UNIVERSITÀ degli STUDI di CATANIA

Multimedia e Laboratorio – Analisi e filtraggio del movimento in video .mp4

Giulia Di Mauro – 1000015058

A.A. 2023/2024

Prof. Dario Allegra

Prof. Filippo Stanco

Sommario

Introduzione *3*

Motivazioni **4**

Obiettivi **5**

Riferimenti teorici *6*

Il movimento nei video **6**

Rappresentazione del movimento 6

Criteri di stima del movimento 6

Sistemi di stabilizzazione video **10**

Sistemi di stabilizzazione digitale 10

Metriche di qualità **13**

Mean square error (MSE) 14

Peak signal to noise ratio (PSNR) 14

Structural Similarity Index (SSIM) 15

Considerazioni finali **16**

Applicazioni pratiche *17*

Fasi del Progetto **17**

Stima del movimento 17

Filtraggio del movimento 21

Deformazione post-processing 22

Valutazione tramite metriche di qualità 25

Risultati ottenuti *27*

Conclusioni *31*

Analisi predittiva ed estrazione di informazioni dai dati: un punto di partenza **32**

Introduzione

Il progetto proposto mira all'analisi e al filtraggio del movimento in video in formato .mp4. Nel contesto dell'era digitale, l'elaborazione multimediale riveste un ruolo importante poiché consente di manipolare e migliorare l'esperienza visiva dei contenuti video.

L'analisi del movimento è un aspetto cruciale in molte applicazioni scientifiche, ad esempio nel contesto della computer vision[1], dove il movimento degli oggetti nella scena viene studiato per monitorare il traffico, per garantire sorveglianza e sicurezza, per riconoscere specifiche azioni umane, per la realtà aumentata. Comprendere il movimento diventa essenziale per estrarre informazioni significative e prendere decisioni intelligenti basate sui dati visivi.

Questo progetto si avvia con un'analisi delle motivazioni e degli obiettivi che lo hanno ispirato, seguita da un'esposizione dei principi teorici fondamentali su cui si fonda. Successivamente, si procede con la rappresentazione pratica del processo di filtraggio del movimento nei video arricchita dall'attenta valutazione della qualità del video attraverso l'applicazione di diverse metriche.

Il lavoro è visibile al seguente link: https://github.com/Giulia341/Multimedia_progetto.git.

[1] Settore dell'Intelligenza Artificiale che si occupa dello sviluppo di algoritmi in grado di interpretare e comprendere il mondo multimediale.

Motivazioni

Di seguito vengono riportate le principali motivazioni per cui nasce questo progetto.

- **Comprensione del movimento**: il movimento è una caratteristica fondamentale dei video; la sua analisi fornisce informazioni preziose su varie scene e oggetti presenti nel video, ad esempio il riconoscimento di azioni umane, monitoraggio del traffico, sicurezza e molto altro.
- **Miglioramento della qualità**: i video possono essere soggetti a vari disturbi durante l'acquisizione o la trasmissione, come sfocatura, tremore della telecamera, artefatti dovuti alla compressione. L'applicazione di tecniche di filtraggio del movimento può contribuire significativamente a migliorare la qualità visiva dei video, rendendoli più chiari e più piacevoli da guardare.
- **Applicazioni pratiche**: le applicazioni pratiche di questo progetto sono molteplici. Ad esempio, nell'ambito cinematografico, il filtraggio del movimento può essere utilizzato per stabilizzare le riprese e ridurre il movimento della telecamera. In campo scientifico può essere utilizzato per studiare il comportamento di oggetti in movimento o per monitorare la progressione di eventi dinamici.

Le moderne tecnologie digitali hanno trasformato radicalmente il modo in cui interagiamo con i media visivi, ponendo al centro l'importanza del movimento nei video. La capacità di catturare, rappresentare e stabilizzare il movimento non solo migliora l'esperienza dell'utente, ma è essenziale per molteplici ambiti, tra cui l'intrattenimento, la sorveglianza, la medicina e l'industria. Comprendere le motivazioni dietro lo studio del movimento video è cruciale per indirizzare le ricerche e sviluppare soluzioni innovative per le sfide attuali

Obiettivi

Il qui presente progetto ha lo scopo di esplorare soluzioni innovative per l'analisi e il filtraggio del movimento nei video .mp4. Attraverso una combinazione di teoria e pratica si cerca di contribuire al miglioramento della qualità visiva dei video digitali e alla comprensione più approfondita del movimento nelle sequenze video.

Il progetto si impegna ad utilizzare algoritmi avanzati di analisi del movimento per individuare e comprendere il tipo di movimento presente nel video. Questo comprende il rilevamento di movimenti lineari, rotazionali e complessi di oggetti che si muovono all'interno del frame.

Utilizzando tecniche impiegate non sono nel contesto della elaborazione multimediale ma anche nella computer vision questo progetto è in grado di estrarre informazioni dettagliate sul movimento presente nel video.

Gli algoritmi utilizzati mirano a ridurre o eliminare i disturbi visivi presenti nei video, come tremore della telecamera, sfocatura o instabilità dell'immagine. Attraverso la progettazione e implementazione di filtri intelligenti, come la stabilizzazione dell'immagine il progetto si impegna a migliorare la qualità visiva complessiva del video.

La parte essenziale di tale progetto è la valutazione delle prestazioni dei metodi proposti. Si utilizzano criteri oggettivi, nonché metriche, per valutare l'efficacia degli algoritmi, misurando parametri di qualità visiva come nitidezza, stabilità dell'immagine e fedeltà al movimento originale. Questo consente di identificare eventuali limitazioni o aree di miglioramento e di raffinare ulteriormente l'approccio.

Tali metodi vengono applicati a scenari concreti, infatti si utilizzano video reali per dimostrare l'utilità e l'applicabilità dei filtri di movimento, migliorando così la qualità visiva e l'interpretazione del contenuto. Questo lavoro permette di convalidare l'efficacia dell'approccio utilizzato e di fornire soluzioni concrete per le possibili sfide pratiche incontrate nell'analisi e nella manipolazione dei video.

Riferimenti teorici

Il movimento nei video

Nei video, il movimento è una componente essenziale che contribuisce alla narrazione visiva e all'esperienza degli spettatori. Oltre ai movimenti degli oggetti all'interno del frame, è importante considerare anche i movimenti della telecamera stessa, poiché influenzano notevolmente la percezione dell'azione e l'immersione dell'utente.

Rappresentazione del movimento

Quando si parla di movimento bisogna innanzitutto considerare se esso avviene in 3D o in 2D.

Nel contesto del movimento 2D la differenza di posizione finale ed iniziale di un punto nello spazio 2D è indicata con *d(x)* e prende il nome di "*displacement*" o "*vettore di movimento 2D*" o "*motion vector*" (MV).

L'insieme dei *d(x)* per ogni punto *x* dell'immagine è chiamato "*Campo di movimento"* (motion field).

I *motion field* possono essere parametrizzati in quattro modi diversi:

- Rappresentazione globale
- Rappresentazione basata su pixel
- Rappresentazione basata su regioni
- Rappresentazione basata su blocchi

Il compromesso per stimare il movimento è tramite la rappresentazione basata su blocchi in quanto offre un buon livello di accuratezza e complessità; è chiaro che anche esso ha degli svantaggi.

Di seguito vedremo un esempio di stima del movimento basata su, blocchi, regioni e features.

Criteri di stima del movimento

Il movimento ha necessità di essere stimato perché l'informazione su di esso nella scena può tornare utile in vari modi, ad esempio si può effettuare il Quality enhancement[2] del video tramite stabilizzazione oppure si può aumentare il fattore di compressione del video oppure si può ricostruire la scena 3D a partire dalla proiezione 2D.

Esistono vari criteri per stimare i parametri del modello di movimento.

In questa prima fase soffermandoci sulla rappresentazione del movimento basata su blocchi: parliamo degli ***Algoritmi di Block-Matching (BMA).***

[2] Miglioramento della qualità complessiva di un qualsiasi tipo di contenuto multimediale.

Algoritmo di Block-Matching (BMA)

I blocchi possono avere qualsiasi forma geometrica (solitamente quadrata) purché valga:

$$\bigcup_{m \in M} Bm = \Lambda \text{ e } B_m \cap Bn = \varnothing, m \neq n$$

Assumiamo che tutti i pixel di un blocco B_m si muovano nella stessa direzione (modello traslazionale blockwise): un MV per blocco.

Dato l'*anchor frame* B_m vogliamo trovare il *target frame* B_m' che minimizza l'errore E_{DFD}.

Dunque, si vuole minimizzare l'errore $E_{m(dm)}$:

$$E_{m(dm)} = \sum_{x \in Bm} |I2(x + dm) - I1(x)|p$$

BMA esaustivo (EBMA)

Per trovare il migliore d_m confrontiamo dopo averli calcolati tutti i d_m fra l'*anchor frame* B_m e tutti i possibili *target frame* B_m' all'interno di una regione di ricerca.

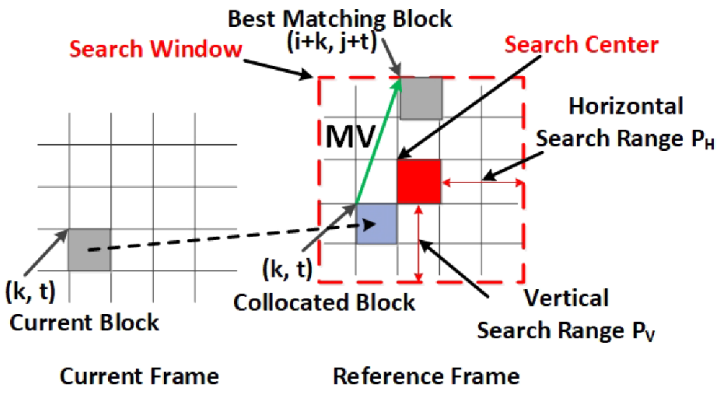

FIGURA 1 – BMA ALGORITHM

Sottrazione di sfondo-movimento

Nel caso di stima del movimento considerando la rappresentazione basata su regioni, si parla di criterio di *sottrazione di sfondo-movimento*. Esso permette di rilevare gli oggetti in movimento rispetto uno sfondo.

Il concetto di base è quello di confrontare ogni frame del video con un modello di sfondo statico per identificare le regioni che sono cambiate rispetto a quest'ultimo. Queste regioni di cambiamento sono poi considerate come oggetti in movimento.

Il processo di sottrazione di sfondo può essere eseguito utilizzando diversi approcci e algoritmi, che possono variare in base alla complessità e alle prestazioni desiderate.

Uno degli algoritmi più comuni utilizzati per la sottrazione di sfondo è l'algoritmo di sottrazione di sfondo basato su modelli misti di Gauss (GMM). Questo algoritmo utilizza un modello statistico per

rappresentare il colore dei pixel dello sfondo e quindi rileva gli oggetti in movimento confrontando ogni pixel del frame corrente con il modello di sfondo.

Altri approcci includono l'uso di metodi di filtraggio morfologico per migliorare la qualità dei risultati, l'integrazione di tecniche di tracciamento dell'oggetto per seguire gli oggetti in movimento attraverso i frame e l'impiego di tecniche avanzate di apprendimento automatico per migliorare l'accuratezza della stima del movimento.

La stima del movimento attraverso la sottrazione di sfondo può essere un processo computazionalmente oneroso specialmente quando si lavora con video ad alta risoluzione o in tempo reale. Tuttavia, sono stati sviluppati metodi efficienti che consentono una rapida elaborazione dei video e una stima accurata del movimento.
un esempio di questo tipo di criterio è il MOG2(Mixture of Gaussian).

MOG2 (Mixture of Gaussians)
MOG2 rappresenta un algoritmo di segmentazione del background ampiamente adottato per individuare le regioni in movimento all'interno di un video.

Questo algoritmo si basa su un modello statistico del background della scena e individua le deviazioni significative da questo modello per rilevare il movimento.

Utilizzando una distribuzione di Gauss per modellare la luminosità dei pixel, MOG2 viene addestrato dinamicamente per adattarsi alle variazioni nel background nel corso del tempo.
Tale approccio consente una segmentazione più accurata anche in presenza di cambiamenti di luce, movimenti della telecamera e altri fattori ambientali.

Grazie alla sua efficacia migliorata rispetto al classico algoritmo MOG, MOG2 è largamente impiegato in applicazioni di sorveglianza, tracciamento oggetti e analisi del movimento.

FIGURA 2 - MOG2 ALGORITHM

Features from Accelerated Segment Test (FAST)

Un altro modo per stimare i parametri del modello di movimento è utilizzando algoritmi basati su features.

Il FAST permette di individuare i punti salienti (corner) nel frame da usare come features per tracciare il movimento.

Si possono individuare due fasi principali dell'algoritmo:

- Si analizza ad ogni passo un insieme di 16 punti con configurazione a cerchio di raggio 3 per classificare se un punto *p* sia o meno di corner
- Data l'intensità I_p del punto *p*, se per almeno 12 punti contigui con intensità I_x si ottiene che $|I_x - I_p| > t$, con *t* soglia, allora *p* sarà considerato corner.

Si può velocizzare il test di *corner detection* con l'***high-speed test***:

Verificare $|I_x - I_p| > t$ per i punti del cerchio in posizione 1 e 9, e successivamente in posizione 5 e 13

- Se per almeno 2 di questi 4 punti il test fallisce e il punto *p* non può essere corner

FAST è computazionalmente efficiente e più veloce di altri metodi (come SIFT[3]), sebbene meno preciso.
Questo lo rende adatto per applicazioni real-time.

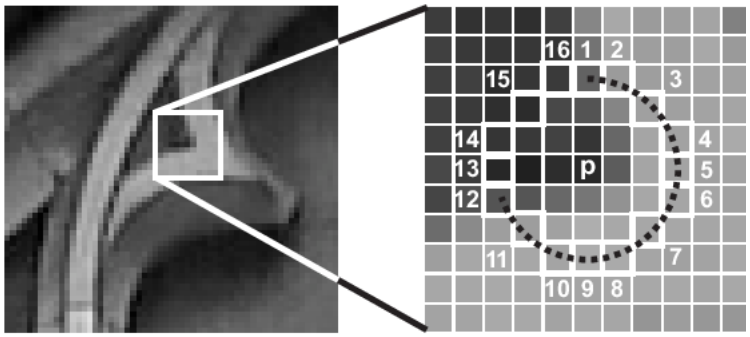

FIGURA 3 - FAST ALGORITHM

[3] Scale-Invariant Feature Transform: algoritmo di Computer Vision usato per rilevare le caratteristiche chiave distinte in un'immagine.

Sistemi di stabilizzazione video

Un sistema di stabilizzazione ha come scopo quello di attenuare i movimenti della camera da una sequenza di immagini.

Si possono distinguere due tipi diversi di movimenti della camera:

- **Padding**: movimenti intenzionali
- **Jitter**: movimenti non intenzionali

In generale, esistono sistemi di stabilizzazione analogici e digitali. Esempi di sistemi di stabilizzazione analogica sono: accelerometri, giroscopi, sensori di velocità angolare e ammortizzatori meccanici.

Per questo progetto interessa attenzionare sistemi di stabilizzazione digitale (DIS).

Sistemi di stabilizzazione digitale

I sistemi di stabilizzazione digitale a sua volta sono divisi in due classi separate:

- DIS real-time
- DIS post-processing

l funzionamento di tali sistemi si basa su tre fasi principali:

1. Stima del movimento
2. Filtraggio (correzione) del movimento
3. Deformazione (post-processing) dell'immagine

Fase 1: Stima del movimento

In questa fase si identifica e traccia il movimento degli oggetti nella sequenza di fotogrammi, permettendo anche di analizzare le dinamiche temporali presenti nei video.

La modalità con cui viene eseguita tale operazione viene enunciata nel capitolo precedente

Fase 2: Filtraggio del movimento

Sono necessari i metodi della Fase 1 per stimare i *motion vector locali* affidabili (LMV).

In questa fase si ottiene il *motion vector globale* (GMV) nel seguente modo:

- Calcolato a partire dai LMV
- Si chiama $v_{act}(n)$ il GMV dell'*anchor frame*.

Nel caso in cui si suppone che la camera debba restare immobile, la stabilizzazione risulta semplice.

Infatti, si possono tracciare uno o più punti e applicare le trasformazioni inverse al GMV in modo che tali punti (e tutti gli altri) vengano riportati sempre nella stessa posizione.

Nel caso di interesse, invece, si vuole riuscire a distinguere i movimenti intenzionali (padding) da quelli casuali (jitter).

Esistono tre tipi diversi di *motion filtering*:

- MVI: Motion Vector Integration
- FPS: Frame Position Smoothing
- Filtro di Kalman

In questa fase attenzioniamo il filtro di Kalman essendo uno strumento molto potente.

Filtro di Kalman

Il filtro di Kalman ha le seguenti caratteristiche:

- Prende in input una serie di misure osservate nel tempo
- Tiene conto di un eventuale rumore casuale
- **Predizione**: fissato un tempo stima la misura
- **Correzione**: osservata una misura può correggerla, eventualmente basandosi su una predizione

La versione discreta di questo filtro si basa su due equazioni differenziali stocastiche in cui lo stato x_k è dato da:

$$x_t = Ax_{t-1} + Bu_t + w_t$$

La misura z_k di x_k è data da:

$$z_t = Hx_t + v_t$$

Dove t rappresenta l'istante temporale; A,B,H sono modelli transazionali; w,v rappresentano il rumore gaussiano e u rappresenta il controllo dell'utente.

La predizione e la correzione si ottengono affiancando a questo filtro un *tracciatore bayesiano* [4] che permette di stimare le probabilità..

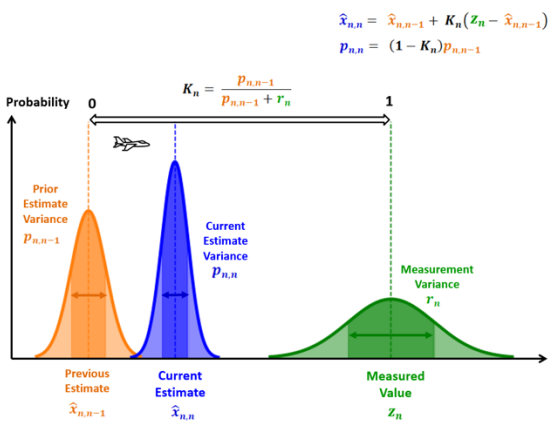

FIGURA 4 - KALMAN FILTER

[4] Algoritmo di tracking che utilizza il teorema di. Bayes per stimare la posizione di un oggetto in un video.

Questi algoritmi possono essere definiti come *standard*. Ad oggi esistono innumerevoli algoritmi basati sul deep learning[5] impiegati nella *computer vision*, i quali risultano computazionalmente più efficienti ed efficaci

Fase 3: Post-processing

Questa fase è necessaria per i frame in quanto il crop dei frame fa perdere informazioni.

Il post processing in generale ha tre scopi ben precisi:

- **Migliorare la qualità dei dati:** rimozione di rumore e artefatti, correzione di distorsioni
- **Estrazione di informazioni:** segmentazione di oggetti, identificazione di caratteristiche
- **Visualizzazione dei risultati:** migliorare la leggibilità e l'interpretabilità dei dati, creare immagini e video.

Dunque si potrebbe pensare di implementare operazioni di filtraggio, smoothing, segmentazioni, e feature extraction.

I vantaggi che ne conseguono sono legati ad una migliore accuratezza e robustezza dei risultati, una maggiore flessibilità e adattabilità a diverse applicazioni e la possibilità di estrarre informazioni più complesse dai dati.

[5] Area dell'Intelligenza Artificiale che lavora sull'addestramento di reti neurali profonde per apprendere modelli complessi di dati.

Metriche di qualità

Una metrica di qualità è un metodo che permette di dare una valutazione qualitativa di un'immagine. È quindi fortemente legata al modello di percezione del sistema visivo umano.

Bisogna distinguere due diverse tipologie di qualità:

- **Qualità assoluta**: essa è un dato oggettivo, completamente descritto dall'immagine e caratterizzabile mediante un insieme di attributi dell'immagine, alcuni semplici, altri composti da sotto-attributi, alcuni misurabili ed altri non misurabili.
- **Qualità percepita**: essa è un parametro soggettivo, dipendente non solo dalle caratteristiche dell'immagine, cioè dai suoi attributi descrittivi, ma anche da altri fattori, ad esempio l'elaborazione effettuata sull'immagine, le caratteristiche tecnologiche del display, la fisiologia e la cognizione della percezione visiva umana.

Dunque, le metriche di qualità assoluta vengono definite *metriche oggettive*, mentre quelle di qualità percepita vengono definite *metriche soggettive*.

Le metriche di qualità possono essere a sua volta classificate in base alla presenza di una maggiore o minore disponibilità del segnale di riferimento, con il quale confrontare l'immagine distorta.

Definiamo quindi altre due tipologie di metriche:

- **Full reference (FR):** metrica di qualità nella quale si ha la completa disponibilità dell'immagine di riferimento e quindi può essere effettuato un confronto completo tra immagine distorta, della quale si vuole conoscere la qualità, e l'immagine di riferimento.
- **No-reference (NR):** metrica di qualità nella quale non si ha alcuna informazione sull'immagine di riferimento. In molte applicazioni video le immagini di riferimento o le sequenze video, spesso non sono accessibili, quindi è desiderabile, per poter gestire tali situazioni, utilizzare metriche che possano valutare la qualità di un'immagine o di un video senza un riferimento.
- **Reduced Reference (RR):** metrica di qualità nella quale l'immagine di riferimento non è interamente disponibile. Quindi si estraggono dall'immagine originale, solo alcune informazioni parziali che vengono in qualche modo elaborate per fornire una valutazione della qualità dell'immagine distorta, presa in esame.

Fra le metriche di qualità oggettiva più comuni si hanno: *PSNR* (Peak Signal to Noise Ratio) e *MSE* (Mean Square Error). Inoltre verrà utilizzato anche l'*OFE* (Optical Flow Error).

Mean square error (MSE)

Questo parametro viene utilizzato per stimare l'errore quadratico medio tra due immagini. Più tale indice è basso minore è la differenza tra le immagini.

In formule si ha:

$$\text{MSE}=\frac{1}{N*M}\sum_{i=1}^{N}\sum_{j=1}^{M}\big(I(i,j)-K(i,j)\big)^2$$

Dove

- N e M sono le dimensioni delle immagini (ad esempio, altezza e larghezza),
- $I(i,j)$ rappresenta il valore del pixel i,j dell'immagine originale,
- $K(i,j)$ rappresenta il valore del pixel i,j dell'immagine compressa o distorta.

Nel contesto di filtraggio del movimento tale parametro permette di misurare la nitidezza e la definizione dei dettagli.

Peak signal to noise ratio (PSNR)

Questo parametro viene utilizzato per misurare la qualità di un'immagine compressa rispetto all'originale. Inoltre dipende dalla differenza tra l'immagine codificata e quella originale; maggiore è il suo valore, maggiore sarà la somiglianza con l'immagine originale.

In formule si ha:

$$PSNR=10\cdot\log_{10}\left(\frac{MAX^2}{MSE}\right)$$

Dove

- MAX rappresenta il valore massimo possibile di un pixel (tipicamente 255 per un'immagine a 8 bit),
- MSE (Mean Square Error) è l'errore quadratico medio tra i pixel dell'immagine originale e quelli dell'immagine compressa o distorta.

Nel contesto di filtraggio del movimento tale parametro valuta il livello di tremore o instabilità dell'immagine nel video. Questa metrica può essere misurata tramite l'analisi del movimento dell'immagine rispetto a un riferimento stabile, come ad esempio il frame precedente.

Structural Similarity Index (SSIM)

I metodi tradizionali PSNR e MSE hanno dimostrato di non essere molto in linea con la percezione dell'occhio umano. Infatti questi approcci stimano gli errori percepiti mentre il SSIM considera la degradazione dell'immagine percepita come cambiamento nelle informazioni strutturali. Il concetto di informazione strutturale nasce dall'idea che i pixel hanno una forte inter-dipendenza soprattutto quando sono spazialmente vicini. Queste dipendenze portano con sé importanti informazioni sulla struttura degli oggetti nella scena visiva.

La Structural SIMilarity (SSIM) misura la similarità tra due immagini (Full-Reference). Si basa sul fatto che il sistema visivo umano è in grado di estrarre *informazioni strutturali* dal campo visivo.

Pertanto, una misurazione della perdita di informazione strutturale può fornire una buona approssimazione alla distorsione della immagine percepita. Sono quindi considerate le informazioni strutturali in un'immagine come quegli attributi che riflettono la struttura degli oggetti sulla scena, indipendenti da luminanza e contrasto medio.

In formule si ha:

$$SSIM(x,y) = \frac{(2\mu_x\mu_y + C_1)(2\sigma_{xy} + C_2)}{(\mu_x^2 + \mu_y^2 + C_1)(\sigma_x^2 + \sigma_y^2 + C_2)}$$

Dove *x,y* sono due segnali non negativi corrispondenti rispettivamente all'immagine campione ed a quella distorta, μ_x è la media di *x*, μ_y è la media di *y*, σ_x^2 la varianza di *x*, σ_y^2 la varianza di *y* e σ_{xy} è la covarianza di *x* e *y*. Inoltre le due costanti C_1 e C_2 servono per i casi limite (divisione per zero) e sono definite come $C_1 = (K_1L)^2$ e $C_2 = (K_2L)^2$, dove L=255 per immagini a 8 bit, e K_1 e K_2 <<1, ovvero sono due costanti molto piccole.

Si ottiene il valore -1 quando ci sono elementi totalmente differenti in termini di frequenza; viceversa, si ottiene 1(indica una migliore qualità dell'immagine).

Considerazioni finali

Per implementare questo progetto si è scelto di effettuare la stima del movimento basata su regioni poiché offre robustezza essendo meno sensibile al rumore e alle variazioni di illuminazione; inoltre, le regioni possono catturare informazioni più globali sul movimento. Si ha anche una migliore discriminabilità in quando le regioni possono essere utilizzate per discriminare tra diversi tipi di movimento come, ad esempio, distinguere il movimento di un oggetto da quello della telecamera. Inoltre le regioni richiedono anche meno dati da elaborare rispetto ad esempio ai pixel. La stima del movimento basata su regioni è anche flessibile e semplice da implementare.

Rispetto alle metriche di qualità delle immagini, è importante valutare le prestazioni di algoritmi e modelli vari utilizzati. È inoltre essenziale considerare la complessità della percezione umana e adoperare una gamma diversificata di metriche per ottenere una valutazione completa.

Un'accurata comprensione e applicazione di tali misurazioni risultano fondamentali per il successo e l'affidabilità di progetti che coinvolgono, in generale, il trattamento di immagini digitali.

Applicazioni pratiche

Il progetto mira a illustrare le diverse fasi dell'analisi e del filtraggio del movimento nei video. A tale scopo, verranno valutate le prestazioni degli algoritmi applicati a campioni di video, i quali rappresentano scenari reali. Questo permetterà di valutare l'applicabilità, in contesti concreti, delle scelte effettuate.

Fasi del Progetto

Al fine di garantire un'analisi esaustiva, il lavoro è stato strutturato in diverse fasi per consentire un'esplorazione approfondita di ciascuna di esse. Questa decisione è stata presa per semplificare il processo e permettere una valutazione completa e dettagliata di ogni aspetto coinvolto.

Stima del movimento

Scelti i video da prendere in esame, il lavoro è iniziato con l'individuazione delle zone salienti all'interno dei vari fotogrammi. Ciò lo si effettua tramite l'impiego del *modello di sottrazione di sfondo* per identificare i pixel in movimento nella regione di interesse chiamata *ROI*.

Inoltre si attenzionano anche i contorni degli oggetti in movimento; infatti, vengono estratti e la loro area viene calcolata.

Vengono conservate solo le aree superiori rispetto al valore di una certa soglia minima chiamata *min_area* per evitare di rilevare rumore e ombre.

In questa fase di stima del movimento viene anche effettuato il *tracking* degli oggetti. Tale fase è un passo fondamentale in molti sistemi di *Computer Vision.* Il suo scopo è quello di monitorare la posizione e lo stato di oggetti di interesse in una sequenza di immagini.

Viene implementato un *tracker* in diverse fasi:

1. **Inizializzazione:** viene creato un oggetto *tracker* di tipo *ObjectTracker*; esso contiene i parametri e le informazioni necessarie per il tracking. Questo tracciamento viene eseguito utilizzando un approccio di *matching di bounding boxes* tra i frame successivi.
   Viene creato un oggetto *object_detector* per rilevare gli oggetti utilizzando il metodo di sottrazione del background MOG2.
2. **Aggiornamento del tracker:** La funzione *tracker.update(detections)* viene chiamata ad ogni frame. Questa funzione riceve in input le nuove detezioni di oggetti (detections) dal modulo di rilevamento. Il tracker associa le nuove detezioni alle tracce esistenti o crea nuove tracce per le detezioni non associate.
3. **Associazione tra tracce e detezioni:** l'associazione viene eseguita utilizzando una implementazione manuale del filtro di Kalman. Esso predice la posizione di ogni traccia nel frame corrente. La distanza tra la posizione predetta e la posizione di una nuova detezione viene utilizzata per determinare se le due sono associate.
4. **Aggiornamento delle tracce:** le tracce associate alle nuove detezioni vengono aggiornate con la nuova posizione e l'ID dell'oggetto. Le tracce che non sono associate a nessuna nuova

detezione vengono considerate perse e vengono eliminate. Questo passaggio in particolare viene implementato nel file chiamato *tracker.py*.

5. **Visualizzazione dei risultati:** le tracce vengono visualizzate sul frame corrente come rettangoli colorati. L'ID dell'oggetto viene visualizzato all'interno di ogni rettangolo.

Inoltre il tracker ha tre parametri non espliciti incorporati implicitamente nell'algoritmo:

1. *distance_threshold*: Soglia di distanza per l'associazione tra tracce e detezioni.
2. *kalman_gain*: Fattore di guadagno del filtro di Kalman.
3. *min_hits*: Numero minimo di frame consecutivi in cui una traccia deve essere associata a una detezione per essere considerata valida.

Il risultato finale è una finestra di visualizzazione in cui viene mostrato il frame originale insieme alla regione di interesse (ROI) dove vengono tracciati gli oggetti e il frame della maschera che rappresenta gli oggetti rilevati.

Queste operazioni si trovano sviluppate all'interno del file chiamato *main.py*.

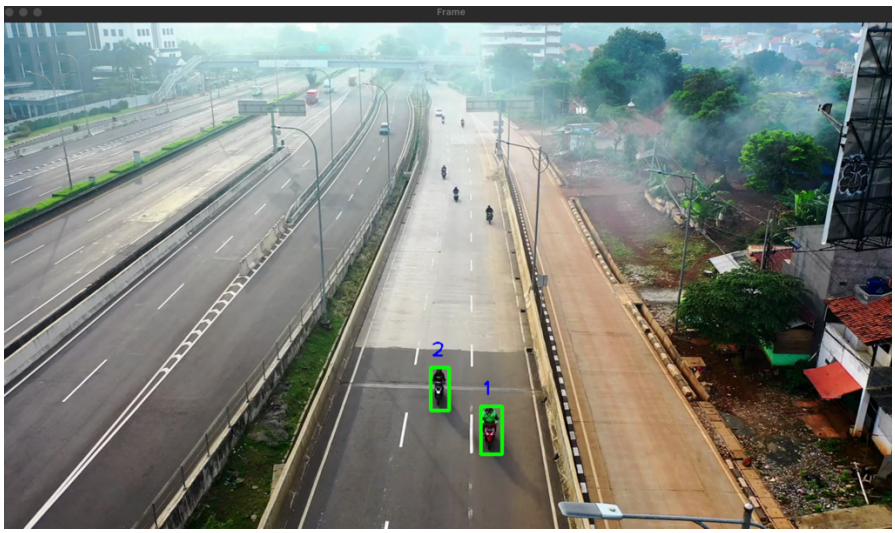

FIGURA 5 - VISUALIZZAZIONE FINESTRA RAPPRESENTANTE FASE STIMA DEL MOVIMENTO

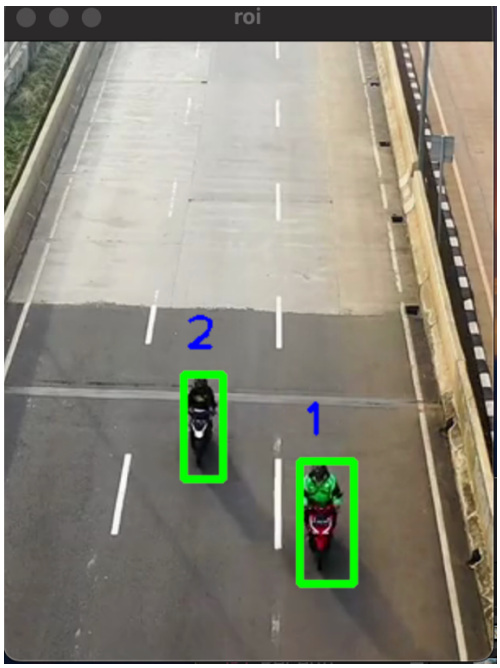

FIGURA 6 - VISUALIZZAZIONE FINESTRA RAPPRESENTANTE "ROI"

All'interno del file chiamato *tracker.py* viene sviluppato un algoritmo che implementa il concetto di stima del movimento, in quanto tiene traccia degli oggetti rilevati da un sistema di visione artificiale durante il tempo, assegnando loro un ID univoco e aggiornando le loro posizioni all'interno di un dizionario chiamato *center_points*. In questo senso, si potrebbe interpretare come una forma di stima del movimento grezza.

Nel dettaglio:

- **Inizializzazione**: viene creato un oggetto *ObjectTracker,* che inizializza due attributi: *center_points*, un dizionario che memorizza le posizioni centrali degli oggetti tracciati insieme ai loro ID, e *id_count*, che tiene traccia del numero di oggetti tracciati.
- **Metodo update**: Questo è il metodo principale per l'aggiornamento del tracciamento degli oggetti. Prende come input i rettangoli (bounding box) che rappresentano gli oggetti rilevati nel frame corrente del video o dell'immagine. Per ciascun rettangolo, calcola il centro (*cx, cy*) e cerca tra gli oggetti tracciati precedentemente se è già presente con posizioni vicine. Se trova un oggetto abbastanza vicino, aggiorna la sua posizione e il suo *bounding box*, altrimenti considera il rettangolo come un nuovo oggetto e gli assegna un nuovo ID. dunque si aggiorna la lista *objects_bbs_ids* con i bounding box degli oggetti e i relativi ID.
- **Metodo _cleanup_unused_objects**: Questo metodo rimuove gli oggetti tracciati che non sono più presenti nel frame corrente. Trova gli ID degli oggetti utilizzati nel frame corrente e rimuove quelli non utilizzati dal dizionario *center_points.*
  Infine, la funzione restituisce la lista aggiornata di bounding box degli oggetti con i relativi ID.

Quindi, questo algoritmo può essere considerato una stima del movimento degli oggetti, poiché tiene traccia delle loro posizioni tra frame successivi in un video, in particolare dei rispettivi bounding box degli oggetti rilevati, i quali cambiano posizione nel tempo.

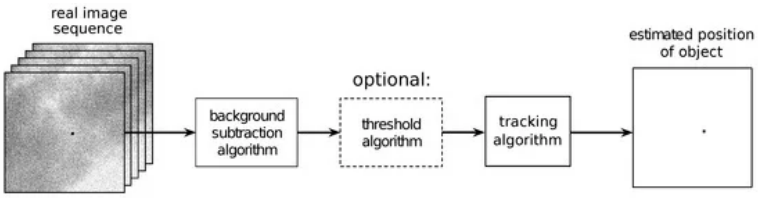

FIGURA 7 – STIMA DEL MOVIMENTO

Filtraggio del movimento

Dopo aver stimato i parametri del movimento si passa alla fase di filtraggio (correzione) del movimento presente nel video.

Sostanzialmente vengono eliminate le aree con dimensioni inferiori ad una soglia chiamata *min_area* durante il rilevamento degli oggetti. Dunque si sottrae lo sfondo e si isolano gli oggetti in movimento, che aiuta a rimuovere il rumore e le ombre dalla maschera di movimento. Questa operazione viene effettuata grazie al metodo *cv2.createBackgroundSubstractorMOG2* definita nel file *main.py*.

In questa fase si implementa la funzione *object_detector.apply(roi)* che restituisce una maschera binaria con i pixel in movimento.

Dopo la sottrazione dello sfondo, la funzione *cv2.findContours(mask, ...)* estrae i contorni degli oggetti in movimento dalla maschera.

L'area di ciascun contorno viene calcolata e confrontata con il parametro *min_area* e quelle di valore inferiore ad esso vengono eliminate.

Il risultato finale è una finestra di visualizzazione in cui vengono mostrati, tramite binarizzazione, i contorni rilevati.

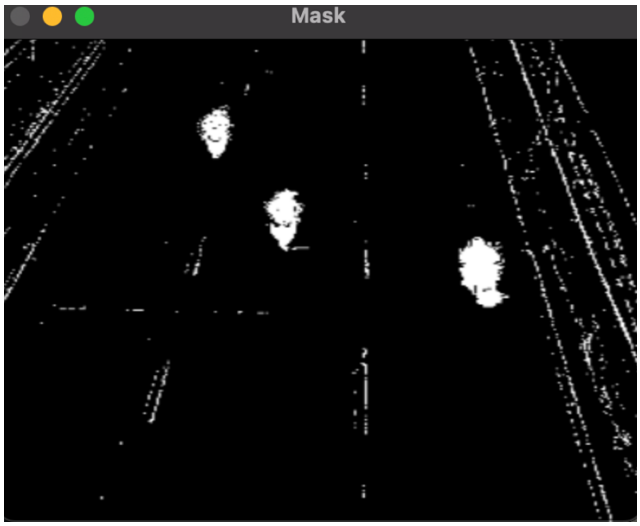

FIGURA 8 - VISUALIZZAZIONE FINESTRA RAPPRESENTANTE FASE FILTRAGGIO DEL MOVIMENTO

Deformazione post-processing

Dopo aver stimato e filtrato il movimento, si passa alla fase di post-processing. Tale passaggio è fondamentale nel contesto di trattamento di immagini e video dopo essere stati acquisiti e processati inizialmente. Questa fase è dedicata a migliorare, ottimizzare o modificare l'aspetto dei video utilizzati, in base agli obiettivi specifici dell'applicazione. La post-elaborazione può coinvolgere una serie di tecniche e operazioni che vengono applicate ai dati visivi grezzi per ottenere risultati desiderati.

In questo caso specifico è stato scelto di implementare:

- **Sfocatura** tramite il metodo *apply_blur*: questa funzione applica un effetto di sfocatura al frame tramite Gauss. La sfocatura può essere utile per ridurre il rumore nell'immagine o per creare effetti estetici.
- **Correzione del colore** tramite il metodo *correct_color*: questa funzione corregge eventuali problemi di colore nel frame. Può includere la correzione del bilanciamento del bianco, la saturazione del colore, o altri aggiustamenti per migliorare la resa cromatica dell'immagine.
- **Regolazione della luminosità e del contrasto** tramite il metodo *adjust_brightness_contrast*: questa funzione regola la luminosità e il contrasto del frame. Può essere utilizzata per migliorare la visibilità dell'immagine o per adattarla a specifiche esigenze di visualizzazione.
- **Applicazione di maschere** tramite il metodo *apply_mask*: questa funzione applica una maschera al frame. Le maschere vengono utilizzate per nascondere o evidenziare determinate parti dell'immagine. Nel contesto del tracciamento degli oggetti, è utile per concentrare l'attenzione su determinate regioni o per escludere parti non desiderate dell'immagine.

Inoltre, per rendere più efficiente il codice è stata implementato un *meccanismo di debugging* che consente di visualizzare i risultati di alcune delle operazioni di post-processing per il frame corrente. Questo consente di controllare se le singole operazioni di post-processing producono i risultati desiderati. È importante notare che l'efficacia delle operazioni di post-processing dipende dalle specifiche esigenze dell'applicazione e dalla natura dei dati di input.

Queste operazioni sono state effettuate in un file chiamato *post_processing.py*.

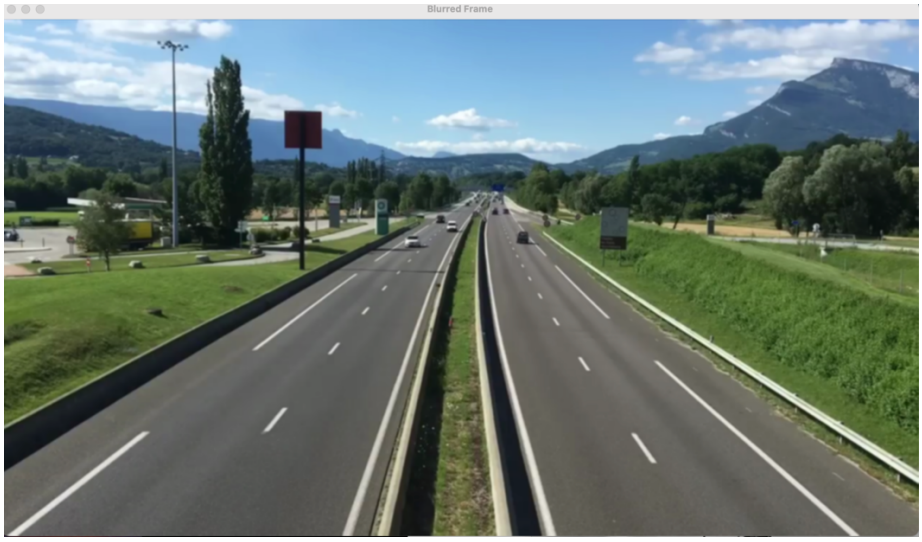

FIGURA 9 - VISUALIZZAZIONE FINESTRA FILTRO GAUSSIAN BLUR

FIGURA 10 - VISUALIZZAZIONE FINESTRA FILTRO CORRECT COLOR

FIGURA 11 - VISUALIZZAZIONE FINESTRA FILTRO ADJUST BRIGHTNESS CONTRAST

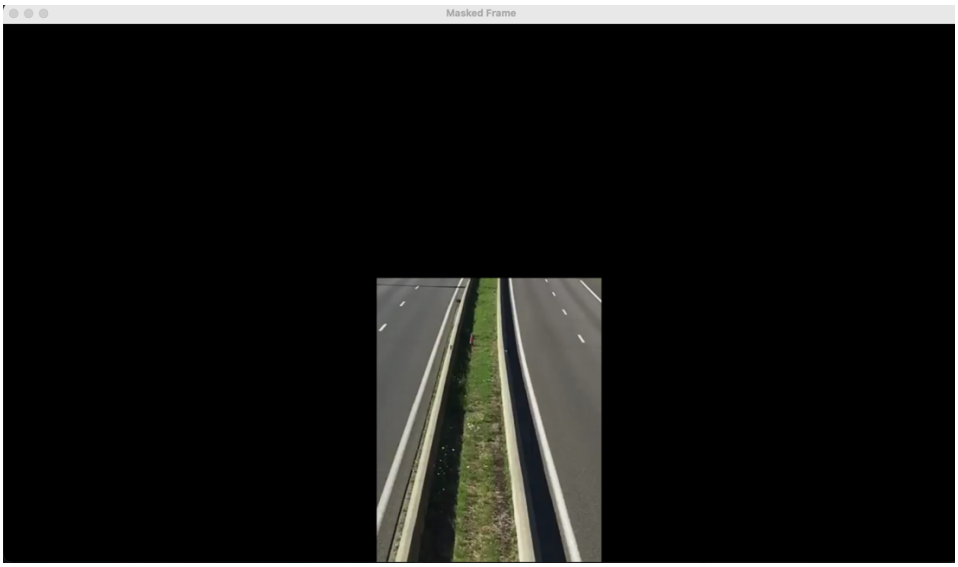

FIGURA 12 - VISUALIZZAZIONE FINESTRA FILTRO APPLY MASK

FIGURA 13 - VISUALIZZAZIONE FINESTRA ORIGINAL FRAME

Valutazione tramite metriche di qualità

Come ultima fase di lavoro si è scelto di implementare alcune delle più importanti metriche di qualità per valutare i risultati ottenuti dal filtraggio dei video presi in esame.

È stato creato un file chiamato *metrics.py* in cui sono state implementate le metriche: PSNR, MSE e SSIM.

All'inizio viene implementato un metodo chiamato *calculate_video_metrics* in cui vengono dichiarate le variabili che permetteranno di ottenere in output i valori delle metriche applicate ad ogni coppia di frame. È stato inserito, inoltre, un controllo per evitare che il codice ritorni un errore in assenza di frame.

```python
import numpy as np
from skimage.metrics import structural_similarity as ssim

def calculate_video_metrics(frames):
    total_psnr = 0
    total_mse = 0
    total_ssim = 0
    frame_count = len(frames)

    # Verifica che ci siano almeno due frame per calcolare le medie
    if frame_count < 2:
        print("Non ci sono abbastanza frame per calcolare le metriche.")
        return None, None, None
```

FIGURA 14 - CODICE DICHIARAZIONE FUNZIONE CALCULATE METRICS

Successivamente sono state calcolate le due metriche PSNR e MSE mostrate in figura 15:

```python
# Calcolo PSNR, MSE tra frame adiacenti
for i in range(1, frame_count - 1):
    mse = np.mean((frames[i + 1] - frames[i]) ** 2)
    if mse == 0:
        psnr = float("inf")
    else:
        max_pixel = 255.0
        psnr = 20 * np.log10(max_pixel / np.sqrt(mse))
    total_mse += mse
    total_psnr += psnr
```

FIGURA 15 - CODICE METRICHE PSNR E MSE

È stata implementata anche la metrica SSIM visibile in figura 16:

```python
# Calcolo del SSIM tra due frame adiacenti (conversione in scala di grigi)
ssim_score, _ = ssim(cv2.cvtColor(frames[i + 1], cv2.COLOR_BGR2GRAY),
                     cv2.cvtColor(frames[i], cv2.COLOR_BGR2GRAY),
                     full=True)
total_ssim += ssim_score
```

FIGURA 16 - VISUALIZZAZIONE METRICA SSIM

Infine, per ogni metrica implementata, è stata calcolata la media dei valori su tutti i frame escluso il primo, in quanto non possiede un frame precedente con cui effettuare il confronto.

```
#calcola la media dei valori per ogni metrica su tutti i frame escluso il primo
avg_mse = total_mse / (frame_count - 1)
avg_psnr = total_psnr / (frame_count - 1)
avg_ssim = total_ssim / (frame_count - 1)

return avg_mse, avg_psnr, avg_ssim
```

FIGURA 17 - CODICE MEDIA DEI VALORI MSE, PSNR E SSIM

Risultati ottenuti

Analisi ottenuta dai video originali senza fare alcun tipo di operazione.

Video utilizzati	Media dei valori PSNR dei frame	Media dei valori MSE dei frame	Media dei valori SSIM dei frame
auto_2	40.796	6.218	0.960
auto_4	44.905	2.629	0.983
auto_5	42.530	3.984	0.985
auto_6	39.956	7.073	0.954

Questi dati sono stati ottenuti dalla compilazione del codice chiamato *metriche_video_originale.py*.

In generale, i valori di PSNR più alti indicano una migliore qualità dell'immagine.
I valori di MSE e SSIM più bassi indicano anche una migliore qualità dell'immagine.

- **PSNR:**
    - **auto_2:** un valore di PSNR di 40,796 dB indica una buona qualità dell'immagine, con un livello di rumore relativamente basso.
    - **auto_4:** un valore di PSNR di 44,905 dB indica un'eccellente qualità dell'immagine, con un livello di rumore quasi impercettibile.
    - **auto_5:** un valore di PSNR di 42,530 dB indica una buona qualità dell'immagine, con un livello di rumore leggermente più alto rispetto ad **auto_4**.
    - **auto_6:** un valore di PSNR di 39,956 dB indica una qualità dell'immagine accettabile, con un livello di rumore più evidente.
- **MSE:**
    - **auto_2:** un valore di MSE di 6,218 indica un livello di distorsione moderato tra l'immagine originale e l'immagine acquisita.
    - **auto_4:** un valore di MSE di 2,629 indica un livello di distorsione molto basso, con una fedeltà molto alta all'immagine originale.
    - **auto_5:** un valore di MSE di 3,984 indica un livello di distorsione leggermente più alto rispetto a **auto_4**, ma comunque accettabile.
    - **auto_6:** un valore di MSE di 7,073 indica un livello di distorsione più evidente, con una minore fedeltà all'immagine originale.

- **SSIM:**
    - **auto_2:** un valore di SSIM di 0,960 indica una forte somiglianza tra l'immagine originale e l'immagine acquisita, con una buona conservazione delle informazioni strutturali.
    - **auto_4:** un valore di SSIM di 0,983 indica un'altissima somiglianza tra le due immagini, con una fedeltà quasi perfetta all'originale.
    - **auto_5:** un valore di SSIM di 0,985 indica un'eccellente somiglianza, con una minima perdita di informazioni strutturali.
    - **auto_6:** un valore di SSIM di 0,954 indica una somiglianza ancora buona, ma con una distorsione più evidente rispetto agli altri video.

In questo caso, i video *auto_4* e *auto_5* hanno i valori di PSNR più alti, il che indica che hanno la migliore qualità del video. Il video *auto_6* ha il valore di PSNR più basso, il che indica che ha la peggiore qualità del video.

Analisi video ottenuta dopo aver effettuato il filtraggio del movimento nei video e operazioni di post-processing.

Il calcolo delle metriche viene effettuato nell'intero frame e non solo nella regione di interesse (ROI).

Inoltre è stato tenuto in considerazione il fatto che le metriche lavorano su frame adiacenti (frame i e il successore i+1) perché, se pur con approcci differenti, cercano di stimare l'errore percepito. In altre parole, l'immagine distorta viene considerata come il cambiamento percepito nella informazione strutturale. L'informazione strutturale è l'idea che i pixel hanno una forte interdipendenza specialmente quando sono spazialmente vicini. Questa dipendenza trasporta importanti informazioni riguardo la struttura degli oggetti che compongono la scena visuale.

Video utilizzati	Operazioni effettuate	Filtri applicati post-processing	Media dei valori PSNR dei frame	Media dei valori MSE dei frame	Media dei valori SSIM dei frame
auto_2	• Stima del movimento • Filtraggio del movimento	• Blurring • correct_color • adjust contrast • apply_mask	44.264	4.519	0.964
auto_4	• Stima del movimento • Filtraggio del movimento	• Blurring • correct_color • adjust contrast • apply mask	43.529	3.773	0.974
auto_5	• Stima del movimento • Filtraggio del movimento	• Blurring • correct_color • adjust contrast • apply mask	42.049	4.319	0.977
auto_6	• Stima del movimento • Filtraggio del movimento	• Blurring • correct_color • adjust contrast • apply mask	39.782	10.090	0.920

Nel caso in cui nel calcolo del PSNR il valore di MSE risulta 0 allora PSNR ritorna un valore pari a inf (non c'è alcuna perdita di informazione).

In generale, i valori di PSNR, MSE e SSIM sono migliorati dopo il filtraggio del movimento e post-processing. Il post-processing può avere anche un impatto negativo sulla qualità dei video, in termini di PSNR, MSE e SSIM. Tuttavia, l'entità di questo impatto varia a seconda del video. L'*auto_6* è la più sensibile al post-processing, mentre l'*auto_4* è la meno sensibile.

- **PSNR:**
    - **auto_2:** un aumento di 3,481 dB indica un miglioramento significativo della qualità dell'immagine, con una riduzione del rumore e una migliore definizione dei dettagli.
    - **auto_4:** un aumento di 1,376 dB indica un miglioramento moderato, con un'ulteriore riduzione del rumore già basso presente nell'originale.
    - **auto_5:** un aumento di 0,481 dB indica un miglioramento minimo, con una qualità dell'immagine già buona in partenza.
    - **auto_6:** un aumento di 0,174 dB indica un miglioramento marginale, con la qualità dell'immagine che rimane comunque la più bassa tra i video.
- **MSE:**
    - **auto_2:** una diminuzione di 1,699 indica una riduzione significativa della distorsione, con un'immagine più fedele all'originale.
    - **auto_4:** una diminuzione di 1,144 indica una riduzione moderata della distorsione, con un ulteriore perfezionamento dell'immagine.
    - **auto_5:** una diminuzione di 0,335 indica una riduzione minima della distorsione, con un'immagine già di buona qualità.
    - **auto_6:** una diminuzione di 0,335 indica una riduzione marginale della distorsione, con la distorsione ancora evidente.
- **SSIM:**
    - **auto_2:** un aumento di 0,004 indica un leggero miglioramento della somiglianza tra le due immagini, con una migliore conservazione dei dettagli.
    - **auto_4:** un aumento di 0,009 indica un miglioramento minimo.
    - **auto_5:** un aumento di 0,008 indica un miglioramento minimo.
    - **auto_6:** un aumento di 0,034 indica un miglioramento molto minimo.

Conclusioni

In base all'analisi approfondita dei risultati, possiamo trarre le seguenti conclusioni:

Qualità dei video originali**:**

- I video **auto_4** e **auto_5** hanno la migliore qualità dell'immagine in termini di PSNR, MSE e SSIM.
- Il video **auto_6** ha la qualità dell'immagine più bassa.

Effetti del filtraggio del movimento e post-processing:

- Il filtraggio del movimento e il post-processing non hanno migliorato la qualità di tutti i video.
- Il miglioramento più significativo è stato ottenuto per il video **auto_2**, con un aumento di PSNR di 3,481 dB e una diminuzione di MSE di 1,699.
- Il miglioramento minimo è stato ottenuto per il video **auto_6**, con un aumento di PSNR di 0,174 dB e una diminuzione di MSE di 0,335.

L'analisi condotta ha evidenziato come il filtraggio del movimento e il post-processing, in generale, rappresentino due tecniche efficaci per il miglioramento della qualità dei video. L'entità del suddetto miglioramento si dimostra direttamente proporzionale alla qualità del video originale.

In ottica di un'applicazione specifica, la scelta del video da utilizzare dipenderà dalle specifiche esigenze dell'utente. Qualora si ricerchi la massima qualità dell'immagine, i video **auto_4** e **auto_5** si configurano come la scelta ottimale. In alternativa, se si necessita di un buon compromesso tra qualità e complessità computazionale, il video **auto_2** rappresenta un'opzione valida. Infine, il video **auto_6** dovrebbe essere utilizzato solo nel caso in cui la qualità dell'immagine non rivesta un ruolo critico.

Raccomandazioni

Si consiglia di eseguire ulteriori test con diverse impostazioni di filtraggio del movimento e post-processing per ottimizzare i risultati per l'applicazione specifica.
È importante valutare la qualità dei video non solo in base a metriche quantitative come PSNR, MSE e SSIM, ma anche con un'analisi visiva soggettiva.

Rispetto al codice, esso è un esempio semplice e potrebbe essere migliorato con tecniche di filtraggio più avanzate, basti pensare ai nuovi algoritmi sviluppati nel contesto della Computer Vision, più specificatamente nel settore del *Machine and Deep Learning*[6], in grado di lavorare con reti neurali convoluzionali per stimare il movimento o per addestrare dei set di dati prestabiliti.

Rispetto questo codice si possono delineare punti di forza quali:

- Struttura: il codice è ben strutturato e organizzato.
- Funzioni: sono ben definite e documentate.
- Efficienza: il codice risulta efficiente e computazionalmente poco costoso.
- Modularità: il codice è facilmente estensibile.

Il codice è un buon punto di partenza per un sistema di rilevamento e tracciamento di oggetti in video. Tuttavia, sono necessari alcuni miglioramenti, pocanzi esplicitati, per renderlo più completo e robusto.

Analisi predittiva ed estrazione di informazioni dai dati: un punto di partenza

Da considerare è il "dizionario" chiamato *objects_bbs_ids* definito all'interno del codice *tracker.py*. Esso è una struttura dati che memorizza i *bounding box* degli oggetti e i relativi ID. I bounding box vengono aggiornati mentre la posizione dell'oggetto varia nel tempo. Questo è utile in quanto si controlla se l'oggetto è già stato rilevato precedentemente (il counter ignora l'oggetto) o se è nuovo, in questo caso gli si assegna un nuovo ID (il counter viene incrementato di una unità); nel caso in cui nel dizionario sono presenti oggetti non utilizzati essi vengono eliminati. Queste operazioni sono implementate nel file *tracker.py* e possono essere considerate come un punto di partenza per lo sviluppo di un modello predittivo che lavori sui dati e che prenda delle decisioni.

Inoltre, l'analisi sul movimento delle automobili che attraversano le strade prese in esame dai video, come anche il conteggio di esse, ci permette di sviluppare sistemi che possano, ad esempio, analizzare e prevedere l'impatto del traffico sull'ambiente, oppure ottimizzare i percorsi e la velocità dei veicoli riducendo i tempi di percorrenza e i costi di carburante. Permette ancora di monitorare l'uso delle macchine dal punto di vista industriale rispetto la produzione e la logistica. Può aiutare a identificare i modelli di comportamento pericolosi e a prevedere il rischio di incidenti; quindi, in ambito urbano può migliorare la sicurezza dei pedoni e dei ciclisti.

[6] Sottocampi dell'Intelligenza Artificiale che permettono ai computer di apprendere informazioni sui dati ed a prendere decisioni.