Appunti Completi di Guida Autonoma

Dalla Percezione al Controllo

Sintesi del Corso

Indice

1 Introduzione al Corso e Storia della Guida Autonoma

1.1 Dettagli del Corso ed Esame

Il corso di "Perception, Localization and Mapping for Autonomous Driving" nasce dalla fusione sinergica di due insegnamenti precedenti: *Autonomous Driving and ADAS Technologies* (tenuto dal Prof. Pietro Cerri) e *Visual Perception for Self-Driving Cars* (tenuto dal Prof. Paolo Medici). L'obiettivo didattico è preservare integralmente i contenuti di entrambi i moduli, offrendo una formazione completa che spazia dalle tecnologie di assistenza alla guida fino alla percezione visiva avanzata per veicoli autonomi.

La valutazione finale dello studente è calcolata come media aritmetica tra due componenti fondamentali. La prima è un **Test Scritto**, strutturato con domande a risposta multipla per la verifica delle conoscenze di base e domande aperte per approfondire la teoria. La seconda è un **Progetto di Laboratorio**, che consiste nella presentazione di un lavoro pratico svolto individualmente o in coppia.

I progetti di laboratorio vertono principalmente sull'applicazione dell'Intelligenza Artificiale ai problemi della guida autonoma. Gli studenti possono scegliere tra una vasta gamma di argomenti, tra cui: la calibrazione complessa tra sensori diversi (es. LiDAR/Camera) o tramite reti neurali (es. *CalibNet*); la segmentazione semantica sia in 2D (usando architetture come *Mask R-CNN* o *SegNet*) che in 3D (con *PointNet* o *Semantic-KITTI*); la ricostruzione di scene 3D tramite tecnologie di frontiera come i *NeRF* (Neural Radiance Fields) o il *3D Gaussian Splatting*. Altri temi includono la pianificazione delle traiettorie tramite *Reinforcement Learning*, il riconoscimento della segnaletica stradale e la *Lane Detection*. È inoltre prevista la possibilità di svolgere progetti specifici legati alle attività del team *UniPR Racing*.

1.2 Storia della Guida Autonoma (AD)

La storia dell'Autonomous Driving (AD) non è lineare, ma caratterizzata da diverse ondate di interesse e sviluppo tecnologico che affondano le radici molto indietro nel tempo.

1.2.1 Le Origini (1920s - 1990s)

Già dagli anni '20 esistevano concetti embrionali di guida autonoma, come veicoli radiocomandati, sistemi di *platooning* o guida magnetica. Tuttavia, è negli anni '80 che la tecnologia inizia a maturare, entrando anche nella cultura popolare (basti pensare a KITT nel 1982). Una pietra miliare tecnica viene posta nel **1986** da Ernst Dickmanns della Mercedes-Benz, che dimostra la capacità di guida autonoma su strade ben marcate in assenza di traffico. Il decennio successivo vede progressi enormi: nel **1995**, il progetto europeo *EUREKA Prometheus* riesce a guidare nel traffico reale da Monaco di Baviera a Odense (Danimarca), raggiungendo velocità fino a 175 km/h. L'intervento umano fu richiesto solo per il 5% della distanza, principalmente per innescare manualmente i cambi di corsia. Un contributo fondamentale arriva dall'Italia nel **1998** con la missione "Mille Miglia in Automatico" del **VisLab (Università di Parma)**. Il veicolo sperimentale percorse 1860 km sulle strade pubbliche italiane per 6 giorni, operando in modalità autonoma per il **94%** del tragitto e toccando una velocità massima di 123 km/h.

1.2.2 Anni 2000: Le DARPA Challenges

L'inizio del millennio segna il passaggio dalla ricerca accademica a quella competitiva e industriale, grazie alle sfide organizzate dalla DARPA negli Stati Uniti. La prima **Grand Challenge (2004)** fu un fallimento apparente, poiché nessun team riuscì a completare il percorso off-road. Tuttavia, servì da catalizzatore: nell'edizione successiva del **2005**, ben 5 team tagliarono il traguardo. Il vincitore fu *Stanley* dell'Università di Stanford, seguito da altri veicoli notevoli come *Sandstorm/H1ghlander* della Carnegie Mellon e *TerraMax*, frutto della collaborazione tra Oshkosh e il VisLab. La maturità tecnologica fu sancita dalla **Urban Challenge (2007)**, che spostò la sfida in un ambiente urbano simulato richiedendo il rispetto del codice della strada e l'interazione con altri veicoli; in questa occasione, 6 team completarono la missione.

1.2.3 Anni 2010: La Grande Crescita

In questo decennio, l'interesse dei grandi costruttori automobilistici si consolida e nel 2009 Google entra nel settore (progetto che diventerà Waymo). Il VisLab continua a segnare record: nel **2010** organizza la spedizione intercontinentale **VIAC** (VisLab Intercontinental Autonomous Challenge), portando veicoli elettrici senza conducente dall'Italia alla Cina (Parma-Shanghai). La spedizione coprì 15.926 km in 100 giorni attraversando 9 paesi. Nel **2013**, il test **PROUD** dimostrò la capacità di guidare nel centro di Parma gestendo traffico reale, rotatorie e pedoni senza alcun intervento umano.

1.2.4 Anni 2020: Robotaxi e Livello 3

Oggi il panorama è diviso. Da un lato vi sono i servizi di **Robotaxi (Livello 4)**, come quelli operati da Waymo, Cruise e Zoox, che offrono corse senza conducente in aree specifiche (geofence) come San Francisco e Phoenix. Dall'altro, il mercato consumer vede l'arrivo dei primi sistemi di **Livello 3** (es. BMW Personal Pilot L3, Mercedes Drive Pilot) e la diffusione massiccia di sistemi di **Livello 2** avanzati, come il *Tesla Autopilot* e *FSD (Full Self-Driving)*. A titolo statistico, nel 2024 Tesla ha riportato una media di un incidente ogni 7.63 milioni di miglia con Autopilot attivo. Parallelamente, aziende come Aurora, Kodiak e Gatik stanno sviluppando camion autonomi per rivoluzionare la logistica.

1.3 Definizioni e Livelli SAE (J3016)

La SAE International ha definito una tassonomia standard per classificare l'automazione dei veicoli. Per comprenderla, è necessario definire alcuni concetti chiave:

- **DDT (Dynamic Driving Task):** È l'insieme di tutte le manovre operative (sterzo, freno, accelerazione) e tattiche necessarie per guidare il veicolo.
- **OEDR (Object and Event Detection and Response):** È la capacità di monitorare l'ambiente circostante e reagire appropriatamente agli altri utenti della strada e agli eventi.
- **ODD (Operational Design Domain):** Definisce le condizioni specifiche (es. meteo, tipo di strada, range di velocità) entro le quali il sistema è progettato per funzionare correttamente.

- **Fallback:** È la procedura di sicurezza (o *Minimal Risk Condition* - MRC) che il sistema o il guidatore deve attuare in caso di guasto o quando il veicolo esce dal suo ODD.

La classificazione si articola in **6 livelli**, con una distinzione fondamentale tra i sistemi di supporto (L0-L2, supervisione umana richiesta) e i sistemi automatizzati (L3-L5):

- **Livello 0 (No Automation):** Il guidatore esegue l'intero DDT. Il sistema fornisce solo avvisi momentanei (es. Blind Spot, Lane Departure) o interventi di emergenza (AEB).
- **Livello 1 (Driver Assistance):** Automazione di *una sola* funzione del moto longitudinale o laterale (o sterzo o velocità), ma mai entrambe contemporaneamente. Esempi sono l'ACC (Adaptive Cruise Control) o il Lane Centering presi singolarmente.
- **Livello 2 (Partial Automation):** Automazione simultanea di sterzo e velocità (es. ACC + Lane Centering). Il guidatore rimane responsabile e deve supervisionare costantemente l'ambiente. Esempi commerciali includono Tesla Autopilot e GM Super Cruise.
- **Livello 3 (Conditional Automation):** L'auto esegue l'intero DDT e l'OEDR autonomamente, ma solo in un ODD limitato (es. ingorghi in autostrada). Il guidatore non è tenuto a supervisionare, ma deve rimanere pronto a riprendere il controllo (*Fallback*) quando il sistema lo richiede.
- **Livello 4 (High Automation):** L'auto gestisce autonomamente guida e *Fallback* in un ODD specifico. Non è richiesto alcun intervento umano. Esempi sono i Robotaxi o le navette in aree delimitate.
- **Livello 5 (Full Automation):** Automazione completa in qualsiasi condizione stradale e meteorologica, equivalente alle capacità di un guidatore umano esperto, senza limiti di ODD.

1.4 Componenti del Veicolo Autonomo

Un veicolo a guida autonoma è un sistema complesso che può essere suddiviso in tre macro-blocchi logici:

1. **Sensori:** Gli "occhi" del veicolo, necessari per percepire il mondo esterno.
2. **Elaborazione (Il Cervello):** L'unità di calcolo che processa i dati grezzi per comprendere l'ambiente e prendere decisioni. Questo stack software include moduli per la Percezione, la Fusione Sensoriale, la Localizzazione (uso di Mappe), la Previsione delle intenzioni altrui (*Forecasting*), la Pianificazione (*Planning*) e il Controllo.
3. **Attuatori (Drive-by-wire):** L'interfaccia elettronica che converte le decisioni digitali in azioni fisiche. Si distinguono in attuatori *primari* (Steer-by-wire per lo sterzo, Brake-by-wire per i freni, Throttle-by-wire per l'acceleratore, Shift-by-wire per il cambio) e *secondari* (gestione di luci, clacson, frecce e tergicristalli).

2 Il Settore in Trasformazione (Modulo 2)

La Guida Autonoma (AD) si posiziona oggi all'intersezione tra diverse discipline avanzate: Intelligenza Artificiale, robotica, ingegneria dei sistemi automotive e studi sul cambiamento sociale. Non si tratta più di un concetto futuristico, bensì di una realtà in fase di sviluppo attivo, sebbene la strada verso la piena maturità sia ancora costellata di sfide irrisolte.

Attualmente, il settore sta vivendo una fase di **consolidamento**. Dopo anni di grande entusiasmo mediatico ("hype"), il mercato sta operando una selezione naturale: molte startup stanno chiudendo o vengono acquisite, e l'attenzione generale si sta spostando dalla ricerca pura al dispiegamento pratico delle tecnologie. In questo scenario competitivo, solo i grandi attori dotati di capitali ingenti riescono a sopravvivere e a sostenere i costi di sviluppo.

2.1 Evoluzione Storica

La storia recente dell'AD può essere suddivisa in fasi distinte che evidenziano il passaggio progressivo dal mondo accademico a quello industriale.

2.1.1 Gli Inizi (Anni '90)

In una prima fase, lo sviluppo dell'AD era guidato principalmente da istituzioni accademiche visionarie, tra cui spiccano l'Università di Parma (UniPR), la Carnegie Mellon University (CMU) e l'Università delle Forze Armate Federali di Monaco (UBW). Le soluzioni di questo periodo erano caratterizzate da idee concettualmente semplici e dall'uso di algoritmi basati su regole (non esisteva ancora il Deep Learning applicato a questo campo). L'hardware si affidava a sensori *off-the-shelf* non specifici per l'automotive e il principale limite allo sviluppo era rappresentato dalla potenza di calcolo, che costituiva un vero e proprio collo di bottiglia per l'elaborazione in tempo reale.

2.1.2 Le DARPA Challenges (2004-2007)

Un punto di svolta fondamentale è stato rappresentato dalle competizioni organizzate dalla DARPA (*Grand Challenge* e *Urban Challenge*). Questi eventi hanno avuto il merito di creare cooperazione tra i ricercatori e di stimolare l'interesse industriale, segnando il passaggio dai laboratori universitari alla realizzazione di prototipi di ricerca decisamente più costosi e performanti.

2.1.3 L'Era Industriale e del Deep Learning (2010-Oggi)

L'ingresso ufficiale dell'industria Tech avviene nel **2010**, quando Google annuncia il suo programma di guida autonoma. Il paradigma tecnologico subisce una rivoluzione con l'avvento del **Deep Learning (DL)**. L'importanza si sposta drasticamente dagli algoritmi ai **dataset**: la necessità di enormi moli di dati per il training, la generazione costosa di *Ground Truth* (GT) e le risorse necessarie per la simulazione hanno creato una barriera all'ingresso insormontabile per le piccole startup. Tuttavia, a questo sviluppo è seguita una fase di **disillusione**. Se nel 2017 le previsioni indicavano un dispiegamento di massa per il biennio 2020-2021, la realtà si è rivelata molto più complessa. Regolamentazioni

stringenti, costi elevati e difficoltà tecniche nella gestione dei casi limite ("edge cases") hanno portato a ritardi significativi rispetto alle roadmap originali.

2.2 I Player e i Segmenti di Mercato

Il mercato della guida autonoma non è un monolite uniforme, ma vede l'interazione di tre categorie principali di attori, ognuna con obiettivi specifici: i **Car Makers (OEM)** come Ford, GM e Toyota, che controllano l'ingegneria del veicolo; i **Tier 1 Providers** come Bosch e Continental, che forniscono componenti hardware e software agli OEM; e infine i **Tech Giants** come Waymo (Google), Zoox (Amazon) e Mobileye, il cui focus è sul software, l'AI e la gestione dei dati.

È fondamentale comprendere che non tutti questi attori cercano di risolvere lo stesso problema. Esistono quattro segmenti distinti, ciascuno governato da un diverso modello di business.

2.2.1 1. Mobilità Personale (Private Vehicles)

Rappresentata da Tesla, BMW e dagli OEM tradizionali, questo segmento si basa sulla vendita del veicolo al consumatore finale, integrata da servizi software opzionali (SaaS). Le priorità progettuali sono il **costo** contenuto, il basso consumo energetico e la perfetta integrazione estetica dei sensori. Il livello di automazione target è generalmente limitato a **L2**/**L3** (assistenza alla guida).

2.2.2 2. Mobilità Condivisa (Shared Mobility/Robotaxi)

Dominata da aziende come Waymo, Cruise e Zoox, punta al modello *Mobility-as-a-Service* (MaaS). I ricavi provengono dalle corse e dalla potenziale monetizzazione del tempo e dei dati dei passeggeri. La priorità assoluta è l'**efficacia** del sistema e la rimozione del guidatore umano. In questo contesto, il costo del singolo veicolo è un fattore meno critico, poiché l'investimento viene ammortizzato su una flotta operativa 24/7. Il target tecnologico è il livello **L4** in aree definite.

2.2.3 3. Logistica a Lungo Raggio (Trucking)

Aziende come Aurora e Kodiak operano in questo settore con l'obiettivo di generare risparmio operativo tramite la rimozione dell'autista, l'ottimizzazione del carburante e l'operatività 24 ore su 24. Le priorità sono l'efficienza e la robustezza del sistema. Poiché il veicolo è uno strumento di lavoro costoso, il prezzo dell'hardware aggiuntivo per l'automazione è trascurabile rispetto ai benefici economici. L'ambiente operativo è strutturato (autostrade).

2.2.4 4. Consegne Locali (Last Mile)

Esemplificato da Nuro, questo segmento mira a sostituire i corrieri umani per le consegne urbane, applicando un premium per la rapidità del servizio. Le caratteristiche chiave dei veicoli sono il basso costo, la bassa velocità e le dimensioni ridotte.

2.3 Analisi Finanziaria: La Realtà dei Costi

L'analisi dei casi studio finanziari rivela che la redditività del settore è ancora un obiettivo lontano. **Waymo (Alphabet)**, ad esempio, nel 2024 ha generato ricavi stimati per soli 50-75 milioni di dollari, a fronte di perdite massicce comprese tra **1.2 e 1.5 miliardi**. La valutazione dell'azienda ha subito un drastico ridimensionamento, crollando dai 175 miliardi del 2018 a circa 30 miliardi nel periodo 2020-23. Similmente, **Cruise (GM)** ha accumulato perdite per oltre **10 miliardi** di dollari dal 2016. Questa emorragia finanziaria ha portato General Motors, nel 2024, ad annunciare un taglio dei fondi destinati al progetto robotaxi per rifocalizzare le risorse sui sistemi di assistenza alla guida più tradizionali.

2.4 Design Goals e Trade-offs

La progettazione di un sistema autonomo non è universale, ma dipende strettamente dal modello di business e dall'**ODD** (*Operational Design Domain*). Le scelte ingegneristiche comportano diversi compromessi (*trade-offs*).

A livello di **Sensori**, si deve bilanciare la ridondanza con il costo: un robotaxi può permettersi una suite sensoriale costosa per garantire la massima sicurezza, mentre un'auto privata deve rimanere economicamente accessibile. Per quanto riguarda l'**Elaborazione**, la scelta è tra architetture **Edge** (calcolo locale), che garantiscono bassa latenza e privacy, e architetture **Centralized** (cloud), che offrono scalabilità ma introducono una dipendenza critica dalla rete. Nell'ambito dell'**AI**, il dibattito contrappone le Reti Neurali (estremamente performanti ma opache, "black box") agli algoritmi tradizionali (più interpretabili e validabili). Infine, l'architettura **Software** può essere di tipo *End-to-End* (dai sensori direttamente ai comandi) oppure *Modulare* (composta da blocchi logici separati).

2.5 Regolamentazione e Società

Il quadro normativo gioca un ruolo cruciale nell'adozione della tecnologia. La **Cina** esercita un controllo rigido sulla qualificazione dei veicoli e sull'acquisizione dei dati per motivi di sicurezza nazionale, ma offre al contempo un forte supporto strategico al settore. Negli Stati Uniti, la situazione è frammentata: la **California** è leader globale per trasparenza e volume di test, mentre altri stati mostrano approcci divergenti (permissivi come il Nevada o restrittivi come New York), evidenziando la mancanza di uno standard federale unificato.

Il settore è inoltre animato da diverse controversie tecniche e sociali. Si discute se l'auto debba dipendere da un'**Infrastruttura Smart (V2X)** e dal 5G, rischiando però di creare un "single point of failure". Anche l'uso delle **Mappe HD** divide i costruttori: alcuni (come Waymo) le ritengono essenziali, altri (come Tesla) vogliono evitarle a causa degli alti costi di manutenzione. Un'altra questione riguarda l'affidabilità del **Crowdsourcing** per sistemi critici di sicurezza.

Un tema delicato è l'**Adattamento Culturale**: l'auto autonoma deve rispettare rigidamente le regole o adattarsi alle consuetudini locali, inclusa l'aggressività di guida? Un comportamento troppo rigido può risultare pericoloso o paralizzare il traffico (si cita l'esempio di un veicolo che ha scambiato una barriera stradale per una tigre, bloccandosi).

Sul piano etico, il concetto di **"Zero Incidenti"** è considerato da molti un mito: la perfezione è impossibile in un sistema aperto e l'obiettivo realistico deve essere la riduzione statistica degli incidenti rispetto alla guida umana. Rimangono irrisolti dilemmi morali come il "Trolley Problem" (chi salvare in caso di incidente inevitabile) e lo scetticismo sulla reale fattibilità economica e temporale del **Livello 5** (automazione totale).

2.6 Conclusioni del Modulo

In conclusione, il settore della guida autonoma continua a generare una forte domanda di ingegneri con competenze specialistiche in AD e AI. Tuttavia, stiamo vivendo una fase di consolidamento e disillusione produttiva: la produzione di massa di veicoli L4 è ancora lontana. L'impatto di questa tecnologia sarà comunque disruptivo, trasformando non solo il panorama tecnologico, ma anche l'economia, il sistema legale e l'urbanistica delle nostre città.

3 Percezione e Sensori (Modulo 3)

La percezione in un veicolo autonomo si basa su una suite eterogenea di sensori. La premessa fondamentale è che nessun sensore è perfetto singolarmente; pertanto, la **ridondanza** e la diversità fisica dei principi di funzionamento sono essenziali per garantire la sicurezza in ogni condizione operativa (ODD). Le categorie principali analizzate sono Telecamere, LiDAR, Radar, Ultrasuoni e sistemi di posizionamento.

3.1 Telecamere

Le telecamere rappresentano i sensori più vicini alla percezione umana, essendo gli unici in grado di interpretare informazioni dense come colori, segnaletica orizzontale, cartelli stradali e testi.

3.1.1 Funzionamento e Ottica

Il sistema di acquisizione è composto da tre elementi: un'ottica (lenti), un sensore (tipicamente CMOS con un filtro *Bayer* per catturare i colori) e un processore di segnale (ISP). Parametri fondamentali per la progettazione sono la **Lunghezza Focale** e l'**Apertura** (f-number), che determinano il campo visivo (FOV) e la quantità di luce in ingresso. L'apertura influenza anche la **profondità di campo** (DOF), ovvero l'intervallo di distanze in cui gli oggetti appaiono nitidi.

Un aspetto critico nell'automotive è la gestione dell'otturatore (*Shutter*). La maggior parte dei sensori utilizza il **Rolling Shutter**, che espone e legge l'immagine riga per riga. Sebbene sia una soluzione economica, introduce distorsioni geometriche significative (effetto *Jello*) quando si riprendono oggetti in movimento rapido. Per applicazioni ad alta velocità o precisione, si preferisce il **Global Shutter**, che espone tutti i pixel simultaneamente, eliminando queste distorsioni ma a un costo superiore.

Inoltre, l'ambiente stradale presenta variazioni di luce estreme (es. l'uscita da un tunnel buio in pieno giorno). Poiché l'occhio umano gestisce circa 24 stop di range dinamico mentre una fotocamera standard ne copre solo 12, è indispensabile utilizzare tecniche **HDR (High Dynamic Range)** per evitare immagini sovraesposte o sottoesposte che renderebbero cieco l'algoritmo di percezione.

3.1.2 Spettro e Tipologie Avanzate

Oltre alla luce visibile (380-750 nm), si utilizzano altre bande dello spettro elettromagnetico. Le telecamere nel **Vicino Infrarosso (NIR)** sono usate per la visione notturna attiva o per i sistemi di monitoraggio del guidatore (DMS), mentre quelle nel **Tardo Infrarosso (Termiche)** rilevano il calore emesso dai corpi, risultando eccellenti per individuare pedoni nel buio completo.

Per la percezione 3D, si adottano configurazioni specifiche:

- **Stereo Camere:** Utilizzano due obiettivi separati da una distanza nota (*baseline*). Attraverso la triangolazione, permettono di calcolare una mappa di disparità e quindi la profondità. Richiedono una calibrazione precisa e molta potenza di calcolo, e soffrono su superfici prive di texture (es. muri bianchi).
- **Time of Flight (ToF):** Emettono luce modulata e misurano il tempo di ritorno per ogni singolo pixel, fornendo una mappa di profondità densa ad alta frequenza.

3.2 LiDAR (Light Detection and Ranging)

Il LiDAR è un sensore attivo che emette impulsi laser per misurare la distanza tramite il principio del *Time of Flight*. Il risultato è una **Point Cloud** (nuvola di punti) 3D ad alta precisione, dove ogni punto possiede coordinate (x, y, z) e un valore di **intensità** (riflettività), utile per distinguere i materiali (es. asfalto vs linee verniciate).

Esistono due architetture principali. I **LiDAR Meccanici** (Scanning) utilizzano specchi rotanti per coprire un campo visivo di 360°. Offrono ottime prestazioni ma sono costosi, ingombranti e le parti mobili soffrono le vibrazioni del veicolo. I **LiDAR Solid State**, invece, sono privi di parti mobili macroscopiche; risultano più robusti ed economici, ma offrono un campo visivo (FOV) limitato, simile a quello di una telecamera. Nonostante la precisione geometrica, il LiDAR vede le sue prestazioni degradare significativamente in presenza di particolato atmosferico come pioggia forte, nebbia o polvere.

3.3 Radar (Radio Detection and Ranging)

Il Radar utilizza onde radio (nello standard automotive a 77 GHz) per rilevare oggetti metallici. Il suo vantaggio unico risiede nella capacità di misurare direttamente la **velocità radiale** degli oggetti sfruttando l'**Effetto Doppler**, senza bisogno di differenziare la posizione nel tempo. Inoltre, è estremamente robusto alle condizioni meteo avverse, riuscendo a "vedere" attraverso nebbia e pioggia. I limiti storici del Radar sono la bassa risoluzione angolare e la difficoltà nel distinguere oggetti statici (spesso filtrati per evitare falsi allarmi, come i tombini). Tuttavia, la nuova generazione di **Radar 4D (Imaging Radar)** utilizza antenne MIMO per fornire risoluzione anche in elevazione (oltre a distanza, azimut e velocità), avvicinandosi alla densità di punti di un LiDAR.

3.4 Altri Sensori e Posizionamento

La suite sensoriale è completata dagli **Ultrasuoni (Sonar)**, sensori economici basati su onde sonore, utilizzati quasi esclusivamente per manovre a bassa velocità e breve raggio come il parcheggio. Per la localizzazione, si utilizza l'**IMU (Inertial Measurement Unit)**, che combina accelerometri, giroscopi e magnetometri per stimare assetto e accelerazioni; soffre però di una deriva (*drift*) che cresce nel tempo e richiede correzione. Il

posizionamento assoluto è fornito dal **GNSS**, che però, nell'uso civile standard (precisione 3-10m), è insufficiente per la guida autonoma. Si utilizzano quindi tecniche **RTK (Real Time Kinematic)** che, tramite stazioni di terra fisse, correggono l'errore portando la precisione a livello centimetrico. Infine, il **V2X (Vehicle-to-Everything)** permette la comunicazione wireless (tramite standard DSRC o C-V2X/5G) tra veicoli e infrastrutture, estendendo la percezione oltre il raggio visivo dei sensori fisici.

4 Introduzione alla Calibrazione (Modulo 4)

La calibrazione è il processo fondamentale per identificare i parametri del modello matematico di un sensore (come il modello proiettivo o *pinhole*) al fine di correlare i punti misurati sull'immagine alle corrispondenti coordinate metriche nel mondo reale. L'affidabilità di questa correlazione dipende direttamente dall'accuratezza del modello scelto e dalla precisione della stima dei suoi parametri.

4.1 Tipologie di Parametri

Nel contesto del modello Pinhole, i parametri si dividono in due categorie distinte:

- **Parametri Estrinseci (Extrinsic):** Definiscono la posa del sensore nello spazio 3D, ovvero la trasformazione rigida tra il sistema di riferimento della camera e quello del mondo. Sono indipendenti dalla costruzione interna del dispositivo e comprendono 6 Gradi di Libertà (DOF): 3 per la **Traslazione** e 3 per la **Rotazione**.
- **Parametri Intrinseci (Intrinsic):** Sono specifici della fisica del sensore e dell'ottica. Includono la **Lunghezza Focale** (f, che può avere 1 o 2 DOF se i pixel non sono perfettamente quadrati), il **Punto Principale** (u_0, v_0, 2 DOF) che rappresenta l'intersezione dell'asse ottico col piano immagine, i coefficienti di **Distorsione della lente** e, nel caso di sensori CMOS comuni, il tempo di lettura (*Rolling Shutter*).

5 Modello Matematico della Camera

5.1 Matrice dei Parametri Intrinseci (K)

La proiezione di un punto 3D con coordinate (x, y, z) nel sistema camera alle coordinate pixel 2D (u, v) è descritta linearmente dall'equazione:

$$\begin{pmatrix} u \\ v \\ 1 \end{pmatrix} \equiv \mathbf{K} \begin{pmatrix} x/z \\ y/z \\ 1 \end{pmatrix} \tag{1}$$

La matrice $\mathbf{K}$ incapsula i parametri interni ed è definita come:

$$\mathbf{K} = \begin{bmatrix} k_u & k_\gamma & u_0 \\ 0 & k_v & v_0 \\ 0 & 0 & 1 \end{bmatrix} \tag{2}$$

Espandendo l'equazione (e ignorando per ora la distorsione), le coordinate immagine si ottengono come:

$$u = f\frac{x}{z} + u_0, \quad v = f\frac{y}{z} + v_0$$

5.2 Distorsione della Lente

Le lenti reali, specialmente quelle grandangolari usate nell'automotive, introducono distorsioni non lineari che curvano le linee rette. Il modello standard per correggere questo fenomeno è la distorsione radiale di **Brown-Conrady**, che approssima la deformazione tramite una serie di Taylor in funzione della distanza radiale r dal centro ottico:

$$f(r) = r(1 + k_1 r^2 + \cdots + k_n r^{2n})$$

Nella pratica ingegneristica, per evitare di calcolare questa funzione inversa complessa a ogni frame, si pre-calcolano delle *warp-tables* che permettono di rimuovere la distorsione via software in modo efficiente, lavorando poi su immagini "rettificate" ideali.

6 Sistemi di Riferimento e Rotazioni

La percezione richiede di trasformare dati attraverso diverse catene di sistemi di riferimento (Frame), tipicamente:

$$\text{Image} \leftrightarrow \text{Camera} \leftrightarrow \text{Sensor} \leftrightarrow \text{Body/Vehicle} \leftrightarrow \text{World}$$

6.1 World Frames: ENU vs NED

Esistono standard contrastanti per definire il sistema di coordinate globale:

- **ENU (East-North-Up):** È lo standard de facto per i veicoli terrestri e la robotica (ISO 8855). L'asse X punta a Est, Y a Nord e Z verso l'Alto.
- **NED (North-East-Down):** È lo standard storico per l'aerospazio e la nautica (ISO 1151). L'asse Z punta verso il Basso (allineandosi alla gravità), X a Nord e Y a Est.

6.2 Rappresentazione delle Rotazioni

La rotazione nello spazio 3D (appartenente al gruppo $SO(3)$) è un concetto matematico che può essere rappresentato in vari modi, ciascuno con vantaggi specifici.

Gli **Angoli di Eulero** (es. Roll, Pitch, Yaw) decompongono la rotazione in tre movimenti sequenziali attorno agli assi. Sono molto intuitivi per l'uomo e facili da linearizzare per piccoli angoli, ma soffrono del grave problema del **Gimbal Lock** (perdita di un grado di libertà se due assi si allineano) e dipendono dall'ordine di applicazione.

Le **Matrici di Rotazione** (3×3) eliminano le ambiguità e sono facili da comporre tramite moltiplicazione, ma sono inefficienti (sovra-parametrizzate: usano 9 numeri per descrivere 3 DOF) e difficili da interpolare fluidamente.

La rappresentazione **Asse-Angolo** usa un vettore $\mathbf{k}$ (asse di rotazione) e uno scalare θ (angolo). La matrice di rotazione si ottiene tramite la formula di Rodrigues:

$$\mathbf{R} = \mathbf{I} + (\sin\theta)\mathbf{K} + (1 - \cos\theta)\mathbf{K}^2$$

dove $\mathbf{K}$ è la matrice antisimmetrica del vettore $\mathbf{k}$. Sebbene geometricamente chiara, soffre di singolarità nell'origine.

I **Quaternioni** (estensione dei numeri complessi: $q = q_w + q_v\mathbf{i} + q_y\mathbf{j} + q_z\mathbf{k}$) rappresentano spesso la scelta migliore per il software di navigazione. Non soffrono di Gimbal Lock, sono computazionalmente efficienti da comporre e permettono un'interpolazione sferica fluida (**SLERP**). Lo svantaggio è la scarsa intuitività per l'operatore umano.

7 Proiezione Completa e Metodi di Calibrazione

7.1 Equazione di Proiezione

Combinando i parametri intrinseci ed estrinseci, possiamo descrivere la proiezione di un punto generico nel mondo (x_i, y_i, z_i) sul piano immagine (u_i, v_i) tramite una singola matrice di proiezione $\mathbf{P}$ di dimensione 3×4:

$$(u_i, v_i)^T \equiv \mathbf{P}(x_i, y_i, z_i, 1)^T \tag{3}$$

Dove $\mathbf{P}$ è composta come $\mathbf{P} = \mathbf{K}[\mathbf{R}|\tilde{\mathbf{t}}_0]$, con $\tilde{\mathbf{t}}_0 = -\mathbf{Rt}$ che rappresenta la posizione della camera nel sistema di riferimento mondo.

7.2 Metodi di Calibrazione

Esistono approcci diversi per stimare questa matrice:

La **Calibrazione Implicita** stima direttamente la matrice $\mathbf{P}$ (o l'omografia $\mathbf{H}$) risolvendo un sistema lineare omogeneo (tipicamente con SVD), senza preoccuparsi di separare i parametri fisici interni da quelli esterni. Poiché la matrice $\mathbf{P}$ ha 11 gradi di libertà, sono necessari almeno **6 punti** noti non complanari (algoritmo **DLT** - *Direct Linear Transformation*).

L'**Omografia (H)** è un caso particolare (matrice 3×3, 8 DOF) che descrive la trasformazione proiettiva tra due piani. In guida autonoma è fondamentale per l'**IPM (Inverse Perspective Mapping)**, che rimuove la prospettiva frontale per creare una vista dall'alto (*Bird's Eye View*) utile alla *Lane Detection*. Richiede almeno **4 punti** complanari.

La **Calibrazione Esplicita** mira invece a recuperare i parametri fisici reali ($f, \mathbf{R}, \mathbf{t}$, ecc.) minimizzando un errore geometrico di riproiezione non lineare (**MLE** - *Maximum Likelihood Estimator*):

$$S = \sum ||(u_i, v_i) - f_\beta(x_i, y_i, z_i)||^2$$

Questo approccio è più robusto al rumore e gestisce correttamente la distorsione della lente.

Il metodo standard de-facto è il **Metodo di Zhang**, che utilizza una scacchiera (*checkerboard*) piana osservata da diverse posizioni. Sfruttando i vincoli di orto-normalità della rotazione, il metodo risolve i parametri intrinseci. Matematicamente, ogni vista della scacchiera fornisce 8 vincoli ma introduce nuove incognite di posa; sono necessarie almeno **3 viste** (o 2 assumendo pixel perfettamente rettangolari) per una soluzione univoca.

Infine, la **Calibrazione Multi-Sensore** allinea sensori diversi (es. LiDAR e Camera). Può essere **Target-Based**, usando oggetti visibili a entrambi (es. scacchiere con fori), oppure **Targetless**, sfruttando la coerenza del movimento (odometria) o feature naturali della scena per l'allineamento automatico.

8 Tracking e Stima dello Stato (Modulo 5)

In un sistema di percezione autonomo, il rilevamento di oggetti (*Object Detection*) risponde puntualmente alla domanda "Dove sono i candidati in questo singolo frame?". Tuttavia, per navigare in sicurezza, è indispensabile il **Tracciamento (Tracking)**, che risponde alla domanda "Quale rilevamento corrisponde a quale oggetto fisico nel tempo?" e ne stima lo stato dinamico più probabile (posizione, velocità, accelerazione), correggendo gli errori di misura.

8.1 Necessità e Sfide del Tracking

Il tracking è necessario per modellare la continuità degli oggetti quando il rilevatore istantaneo fallisce. Le cause comuni di fallimento includono le **occlusioni** (totali o parziali), le variazioni rapide di punto di vista o illuminazione, il **clutter** (rumore di sfondo scambiato per oggetto) e la necessità di predire il comportamento futuro (es. la traiettoria di un pedone che attraversa).

Negli scenari complessi di guida urbana nasce il problema del **Multi-Object Tracking (MOT)**, dove la presenza di molti oggetti simili vicini tra loro genera ambiguità. Le sfide principali sono:

- **Predizioni senza misure:** Un oggetto tracciato non viene più rilevato (è sparito, è uscito dal campo visivo o è momentaneamente occluso?).
- **Misure inaspettate:** Il sensore rileva qualcosa dove non era previsto nulla (è un nuovo oggetto che entra in scena o è un falso positivo/rumore?).
- **Ambiguità di associazione:** Una singola predizione può avere più misure candidate nelle vicinanze, o viceversa.

8.2 Data Association

Per risolvere queste ambiguità, si utilizzano algoritmi di associazione dati. Una tecnica di pre-processing comune è il **Gating**, che elimina a priori le associazioni geometricamente improbabili per ridurre il carico computazionale. Per l'associazione vera e propria, l'approccio più semplice è il **Nearest Neighbor (NN)**, che associa la misura più vicina alla predizione; tuttavia, è fragile in ambienti affollati. L'approccio ottimale per l'assegnazione globale è l'**Algoritmo Ungherese (Kuhn-Munkres)**, che risolve il problema di assegnazione lineare minimizzando il costo totale su una matrice di costo o un grafo bipartito pesato.

Esiste inoltre una distinzione operativa: il **Tracking Online** stima lo stato corrente basandosi solo sulle osservazioni passate e presenti ed è l'unico utilizzabile per la guida real-time, pur essendo soggetto a deriva (*drifting*). Il **Tracking Offline (Batch)**, invece, utilizza l'intera sequenza temporale (inclusi i frame futuri) per ottimizzare le traiettorie globalmente; gestisce meglio le occlusioni ma è limitato all'analisi video a posteriori.

9 Framework Bayesiano e Filtri

La stima dello stato in presenza di incertezza si fonda sulla teoria della probabilità e sulla **Regola di Bayes**:

$$p(x|z) = \frac{p(z|x)p(x)}{p(z)}$$

Dove la distribuzione *Posterior* $p(x|z)$ (stato stimato dopo la misura) è proporzionale al prodotto della *Likelihood* $p(z|x)$ (quanto è verosimile la misura dato lo stato) per la *Prior* $p(x)$ (stima precedente).

Si assume generalmente che il sistema sia **Markoviano**, ovvero che lo stato futuro dipenda esclusivamente dallo stato corrente e non dall'intera storia passata. Il filtro ricorsivo opera ciclicamente in due passi:

1. **Predizione (Time Update):** Si proietta lo stato in avanti nel tempo ($\hat{x}_k^-$) usando il modello fisico di moto. L'incertezza aumenta.
2. **Correzione (Measurement Update):** Si corregge la predizione usando la nuova misura z_k, ottenendo lo stato a posteriori ($\hat{x}_k$). L'incertezza diminuisce.

9.1 Il Filtro di Kalman (KF)

Il Filtro di Kalman rappresenta la soluzione ottima (in senso di minimi quadrati) per sistemi **lineari** affetti da rumore **Gaussiano**. Il sistema è modellato dalle equazioni:

$$\begin{aligned} x_k &= Ax_{k-1} + Bu_{k-1} + \omega_{k-1} \quad \text{(Dinamica)} \\ z_k &= Hx_k + \nu_k \quad \text{(Misura)} \end{aligned}$$

dove $\omega \sim \mathcal{N}(0, Q)$ e $\nu \sim \mathcal{N}(0, R)$ sono i rumori di processo e misura.

L'algoritmo calcola il **Guadagno di Kalman** K_k, un fattore di peso che decide quanto fidarsi della nuova misura rispetto alla predizione del modello:

$$K_k = \frac{P_k^- H^T}{HP_k^- H^T + R}$$

Lo stato viene aggiornato sommando alla predizione l'errore tra misura reale e attesa (detto *residuo* o *innovazione*), pesato per K_k:

$$\hat{x}_k = \hat{x}_k^- + K_k(z_k - H\hat{x}_k^-)$$

9.2 Filtri per Sistemi Non Lineari

La realtà fisica della guida (es. cinematica del veicolo) è raramente lineare. Per questo si adottano varianti avanzate.

9.2.1 Extended Kalman Filter (EKF)

È lo standard industriale. Gestisce le non-linearità $f(x)$ e $h(x)$ linearizzandole localmente attorno alla stima corrente tramite l'espansione in serie di Taylor al primo ordine. Le matrici costanti A e H vengono sostituite dai **Jacobiani**:

$$A_k = \left.\frac{\partial f}{\partial x}\right|_{\hat{x}_{k-1}}, \quad H_k = \left.\frac{\partial h}{\partial x}\right|_{\hat{x}_k^-}$$

Sebbene efficace, il calcolo dei Jacobiani può essere matematicamente complesso e costoso, e il filtro può divergere se le non-linearità sono forti.

9.2.2 Unscented Kalman Filter (UKF)

L'UKF affronta il problema senza calcolare Jacobiani, basandosi sull'intuizione che "è più facile approssimare una distribuzione di probabilità che una funzione non lineare arbitraria". Utilizza la **Unscented Transform (UT)**: seleziona deterministicamente un set di punti campione (**Sigma Points**) che catturano media e covarianza della distribuzione, li propaga attraverso la funzione non lineare vera e poi ricalcola le statistiche dai punti trasformati. Spesso risulta più accurato dell'EKF.

9.2.3 Particle Filter (Sequential Monte Carlo)

Quando il sistema è fortemente non lineare e, soprattutto, il rumore è **Non Gaussiano** (es. distribuzioni multimodali, come l'incertezza "sono nella corsia A o B?"), si usa il Particle Filter. Invece di usare parametri fissi (media/covarianza), rappresenta la distribuzione di probabilità come un insieme di campioni casuali (**particelle**) pesati. L'algoritmo **SIR (Sequential Importance Resampling)** opera in tre fasi:

1. **Prediction:** Le particelle vengono spostate casualmente secondo il modello di moto.
2. **Update:** Il peso di ogni particella viene aggiornato in base alla verosimiglianza della misura osservata.
3. **Resampling:** Per evitare la degenerazione (dove poche particelle hanno tutto il peso), si eliminano le particelle improbabili e si duplicano quelle probabili.

Il costo computazionale è elevato, proporzionale al numero di particelle.

10 Reti Neurali e Percezione (Modulo 6)

Le Reti Neurali Artificiali (ANN) costituiscono il cuore pulsante dei moderni sistemi di percezione per la guida autonoma. Esse sono composte da unità elementari, dette "neuroni artificiali", che apprendono ad approssimare funzioni matematiche complesse in grado di mappare input eterogenei (come immagini o dati LiDAR) in output decisionali (come la classificazione di oggetti o comandi di sterzo).

10.1 Principi di Funzionamento

10.1.1 Il Neurone Artificiale

Ogni singolo neurone esegue due operazioni matematiche fondamentali in sequenza. Prima calcola una **somma pesata** degli input (x) moltiplicati per i rispettivi pesi (w) e addizionati a un termine di bias (b), ottenendo $z = \sum w \cdot x + b$. Successivamente, applica al risultato una **funzione di attivazione** non lineare $f(z)$ (come la ReLU o la Sigmoide). Senza questa non-linearità, l'intera rete, per quanto profonda, collasserebbe matematicamente in un semplice modello lineare, perdendo la capacità di apprendere relazioni complesse.

10.1.2 Architetture Principali

Le reti si sono evolute in diverse famiglie architetturali specializzate:

- **Fully Connected (MLP):** Composte da strati densi generici, dove ogni neurone è connesso a tutti quelli dello strato precedente.
- **Convolutional Neural Networks (CNN):** Progettate specificamente per elaborare dati strutturati a griglia come le immagini. Utilizzano filtri (kernel) che scorrono sull'input per estrarre feature spaziali locali (bordi, forme), garantendo invarianza alla traslazione.
- **Recurrent Neural Networks (RNN):** Ideali per dati sequenziali o temporali (es. predizione di traiettorie), poiché mantengono una "memoria" degli stati passati.
- **Transformers:** Originariamente nati per l'elaborazione del linguaggio naturale (NLP), sono oggi lo stato dell'arte anche nella Visione Artificiale. Basati sul meccanismo di **Attention**, permettono di modellare relazioni globali tra parti distanti dell'input meglio delle CNN.

10.1.3 Il Processo di Apprendimento (Learning)

L'addestramento supervisionato avviene attraverso un ciclo iterativo composto da quattro fasi:

1. **Forward Pass:** L'input attraversa tutti i livelli della rete fino a generare una predizione finale.
2. **Calcolo della Loss:** Si misura l'errore calcolando la differenza tra la predizione e la verità di base (*Ground Truth*) tramite una funzione di costo. Un esempio comune per la classificazione è la *Cross Entropy*, che quantifica la divergenza tra la distribuzione di probabilità predetta e quella reale.
3. **Backpropagation:** Attraverso la *regola della catena* del calcolo differenziale, si propagano i gradienti dell'errore all'indietro, determinando quanto ogni singolo peso della rete ha contribuito all'errore totale.
4. **Weight Update:** Si aggiornano i pesi utilizzando algoritmi di ottimizzazione come la **Discesa del Gradiente** ($W_{new} = W_{old} - \alpha \frac{dJ}{dW}$), dove α è il tasso di apprendimento (*learning rate*), per minimizzare progressivamente la Loss.

11 La Sfida della Percezione Reale

L'obiettivo della percezione è comprendere semanticamente la scena stradale per alimentare i moduli di Pianificazione e Controllo. Tuttavia, il mondo reale presenta una sfida statistica nota come il problema della **"Long Tail"** (Coda Lunga). Mentre il 99.9% delle situazioni di guida sono standard e ripetitive, esiste una "coda" infinita di eventi rari e imprevedibili (es. un incidente insolito, condizioni meteo estreme, oggetti mai visti prima). I dataset accademici controllati non riescono a rappresentare questa varietà. Per gestire la *Long Tail* è necessario scalare su tre fronti: utilizzare dataset massivi e variegati, sviluppare modelli con un numero enorme di parametri e migliorare la qualità e la ricchezza delle annotazioni (*Ground Truth*).

12 Task di Percezione Fondamentali

12.1 Object Detection

Il compito è identificare e localizzare oggetti dinamici (auto, pedoni, ciclisti). L'output tipico include un **Bounding Box** (2D nell'immagine o 3D nello spazio), la classe dell'oggetto e un punteggio di confidenza. Le architetture si dividono in tre categorie:

- **Two-Stage (es. Faster R-CNN):** Prima generano una serie di regioni candidate (*Region Proposals*) e poi le classificano. Sono generalmente più accurate ma computazionalmente pesanti.
- **Single-Stage (es. YOLO, SSD):** Eseguono localizzazione e classificazione in un unico passaggio denso. Sono molto più veloci e adatte ad applicazioni real-time, sebbene storicamente meno precise su oggetti piccoli.
- **Anchor-Free (es. CenterNet, DETR):** Eliminano la necessità di definire box di ancoraggio predefiniti (*anchors*), semplificando il design. DETR, in particolare, utilizza i Transformer per gestire la detection come un problema di predizione di insiemi.

12.2 Semantic Segmentation

Questo task richiede una comprensione a grana fine, classificando **ogni singolo pixel** dell'immagine nella categoria appropriata (es. strada, cielo, veicolo). L'obiettivo è estrarre il contesto semantico globale senza perdere la precisione dei dettagli spaziali. Le architetture standard (come **U-Net**, **SegNet**, **DeepLabV3+**) adottano una struttura **Encoder-Decoder**: l'encoder riduce la risoluzione spaziale per estrarre feature astratte, mentre il decoder la ripristina per generare la maschera finale. Per il real-time si usano varianti ottimizzate come *Fast-SCNN*. La metrica di valutazione standard è la **mIoU** (*mean Intersection over Union*).

12.3 Lane Detection

Consiste nel rilevare i confini delle corsie e stimare il corridoio di guida. A seconda dell'approccio, le corsie possono essere rappresentate come maschere binarie (simile alla segmentazione), come una serie di punti chiave, o come coefficienti di polinomi/spline (approccio parametrico, es. *3D-LaneNet*).

13 Task Avanzati e Integrazione

13.1 Segmentazione Avanzata

Oltre alla semantica classica, esistono varianti più ricche:

- **Instance Segmentation:** Distingue tra diverse istanze della stessa classe (es. separa "Auto A" da "Auto B"), cosa che la segmentazione semantica non fa (trattandole come un unico blob di pixel "auto"). Spesso si realizza eseguendo una segmentazione locale all'interno dei bounding box rilevati (*Mask R-CNN*).

- **Panoptic Segmentation:** È l'unificazione delle due precedenti. Fornisce una descrizione completa della scena assegnando a ogni pixel una classe semantica (per lo sfondo statico o *stuff*) e un ID istanza (per gli oggetti numerabili o *things*).

13.2 Comprensione dello Spazio e Multi-Task

Per la navigazione è cruciale distinguere tra:

- **Drivable Area:** La regione dove è **legale** guidare (rispettando il codice della strada), derivata dalla comprensione semantica.
- **Free Space:** La regione fisicamente libera da ostacoli, derivata dalla detection geometrica.

Per efficienza computazionale, invece di avere una rete separata per ogni compito, si adottano architetture **Multi-Task**: un singolo **Shared Backbone** estrae le feature dall'input e alimenta diverse "teste" (*Heads*) specializzate per Detection, Segmentazione, Depth Estimation, ecc. Inoltre, la percezione moderna è intrinsecamente **Multi-Modale**, fondendo dati provenienti da Camere, LiDAR, Radar e sensori termici per aumentare la robustezza.

Per addestrare questi modelli complessi sono nati dataset massivi come **NuScenes**, **Argoverse 2** e **Waymo Open Dataset**, che superano i vecchi standard (KITTI, Cityscapes) offrendo dati multimodali, annotazioni 3D a 360° e scenari di guida reali diversificati.

14 Geometria 3D e Visual Odometry (Modulo 7)

Questo modulo approfondisce le tecniche per estrarre informazioni tridimensionali (struttura) e di movimento (cinematica) partendo dalla sola analisi delle corrispondenze tra punti chiave (*keypoints*) nelle immagini. Le metodologie principali sono la **Stereovisione** (ricostruzione tramite coppia di camere fisse), la **Structure from Motion** (ricostruzione tramite camera singola in movimento) e la **Visual Odometry**, cruciale per la navigazione.

14.1 Geometria Epipolare

Quando due telecamere osservano la stessa scena da posizioni diverse, la geometria che le lega è detta geometria epipolare. L'obiettivo è comprendere il movimento relativo tra i sensori o la profondità della scena. Per semplificare il problema computazionale, spesso si ricorre alla **rettificazione** delle immagini: una trasformazione che allinea le due viste in modo che ogni punto su un'immagine si trovi alla stessa coordinata verticale (riga) nell'altra, riducendo la ricerca delle corrispondenze da bidimensionale a monodimensionale. Tuttavia, ciò non è sempre fisicamente possibile.

14.1.1 Il Vincolo Epipolare

Consideriamo un punto P nello spazio 3D proiettato sui piani immagine in p e p'. Se tracciamo i vettori che connettono i due centri ottici O e O' con il punto P, questi devono necessariamente giacere sullo stesso piano, detto piano epipolare. In termini vettoriali, questo si traduce nella nullità del prodotto misto:

$$\vec{Op} \cdot [\vec{OO'} \times \vec{O'p'}] = 0 \tag{4}$$

La conseguenza pratica è fondamentale: data la proiezione p in una vista, la sua corrispondenza p' nell'altra vista deve trovarsi lungo una retta specifica chiamata **linea epipolare**.

14.2 Le Matrici Algebriche Fondamentali

Questo vincolo geometrico è incapsulato in due matrici 3×3 che permettono di passare da una vista all'altra.

La **Matrice Essenziale (E)**, introdotta da Longuet-Higgins nel 1981, descrive la geometria quando i parametri intrinseci delle camere sono noti (ovvero lavorando in coordinate normalizzate). È definita come $\mathbf{E} = [\mathbf{t}]_\times \mathbf{R}$, combinando la matrice antisimmetrica della traslazione $[\mathbf{t}]_\times$ con la matrice di rotazione $\mathbf{R}$. Soddisfa la relazione $m'^T \mathbf{E} m = 0$. Ha proprietà uniche: è una matrice di rango 2, dipende solo dai parametri **estrinseci** e possiede 5 gradi di libertà (i tre angoli di rotazione e la direzione della traslazione). È definita a meno di un fattore di scala, il che significa che da sola non può recuperare le dimensioni assolute degli oggetti.

La **Matrice Fondamentale (F)** estende il concetto al caso di camere non calibrate, lavorando direttamente sui pixel p. È legata alla matrice essenziale dalla relazione $\mathbf{F} = \mathbf{K}'^{-T}\mathbf{E}\mathbf{K}^{-1}$. Anch'essa ha rango 2 ma possiede 7 gradi di libertà. I suoi kernel destro e sinistro definiscono gli **Epipoli** (e_1, e_2), ovvero i punti in cui la linea congiungente i due centri ottici interseca i piani immagine.

14.3 Stima e Algoritmo 8-Point

Per stimare queste matrici dalle corrispondenze immagine, si utilizza l'**Algoritmo 8-Point**. Poiché ogni coppia di punti fornisce un'equazione lineare ($p'^T \mathbf{F} p = 0$), con 8 punti si costruisce un sistema sovradeterminato risolvibile tramite decomposizione ai valori singolari (SVD).

Un dettaglio critico è la **Normalizzazione**. I dati pixel grezzi hanno ordini di grandezza molto diversi (es. coordinate 1000 vs termine omogeneo 1), causando instabilità numerica. L'algoritmo **Normalized 8-point** risolve il problema traslando e scalando i punti nel range $[-1, 1]$ prima del calcolo e denormalizzando il risultato alla fine. Inoltre, poiché la matrice stimata dai dati rumorosi avrà quasi certamente rango 3 (violando la geometria), si impone forzatamente il rango 2 azzerando il valore singolare più piccolo nella SVD.

Per raffinare la stima iniziale lineare, si minimizzano funzioni di costo non lineari come l'**Errore di Trasferimento Simmetrico** (distanza punto-linea epipolare) o l'errore di riproiezione geometrica (**MLE**), che è statisticamente ottimale.

14.4 Triangolazione e Ricostruzione

Una volta nota la geometria delle camere, la posizione 3D di un punto X si ottiene tramite **Triangolazione**. A causa del rumore, le rette proiettanti nello spazio non si intersecheranno mai perfettamente (sono sghembe). Metodi come la **DLT** forniscono una soluzione rapida minimizzando l'errore algebrico, mentre i metodi ottimi minimizzano l'errore di riproiezione sull'immagine.

La **Structure from Motion (SfM)** generalizza questo processo a sequenze di immagini. Se la calibrazione $\mathbf{K}$ è nota, si ottiene una ricostruzione **Euclidea** (corretta in angoli e

forme, a meno della scala). Se **K** è ignota, si ottiene solo una ricostruzione **Proiettiva**, geometricamente distorta.

14.5 Visual Odometry (VO)

La Visual Odometry stima l'ego-motion (movimento del veicolo) fotogramma per fotogramma. Introdotta per i rover marziani (NASA/JPL, 1980) e formalizzata da Nister (2004), offre un vantaggio cruciale rispetto all'odometria delle ruote: non soffre di slittamenti su terreni scivolosi, garantendo errori relativi molto bassi ($0.1\% - 2\%$).

Esistono tre strategie per stimare il movimento incrementale:

- **2D-to-2D:** Usa solo le feature immagine e la matrice Essenziale. Serve per l'inizializzazione.
- **3D-to-3D:** Se disponiamo di punti 3D per entrambi i frame (es. stereo o LiDAR), calcoliamo la trasformazione rigida che allinea le due nuvole di punti (es. metodo dei centroidi/SVD).
- **3D-to-2D (PnP):** È il metodo standard per la VO monoculare. Si associano le feature 2D del frame corrente ai punti 3D già triangolati nella mappa locale, minimizzando l'errore di riproiezione.

Infine, il **Bundle Adjustment (BA)** è la tecnica di ottimizzazione globale che raffina simultaneamente le pose di tutte le camere e le coordinate di tutti i punti 3D. La distinzione chiave tra **VO** e **SLAM** risiede nell'ambizione: la VO è locale e accetta una deriva (*drift*) lenta nel tempo; lo SLAM mira a una mappa globale consistente e corregge la deriva accumulata tramite il **Loop Closure**, ovvero riconoscendo di essere tornati in un luogo già visitato.

15 Navigazione, Mappe HD e Localizzazione (Modulo 8)

La navigazione è definita formalmente come la capacità di un robot (o veicolo) di determinare la propria posizione nello spazio e pianificare un percorso sicuro ed efficace verso una meta desiderata senza intervento umano. Esiste una correlazione diretta e stringente tra il livello di autonomia del veicolo (secondo lo standard SAE) e la precisione richiesta alla cartografia di supporto.

Per i sistemi di assistenza alla guida (**Livello 1-2**), sono sufficienti le cosiddette *Mappe ADAS*, che offrono una precisione sub-metrica spesso opzionale. Al contrario, per garantire la sicurezza nei sistemi di **Livello 3** (Conditional Automation) è necessaria la combinazione di mappe ADAS e Mappe HD con precisione centimetrica. Infine, per la guida totalmente autonoma (**Livelli 4-5**), l'uso di **HD Maps** con accuratezza centimetrica diventa un requisito imprescindibile.

Livello Autonomia	**Tipo di Mappa**	**Accuratezza Richiesta**
Livello 1-2 (Driver Assistance)	Mappe ADAS	Sub-metrica (Opzionale)
Livello 3 (Conditional)	ADAS + HD Map	Centimetrica
Livello 4-5 (High/Full Automation)	HD Map	Centimetrica (Richiesta)

Tabella 1: Requisiti delle mappe in base al livello di automazione.

15.1 Tipologie di Mappe

Non tutte le mappe hanno lo stesso scopo; si distinguono principalmente in topologiche e ad alta definizione.

15.1.1 Mappe Topologiche

Questa tipologia si concentra sulla connettività logica della rete stradale e sulle relazioni spaziali, piuttosto che sulla fedeltà geometrica o sulla scala esatta. Le priorità sono definire cosa è collegato a cosa (connettività), l'ordine di percorrenza e le relazioni di contenimento. Una caratteristica peculiare è che la geometria esatta (distanze, curvature) è spesso intenzionalmente distorta per enfatizzare la struttura del grafo della rete (simile alla mappa di una metropolitana).

15.1.2 Mappe HD (High Definition)

Le Mappe HD sono repliche digitali esatte dell'ambiente, caratterizzate da precisione centimetrica e ricchezza di dettagli semantici. Esse svolgono tre funzioni critiche per il veicolo autonomo:

1. **Localizzazione Precisa:** Forniscono un'impronta digitale dell'ambiente, permettendo al veicolo di rispondere alla domanda "Dove sono esattamente?" con un margine di errore minimo.
2. **Percezione Aumentata (Sensore Virtuale):** La mappa funge da memoria a lungo termine, permettendo al veicolo di "vedere" oltre il raggio dei sensori fisici (ad esempio, anticipando la geometria della strada dopo una curva cieca).
3. **Pianificazione del Percorso (*Path Planning*):** Offrono i vincoli geometrici precisi e le regole legali necessarie per calcolare traiettorie ottimali e sicure.

15.2 Localizzazione

La localizzazione basata su mappa è il processo che determina la posizione del veicolo confrontando i dati acquisiti in tempo reale dai sensori a bordo (LiDAR, Radar, Camera) con gli elementi statici salvati nella Mappa HD. Questo confronto continuo permette di raffinare la stima iniziale fornita dal GPS, che da sola sarebbe troppo imprecisa.

15.2.1 Visual Localization

Gli algoritmi di localizzazione visiva cercano di associare elementi percepiti nelle immagini (come linee di corsia, pali o segnali stradali) ai corrispondenti elementi vettoriali della Mappa HD per stimare la posa ottimale. La pipeline tipica prevede: l'acquisizione di un'immagine di query, l'estrazione di descrittori locali, il *matching* con il database della mappa e infine la verifica geometrica (*Image-to-3D*). Le sfide principali riguardano le variazioni drastiche dell'aspetto della scena dovute all'illuminazione (es. sole diretto contro guida notturna), al meteo (pioggia, neve che coprono la segnaletica) e alle occlusioni dinamiche causate dal traffico o dai pedoni.

15.2.2 Sensor Fusion per la Localizzazione

Poiché nessun sensore è infallibile, si ricorre alla fusione di diverse fonti:

- **IMU (Inertial Measurement Unit):** Utilizza giroscopi e accelerometri per calcolare continuamente variazioni di posizione e orientamento. È immune al *jamming* e agli inganni esterni, ma soffre di una deriva (*integration drift*) che si accumula nel tempo.
- **RTK (Real Time Kinematic):** Un sistema GNSS differenziale che analizza la fase dell'onda portante. Offre altissima precisione in spazi aperti, ma è vulnerabile ai blocchi del segnale (es. tunnel) e al fenomeno del *multi-path* (riflessioni del segnale sui palazzi).
- **LiDAR:** Eccellente se l'ambiente è ricco di feature geometriche 3D uniche, ma può fallire in condizioni di meteo avverso o in ambienti privi di texture.
- **Radar:** Misura la distanza basandosi sul tempo di volo delle onde radio, offrendo robustezza ma minore risoluzione spaziale.

15.3 Architettura e Generazione delle Mappe HD

Una Mappa HD è organizzata in una struttura a livelli (*Layers*) sovrapposti:

- **Road Layer:** Contiene la geometria stradale di base, la topologia e le intersezioni.
- **Lane Layer:** Definisce i modelli precisi delle corsie, il tipo di linee, le larghezze e le regole di connessione.
- **Traffic/Localization Layer:** Include segnaletica verticale, semafori e *landmark* visivi utili alla localizzazione.
- **Dynamic Data:** Un livello transitorio per informazioni in tempo reale come traffico o congestioni.

Esistono diversi formati standard e proprietari, tra cui **OpenDRIVE** (per reti stradali logiche), **Lanelet2** (modulare e basato su XML, popolare in ROS) e **Apollo HD Map** (usato da Baidu). I principali provider commerciali includono HERE, TomTom, NavInfo, Google e Mobileye (con il formato *RoadBook*).

15.3.1 Pipeline di Generazione: Offline vs Online

La **Pipeline Offline Classica** è un processo laborioso che inizia con la *Data Collection* tramite veicoli equipaggiati con sensori ad alta fedeltà. Segue la *Map Production*, dove le nuvole di punti (*Point Cloud*) di passaggi multipli vengono allineate per correggere errori di deriva e creare una mappa coerente. Infine, avviene l'estrazione delle feature (*Labeling*), spesso manuale o semi-automatica. Il problema principale di questo approccio è il costo e la difficoltà nel mantenere la mappa aggiornata rispetto ai cambiamenti stradali (lavori, incidenti).

Per questo motivo, si stanno affermando **Approcci Moderni e Online**:

- **Crowdsourcing:** Sfrutta flotte di veicoli commerciali dotati di sensori (come nell'approccio Mobileye REM o Tesla) per inviare aggiornamenti rapidi al cloud.
- **MapTR / MapTRv2:** Framework innovativi che costruiscono mappe HD vettoriali **online** (in tempo reale a bordo veicolo). Utilizzano architetture basate su

Transformer per convertire i dati dei sensori in una rappresentazione *Bird's Eye View* (BEV) vettoriale (linee, strisce pedonali) in modalità *end-to-end*.

- **LLM-based Generation:** L'uso di *Large Language Models* (come GPT-4) per interpretare dati grezzi o manuali stradali e generare automaticamente il codice o le strutture dati della mappa.

15.4 Dataset e Trend Futuri

Lo sviluppo di queste tecnologie si appoggia su dataset massivi come **NuScenes** (Boston/Singapore), **Argoverse 2** (mappe HD di 6 città USA) e **OpenLane-V2**. Il trend futuro vede un passaggio da mappe statiche pre-costruite a moduli di "Mapping" integrati nelle reti neurali del veicolo, addestrati congiuntamente (*jointly trained*) con la percezione e la pianificazione, verso una guida autonoma veramente *End-to-End*.

16 Planning e Controllo (Modulo 9)

La navigazione autonoma classica è strutturata in un'architettura gerarchica rigorosa, composta da moduli specializzati che operano su orizzonti spaziali e temporali differenti, ciascuno con una propria frequenza di aggiornamento specifica.

Modulo	**Funzione**	**Orizzonte**	**Frequenza**
Global Planner	Route/Mission Planning	Chilometri	Start/Rerouting
Behavioral Planner	Gestione Manovre	~ 100 m	3 Hz
Local Planner	Generazione Traiettoria	~ 70 m	5 Hz
Control	Esecuzione (Lat/Long)	< 70 m	50 Hz

Tabella 2: Specifiche dei moduli di pianificazione.

16.1 Global Planning (Route Planning)

Il Global Planner agisce come lo "schedulatore" del viaggio, simile al navigatore GPS tradizionale. Dato un grafo stradale $G = (V, E)$ (*Road Network*) composto da segmenti (archi) e intersezioni (nodi), il suo compito è risolvere il problema del percorso minimo (*Shortest Path*) dal punto di partenza S al target T. Gli algoritmi principali sono basati sulla teoria dei grafi:

- **Dijkstra (1959):** Trova il percorso matematicamente ottimo esplorando i nodi in ampiezza. Il suo principale svantaggio è la lentezza computazionale quando applicato ad aree vaste con milioni di nodi.
- **A* (1968):** È un'evoluzione di Dijkstra che introduce una funzione **euristica** per guidare la ricerca verso la destinazione, riducendo drasticamente il numero di nodi esplorati. La sfida risiede nel definire un'euristica ammissibile ed efficiente per il problema specifico.

16.2 Behavioral Planning

Questo livello intermedio traduce il percorso globale in decisioni tattiche per gestire il traffico locale. Il modulo costruisce soluzioni basandosi su tre obiettivi concorrenti: la

Valutazione del Rischio (evitare collisioni statiche e dinamiche), l'**Ottimizzazione** (minimizzare tempo e consumi, massimizzare il comfort dei passeggeri) e la **Soddisfazione dei Vincoli** (limiti di velocità, accelerazione, regole del codice della strada). Gli input fondamentali includono la geometria stradale, le regole del traffico, la "Driving Ethics" (es. cortesia verso i pedoni) e il feedback sulla fattibilità dal planner locale.

Le decisioni sono rappresentate come una scelta tra stati discreti, o **Manovre**: *Stop*, *Lane Keeping*, *Car Following* (ACC), *Lane Change*, *Overtaking* (sorpasso), *Merging* (immissione), *Avoidance* ed *Emergency Braking*. Gli algoritmi decisionali spaziano dai metodi **Rule-Based** (come le Macchine a Stati Finiti - FSM), ai metodi di ottimizzazione (Teoria dei Giochi), fino ai recenti approcci di Intelligenza Artificiale (*Learning Based*). Un rischio comune è che la manovra scelta (es. "sorpassa ora") possa rivelarsi infattibile cinematicamente per il livello successivo.

16.3 Trajectory Planning (Local Planner)

Una volta definita la manovra tattica, il Trajectory Planner deve calcolare la curva geometrica esatta (x, y, t) che il veicolo seguirà. Il problema è formulato matematicamente come un **Problema di Controllo Ottimo (OCP)**, dove si cerca di minimizzare una funzione di costo J soggetta a vincoli differenziali e algebrici:

$$\min_{\mathbf{x}(\cdot),\mathbf{u}(\cdot)} \int_0^{t_f} J(\mathbf{x}(\tau), \mathbf{u}(\tau))d\tau + m(\mathbf{x}(t_f)) \tag{5}$$

I vincoli includono la dinamica del sistema $\dot{\mathbf{x}}(t) = f(\mathbf{x}(t), \mathbf{u}(t))$, le condizioni iniziali e i vincoli di disuguaglianza $h(\mathbf{x}, \mathbf{u}) \leq 0$ (che rappresentano gli ostacoli e i limiti fisici degli attuatori). Per risolverlo, il problema continuo viene discretizzato e affidato a solver numerici (es. IPOPT, SNOPT) che utilizzano tecniche come la Programmazione Quadratica Sequenziale (SQP).

16.3.1 Modelli del Veicolo

Per pianificare una traiettoria fattibile, è necessario un modello matematico del veicolo:

- **Simple Bicycle Model (Inertial Frame):** Modello cinematico parametrizzato nel tempo (x, y, θ, v). Gli input sono accelerazione e curvatura dello sterzo. È semplice ma ignora le forze.

$$\dot{x} = v\sin(\theta), \quad \dot{y} = v\cos(\theta), \quad \dot{\theta} = vk, \quad \dot{v} = a$$

- **Simple Bicycle Model (Curvilinear Frame):** Parametrizzato nello spazio curvilineo s (lungo la strada), utile per il *Lane Keeping*. Le variabili di stato includono l'errore laterale w e l'errore di orientamento μ.

$$\dot{s} = \frac{v\cos\mu}{1 - w\bar{\kappa}_{cl}(s)}$$

  dove $\bar{\kappa}_{cl}$ è la curvatura della linea centrale della corsia.

- **Complex Bicycle Model:** Un modello dinamico che considera massa m, inerzia I_z, forze sugli pneumatici e *slip angle*. È indispensabile per pianificare manovre ad alta velocità o in condizioni di bassa aderenza.

16.4 Controllo

Il modulo di controllo "chiude il loop" ad alta frequenza (50Hz+), garantendo che il veicolo segua la traiettoria pianificata minimizzando l'errore istantaneo.

Per il **Controllo Longitudinale** (velocità) si usano spesso semplici controllori **PID** per regolare acceleratore e freno, oppure tecniche **MPC** per ottimizzare anche i consumi.

Per il **Controllo Laterale** (sterzo), le tecniche geometriche classiche includono:

- **Pure Pursuit:** Calcola l'angolo di sterzo necessario per raggiungere un punto target posto a una certa *Look-ahead distance* (l_d) sulla traiettoria. Simula il comportamento umano di "guardare avanti".

$$\delta(t) = \psi(t) + \arctan\left(\frac{ke(t)}{v_{ego}}\right)$$

- **Stanley Controller:** Calcola lo sterzo basandosi sull'asse anteriore, correggendo separatamente l'errore di direzione (*heading error*) e l'errore di posizione laterale (*cross-track error*).

La tecnica più avanzata è il **Model Predictive Control (MPC)**. È un controllo a "orizzonte recessivo": ad ogni passo temporale, il sistema simula il futuro, risolve un problema di ottimizzazione per l'intero orizzonte predittivo, applica solo la **prima** azione di controllo calcolata e ripete l'intero processo al passo successivo.

16.5 Approcci End-to-End

In contrapposizione all'architettura modulare, l'approccio **End-to-End** utilizza una singola rete neurale profonda per mappare direttamente i dati grezzi dei sensori (Camera, LiDAR) ai comandi di attuazione (sterzo, gas), saltando i passaggi intermedi. Le metodologie di addestramento principali sono:

- **Imitation Learning (IL):** Apprendimento supervisionato dove la rete impara a clonare il comportamento di un guidatore esperto umano. È potente per imparare manovre complesse, ma soffre se si trova in situazioni rare mai viste nel training set.
- **Reinforcement Learning (RL):** Un agente autonomo impara per tentativi ed errori (*trial and error*) all'interno di un simulatore, cercando di massimizzare una funzione di ricompensa (*Reward*) definita dal progettista.

17 Deep Planning: Approcci End-to-End (Modulo 10)

Mentre l'architettura classica si basa su moduli ingegnerizzati manualmente, l'Intelligenza Artificiale ha introdotto nuovi paradigmi per la strutturazione della pipeline di guida autonoma.

17.1 Architetture di Navigazione

Esistono tre paradigmi fondamentali:

1. **Architettura Modulare (Classica):** Il sistema è diviso in blocchi sequenziali distinti: Percezione (elabora sensori), Predizione/Localizzazione (stima posa e intenzioni altrui), Planning (decide la manovra e calcola la traiettoria) e Controllo (gestisce gli attuatori). Ogni blocco ha input e output ben definiti e interpretabili.
2. **Architettura End-to-End:** Una singola rete neurale profonda mappa direttamente l'input grezzo dei sensori (immagini, nuvole di punti) agli output di controllo (sterzo, freno, acceleratore), eliminando completamente i passaggi intermedi definiti dall'uomo.
    - *Vantaggi:* Semplifica drasticamente il codice, generalizza meglio a situazioni complesse non codificabili a mano e offre performance real-time.
    - *Svantaggi:* È una "black box" non interpretabile (difficile capire *perché* l'auto ha frenato), complessa da validare e debuggare, sollevando seri dubbi sulla sicurezza.
3. **Architettura Ibrida:** Un compromesso in cui la percezione alimenta un modulo di *Data-driven Path Planning* (basato su AI), il quale genera una traiettoria che viene poi eseguita da un controllore classico robusto.

17.2 Imitation Learning (IL)

Nell'Imitation Learning, l'obiettivo è apprendere una *policy* di guida π_θ imitando il comportamento registrato di un "esperto" umano (rappresentato dalle traiettorie τ_i).

Il metodo più semplice è il **Behavioral Cloning (BC)**, un approccio passivo dove si addestra un classificatore a copiare l'azione dell'esperto dato uno stato. Tuttavia, soffre di un limite critico: se il dataset è parziale, l'agente non sa come comportarsi in stati mai visitati dall'esperto (es. se l'auto devia leggermente dalla corsia, l'esperto non ha mai corretto quell'errore perché non lo ha mai commesso). Questo fenomeno è noto come *distribution shift*.

Per risolverlo, si usano metodi attivi come il **Direct Policy Learning (DPL)**, in particolare l'algoritmo **DAgger (Dataset Aggregation)**. In DAgger, l'agente guida e raccoglie traiettorie; successivamente, un esperto umano (o un oracolo) etichetta retroattivamente cosa *avrebbe dovuto fare* in quelle situazioni. Questi nuovi dati vengono aggregati al dataset originale per ri-addestrare la policy, insegnando all'agente come recuperare dai propri errori. Una variante è il **DAgger by Coaching**, dove un "coach" dimostra policy inizialmente semplici e sub-ottimali, aumentando progressivamente la difficoltà per guidare l'apprendimento.

Un approccio alternativo è l'**Inverse Reinforcement Learning (IRL)**: invece di copiare ciecamente le azioni, l'algoritmo cerca di dedurre la **Funzione di Ricompensa (Reward Function)** implicita che l'esperto sta cercando di massimizzare. Una volta nota la ricompensa, si può calcolare la policy ottima. I metodi principali includono il *Max-Margin* (massimizza la differenza di valore tra la scelta esperta e le altre) e la *Maximum Entropy* (che favorisce l'esplorazione in spazi continui).

17.3 Reinforcement Learning (RL)

Nel Reinforcement Learning, non ci sono dati etichettati. L'agente impara per tentativi ed errori (*trial and error*) interagendo con l'ambiente per massimizzare una ricompensa cumulativa. Il problema è formalizzato come un **Processo Decisionale di Markov (MDP)** definito dalla tupla $(\mathbb{S}, \mathbb{A}, T, R, \gamma)$, dove γ è il fattore di sconto che pesa le ricompense future rispetto a quelle immediate.

17.3.1 Metodi Value-based

Questi algoritmi stimano il valore $Q(s, a)$ di eseguire una certa azione in un certo stato. Il più famoso è il **Q-Learning**, che aggiorna iterativamente la tabella dei valori Q. Per spazi di stato complessi (immagini), si usa il **Deep Q-Learning (DQN)**, dove una rete neurale approssima la funzione Q. Per garantire la stabilità dell'apprendimento, DQN utilizza due tecniche chiave: l'*Experience Replay* (memorizza e riutilizza esperienze passate per rompere le correlazioni temporali) e una *Target Network* separata.

17.3.2 Metodi Policy-based

Adatti per spazi di azione continui (come l'angolo di sterzo), ottimizzano direttamente la policy.

- **DDPG (Deep Deterministic Policy Gradient):** Un'architettura *Actor-Critic* che opera *Off-Policy* (impara anche da vecchie esperienze). L'"Attore" propone un'azione deterministica, mentre il "Critico" ne valuta la bontà stimando il Q-value.
- **PPO (Proximal Policy Optimization):** Un metodo *On-Policy* molto popolare per la sua stabilità. Utilizza un meccanismo di *clipping* nella funzione obiettivo per impedire che l'aggiornamento della policy sia troppo drastico, evitando il collasso delle performance.

17.4 Simulazione e Sim2Real Gap

L'addestramento RL richiede milioni di interazioni, rendendo impossibile l'uso di veicoli reali per motivi di sicurezza e tempo. Si ricorre quindi a simulatori (es. CARLA, NuPlan). Tuttavia, esiste il problema del **Sim2Real Gap**: un modello che guida perfettamente in simulazione spesso fallisce nel mondo reale. Le cause sono le semplificazioni della fisica, la mancanza di rumore sensoriale realistico e la scarsa variabilità degli scenari virtuali (*corner cases* mancanti). Le soluzioni per colmare questo divario includono:

- **Domain Randomization:** Durante il training, si randomizzano texture, luci e parametri fisici per costringere la rete a imparare feature robuste e invarianti.
- **Realistic Simulations:** Migliorare la fedeltà dei sensori (es. simulando pioggia, nebbia e riflessi radar).
- **Transfer Learning:** Pre-addestrare in simulazione e fare un *fine-tuning* finale con pochi dati reali.

18 Il Sistema di Elaborazione: Hardware (Modulo 12)

Questo modulo analizza l'hardware sottostante agli algoritmi di guida autonoma. Il mercato dei semiconduttori per l'automotive è in una fase di rapida espansione, trainato dalla domanda di veicoli con livello di automazione L2 o superiore. Nel 2019 i veicoli L2+ rappresentavano il 23% dei ricavi del settore chip; si stima che entro il 2030 questa quota salirà al 93%. I requisiti chiave per i nuovi processori includono la gestione di una percezione a 360°, ridondanza critica, potenza per l'IA, efficienza energetica e conformità agli standard di *Functional Safety*.

18.1 Evoluzione delle Architetture Elettroniche

Si sta verificando un cambio di paradigma nell'architettura dei veicoli:

1. **Processing at Edge (Passato):** Ogni sensore "intelligente" possedeva il proprio chip di elaborazione interno (es. una telecamera che invia solo la lista degli oggetti rilevati). Questo approccio era sufficiente per ADAS semplici ma limitante per la fusione dati.
2. **Central Domain Controller (Futuro):** Un singolo super-computer centrale riceve i dati grezzi da tutti i sensori. Questa centralizzazione è indispensabile quando il numero di telecamere supera la decina (più LiDAR e Radar) per gestire manovre complesse e sensor fusion di alto livello.

18.2 Calcolo Eterogeneo (Heterogeneous Computing)

I moderni SoC (*System on Chip*) non si affidano più a un solo tipo di processore, ma integrano core specializzati per massimizzare l'efficienza. Un tipico chip automotive include:

- **CPU Generica:** Spesso basata su architettura ARM Cortex in configurazione *Lock-Step* (es. due core che eseguono lo stesso codice e si controllano a vicenda) per garantire tolleranza ai guasti (ASIL-D). Gestisce il sistema operativo e la logica sequenziale.
- **GPU (Graphics Processing Unit):** Offre parallelismo massiccio per algoritmi generali.
- **Acceleratori AI (NPU/DLA):** Hardware dedicato esclusivamente all'inferenza di reti neurali (moltiplicazioni matriciali).
- **Processori Specializzati:** Unità dedicate a compiti specifici come l'**ISP** (Image Signal Processor) per la pulizia delle immagini, encoder video e motori per la stereovisione.

18.3 Analisi dei Competitor

18.3.1 1. Ambarella (CV3-AD)

Ambarella si posiziona come leader nell'efficienza energetica per i controller di dominio. Il chip CV3-AD offre una potenza di calcolo equivalente a 500 eTOPS consumando solo **50 Watt**, grazie a un processo produttivo a 5nm. Il cuore del sistema è l'architettura **CVflow**, un co-processore che esegue algoritmi rappresentati come **DAG** (Grafi Aciclici

Diretti). In questo modello, i nodi sono le operazioni matematiche e gli archi sono i flussi di dati (tensori); l'hardware ottimizza il passaggio dei dati minimizzando l'accesso alla memoria DRAM esterna (che è il principale collo di bottiglia energetico) e sfruttando la memoria on-chip.

Componenti distintivi includono:

- **NVP (Neural Vector Processor):** Un acceleratore AI che supporta nativamente la **Sparsità Casuale** (vedi sotto) e dati a bassa precisione (4-8 bit).
- **ISP Avanzato:** Gestisce HDR e riduzione del rumore direttamente in hardware.
- **Stereo/Optical Flow Unit:** Un motore dedicato per calcolare mappe di disparità dense e flusso ottico, liberando la CPU/GPU da questo carico oneroso.

18.3.2 2. Nvidia (Famiglia Orin AGX)

Nvidia offre una piattaforma scalabile ad altissime prestazioni ma con consumi tendenzialmente maggiori. Il modello di punta **AGX Orin** eroga fino a 275 TOPS (INT8). L'architettura comprende:

- **CPU:** 12 core ARM Cortex-A78AE affiancati da una *Safety Island* (ARM R52) per garantire determinismo e sicurezza.
- **GPU Ampere:** Dotata di 2048 CUDA cores e 64 **Tensor Cores**, unità programmabili specifiche per l'AI.
- **DLA (Deep Learning Accelerator):** Un motore a funzione fissa per l'inferenza di CNN standard (convoluzioni, pooling), che risulta 3-5 volte più efficiente in termini di Watt/Ops rispetto alla GPU.
- **PVA (Programmable Vision Accelerator):** Un processore VLIW (*Very Long Instruction Word*) a 7 vie, capace di eseguire istruzioni multiple in parallelo in un singolo ciclo di clock. È usato per task di visione classica come feature tracking e FFT.

18.3.3 3. Mobileye (EyeQ Ultra)

Mobileye propone soluzioni "chiavi in mano" spesso percepite come *black box*, ma estremamente diffuse. Il chip **EyeQ Ultra** (L4) è basato sull'architettura open-source **RISC-V**. Caratteristiche peculiari:

- **CGRA (Coarse Grained Reconfigurable Architecture):** Un array di unità funzionali interconnesse a maglia, simile a una FPGA ma meno granulare, che offre un ottimo bilanciamento tra flessibilità e prestazioni.
- **Barrel-threaded CPU:** Una CPU progettata per il determinismo temporale, che cambia il thread di esecuzione a ogni ciclo di clock per garantire che nessun processo blocchi gli altri.

18.4 Concetti Tecnici Avanzati

18.4.1 Sparsity (Sparsità)

Le reti neurali sono tradizionalmente "dense" (tutti i neuroni sono collegati), ma tramite tecniche di *Pruning* (potatura) è possibile impostare molti pesi a zero senza perdere accuratezza, riducendo calcoli e memoria di fattori 10-100x. I due approcci hardware principali sono:

- **Random Sparsity (Ambarella):** L'hardware supporta zeri in qualsiasi posizione. È molto flessibile ma richiede circuiti complessi per saltare i calcoli inutili.
- **Fine-Grained Structured Sparsity (Nvidia):** Impone un vincolo strutturale "2:4" (in ogni blocco di 4 pesi, almeno 2 devono essere zero). È più rigido e può impattare leggermente l'accuratezza, ma si adatta meglio all'architettura parallela delle GPU.

18.4.2 Scheduling: Batch vs Stream

La gestione del flusso dati differenzia radicalmente le architetture. Il **Modello GPU** classico preferisce il *Batch Processing*: accumula più immagini insieme per saturare i core paralleli. Questo è efficiente per il training offline, ma in auto aumenta la latenza. Il **Modello CVflow** (Ambarella) utilizza lo *Stream Processing*: divide le reti neurali in piccoli pezzi (*chunks*) che scorrono nell'hardware attraverso buffer circolari. Questo permette di mantenere l'utilizzo dell'hardware alto (70-80%) anche processando una singola immagine alla volta, garantendo la bassissima latenza richiesta dalla guida autonoma.

È importante notare che i **TOPS** (Tera Operations Per Second) dichiarati dai produttori sono spesso teorici. Ad esempio, benché il chip Nvidia Orin abbia sulla carta 8.5 volte i TOPS del predecessore Xavier, nei benchmark reali (FPS applicativi) il miglioramento effettivo misurato è di circa 3.3 volte.

19 La Pipeline di Validazione: Dal Virtuale al Reale (Modulo 13)

Lo sviluppo di veicoli autonomi non può prescindere da una rigorosa pipeline di validazione. La prassi standard adottata dalla maggior parte delle aziende inizia con la raccolta di dati dal mondo reale per costruire ambienti di simulazione fedeli. Lo scopo ultimo è validare il software e l'hardware in termini di sicurezza e funzionalità **prima** del dispiegamento su strada pubblica.

Il processo di sviluppo segue il cosiddetto **V-Model** (Modello a V), uno standard nell'industria automotive guidato dalla normativa **ISO 26262** per la sicurezza funzionale. Questo modello collega ogni fase di progettazione (lato sinistro della V) alla corrispondente fase di test (lato destro della V), garantendo che ogni requisito sia verificato al giusto livello di astrazione.

19.1 Operational Design Domain (ODD)

Un concetto cardine per la sicurezza e la simulazione è l'**ODD** (*Operational Design Domain*). L'ODD descrive le specifiche condizioni operative entro le quali il veicolo autonomo è progettato per funzionare correttamente. Le aziende utilizzano l'ODD per delimitare

il perimetro di sicurezza e incrementare iterativamente le capacità del veicolo. Un ODD tipico è definito da:

- **Geografia:** Aree urbane specifiche, autostrade o zone geofenced.
- **Tipologia di strada:** Presenza di marciapiedi, numero di corsie, tipo di asfalto.
- **Condizioni Ambientali:** Meteo (pioggia, nebbia, sole), orario (giorno/notte).
- **Vincoli Dinamici:** Range di velocità ammissibile e rispetto delle leggi del traffico locali.

19.1.1 Case Studies: La definizione dell'ODD

Le strategie per definire l'ODD variano tra i player:

- **Waymo:** Definisce il proprio dominio basandosi rigorosamente su geografia, condizioni meteo e legislazione statale.
- **Uber:** Ha formalizzato un processo di caratterizzazione in quattro fasi:
  1. **Guida Manuale:** Raccolta di log dettagliati guidando nell'area target.
  2. **Tagging:** Etichettatura dei dati raccolti (es. geometria stradale, attori, regole).
  3. **Sintesi:** Identificazione degli scenari critici e dei comportamenti richiesti.
  4. **Test:** Creazione di test rappresentativi in simulazione e su pista fisica.

19.2 Tassonomia "X-in-the-Loop"

La validazione avviene applicando tecnologie di simulazione a livelli crescenti di integrazione hardware.

19.2.1 MIL (Model In the Loop)

Rappresenta la fase iniziale di sviluppo, spesso definita *Model-based design*. In questo stadio, il sistema non è ancora scritto in codice definitivo, ma è rappresentato da **modelli logici** (es. diagrammi a blocchi in MATLAB/Simulink, PLECS o Dymola). Il modello del veicolo comunica con un modello dell'ambiente in un loop chiuso puramente matematico. È il metodo più economico e veloce per verificare la correttezza algoritmica di base.

19.2.2 SIL (Software In the Loop)

Qui si testa il software reale (in C++ o Python) che verrà poi caricato sul veicolo, ma facendolo girare su piattaforme commerciali standard (PC o Server). L'obiettivo è incorporare lo stack software di produzione nella simulazione.

- **Approccio NVIDIA:** Il simulatore *DRIVE Sim* alimenta un server che ospita l'intero stack AV (lo stesso destinato al computer di bordo *DRIVE AGX*). Il software processa i sensori simulati e restituisce i comandi di controllo.
- **Approccio VisLab:** Si utilizzano simulatori di terze parti (come *dSPACE ASM*) che comunicano via protocolli standard (Ethernet/PCIe) con lo stack software proprietario (*CVPlayer* e *SuperDAG*), il quale gira su hardware generico.

19.2.3 AccSIL / PIL (Accelerated SIL / Processor In the Loop)

Si tratta di una variante del SIL in cui una parte specifica dell'hardware target viene inserita nel loop. Nel caso di **VisLab**, si parla di *Accelerated SIL* quando il software sfrutta il chip reale (es. l'acceleratore neurale del **CV3/AD685**) per eseguire i calcoli pesanti (come l'inferenza delle reti neurali), mentre la logica di controllo e la simulazione rimangono su PC. Questo permette di validare le performance dell'acceleratore hardware senza bisogno dell'intera centralina.

19.2.4 HIL (Hardware In the Loop)

In questa fase, il software gira sull'**hardware finale** (la centralina ECU reale), che è connessa fisicamente a un simulatore in tempo reale. Il simulatore "inganna" la centralina iniettando segnali elettrici sui pin di input che simulano i dati dei sensori (camere, radar, bus CAN). La ECU elabora i dati e invia comandi agli attuatori, che vengono letti dal simulatore per aggiornare lo stato del veicolo virtuale. Nell'integrazione **NVIDIA**, ad esempio, il sistema *DRIVE Sim* (che gestisce fisica e rendering) comunica a bassissima latenza con l'hardware *DRIVE AV* (che esegue Percezione, Planning e Controllo) chiudendo il loop.

19.3 Validazione Fisica Avanzata

19.3.1 High-Fidelity Simulation (VIL - Vehicle In the Loop)

Il VIL rappresenta il punto d'incontro tra virtuale e reale. Utilizza simulatori ad alta fedeltà accoppiati a banchi prova dinamici multi-asse. L'intero veicolo (o il telaio nudo) viene posizionato su questi banchi (rulli o piattaforme mobili) per mimare le sollecitazioni fisiche della strada, sincronizzando perfettamente la dinamica meccanica del veicolo con la visualizzazione grafica che stimola i sensori. Laboratori avanzati come quelli di **BMW** e **GM** utilizzano estensivamente questa tecnica.

19.3.2 Closed-Course Testing

È il passo intermedio obbligatorio prima della strada pubblica. Si svolge in aree private o piste di prova dedicate (come la città simulata "Castle" di Waymo). Qui si possono testare scenari limite o pericolosi (es. pedoni che sbucano all'improvviso, usando manichini) in totale sicurezza, permettendo agli ingegneri di rifinire il software in un ambiente controllato ma fisicamente reale.

20 Panoramica dei Simulatori (Modulo 14)

La simulazione è diventata uno strumento critico e imprescindibile per lo sviluppo di sistemi ADAS e di Guida Autonoma (AV). Il panorama attuale offre diverse soluzioni che variano per modello di licenza (Open Source vs Commerciali), fedeltà fisica e target di utilizzo (Ricerca Accademica vs Ingegnerizzazione Industriale).

20.1 Simulatori Open-Source

20.1.1 CARLA (Car Learning to Act)

CARLA rappresenta lo standard *de facto* in ambito accademico. Il suo obiettivo primario è la "democratizzazione" della ricerca sulla guida autonoma, offrendo uno strumento accessibile, gratuito e altamente personalizzabile. Dal punto di vista tecnico, il motore grafico si basa su **Unreal Engine (versioni 4 e 5.5)**, sfruttando tecnologie avanzate come *Lumen* e *Nanite* per un rendering ad alta fedeltà. Per la fisica, combina un motore proprietario con l'integrazione di *Project Chrono*, garantendo una simulazione accurata della dinamica del veicolo. CARLA include un *Traffic Manager* nativo ma supporta anche l'integrazione con tool esterni come SUMO e InvertedAI. La gestione degli scenari complessi avviene tramite il modulo *ScenarioRunner*, compatibile con lo standard **OpenSCENARIO**, mentre le reti stradali sono definite tramite **OpenDRIVE 1.4**. Le API sono disponibili sia in Python (per scripting rapido) che in C++ (per massime performance).

20.2 Simulatori Industriali e Commerciali

20.2.1 rFpro

Nato nel 2007 con un focus sul motorsport, rFpro è considerato un ambiente di grado ingegneristico (*Engineering-grade*), utilizzato dai maggiori costruttori auto e team di Formula 1. Si distingue per un motore di rendering customizzato con Ray Tracing avanzato, ottimizzato per la velocità e la bassa latenza. A differenza di CARLA, rFpro non possiede un motore fisico interno, ma è progettato come un ambiente agnostico capace di interfacciarsi con i modelli dinamici leader di mercato come **CarMaker**, **CarSim** e *VI-CarRealTime*. Utilizza "Gemelli Digitali" (*Digital Twins*) di strade e circuiti reali in formato OpenDRIVE e supporta workflow completi SIL e HIL.

20.2.2 Applied Intuition

Azienda giovane (fondata nel 2019) ma in rapidissima ascesa, offre una soluzione modulare che copre l'intero ciclo di vita, dalla simulazione alla validazione. Il motore grafico combina Unreal Engine 5 con un modulo proprietario ("Neural Sim") specifico per la simulazione realistica dei sensori. Il suo punto di forza è la sicurezza: il motore fisico per la dinamica del veicolo è certificato **ISO 26262** per sistemi ASIL-D. L'approccio è fortemente orientato al cloud e alla scalabilità, con un'interfaccia grafica (GUI) che permette la creazione di scenari tramite linguaggio naturale.

20.2.3 dSPACE e VTDx

dSPACE, storica azienda tedesca leader nei sistemi HIL, offre una suite basata sul modello **ASM** (*Automotive Simulation Model*), anch'esso certificato ISO 26262. Utilizza Unreal Engine 5 per la visualizzazione e supporta nativamente la generazione procedurale di scenari e l'ingestione di dati reali. **VTDx (Virtual Test Drive)**, sviluppato da Hexagon, è una piattaforma ad alta fedeltà che supporta l'intero spettro di test (MiL, SiL, HiL, DiL). Si caratterizza per il supporto avanzato alla semantica stradale (**OpenDRIVE 1.8**) e la gestione di ambienti multilivello complessi, come i parcheggi multipiano.

20.3 Piattaforme Cloud-Native e AI

20.3.1 NVIDIA Omniverse

NVIDIA propone una piattaforma "cloud-native" focalizzata sulla generazione di dati sintetici e sull'addestramento dell'IA. Basata su Unreal Engine 5, punta al fotorealismo estremo. Il motore fisico integra la simulazione classica con la ricostruzione neurale (**NuRec**) per modellare fedelmente le interazioni sensore-ambiente. Una caratteristica unica è il *Traffic Manager* (**Cosmos**), un sistema guidato dall'IA che utilizza "Foundation Models" per generare automaticamente scenari di traffico realistici e variati, superando i limiti degli script manuali. Utilizza lo standard **OpenUSD** per la rappresentazione della scena, pur non supportando nativamente OpenDRIVE (per il quale si integra con CARLA).

20.4 Criteri di Scelta e Confronto

La selezione del simulatore ideale dipende da una matrice di requisiti:

1. **Obiettivo:** Se serve generare dataset per il training visivo, la priorità è il fotorealismo (Omniverse/Applied Intuition). Se serve validare il controllo dinamico, la priorità è la fedeltà fisica (rFpro/dSPACE).
2. **Performance:** Il test HIL richiede rigoroso real-time, mentre la generazione di dati sintetici può avvenire offline.
3. **Scalabilità:** La capacità di parallelizzare migliaia di istanze nel Cloud è cruciale per la validazione statistica.
4. **Ecosistema:** Supporto agli standard aperti (OpenDRIVE, OpenSCENARIO) e integrazione con tool esistenti (ROS, MATLAB).

Simulatore	**Motore Grafico**	**Motore Fisico**	**APIs**
NVIDIA Omniverse	UE5	Proprietario	Python, C++
CARLA	UE 5.5	Proprietario + Chrono	Python, C++, ROS
rFpro	Custom	Esterno (CarSim, etc.)	C++, MATLAB
Applied Intuition	UE5 (Custom)	Built-in + Esterno	Python, C++
VTDx	UE5	Esterno	Python, C++, MATLAB
dSPACE	UE5	Built-in (ASM)	MATLAB, C++

Tabella 3: Confronto sintetico delle architetture di simulazione.