UNIVERSITÀ DEGLI STUDI DI SALERNO

Dipartimento di Ingegneria dell'Informazione ed

Elettrica e Matematica Applicata

Progetto Autonomous Vehicle Driving

Luglio 2021

Gruppo 02

Angellotto Daniele	**0622701082**
Barbella Michele	**0622701341**
Cavalcante Marco	**0622701209**
Vigliotti Vincenzo	**0622701206**

ANNO ACCADEMICO 2020-2021

Sommario

1. Executive Summary

1.1 Descrizione preliminare sintetica del progetto

In questo progetto di Autonomous Vehicle Driving si è utilizzato un detector per il riconoscimento dei semafori e sono stati implementati controlli basati sulle informazioni perfette fornite dall'ambiente relative a pedoni e veicoli. In base a tali informazioni ricavate si è cercato di ottenere un veicolo a guida autonoma con ODD limitato alla mappa Town01 capace evitare collisioni con altri veicoli e pedoni presenti all'interno della stessa, restando ad una adeguata distanza da essi, ed in grado di rilevare e rispettare i semafori.

1.2 Assunzioni fatte

- Il semaforo si trova sempre nella porzione destra dell'immagine acquisita dalla camera installata sul veicolo.
- I semafori si trovano sempre su vie parallele all'asse delle x o delle y del mondo, per costruzione della mappa Town01.
- Si considerano come Lead Vehicle solo quelli che hanno un angolo compreso tra +/- 45° rispetto all'orientamento del veicolo e che rientrano all'interno della distanza di lookahead.
- Quando troviamo un veicolo in movimento o fermo ad un semaforo davanti a noi, lo consideriamo un Lead Vehicle e ci poniamo dietro di esso, non cerchiamo di effettuare sorpassi.
- Quando un pedone sta attraversando la strada il nostro veicolo si ferma per permettere l'attraversamento senza cercare di aggirarlo.
- Il sistema viene eseguito con qualità "Epic", poiché con questa stessa risoluzione è stato addestrato il detector.

1.3 Problemi affrontati

- Gestione del detector e di errate detection e/o miss detection con riferimento alla risoluzione finale scelta.
- Bounding Box generata dal detector poco precisa.
- Pedoni che si fermano in mezzo alla carreggiata durante l'attraversamento.
- Gestione combinata di pedoni, semafori e Lead Vehicle.

2. Progettazione ed implementazione

Tutte le modifiche apportate sulla baseline fornitaci per il progetto iniziano e finiscono rispettivamente con i tag **#EditGroup2** e **#EndEditGroup2**. Di seguito descriviamo nel dettaglio le parti di codice che abbiamo aggiunto per realizzare il nostro progetto.

2.1 Sensori utilizzati

Sul veicolo sono stati installati una camera RGB (riga 246-254 main.py) ed una camera di profondità (riga 256-265 main.py). Questi vengono utilizzati per la rilevazione dei semafori e per il calcolo della distanza a cui fermarci. In particolare, entrambi i sensori catturano un'immagine di risoluzione 800x800 e hanno una FOV (field of view) pari a 90, con posizione [x, y, z] = [1.8, 0, 1.3].

2.2 Detector di semafori

Il detector integrato all'interno del nostro progetto (inizializzato a riga 841-845 main.py e integrato nella classe *TrafficLightDetection* in *traffic_light_detection.py*) è stato allenato in un primo momento su 5800 immagini del dataset "LISA traffic light detection" ed in seguito su un insieme di circa 1800 immagini estratte dal simulatore Carla in modalità EPIC. L'addestramento è iniziato con i pesi pre-allenati YOLOv2 COCO.

Questo detector è in grado di classificare soltanto due classi:

- Colore verde e giallo codificati con "go"
- Colore rosso codificato con "stop"

Nella classe TrafficLightDetection (file traffic_light_detection.py) sono implementate due funzioni:

detect: (righe 102-162 traffic_light_detection.py) effettua la detection dei semafori sull'immagine in input.
Questa funzione prende in ingresso l'immagine catturata dalla camera RGB, convertita in BGRA, verifica se la predizione supera una determinata soglia (**SCORE_THRESHOLD = 0.20**, si noti che è stata presa in considerazione tale soglia in quanto si è potuto notare che lo score risulta molto variabile nella detection dei semafori) ed in caso affermativo si controlla se questa bounding box appartiene ad un semaforo visto per la prima volta oppure uno già rilevato precedentemente.
Nel primo caso vengono inizializzati i contatori che si riferiscono alla classe semaforo verde o rosso, nel secondo caso si controlla se la bounding box si riferisce allo stesso oggetto tra due frame consecutivi in quanto la detection di due bounding box troppo distanti tra loro indica una errata classificazione. Se ciò è verificato viene incrementato il contatore che si riferisce all'attuale classe del semaforo, altrimenti viene incrementato il contatore relativo alle missdetection.
Altro caso preso in considerazione è quello in cui lo score predetto dal detector sia minore della soglia scelta oppure non siano stati rilevati semafori, nonostante in frame precedenti fosse avvenuta una detection. Se si presentano queste condizioni, si ha una missdetection e viene incrementato di conseguenza il contatore della stessa.
Questa funzione restituisce tre variabili: *traffic_light_detected, boxes, is_green.*

- Traffic_light_detected: indica il corretto rilevamento di un semaforo.
- Boxes: contiene quattro valori relativi ai vertici della bounding box
- Is_green: indica il rilevamento di un semaforo verde o il verificarsi di un determinato numero di missdetection (**_missdetection_repetition = 3**).

Il corretto rilevamento di un semaforo si ha quando si raggiungono NUM_SEMAPHORE_CHECK (settato a 21) detections della stessa classe. Si è deciso di prendere in considerazione tale soglia per minimizzare il più possibile i casi in cui venga considerata una errata detection come un semaforo.
Se sono stati rilevati un numero di frame sufficienti per affermare di essere in presenza di un semaforo rosso o un semaforo verde si resettano tutti i contatori.
Se sono stati rilevati un numero di frame sufficienti per affermare che non si è in presenza di un semaforo azzeriamo i contatori relativi alle missdetection ed alle due classi del semaforo e viene incrementato *self._missdetection_repetition*. Successivamente, tramite questa variabile, si effettua un controllo per evitare di rimanere bloccati ad un semaforo nel caso in cui non riuscissimo più a rilevarne uno con la nostra camera.
È stato quindi necessario settare *is_green* a True anche nel caso in cui si hanno un certo numero di missdetection in quanto sono stati riscontrati casi in cui un pedone che attraversa viene rilevato dal detector come un semaforo rosso per un elevato numero di frame. Se non fosse stato previsto questo controllo, il nostro veicolo, non vedendo un semaforo verde, sarebbe rimasto fermo indefinitamente dopo il passaggio del pedone. Infine, si è deciso di ripartire in caso di miss detection se e solo se viene raggiunto un numero di miss detection pari a NUM_SEMAPHORE_CHECK per tre volte consecutive. Ciò è stato deciso in quanto si è potuto verificare sperimentalmente che il detector diventa particolarmente fallace quando all'interno dell'immagine su cui viene effettuata la detection sono presenti dei pedoni.

***get_traffic_light_fences**:* (righe 165-272 traffic_light_detection.py) permette di calcolare una linea (individuata da quattro punti) perpendicolare alla yaw del veicolo e ad una distanza di sicurezza rispetto al semaforo.
Questa funzione prende in ingresso l'immagine catturata dalla camera Depth, la posizione e l'orientamento del veicolo. Dopo le opportune conversioni che permettono di trasformare i punti dei pixel dell'immagine in coordinate del mondo siamo in grado di calcolare la distanza che separa in centro del veicolo dal semaforo rilevato (a partire da riga 202). Dal momento che il semaforo si trova alla nostra destra, le misure della depth, che sono considerate rispetto alla posizione della camera, richiedono una proiezione della depth del semaforo sull'asse centrale dell'immagine.

Si noti che in alcuni casi la fence del semaforo viene tracciata diversi metri prima rispetto ad esso. Questo si verifica in particolare quando ci sono oggetti in prospettiva molto vicini al semaforo, ma spazialmente più vicini al veicolo. Questo è dovuto al fatto che, quando si calcola la depth del semaforo rispetto al veicolo, viene presa in considerazione una bounding box allargata. Tale scelta progettuale è stata implementata (righe 176-181 in traffic_light_detection.py) per sopperire ai casi in cui essa venga disegnata completamente al di fuori del semaforo. Ciò portava a considerare una depth molto alta relativa ad elementi dello sfondo (si noti che questa casistica era molto comune). All'interno della bounding box allargata, si prende quindi il punto con la minima depth rispetto al veicolo. Quindi è chiaro che, se all'interno della bounding box vi è un palo, si potrebbe prendere come depth quella del palo piuttosto che quella del semaforo, tracciando quindi una fence all'altezza di esso. In ogni caso, si è preferita questa soluzione in quanto un tracciamento della fence vari metri prima del semaforo permette comunque un corretto rispetto di esso.

2.3 Gestione degli ostacoli

Le informazioni sugli ostacoli sono prese dal simulatore e pertanto non risultano essere affette da rumore. Per mezzo di posizione, orientamento e velocità degli agenti è possibile prevedere la posizione futura di ciascun ostacolo dinamico a meno di fermate improvvise.

2.3.1 Collisioni con pedoni

La prima parte della gestione dei pedoni avviene nel file *main.py*. Dopo aver prelevato le informazioni relative ai pedoni provenienti dal simulatore (riga 984 main.py), si effettua un prefiltraggio andando a compiere i controlli solo su quelli che si trovano nelle vicinanze del veicolo; in particolare si prendono in considerazione quelli la cui distanza euclidea tra essi ed il veicolo è al di sotto di una certa threshold data dalla somma della lunghezza della lookahead e della fence (riga 992 main.py), al fine di garantire una adeguata distanza di frenata. Inoltre, vengono esclusi dai controlli i pedoni che iniziano ad attraversare ma si fermano lateralmente al veicolo (riga 993 main.py), in quanto questi non possono essere investiti, ma, includendoli, il veicolo resterebbe bloccato al centro della corsia per un tempo non definito. Successivamente si calcolano le bounding box in coordinate mondo per ogni pedone non escluso (riga 1009 main.py).
Per quanto riguarda il controllo delle collisioni con i pedoni si fa riferimento alla funzione *pedestrian_collision_check* (riga 104 di *collision_checker.py*). In questa funzione avviene un secondo filtraggio dei pedoni riutilizzando la logica della funzione *collision_check* che era già fornita dalla baseline. Tale logica però è stata modificata al fine di ricavare i pedoni che intersecano un nostro path locale, invece di ricavare i path che non collidono con i pedoni. Successivamente si procede ad iterare su tutti i pedoni che sono in collisione con i nostri path locali. Per ogni pedone verrà creata una fence (lunga 6 metri in avanti e 3 metri dietro rispetto al centro del pedone) e si controlla se questa interseca con il best path del veicolo. In tal caso si deduce che il pedone sta effettivamente attraversando la strada. Di conseguenza si calcola il punto in cui potrebbe avvenire la collisione, passando tale informazione al behavioral planner che provvede ad impostare il goal state in modo che il veicolo possa fermarsi prima del pedone e non investirlo.

Di seguito si descrivono i possibili casi che si possono verificare quando il veicolo circola per le strade di Town01:

Pedoni con traiettoria che interseca il veicolo: una volta trovata un'intersezione con un pedone, si salva il goal state settato in seguito alla stessa (l'ultimo elemento di path_with_obstacle) e nei frame successivi si calcolano i path locali relativi a questo goal state salvato, oltre che al goal state presente nel behavioral planner (righe 1049-1053 main.py). Inoltre, viene anche salvato il best path sul quale è stata individuata la collisione, provvedendo nei frame successivi ad escludere i punti già superati dal veicolo (righe 1090-1094 main.py). I path locali relativi al goal state salvato e il best path salvato vengono successivamente utilizzati solo per il controllo delle collisioni con i pedoni.

Pedoni che attraversano ma si fermano al centro della nostra corsia: simile al caso precedente, si calcola il goal state entro cui fermarci per permettere l'attraversamento.

Pedoni con traiettoria che non interseca il veicolo: se non ci sono pedoni lungo il nostro percorso al frame precedente, si controlla se al frame corrente ci siano pedoni che intersecano il nostro best path attuale (righe 1079-1082 main.py).

Pedoni che iniziano ad attraversare ma si fermano lateralmente al veicolo: esclusi dai controlli come già descritto precedentemente.

Nei primi due casi è stato previsto il salvataggio del goal state e del best path su cui si era individuata la collisione. Questa scelta progettuale è stata fatta perché, al fine di fermarci correttamente, viene settato come goal state nel behavioral planner il punto di fermata desiderato. Il goal state settato nel behavioural planner comportava il calcolo dei path relativi ad un punto spazialmente più vicino (quindi più corto) rispetto al pedone, che di conseguenza non avrebbero previsto l'intersezione con lo stesso, causando continui cambi di stato da Follow Lane a Decelerate_to_Stop e viceversa.

2.3.2 Collisioni con veicoli

Anche in questo caso prendiamo le informazioni relative ai veicoli provenienti dal simulatore (righe 977-982 main.py). Queste informazioni vengono poi passate alla funzione *check_for_lead_vehicle* (riga 1038 main.py) che è definita nel behavioral planner (riga 361 behavioral_planner.py). In questa funzione si controlla che per ogni veicolo la distanza sia minore della variabile *_follow_lead_vehicle_lookahead* (data da LEAD_VEHICLE_LOOKAHEAD_BASE + (current_speed * 3.6 /10)^2, dove LEAD_VEHICLE_LOOKAHEAD_BASE è pari a 10 m) e che il veicolo viaggi nella nostra stessa direzione. Tra i veicoli che verificano queste condizioni viene scelto quello a minima distanza come lead vehicle. In seguito, queste informazioni sono passate alla funzione *compute_velocity_profile* (riga 1110 main.py) che provvede ad impostare la nostra velocità come il minimo tra la velocità desiderata e quella del lead vehicle.

2.4 Descrizione dell'architettura del hierarchical planner

In generale lo hierarchical planner in un veicolo a guida autonoma si compone di tre moduli:

- Mission planner
- Behavioral planner
- Local planner

Nell'ambito del nostro progetto è stata mantenuta questa struttura. In particolare, viene utilizzato il mission planner di Carla (come da baseline) per il calcolo dei waypoint che il veicolo deve seguire per andare da un punto A ad un punto B. Il progetto si è concentrato sullo sviluppo del behavioral planner e del local planner.

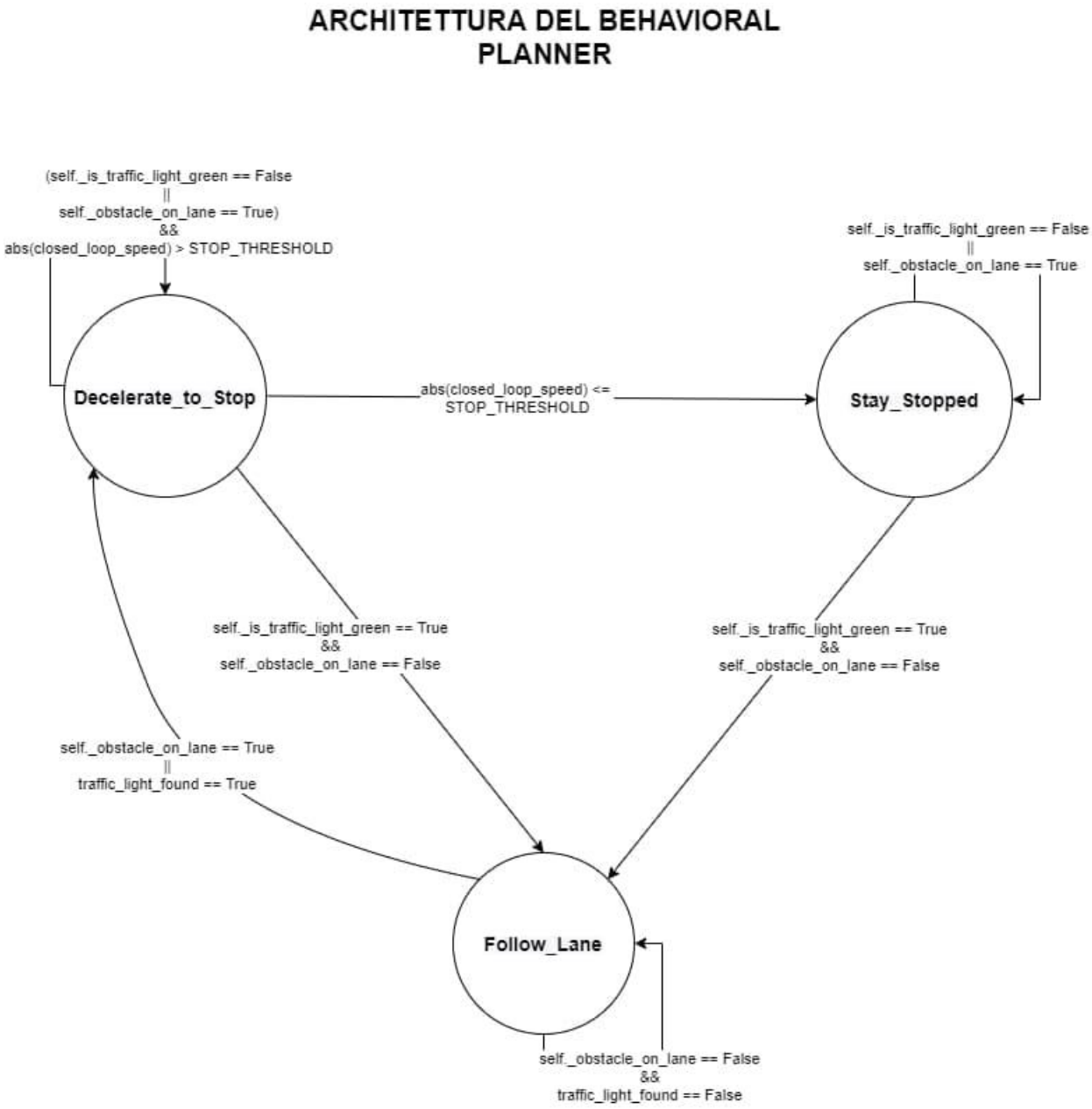
Il main funge da centro di smistamento delle informazioni: preleva le informazioni provenienti dal planner di Carla per passarle al behavioral planner, dal behavioral planner riceve informazioni sul comportamento che il veicolo deve assumere (cioè se deve mantenere una certa velocità oppure iniziare a decelerare oppure rimanere fermo); tali informazioni vengono poi passate dal main al local planner per effettuare il check di eventuali collisioni con veicoli e pedoni presenti nella scena e per calcolare i giusti profili di velocità e accelerazione che il veicolo deve seguire per rispettare il comportamento assegnato.

Di seguito vengono descritti i moduli implementati, con particolare riferimento al Behavioral planner.

2.4.1 Behavioral Planner

Il Behavioral Planner proposto è una macchina a stati finiti costituita di tre stati: **Follow_Lane**, **Decelerate_to_Stop** e **Stay_Stopped**. Di seguito viene descritto in dettaglio il funzionamento dei tre stati e le condizioni che ci permettono di passare da uno stato all'altro, come anche mostrato nella figura.

Indipendentemente dallo stato in cui si trova, il veicolo è anche capace di gestire eventuali lead vehicle come descritto in precedenza.



Nota: Follow Lead Vehicle non prevede cambi di stato

2.4.2 Descrizione di dettaglio di ogni stato

FOLLOW_LANE

Nello stato di Follow_Lane il veicolo si muove seguendo i waypoint calcolati dal planner di Carla.

Dallo stato di Follow_Lane possiamo passare nello stato di Decelerate_to_Stop; ci sono diverse condizioni che portano a questo passaggio di stato:

- C'è un semaforo rosso e non ci sono pedoni sulla lane; in questo caso modifichiamo il goal state per fermarci prima del semaforo (in particolare il nostro obiettivo è fermarci a circa 2 metri dalla linea tracciata per il semaforo, questa distanza viene settata dalla costante DISTANCE_FROM_TRAFFIC_LIGHT in behavioral_planner.py)
- Non c'è un semaforo rosso, ma c'è un pedone sulla lane; in questo caso modifichiamo il goal state per fermarci prima del pedone e non investirlo
- C'è sia un semaforo rosso sia un pedone sulla lane; in questo caso controlliamo quale dei due è più vicino al nostro veicolo e modifichiamo il goal state per fermarci prima del pedone/semaforo

DECELERATE_TO_STOP

Nello stato di Decelerate_to_Stop il veicolo sta rallentando allo scopo di fermarsi. Il veicolo decelera nel caso in cui sia stato rilevato un semaforo rosso o se è stato rilevato un pedone che sta attraversando la strada.

Dallo stato di Decelerate_to_Stop possiamo passare allo stato di Stay_Stopped nel caso in cui la nostra velocità scenda al di sotto di una certa soglia (STOP_THRESHOLD).

Dallo stato di Decelerate_to_Stop possiamo inoltre passare allo stato di Follow_Lane, se si verifica la seguente condizione:

- Non ci sono pedoni sulla lane e non c'è un semaforo rosso in vista (questo vuol dire che c'è un semaforo verde se siamo fermi ad un incrocio, oppure che vediamo affatto un semaforo e quindi possiamo ripartire)

STAY_STOPPED

Quando entriamo nello stato di Stay_Stopped la nostra velocità è scesa al di sotto di una certa soglia (nel codice viene definita come STOP_THRESHOLD a riga 11 di behavioral_planner.py, con un valore pari a 0.02). Quindi la velocità non è esattamente 0, ma è davvero bassa quindi possiamo ritenere il veicolo fermo.

Dallo stato di Stay_Stopped possiamo passare nello stato di Follow_Lane se si verifica la seguente condizione:

- Non ci sono pedoni sulla lane e non c'è un semaforo rosso in vista (questo vuol dire che c'è un semaforo verde se siamo fermi ad un incrocio, oppure che vediamo affatto un semaforo e quindi possiamo ripartire)

2.4.3 Local Planner

PATH GENERATOR

È stato mantenuto il path generator presente nella baseline del progetto

VELOCITY PLANNER
In questo modulo abbiamo apportato una modifica che permette di controllare se nello stato di Decelerate_to_Stop oltre ad un lead_vehicle ci sono anche uno o più pedoni, in tal caso tramite la funzione min_distance viene settato il punto dove fermarci per evitare incidenti (righe 121-130 velocity_planner.py).

COLLISION CHECKER

Questo modulo è stato modificato aggiungendo ad esso una funzione per la gestione delle collisioni con i pedoni, pedestrian_collision_check, già spiegata approfonditamente nel paragrafo 2.3.1.

3. Analisi sperimentale del sistema

3.1 Analisi quantitative dei risultati

Il sistema è stato testato con diversi punti di partenza e arrivo, e di conseguenza su diversi percorsi della mappa Town01 fornita. Nei test effettuati è stato inserito un numero massimo di pedoni e veicoli uguale o inferiore a **450** e **150** rispettivamente. Inoltre, in ogni fase di testing, sono stati testati più percorsi al fine di controllare il corretto funzionamento in ogni direzione possibile, con ogni combinazione possibile di curva.

Il numero di test effettuati sulla versione finale del progetto è stato di **42** test. Tra questi, **28** sono stati portati a termine con successo, senza alcun comportamento errato. In quelli rimanenti ci sono stati degli errati comportamenti del veicolo.

Tuttavia, non tutti i comportamenti errati hanno portato a situazioni critiche o fatali, quali l'incidente con un altro veicolo o l'investire un pedone. In particolare, **3** test sono stati invalidati a causa di problemi del simulatore (ad esempio incidente tra due veicoli di Carla che crea un ingorgo nel simulatore). **6** test invece hanno riportato dei comportamenti errati, ma che sostanzialmente possono essere considerati come errori non gravi, in quanto non portano a situazioni critiche. Nei **5** test rimanenti si sono riscontrati errori fatali. Si noti che 2 casistiche che portavano ad errori gravi o non gravi sono state risolte.

Riportiamo di seguito il link ad un foglio Excel dove sono elencati tutti i test da noi effettuati, con i dati inseriti nel simulatore (punto di partenza e arrivo, numero di pedoni e veicoli, seed di pedoni e veicoli), i giorni in cui sono stati effettuati questi test e i problemi riscontrati per quelli che non sono andati a buon fine.

Link Excel: https://docs.google.com/spreadsheets/d/1JL7ccKuVICHYktr0lUIeT7mVv-LI-wmfJgGCF4Vud4k/edit?usp=sharing

3.2 Analisi qualitative dei risultati

I test sono stati effettuati in diverse fasi dello sviluppo del progetto. In particolare, lo sviluppo è stato diviso in diverse fasi, ognuna delle quali aveva l'obiettivo di aggiungere un grado di complessità al sistema proposto.

Inizialmente è stato integrato il detector per i semafori e sono stati effettuati dei test senza veicoli e senza pedoni allo scopo di testare il corretto funzionamento dei semafori, cioè il corretto rilevamento del semaforo e dello stato dello stesso (sia verde che rosso), ma anche il fatto che il

veicolo si fermi correttamente prima di un semaforo rosso e che riparta quando questo diventa verde.

Successivamente è stato integrato il controllo per evitare le collisioni con i lead vehicle. Il sistema è stato testato con questa ulteriore complessità per verificare che funzionasse bene in entrambe le condizioni. In questa fase il sistema è stato testato con pochissimi pedoni, generalmente 0-1, e con un numero abbastanza alto di veicoli, così da verificare che il nostro veicolo si metta correttamente in coda e prosegua il suo tragitto alla stessa velocità del veicolo davanti, cercando di mantenere un'adeguata distanza di sicurezza.

Infine, è stato sviluppato il controllo per evitare le collisioni con i pedoni e il sistema è stato testato senza veicoli ma con un numero abbastanza elevato di pedoni per verificare che il veicolo rilevi correttamente che un pedone stia attraversando la strada e si fermi per permettergli di attraversare.

Infine, il sistema è stato testato nella sua interezza per verificare il corretto funzionamento anche in situazioni combinate, ad esempio:

- Il veicolo è fermo ad un semaforo rosso, scatta il verde mentre un pedone sta attraversando la strada (l'obiettivo è rimanere fermi per far attraversare il pedone e poi ripartire)

3.2.1 Problemi riscontrati nei test

- Il veicolo in alcuni casi effettua una curva troppo larga e potrebbe collidere con un altro veicolo.
- Al fine di evitare uno o più pedoni, il veicolo potrebbe fermarsi al centro dell'incrocio, questo potrebbe causare un incidente con i veicoli nelle circostanze.
- In caso di stop al semaforo, è capitato che un veicolo in arrivo alle nostre spalle spinga l'auto oltre il semaforo, in questo modo il detector non vede più lo stesso e ripartiamo col rosso.
- Dato che il semaforo giallo viene classificato come "go", in alcuni casi potrebbe scattare il rosso quando ormai il veicolo è impossibilitato a fermarsi in tempo. Ciò potrebbe anche causare un incidente nell'incrocio.
- La presenza di pedoni all'interno dell'immagine BGRA può rendere più fallace il detector. In particolare, in alcuni casi, si sono verificate ripetute miss detection che hanno causato una ripartenza con semaforo rosso.
- In alcuni casi i pedoni iniziano ad attraversare quando ormai il veicolo è troppo vicino e non ha il tempo e lo spazio per fermarsi in maniera adeguata.

3.3 Screenshot di casi d'uso

In questo paragrafo vengono mostrati alcuni casi che si verificano più frequentemente durante le simulazioni e che il nostro software gestisce in maniera corretta.

Figura 1 - Semaforo Rosso: velocità 0 km/h

Figura 2 - Semaforo Verde: velocità 14 km/h

Figura 3 - Pedone che attraversa orizzontalmente, l'auto si ferma per permettere l'attraversamento

Figura 4 - Pedone che attraversa diagonalmente, l'auto si ferma per permettere l'attraversamento

Figura 5 - Pedoni che attraversano contemporaneamente, l'auto si ferma per permettere l'attraversamento di tutti i pedoni

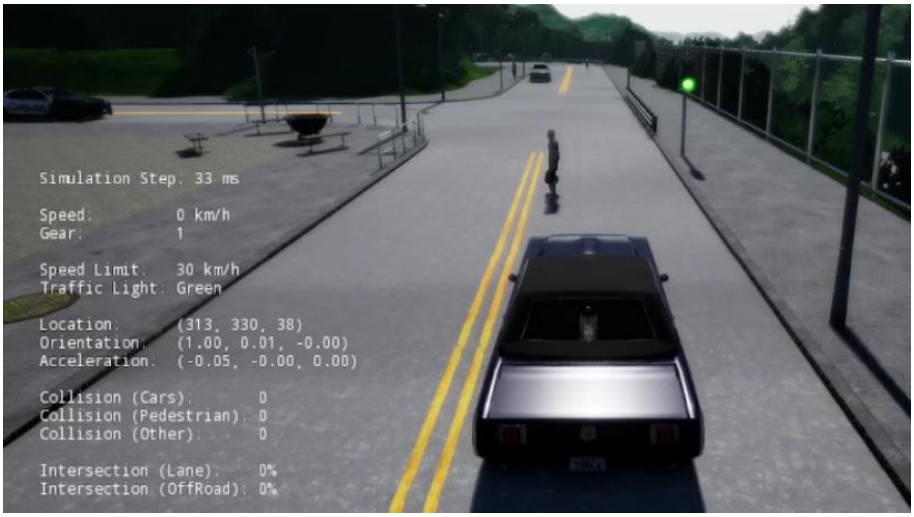


Figura 6 - Pedone davanti al semaforo verde, l'auto si ferma

Figura 7 - Auto continua la corsa nel caso in cui il pedone si ferma al lato

Figura 8 - Auto che si mette in coda assumendo la velocità di chi lo precede

Figura 9 - Gestione combinata di pedoni, lead vehicle e semaforo

3.4 Esempi video di funzionamento

Di seguito vengono riportati i link a 3 video, che mostrano il funzionamento in tempo reale del sistema proposto in Carla nelle seguenti tre situazioni: stop al semaforo rosso e passaggio con semaforo verde, gestione collisioni con altri veicoli, gestione collisioni con pedoni.

***Link Video* 1**: stop al semaforo rosso e passaggio con semaforo verde

https://drive.google.com/file/d/1Jxj1CApgbv_gG07vLSRTzIIlhZgBLgAI/view?usp=sharing

***Link Video* 2**: gestione collisioni con pedoni

https://drive.google.com/file/d/1t3Blp0PN-TumKl5qHoWlXYChIK7k_FT5/view?usp=sharing

***Link Video* 3**: gestione collisioni con altri veicoli

https://drive.google.com/file/d/1m42y_uaNvZslVV5IZIljXS5RfDNOwOy9/view?usp=sharing