sum_t g_t}$$

Dove  $fn_t$  è il numero di falsi negativi,  $fp_t$  è il numero di falsi positivi,  $mme_t$  è il numero di mismatches. Un mismatch avviene quando durante l'associazione detection-tracker un tracker viene associato ad una detection che rappresenta un oggetto diverso da quello associato nel frame precedente. Infine  $g_t$  è il numero di oggetti presenti nel frame  $t$ .

#### 7.4.7 IDF1

IDF1 è una metrica molto simile ad F1 utilizzata come metrica per singole immagini. Questa metrica applicata nell'ambito del tracking fornisce un bilanciamento tra la *precision* media e la *recall* media attraverso la loro media armonica.

$$IDF1 = \frac{2IDTP}{2IDTP + IDFP + IDFN}$$

Dove IDTP, IDFP, IDFN rispettivamente sono la media del numero di detections avvenute correttamente, falsi positivi e falsi negativi calcolati su tutti i frames.

### 7.5 Dataset utilizzato per tracking su video

Per testare la correttezza del tracciamento sono stati utilizzati alcuni video messi a disposizione dalla sfida VisDrone2019, così come lo è anche lo script utilizzato per calcolare le metriche. I video sono stati girati da alcune telecamere montate su dei droni fatti volare riprendendo diversi scenari che spaziano da aree urbane a zone rurali. I video sono stati girati sotto diverse condizioni meteo, di luminosità a diverse altezze ed angolazioni rendendo quindi particolarmente impegnativi sia il tracking che la detection. Un' ulteriore difficoltà è dovuta al fatto che alcuni frames presentano effetti di sfocatura causata dal movimento.

---

Le dimensioni dei frames variano da video a video così come varia la lunghezza in frames di ogni video. Questa volta sono presenti solamente 5 categorie di oggetti diversi e facilmente distinguibili tra loro:

- Auto;
- Bus;
- Camion;
- Persona;
- Furgone.

E' da notare la categoria Persona è l'unica categoria che non rappresenta un mezzo di trasporto.

### 7.5.1 Impostazione parametri tracking

Nel tracking entrano in gioco 3 parametri variabili ovvero:

- **Soglia di cancellazione;**
- **Soglia di validazione;**
- **Soglia di assegnazione.**

Una spiegazione più dettagliata riguardo ai parametri la si può trovare in sezione 4.2 o nel *Glossario*. Per effettuare i test, durante la detection è stata tenuta una **Soglia di affidabilità** pari a 0.3 e non è stata applicata la frammentazione ai singoli frames in quanto l'operazione sarebbe stata troppo onerosa in termini di tempo mentre usando una Faster-RCNN su ogni frame completo si possono raggiungere qualità di frame-rate accettabili anche per elaborazioni in tempo reale (circa 2 frames al secondo senza utilizzare GPU e con CPU Intel core i5). Il test è stato effettuato su 7 video, con 3 valori di **soglia di validazione**=[1, 3, 5], 3 valori di **soglia di cancellazione**=[1, 5, 10] e due valori di **soglia di assegnazione**=[0.1, 0.5] per un totale di 18 combinazioni di parametri diverse. Le metriche sono state calcolate su ogni video, su ogni categoria e come media delle metriche per ogni combinazione di parametri.

Per quanto riguarda questione dello scambio dell'ID, è da sottolineare che se due oggetti che, incrociando le loro traiettorie, si dovessero ritrovare con un ID diverso, questo evento conterà come 2 scambi di ID invece di 1.

---

## 7.6 Risultati ottenuti nel tracking su video

Nella seguente sezione vengono riportati i risultati ottenuti per misurare la qualità del sistema di tracking. In quanto l'obiettivo è quello di misurare l'efficacia del tracking e non della detection, quest'ultima non è stata effettuata ma le labels di ogni frame sono state lette da un file di testo contenente le detections esatte, uguali a quelle contenute nel file con il quale vengono confrontati i risultati del tracking ottenuti ai fini di ricavarne le metriche. Successivamente verranno mostrati dei risultati eseguendo le detections con un modello vero e proprio. Le metriche IDF1, Rcll, Prcn, MT, ML, MOTA e MOTP sono espresse sotto forma di percentuali e, ad eccezione di ML, indicano risultati migliori per valori maggiori. Al contrario, le metriche FP, FN, IDs ed FM sono espresse in valore quantitativo ed indicano risultati migliori per valori minori.

### 7.6.1 Risultati per categoria

Nella seguente tabella vengono riportate tutte le cinque categorie insieme ai relativi risultati. Le due metriche Rcll (recall) e Prcn (precision) sono le stesse che sono state utilizzate nella detection su singole immagini ma in questo ambito sono state calcolate come recall e precisione media su ogni frame e su ogni video. I seguenti valori sono stati ottenuti impostando una **soglia di cancellazione=1**, **soglia di validazione=1** e **soglia di assegnazione=0.1** in quanto questa combinazione di parametri ha permesso di ottenere risultati migliori rispetto alle altre per quanto riguarda la maggior parte delle metriche. In generale è la categoria

Categoria	IDF1	Rcll	Prcn	MT	LT	FP	FN	IDs	FM	MOTA	MOTP
Auto	91.0	98.4	98.8	96.0	1.5	380	505	312	119	96.2	93.1
Bus	93.6	95.5	99.6	100.0	0	1	12	7	7	92.4	95.6
Camion	98.7	99.3	99.6	100.0	0	6	10	5	2	98.5	95.0
Persona	84.7	97.8	98.7	92.4	2.2	420	704	462	200	95.0	90.1
Furgone	93.4	98.6	99.1	95.0	0	58	97	32	21	97.2	93.5

Tabella 3: Risultati tracking per categoria

Persona quella più difficile da tracciare sia per la numerosità delle persone stesse sia per il numero di sovrapposizioni che si verificano tra di esse, lo si può notare dal fatto che le relative metriche sono leggermente più basse rispetto a quelle delle altre categorie. Le categorie Auto e Persona sono le più frequenti sia nei video sia nel mondo reale perciò è più difficile tracciarle ed aumenta la probabilità che vengano commessi degli errori.

### 7.6.2 Risultati per parametri

La tabella sottostante riporta i risultati ottenuti nel tracking utilizzando diverse combinazioni parametri SC (**soglia di cancellazione**), SV (**soglia di validazione**) ed SA (**soglia di assegnazione**). Per quanto riguarda il valore dei pesi delle metriche per decidere se validare un assegnazione o meno, è stato dato maggior peso all'IoU ed un peso minore alle metriche che misurano la differenza tra le aree e le dimensioni dei lati.

SV	SC	SA	IDF1	Rcll	Prcn	MT	ML	FP	FN	IDs	FM	MOTA	MOTP
1	1	0.1	88.6	98.2	98.8	94.3	1.7	865	1328	818	349	95.8	91.9
1	1	0.5	87.7	98.1	98.8	94.3	1.9	875	1359	834	371	95.7	91.9
1	5	0.1	86.9	98.2	95.0	94.7	1.5	3684	1289	889	335	91.8	91.9
1	5	0.5	86.3	98.2	94.8	94.7	1.7	3833	1318	898	368	91.6	91.9
1	10	0.1	84.8	98.2	90.8	95.0	1.3	7144	1265	927	359	87.0	91.9
1	10	0.5	84.3	98.2	90.4	95.0	1.5	7463	1279	956	370	86.5	91.9
3	1	0.1	88.5	95.9	99.1	88.7	3.2	638	2948	459	270	94.4	92.0
3	1	0.5	87.8	95.8	99.1	88.7	3.2	633	3025	461	275	94.3	92.0
3	5	0.1	87.3	96.0	96.3	89.1	3.2	2678	2903	513	285	91.5	92.0
3	5	0.5	86.7	95.9	96.2	89.1	3.2	2734	2954	522	294	91.4	92.0
3	10	0.1	85.7	96.0	93.0	89.1	2.7	5216	2883	541	298	88.0	91.9
3	10	0.5	85.3	95.9	92.8	89.1	2.9	5337	2921	565	304	87.7	92.0
5	1	0.1	88.2	94.1	99.2	81.3	5.0	557	4240	362	233	92.8	92.1
5	1	0.5	87.3	94.0	99.2	80.9	5.3	553	4340	366	236	92.7	92.1
5	5	0.1	87.1	94.2	96.7	81.7	4.8	2287	4189	400	254	90.4	92.0
5	5	0.5	86.5	94.1	96.7	81.5	4.8	2340	4239	413	258	90.3	92.1
5	10	0.1	85.8	94.2	93.8	81.7	4.6	4437	4172	423	263	87.4	92.0
5	10	0.5	85.3	94.1	93.6	81.7	4.6	4594	4214	440	265	87.1	92.1

Tabella 4: Risultati tracking per parametri

Nella tabella si può notare che all'aumentare del valore della **soglia di validazione** diminuisce il numero di falsi positivi, scambi di ID e di traiettorie frammentate in quanto vengono tracciati meno elementi ma in modo più efficace. Allo stesso tempo, però, le altre metriche tendono a diminuire in quanto elementi di breve comparsa nei video verranno ignorati o solo parzialmente tracciati. All'aumentare della **soglia di cancellazione** aumenta di molto il numero di falsi positivi al guadagno di una leggera riduzione del numero di falsi negativi ma le altre metriche tendono a rimanere costanti o addirittura peggiorare. Questo è dovuto anche al fatto che in questo caso le detections sono perfette ed il rischio di perdere un

---

elemento tracciato si verifica solo in caso di forte sovrapposizione con altri elementi. Inoltre, le metriche ottenute con **soglia di accettazione** maggiore sono quasi sempre leggermente inferiori a quelle ottenute con la soglia più bassa, questo perché una soglia alta tende a scartare molte coppie tracker-detection che in realtà sarebbero state corrette, d'altra parte una soglia bassa comporta il rischio di associazioni errate.

### 7.6.3 Risultati per parametri con detection

La seguente tabella mostra i risultati ottenuti applicando il tracking sul set di video utilizzato in precedenza, questa volta però, gli oggetti sono stati individuati utilizzando un modello per eseguire la detection piuttosto che tramite le loro labels esatte. In questo modo si va a testare il sistema di tracking completo in condizioni reali dove le labels individuate per mezzo della detection non sono esatte ma possono contenere degli errori in grado di influenzare il risultato del tracking stesso. Anche le combinazioni di parametri sono le stesse in modo da poter confrontare i risultati di questa tabella con quelli ottenuti nella tabella precedente.

SV	SC	SA	IDF1	Rcll	Prcn	MT	ML	FP	FN	IDs	FM	MOTA	MOTP
1	1	0.1	50.0	68.2	81.2	69.6	16.1	5007	10076	702	477	50.2	84.1
1	1	0.5	50.0	68.2	81.2	69.6	16.1	5009	10079	702	473	50.2	84.1
1	5	0.1	50.2	70.0	76.5	70.5	15.6	6832	9518	609	431	46.5	83.6
1	5	0.5	49.9	70.0	76.4	70.5	15.6	6867	9512	606	426	46.4	83.6
1	10	0.1	49.9	70.7	73.3	71.4	15.2	8155	9299	578	420	43.1	83.4
1	10	0.5	49.7	70.7	73.2	71.4	15.2	8206	9287	575	418	43.0	83.4
3	1	0.1	50.9	66.8	83.3	69.2	16.1	4251	10535	642	444	51.3	84.5
3	1	0.5	50.9	66.7	83.3	69.2	16.1	4249	10543	641	440	51.3	84.5
3	5	0.1	51.0	69.1	78.5	69.6	16.1	5996	9795	584	412	48.3	83.8
3	5	0.5	50.7	69.1	78.5	70.1	16.1	6006	9794	581	405	48.3	83.8
3	10	0.1	50.6	69.9	75.2	70.5	16.1	7324	9541	561	409	45.0	83.6
3	10	0.5	50.4	69.9	75.2	70.5	16.1	7316	9538	558	406	45.1	83.6
5	1	0.1	51.5	65.6	84.6	68.7	16.1	3801	10893	613	420	51.7	84.8
5	1	0.5	51.5	65.6	84.6	68.7	16.1	3797	10900	614	418	51.7	84.8
5	5	0.1	51.5	68.4	80.1	69.2	16.1	5398	10021	574	400	49.5	84.0
5	5	0.5	51.2	68.4	80.0	69.2	16.1	5403	10021	571	396	49.5	84.0
5	10	0.1	51.1	69.3	76.7	69.6	16.1	6674	9721	552	339	46.5	83.7
5	10	0.5	50.9	69.3	76.7	69.6	16.1	6658	9725	549	397	46.6	83.7

Tabella 5: Risultati tracking per parametri

Come è facile aspettarsi, quasi tutte le metriche mostrano risultati sensibilmente peggiori rispetto a quelli mostrati nella tabella precedente. In particolare, a causa degli errori introdotti dalla detection risultano esserci molti più falsi negativi e falsi positivi rispetto al tracciamento eseguito con le labels delle detections esatte. A causa del significativo aumento dei falsi negativi è presente una lieve diminuzione del numero di scambi degli ID. Questo comporta un valore MOTA praticamente dimezzato rispetto alla versione precedente mentre

---

la metrica MOTP subisce solo una moderata diminuzione. Per lo stesso motivo anche le metriche Rcll e Prcll presentano valori più bassi riducendo di conseguenza il valore di IDF1. Anche in questo caso i parametri giocano un ruolo fondamentale nel determinare il risultato migliore. Nello specifico, valori alti della **soglia di validazione** e della **soglia di cancellazione** aumentano la robustezza del tracciamento a discapito della sua qualità. Infatti i valori di IDs e FM minimi si hanno in corrispondenza dei valori dei parametri SV e SC maggiori. Valori maggiori di SV, filtrando le detections errate, diminuiscono il numero dei falsi positivi mentre valori maggiori di SC, prolungando la durata del tracciamento anche in caso di scomparsa dell'elemento tracciato, riducono il numero di falsi negativi. La metrica IDF1 rimane più o meno stabile per tutte le combinazioni di parametri significando quindi che essa è più influenzata dalla qualità della detection che da quella del tracking. In generale, aumentando le soglie di cancellazione e di validazioni si ha un tracciamento migliore e più resistente agli errori mentre diminuendole si ha un tracking meno preciso ma più completo in quanto verrebbero tracciati un numero maggiore di elementi.

---

## 8 Glossario

### A

**Affidabilità** Soglia di probabilità minima che lo score di una labels deve possedere per essere considerata come affidabile e non venire scartata.

### B

**Bounding box** E' un rettangolo che identifica il perimetro entro quale l'oggetto riconosciuto si trova.

### C

**Categoria** L'obiettivo della classificazione è quello di assegnare all'oggetto individuato la sua categoria di appartenenza.

### L

**Label** Etichetta che viene data ad ogni elemento riconosciuto e ne indica la categoria di appartenenza, una bounding box ed uno score.

### O

**Overlap** L'overlap indica lo spessore in pixels dei confini delle regioni. E' usato per sapere se un' entità interseca un confine o meno.

### R

**Raggruppamento** Insieme di labels correlate tra loro tale che da una qualsiasi label del gruppo sia possibile raggiungere qualsiasi altra label dello stesso insieme passando solo per label *vicine*<sup>6</sup>.

**Regione** Una sezione di un'immagine con un'area di dimensione minore dell'area dell'immagine originale.

---

<sup>6</sup>Due label sono considerate vicine se sono intersecate tra di loro ed intersecano lo stesso confine di regione

---

## S

**Score** La probabilità che la categoria associata all'elemento riconosciuto sia quella corretta.

**Soglia di matching** Proporzione entro la quale le dimensioni di due bounding boxes devono combaciare per poter essere unite.

**Stride** E' un attributo da tenere conto in fase di frammentazione di un'immagine ed indica quanti pixels della regione tralasciare, sia in verticale che in orizzontale, prima che cominci quella successiva. Lo stride orizzontale può avere dimensione diversa da quello verticale. La sua applicazione ha come scopo quello di creare delle zone di sovrapposizione tra le varie regioni.

## T

**Tolleranza** La distanza in pixels per la quale una label può discostare da una zona di interesse per cui sia ancora considerata essere intersecata con la zona di interesse.

---

## 9 Bibliografia

### Riferimenti bibliografici

- [1] Sito web dell'azienda Studiomapp.  
<https://www.studiomapp.com/en/>
- [2] Alex Krizhevsky, Ilya Sutskever, Geoffrey E. Hinton. *ImageNet Classification with Deep Convolutional Neural Networks*. University of Toronto.
- [3] Ross Girshick, Jeff Donahue, Trevor Darrell, Jitendra Malik. *Rich feature hierarchies for accurate object detection and semantic segmentation*. UC Berkeley.
- [4] Shaoqing Ren, Kaiming He, Ross Girshick, Jian Sun. *Faster R-CNN: Towards Real-Time Object Detection with Region Proposal Networks*. Microsoft Research.
- [5] Vit Ruzicka, Franz Franchetti. *Fast and accurate object detection in high resolution 4K and 8K video using GPUs*. Carnegie Mellon University.
- [6] David S.Bolme, J.Ross Beveridge, Bruce A.Draper, Yui Man Lui. *Visual Object Tracking using Adaptive Correlation Filters*. Colorado State University.
- [7] Guida all'utilizzo del framework Tensorflow.  
<https://www.tensorflow.org/>
- [8] Guida all'utilizzo del modulo matplotlib.  
<https://matplotlib.org/>
- [9] Guida all'utilizzo del modulo numpy.  
<https://www.numpy.org/>
- [10] Guida all'utilizzo del modulo OpenCV.  
<https://opencv.org/>
- [11] Studio delle metriche per l'object detection.  
[https://medium.com/@jonathan\\_hui/map-mean-average-precision-for-object-detection-45c121a31173](https://medium.com/@jonathan_hui/map-mean-average-precision-for-object-detection-45c121a31173)
- [12] Tracking con detection  
<https://towardsdatascience.com/computer-vision-for-tracking-8220759eee85>

- 
- [13] Rudolf Emil Kálmán. *A New Approach to Linear Filtering and Prediction Problems*. Research Institute for Advanced Study, Baltimore.
  - [14] Harold William Kuhn. *The Hungarian method for the assignment problem*. Princeton University.
  - [15] Yuan Li, Chang Huang, Ram Nevatia. *LearningtoAssociate: Hybrid Boosted Multi-Target Tracker for Crowded Scene*. University of Southern California.
  - [16] Drone Challenge 2019  
<http://aiskyeye.com/views/index>
  - [17] Keni Bernardin, Rainer Stiefelhagen *Evaluating Multiple Object Tracking Performance: The CLEAR MOT Metrics*. Karlsruhe Institute of Technology.
  - [18] Ergys Ristani1, Francesco Solera, Roger S. Zou, Rita Cucchiara, Carlo Tomasi. *Performance Measures and a Data Set for Multi-Target, Multi-Camera Tracking*. Duke University, University of Modena and Reggio Emilia.

---

## **10 Appendice**