

# Sistema di object tracking di veicoli

Pagnini Lorenzo - 0000942265  
{lorenzo.pagnini2@studio.unibo.it}

Ottobre 2021

# Indice

<b>1</b>	<b>Introduzione</b>	<b>2</b>
<b>2</b>	<b>Progetto</b>	<b>3</b>
2.1	Struttura del progetto . . . . .	3
2.2	Eseguire il programma . . . . .	3
<b>3</b>	<b>Tracking</b>	<b>5</b>
3.1	Calibrazione . . . . .	5
3.2	Pre-processing . . . . .	5
3.3	Optical Flow . . . . .	6
3.4	Operazioni Morfologiche . . . . .	6
3.5	Algoritmo di tracking basato sulle distanze minime dei centroidi	7
3.5.1	Valori di soglia considerati . . . . .	9
3.6	Parametri di tracking stimati . . . . .	10
3.6.1	Velocità . . . . .	11
3.6.2	Direzione del movimento . . . . .	12
3.7	Principali problematiche . . . . .	12
<b>4</b>	<b>Conclusioni</b>	<b>14</b>

# Capitolo 1

## Introduzione

L'obiettivo di questo elaborato è la realizzazione di un sistema software per il tracciamento dei veicoli utilizzando l'optical flow (*flusso ottico*). Il tracking, utilizza una telecamera per il riconoscimento di uno o più oggetti in movimento (in questo caso veicoli). Un algoritmo, analizza i fotogrammi (o frames) del video e in output produce la posizione (bounding box e relativo centroide) degli oggetti individuati. Come accennato, il tracking si basa sul flusso ottico che permette di identificare quali pixel cambiano di intensità nel tempo e quindi tra i vari frames, riuscendo ad identificare il movimento di più oggetti.

L'obiettivo principale è quello di identificare e tracciare i veicoli in moto e stimare alcuni parametri di essi come la velocità e la direzione di movimento. Come spiegato in seguito, l'algoritmo di tracking implementato, si basa sulle distanze minime, calcolate sulla base dei centroidi delle varie bounding box, tra i frames.

La telecamera presa in considerazione mostra un incrocio di *Cambridge (Market Central, MA)* consultabile al seguente *link*.

## Capitolo 2

# Progetto

### 2.1 Struttura del progetto

In questo capitolo viene brevemente illustrata la struttura del progetto. Il codice è possibile consultarlo al seguente *repository* di GitHub.

Il progetto è organizzato nelle seguenti directory:

- **Calibration**: contiene il codice per effettuare la calibrazione della camera;
- **Common**: contiene il codice condiviso da più classi all'interno del progetto;
- **MotionTracking**: è la directory più importante.

In essa è contenuta la classe *Vehicle* che rappresenta l'oggetto da tracciare per questo contesto. La classe *Table* mostra i parametri stimati, relativi al tracciamento, nella tabella. La classe *Motion* implementa l'algoritmo di tracking ed effettua il calcolo dei parametri stimati. Sono inoltre contenute tutte le funzioni di utilità (*Utility.py*);

- **OpticalFlow**: contiene il codice per realizzare ed ottenere il flusso ottico;
- File **main.py** è il file con cui far partire il programma.

**NB:** nel repository fornito non è presente la directory video poiché i file video sono troppo pesanti da caricare.

### 2.2 Eseguire il programma

Il progetto può essere eseguito dal file *main.py*.

Una volta avviato, verranno mostrate le seguenti finestre:

- View principale che mostra, a sinistra, il video con i veicoli che verranno tracciati e a destra, il video originale;
- Maschere ottenute dall'applicazione dell'optical flow. A destra è presente la maschera codificata secondo il modello colore HSV e a sinistra, la stessa maschera binarizzata a cui sono state applicate precedentemente le tecniche morfologiche;
- Tabella che mostra i parametri stimati.

All'interno del *main* possono essere settati i seguenti parametri:

- *video\_url*: è una stringa ed indica l'URL del video di youtube o il suo percorso locale della città specificata. Per cambiare il path, aprire il file *url.py* nella directory *Common* e modificare la voce *path* della città desiderata;
- *excluded\_area*: è un booleano. Se si vuole considerare solo la porzione di area di interesse per il tracciamento (creata di default per le varie città disponibili) settarlo a *true* altrimenti impostarlo a *false*;
- *show\_log*: è un booleano. Impostarlo a *true* se si vuole mostrare i log durante l'esecuzione del progetto altrimenti impostarlo a *false*.

A questo punto, per avviare il video, e quindi il tracking, è necessario cliccare il pulsante *space* (barra spaziatrice) della tastiera.

Al seguente link è possibile scaricare i video delle città disponibili: *link*. Il video è stato registrato con un ritardo di circa 22 millisecondi per ogni frame.

**NB:** si consiglia di utilizzare un video locale, poichè se si imposta un URL che corrisponde ad un *link* di youtube, il tracking potrebbe presentarsi a scatti a causa del buffering nello scaricare i pacchetti del video della libreria *pafy*.

## Capitolo 3

# Tracking

In questo capitolo vengono descritti tutti i processi coinvolti nella fase di sviluppo per ottenere l'identificazione dei veicoli, il loro tracciamento e la stima dei parametri.

### 3.1 Calibrazione

La calibrazione della camera è quell'operazione necessaria per eliminare la distorsione provocata dalla lente del sensore. In questo modo le linee curve appaiono rette perciò è importante applicarla al fine di migliorare la precisione degli oggetti individuati, il tracciamento o qualunque altra operazione eseguita sui singoli frame. Tale operazione è di fondamentale importanza poiché permette di fare la differenza in un sistema di *computer vision*.

Poiché impossibilitato ad applicare la calibrazione della telecamera prima dell'acquisizione, l'operazione è stata fatta in un secondo momento direttamente sulle immagini. I coefficienti di distorsione sono stati calcolati utilizzando una scacchiera quadrata di dimensione  $9 \times 6$  acquisendo 30 punti con un errore pari a  $0.0224$ . Per ogni scatto acquisito, la scacchiera è stata posta in diverse posizioni (verticale, orizzontale e diagonale) e a diverse distanze dalla telecamera.

### 3.2 Pre-processing

Prima di determinare il flusso ottico su ogni frame, viene applicato un filtro gaussiano con dimensione del kernel  $9 \times 9$  e sigma pari a 4 per smussare l'immagine. L'applicazione di un filtro permette di rimuovere del rumore presente e far emergere le informazioni salienti rispetto agli oggetti da tracciare.

### 3.3 Optical Flow

L'optical flow è la tecnica utilizzata per individuare gli oggetti nella scena e tracciarli successivamente. Si basa sull'analisi delle intensità dei singoli pixel di ogni frame considerando tre presupposti:

1. Le intensità dei pixel di un oggetto non cambiano tra frames consecutivi;
2. Lo spostamento (inteso come movimento dell'oggetto) deve essere piccolo;
3. Coerenza spaziale: i pixel vicini hanno un movimento simile;

Quindi le intensità di ogni pixel dell'immagine di coordinate  $(x, y)$  al tempo  $t$  vengono confrontate con le intensità del frame successivo in direzione  $(dx, dy)$  al tempo  $dt$ . Se tali intensità sono uguali significa che non c'è movimento altrimenti si avrà una variazione d'intensità.

Il flusso ottico è rappresentato da un campo vettoriale-2D in cui ogni vettore è un vettore spostamento che mostra il movimento dei punti tra frames consecutivi.

In questo elaborato si è utilizzato l'optical flow denso implementando l'algoritmo di *Gunner Farneback*. Quest'ultimo permette di ricavare le features dense ovvero calcolate su tutti i pixel del frame. Sebbene questo metodo è più lento, permette di ottenere un risultato più accurato, riuscendo a stimare la velocità e direzione degli oggetti che invece non era possibile ricavare con l'algoritmo di *Lucas-Kanade* poiché determina un insieme di features sparse. L'algoritmo utilizzato restituisce un array-2D da cui viene ricavata la direzione e la magnitudo. Attraverso il modello di colore HSV, il movimento può essere codificato per ottenere una maschera su cui poter estrarre delle features. Secondo tale modello, la direzione (angolo del flusso) è rappresentata dalla tonalità (componente H - *Hue*), la magnitudo (distanza del flusso) è rappresentata dal valore (componente V - *Value*) mentre la saturazione (componente S - *Saturation*) è stata impostata al valore massimo per ottenere una migliore visualizzazione della maschera. Quindi solo le componenti H e V dipendono dal movimento apparente.

La figura 3.1, mostra un esempio di una maschera ottenuta dall'applicazione dell'optical flow. Come si può notare, dove sono presenti delle zone colorate, vi è una variazione d'intensità dei pixel con presenza di movimento degli oggetti.

### 3.4 Operazioni Morfologiche

Una volta ottenuta la maschera, questa viene binarizzata e successivamente vengono applicate le operazioni morfologiche. In particolare nel progetto è stata applicata un'apertura che realizza un'erosione seguita da una dilatazione. Il

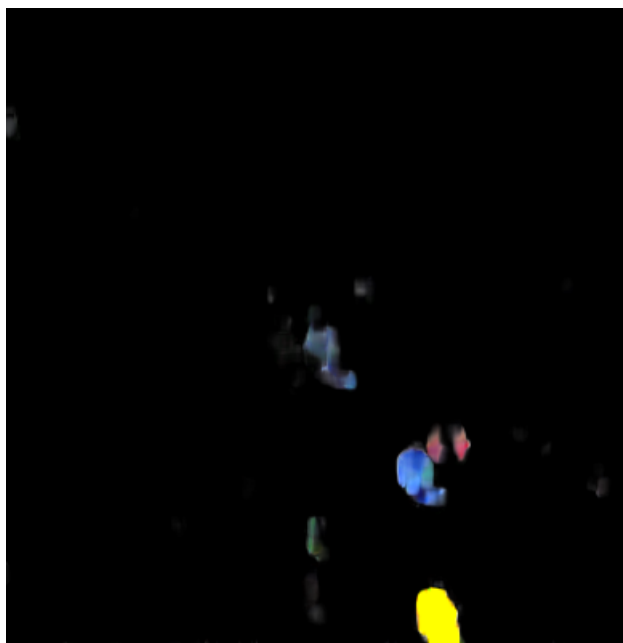


Figura 3.1: Esempio del flusso ottico su una maschera codificata secondo il modello HSV.

risultato complessivo è quello di separare le componenti debolmente connesse, rimuovere piccole regioni facendo emergere componenti (poligoni) di grandi dimensioni.

### 3.5 Algoritmo di tracking basato sulle distanze minime dei centroidi

Questa fase ha come scopo quello di identificare i vari oggetti presenti nella scena. Viene quindi descritto brevemente l'algoritmo di tracking (nella pagina seguente è possibile consultare lo pseudo-codice).

In primo luogo, vengono individuati i contorni dei poligoni, la loro area e il numero di lati. Poligoni molto piccoli o con un numero di lati inferiori a quattro vengono scartati poichè non corrispondono alle caratteristiche aventi da un veicolo. A questo punto, per ogni frame, viene eseguito uno dei due *step*:

- si ha questa fase, se siamo alla prima iterazione (inizio del video) o se nel frame precedente non sono stati individuati dei veicoli. Se nel frame corrente sono presenti poligoni di interesse, questi vengono considerati come nuovi veicoli e vengono memorizzate le seguenti informazioni nella



---

**Algorithm 1** Vehicles tracking

---

```
1: for frame in all frames do
2:   if prev_vehicles = 0 then
3:     Start of the video or there aren't vehicles in the previous frame
4:     for cnt in all contours do
5:       Check area and sides to discard the area
6:       if if it is not discarded then
7:         Check cnt if was previously tracked
8:         if if it has not been tracked then
9:           Create a new vehicle
10:        end if
11:      end if
12:      Update table Qt and draw vehicles in img
13:    end for
14:  else
15:    There are vehicles in the previous frame
16:    for cnt in all contours do
17:      Check area and sides to discard the area
18:      if if it is not discarded then
19:        Find the minimum distance among all contours
20:        if There is a distance then
21:          if Check if vehicle has left the scene then
22:            Remove the vehicle
23:          end if
24:          if Check if vehicles is stationary then
25:            Add vehicle as stationary
26:            Update parameters
27:          end if
28:          if distace < max_distance then
29:            Update parameters
30:          end if
31:        else
32:          Check cnt if was previously tracked
33:          if if it has not been tracked then
34:            Create a new vehicle
35:          end if
36:        end if
37:      end if
38:      Update table Qt e draw vehicles in img
39:    end for
40:  end if
41:  Update vehicles lists
42: end for
```

---

classe *Vehicle*: numero (incrementato in ordine crescente per ogni nuovo veicolo identificato), le 4 coordinate della bounding box, le coordinate del centroide, la velocità (che in questo caso è pari a 0 km/h poiché non si è ancora tracciato il veicolo su più frame consecutivi), se è stazionario e la sua direzione di movimento. Dopo aver identificato tutti i poligoni si passa all'iterazione successiva.

Ad ogni nuovo veicolo creato gli verrà associato un colore casuale. Per ridurre l'ambiguità, non è possibile associare a più macchine presenti nella stessa scena lo stesso colore.

- si ha questa fase se nel frame precedente sono stati individuati dei veicoli. L'obiettivo ora, è quello di far 'matchare' gli oggetti del frame corrente con quelli del frame precedente ovvero riconoscere lo stesso veicolo. Per fare ciò, viene calcolata la distanza di ogni nuovo poligono (a partire dal suo centroide), con tutti i centroidi dei veicoli individuati precedentemente. Se è presente, viene considerato il veicolo con la distanza minima tra tutti i veicoli, altrimenti sarà considerato come nuovo veicolo.

Se la distanza minima è prossima allo zero il veicolo viene etichettato come stazionario (fermo) e vengono aggiornati i parametri sulla coordinate e sulla la velocità.

Se il veicolo è vicino ai *margini* dell'immagine verrà considerato come fuori dalla scena e quindi non più tracciato.

Per gestire veicoli stazionari e possibili occlusioni sono stati definiti un numero di frame massimi prima che l'algoritmo elimini il veicolo dalle liste di gestione degli oggetti tracciati. Questo significa che se un veicolo rimane stazionario o si ripresenta dopo un'occlusione per un numero di frame minore o uguale rispetto al numero di frame massimo allora l'algoritmo sarà ancora in grado di identificarlo con il suo numero altrimenti verrà etichettato come nuovo veicolo.

La figura 3.2 mostra un esempio di tracking.

### 3.5.1 Valori di soglia considerati

I valori di soglia considerati nel progetto sono:

- i poligoni (ricavati dall'applicazione delle operazioni morfologiche) per essere accettati, quindi considerati come veicoli, devono avere un'area maggiore o uguale di 40 pixel e un numero di lati maggiore o uguale a 4;
- la massima distanza (ottenuta dalle differenze dei centroidi) per eseguire il *matching* di un veicolo tra frame consecutivi è impostata a 30 pixel. Superata questa distanza si considera come nuovo veicolo;

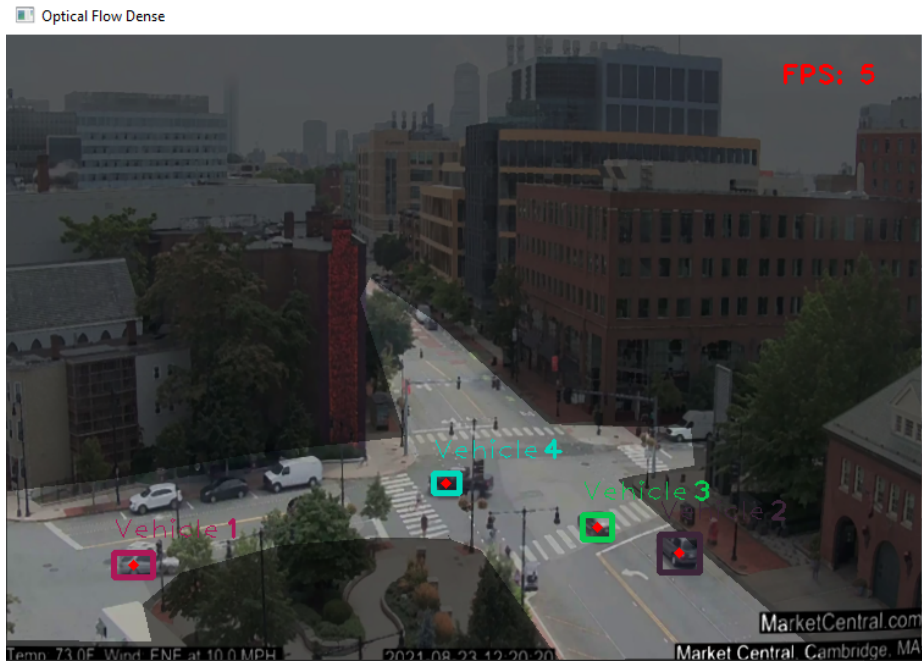


Figura 3.2: Esempio di tracking

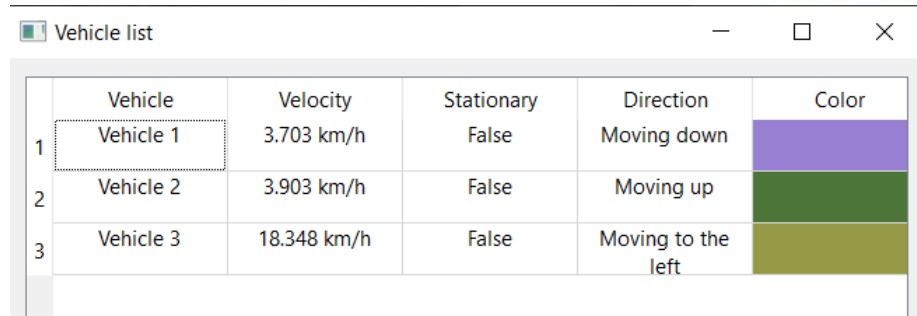
- la massima distanza per marcare un veicolo come stazionario è pari 1.5 pixel;
- vettori dell'optical flow con magnitudo inferiori a 10, vengono scartati per il calcolo della direzione di movimento dei veicoli;
- il numero di frames massimo per cui un veicolo stazionario (quindi non in movimento) continua ad essere tracciato è impostato a 25 frames. Se dopo 25 frames il veicolo continua ad essere ancora stazionario non verrà più tracciato. Questa scelta è dettata dal fatto di evitare di ridurre le performance dell'algoritmo;
- il numero di frames massimo per cui un veicolo, che non si è più in grado di tracciarlo ma che non è uscito dalla scena (caso tipico di un'occlusione) è impostato a 25 frames. Anche in questo caso la scelta è dettata dalle performance dell'algoritmo di tracking.

### 3.6 Parametri di tracking stimati

In questa sezione vengono descritti i parametri stimati ottenuti dal tracciamento degli oggetti.

Per una migliore visualizzazione ho deciso di realizzare una tabella con la libreria *Qt* in cui sono presenti le seguenti colonne: nome e numero del veicolo, velocità, direzione, se è stazionario e il suo colore.

Inoltre, avviando il progetto, viene proposta la maschera HSV e la relativa maschera binaria utilizzate dall'algoritmo di tracciamento.



	Vehicle	Velocity	Stationary	Direction	Color
1	Vehicle 1	3.703 km/h	False	Moving down	Blue
2	Vehicle 2	3.903 km/h	False	Moving up	Red
3	Vehicle 3	18.348 km/h	False	Moving to the left	Green

Figura 3.3: Esempio della tabella di tracking.

### 3.6.1 Velocità

Poiché non è stato possibile misurare le distanze reali della scena (ad esempio lunghezza e larghezza della carreggiata) né avere alcun tipo di informazione della telecamera (a quale altezza è stata installata, parametri ottici, ...) si è deciso di calcolare la velocità secondo la seguente formula:

$$velocità = fps * distance * m2km * m2px$$

in cui:

- *fps*: frame per secondi correnti;
- *distance*: distanza in pixel del veicolo calcolata come differenza dei suoi centroidi tra il frame corrente e quello precedente;
- *m2km*: fattore di conversione dei m/s in km/h pari a 3.6;
- *m2px*: fattore di conversione della distanza da pixel in metri pari a 0.075.

Naturalmente questa metodologia risulta essere meno accurata rispetto all'utilizzo delle distanze reali infatti non si sono avuti grandi risultati sul calcolo di questo parametro dato anche dalla prospettiva del video (si veda l'analisi degli errori: paragrafo 3.7).

### 3.6.2 Direzione del movimento

La direzione dei veicoli è calcolata a partire dall'angolo e dalla magnitudo del flusso. Le direzioni individuate dagli oggetti sono: in alto, in basso, a sinistra e a destra.

Per facilitare il calcolo della direzione, gli angoli e la magnitudo sono stati convertiti in gradi ed equivalgono a:

- $0^\circ$  verso sinistra;
- $180^\circ$  verso destra;
- $90^\circ$  verso il basso;
- $270^\circ$  verso l'alto;

Per ogni direzione è stato definito un range di valori (con un intervallo di circa  $70^\circ$ ) e rappresentano gli angoli che discriminano tale direzione. Tali range sono stati rimodulati in base alla prospettiva del video. La magnitudo è utilizzata per rimuovere alcuni rumori di movimento (intesi come piccole variazioni d'intensità dei pixel). A questo punto, per ogni veicolo si considera la direzione predominante (eseguendo una media delle direzioni individuate nella porzione di interesse del frame). Le direzioni possono essere consultate nella tabella descritta.

## 3.7 Principali problematiche

Le problematiche principali riscontrate nel tracking dei veicoli sono:

- **Occlusioni.** Nel caso di occlusioni parziali (ad esempio lampioni artificiali) il tracking è abbastanza robusto. In caso di occlusioni totali (ad esempio alberi) l'algoritmo non riesce più ad identificare il movimento e quindi il veicolo che stava tracciando. Una possibile soluzione è l'utilizzo del filtro di *Kalman*, robusto alle occlusioni. Tale filtro permette di stimare lo stato di un sistema dinamico a partire da una serie di osservazioni, specificando un modello matematico che descrive il movimento degli oggetti. Sono state fatti alcuni tentativi ma la sua implementazione non è riuscita, forse a causa del fatto di non essere riuscito nello specificare correttamente il modello matematico rispetto al problema considerato;
- **Oggetti multipli.** Se i veicoli sono molto vicini, il sistema con molta probabilità li identificherà come un unico oggetto. Per risolvere parzialmente il problema si è agito sulle operazioni morfologiche al fine di ottenere tante componenti connesse quanti sono i veicoli presenti nella scena. Anche in questo caso il problema potrebbe essere ridotto utilizzando il filtro di Kalman;

- **Prospettiva.** la webcam scelta per la stima della velocità dei veicoli non è perfettamente adeguata poiché vi è una prospettiva accentuata tale per cui i veicoli in primo piano (quelli in basso a destra) a parità di distanza percorsa, si muovono più velocemente rispetto a veicoli in secondo piano (e quindi in lontananza) che si muoveranno più lentamente. Il calcolo della stima della velocità non tiene in conto della prospettiva, risultando che i veicoli in primo piano avranno una velocità maggiore rispetto a quelli in secondo piano che appaiono quasi fermi.

## Capitolo 4

# Conclusioni

Al termine dell'elaborato, posso sostenere di non essere pienamente soddisfatto del lavoro ottenuto. Sapevo già dall'inizio che la complessità, per la tipologia del *task* scelto, non era facile. Sono comunque riuscito ad ottenere un discreto algoritmo di tracking per i veicoli. Alcune criticità riscontrate sono date dall'occlusione degli oggetti, causando la perdita del tracking del veicolo o la 'frammentazione' del veicolo in più di uno. L'occlusione rappresenta uno dei problemi più difficili da risolvere per quanto riguarda la *computer vision*. Un'altra criticità rilevata è data dal fatto che l'algoritmo non riesce a distinguere con elevata precisione le singole istanze dei veicoli, soprattutto quando questi si trovano molto vicini l'uno all'altro. Per quanto riguarda i parametri stimati, la velocità risulta essere poco accurata mentre la direzione del movimento dei veicoli risulta essere ottima.

Nonostante tutto ciò, la realizzazione di questo elaborato mi ha permesso di imparare e approfondire nuove tecniche legate alla visione artificiale, essendo la prima volta che mi approcciavo ad un task di *tracking*.