Politecnico di Torino

Corso di Laurea Magistrale in Ingegneria Meccanica

Algoritmo di guida autonoma per rover elettrico in campo agricolo-industria 4.0

Relatore:
Prof. Aurelio Somà

Candidato:
Jordi Rivella

Correlatori:
Ing. Francesco Mocera
Dott. Salvatore Martelli

Anno accademico 2022/2023

Indice

Elenco delle figure

1 Introduzione

Negli ultimi decenni sono avvenuti una serie di progressi tecnologici che hanno il potenziale di cambiare radicalmente il mondo in cui viviamo. Al contempo, sono diventate sempre di maggior rilievo alcune tematiche come il cambiamento climatico, la sostenibilità e il benessere dei lavoratori. Il processo di digitalizzazione e automazione, iniziato già in ambito industriale, è destinato nel prossimo futuro a rivoluzionare il settore agricolo. Lo sviluppo di piattaforme robotiche autonome, in grado di sostituire la manodopera umana in operazioni ripetitive e faticose, può portare numerosi vantaggi anche da un punto di vista economico, di efficienza del processo produttivo e di qualità del prodotto finale. In quest'ottica, in collaborazione con la società Ecothea S.r.l, è stato avviato un progetto per la creazione di un rover elettrico autonomo. L'attività di questa tesi è incentrata sulla tematica relativa alla guida autonoma, che comprende tutte le azioni volte all'orientamento e spostamento del veicolo, senza l'ausilio dell'uomo, in un ambiente complesso, come quello di un terreno agricolo. La prima fase di un algoritmo di guida autonoma è costituita dal global path planning, ossia l'attività di pianificazione dell'intera missione, che prevede la creazione, a partire da una mappa dell'ambiente, di una traiettoria suddivisa in waypoint, da un punto di inizio ad uno di fine. Una volta creato il percorso, è essenziale seguirlo: un algoritmo di path following è necessario per definire le velocità da applicare alle ruote del veicolo, affinchè quest'ultimo segua la traiettoria precedentemente definita. Infine, durante l'effettivo movimento del robot è da tenere in considerazione la possibile presenza di ostacoli non previsti. Si rende quindi necessaria l'implementazione di una logica di obstacle avoidance, che effettua una riprogrammazione della traiettoria nel breve raggio, calcolando una deviazione volta a evitare tali ostacoli in maniera sicura ed efficiente. L'attività della tesi prevede una parte di analisi sullo stato dell'arte degli algoritmi già presenti in letteratura, una parte di confronto e di valutazione sulle scelte che meglio si adattano al progetto e una fase di implementazione e scrittura di codice effettuata in ambiente MATLAB®.

1.1 Contesto socioeconomico

Negli ultimi decenni è iniziata una rivoluzione del settore agricolo, che è destinata ad avere un grande sviluppo negli anni a venire. L'agricoltura ricopre un ruolo fondamentale nell'economia della società moderna. Seppur essa abbia un peso, in termini di profitti, inferiore rispetto al settore secondario e terziario, il suo contributo non è affatto trascurabile. Secondo la FAO, l'organizzazione delle nazioni unite per l'alimentazione e l'agricoltura, nel 2020 il settore dell'agricoltura, pesca e silvicoltura contava circa 900 milioni di lavoratori a livello mondiale, il 27% della forza lavoro totale, ha prodotto 9,4 miliardi di tonnellate di beni primari, e ha generato un valore economico aggiunto pari a 3,5 trilioni di dollari [1]. L'importanza economica di questo settore è in continua crescita, dovendo soddisfare il fabbisogno di una popolazione sempre più numerosa, che secondo le stime [2] nel 2050 raggiungerà i 9,8 miliardi di persone, rispetto ai circa 8 attuali. La capacità produttiva agricola è destinata addirittura a raddoppiare [3], se si considera anche che oggigiorno quasi una persona su otto nel modo vive in condizioni di denutrizione. Oltre alla quantità, il mercato, in particolare nei paesi in via di sviluppo, richiede cibi più sani e perciò di maggiore qualità. La crescita della popolazione sfida quindi gli agricoltori ad apportare cambiamenti nel controllo, monitoraggio e gestione della produzione agricola per soddisfare la crescente domanda di cibo.

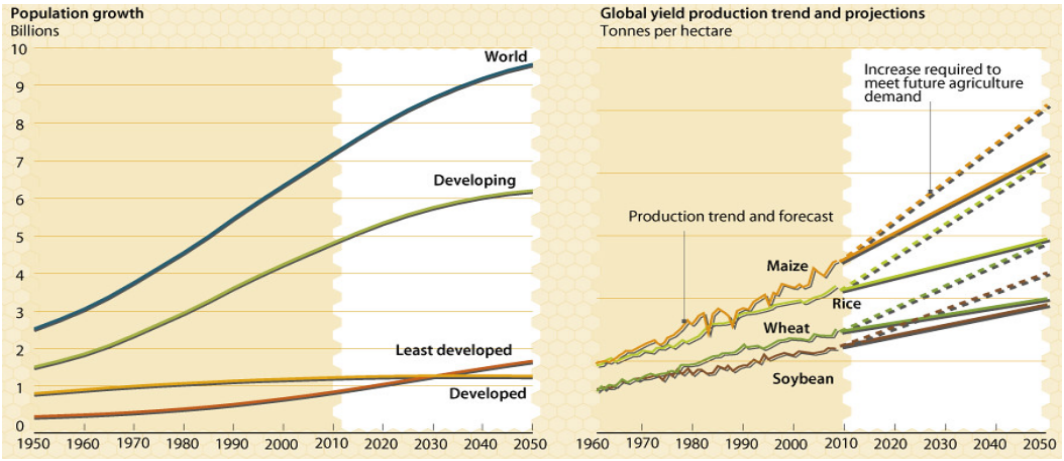

Figura 1: Stime della crescita della popolazione mondiale e del fabbisogno delle principali colture agricole. [2],[3]

Un altro tema che ha un grande impatto sull'agricoltura è quello relativo al cambiamento climatico. Esso minaccia la nostra capacità di garantire la sicurezza alimentare globale, sradicare la povertà e raggiungere uno sviluppo sostenibile. Da un lato il settore primario è responsabile della produzione di una grossa fetta di gas ad effetto serra, perciò sarà sempre più necessario ridurre l'impatto ambientale di tutti i sistemi di produzione, adottando strategie alternative e rivoluzionando i metodi produttivi. Nel 2016, come si vede in figura 2, il 18,4% delle emissioni globali originate dall'attività umana proveniva dai sistemi agroalimentari, includendo la deforestazione, l'allevamento, la gestione del suolo e dei concimi, la perdita e lo spreco di cibo. Dall'altro lato, il riscaldamento globale ha un impatto diretto sui sistemi agroalimentari, a causa di precipitazioni e temperature mutevoli e imprevedibili, di una maggiore incidenza di eventi meteorologici estremi e di disastri come siccità, inondazioni, epidemie e malattie, che rendono sempre più complicata la sopravvivenza di certe tipologie di colture.

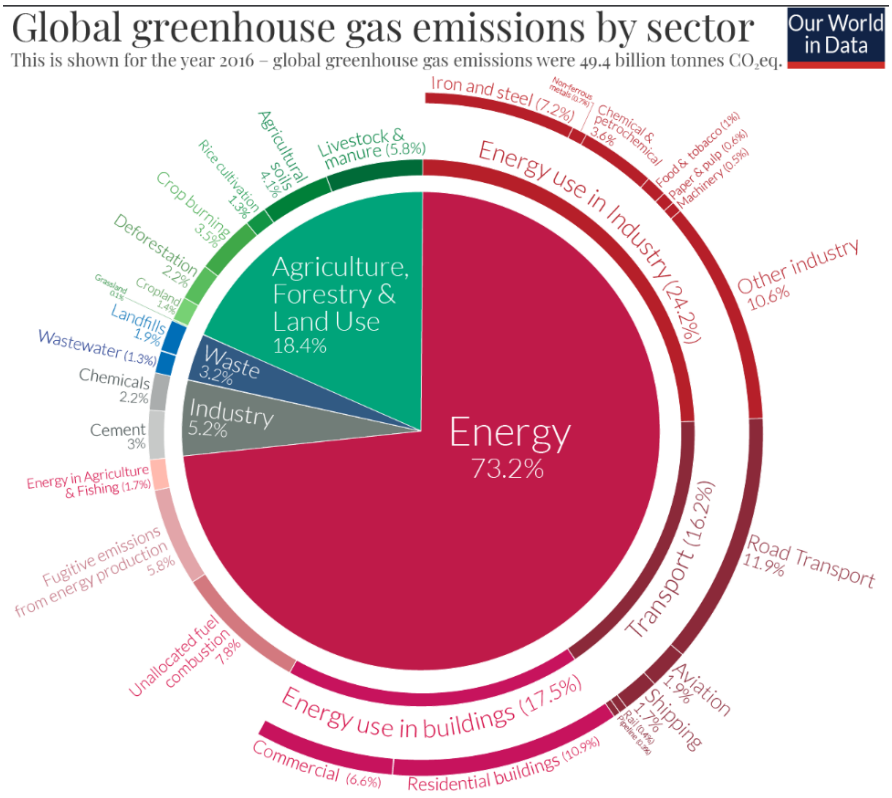

Figura 2: Emissioni globali di gas ad effetto serra divise per settore, dell'anno 2016. [4]

Un altro fattore di fondamentale importanza per il settore agricolo è quello umano. In agricoltura ci sono numerose operazioni sul campo che richiedono molta manodopera e che hanno una certa complessità e ripetitività. Nello scorso secolo, in seguito all'introduzione e diffusione delle macchine agricole, specialmente nei paesi sviluppati, la mole di lavoro e il numero di lavoratori in questo settore è diminuito. Un discorso diverso vale per i paesi in via di sviluppo, come ad esempio quelli del continente africano, che basano ancora gran parte della loro produzione agricola sulla manodopera umana, che quindi costituisce la maggior percentuale della forza lavoro totale. D'altra parte, il processo di urbanizzazione globale sta trasformando i paesaggi rurali in urbani, facendo sì che il 68% della popolazione risiederà in ambienti urbani entro il 2050 [2], con possibili cause di mancanza di manodopera qualificata nelle zone extraurbane, in cui è concentrata l'attività agricola. Si può quindi concludere, facendo osservazioni sia di tipo storico, analizzando l'evoluzione temporale nei secoli, sia di tipo scientifico ed economico, come il progresso e l'innovazione scientifica tendano a diminuire il coinvolgimento della forza lavoro umana nel settore agricolo, che, come detto, richiede molte operazioni faticose e ripetitive.

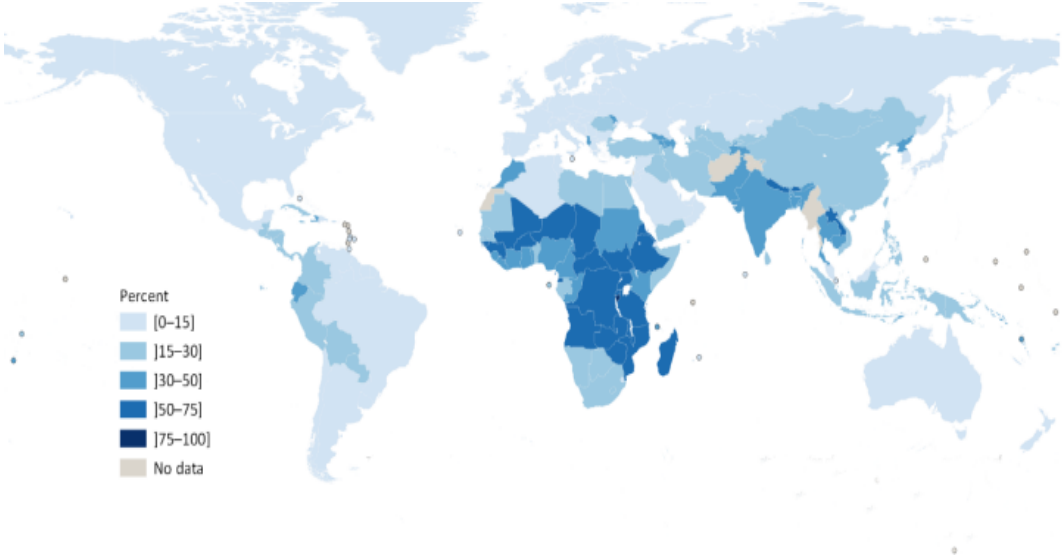

Figura 3: Mappa che esprime la percentuale di lavoratori in agricoltura, pesca e foresteria rispetto alla forza lavoro totale nei vari paesi del mondo. Si nota come nei paesi sviluppati, grazie all'ausilio e alla diffusione della tecnologia, la percentuale sia inferiore rispetto ai paesi in via di sviluppo. [5]

Storicamente, in risposta all'aumento della capacità produttiva del settore agricolo, si è adottata sempre una strategia di intensificazione delle colture, seguendo la convinzione del "più grande è meglio". L'utilizzo di colture intensive su vasta scala, da un lato ha vantaggi pratici per l'uomo, per la sua comodità e la standardizzazione dei metodi di lavorazione, che si avvicina molto alla filosofia del mondo industriale. Dal punto di vista naturale, però, questa tipologia di produzione causa notevoli danni, dovuti allo sfruttamento indiscriminato del suolo, che limita moltissimo la biodiversità. Questa strategia non può essere adottata all'infinito, in quanto la superficie terrestre sfruttabile per scopi agricoli è limitata e in gran parte già utilizzata. È necessario quindi aumentare l'efficienza delle colture, senza trascurare la sostenibilità, e la qualità del prodotto finale. L'efficienza finora è stata migliorata, tra le altre cose, tramite l'utilizzo diffuso di pesticidi e altre sostanze chimiche contro malattie e parassiti, che però secondo la FAO causano ancora oggi la perdita del 20-40% del raccolto globale.

1.2 Prospettiva di sviluppo del settore agricolo

Fortunatamente, per superare le sfide imposte dalla crescita della popolazione, dall'innalzamento della vita media della popolazione, dal cambiamento climatico, da un modello agroalimentare vecchio, inefficiente e poco sostenibile, dall'accelerazione dell'urbanizzazione e dai problemi relativi alla manodopera umana, i progressi scientifici e tecnologici stanno trasformando il modo di gestire le attività agricole, riducendo sempre più l'intervento umano [6]. Nel prossimo futuro, lo sviluppo dell'agricoltura sarà strettamente legato ai progressi avvenuti negli ultimi decenni in un ampio gruppo di campi scientifici, come ad esempio nell'intelligenza artificiale, Internet of Things e nella robotica [7]. L'automazione dei sistemi di produzione e coltura è destinata ad essere la prossima grande rivoluzione del settore agricolo. Le più moderne e innovative scoperte in campo scientifico e tecnologico offrono l'opportunità di sviluppare nuove attrezzature agricole, basate su macchinari intelligenti che permettono la riduzione gli sprechi, migliorano la produttività e l'efficienza, riducono l'impatto ambientale e aumentano la sostenibilità alimentare.

L'intelligenza artificiale può essere utilizzata per numerosi compiti in campo agricolo. Ad esempio, gli algoritmi di computer vision sono utili nella classificazione dei frutti e nel rilevamento delle erbe infestanti in ambienti complessi, ovvero con illuminazione ambientale variabile, sfondi eterogenei, considerando anche la variazione di forme e colori di frutti o erbe. L'Internet of Things, d'altro canto, permette di monitorare le aziende agricole attraverso sensori di diverso tipo (ottici, meccanici, elettrochimici, dielettrici). Le informazioni registrate, provenienti da numerosi elementi interconnessi tra di loro e distribuiti in maniera capillare su tutta l'azienda agricola, permettono una diffusa raccolta di dati. Quest'ultimi, trasmessi ad un calcolatore centrale grazie a tecnologie di comunicazione a breve/lungo raggio, possono essere elaborati tramite specifici algoritmi e costituiscono una fonte preziosa per compiere previsioni, decisioni e in generale per la gestione dell'azienda agricola.

Un aspetto in comune tra le tecnologie sopra citate è che la maggior parte di esse basano il loro funzionamento sull'utilizzo di piattaforme robotiche. La robotica e l'automazione sono già ampiamente utilizzate nel settore della trasformazione dell'industria alimentare, ma non vengono sfruttate nella stessa misura per la produzione sui terreni agricoli. Le tecnologie robotiche hanno un notevole potenziale per ampliare le aree di intervento dei macchinari in campo agricolo, ad esempio avendo la possibilità di muoversi su terreni umidi, di lavorare di notte e di compiere azioni ripetitive mantenendo sempre la stessa efficienza. I dati raccolti dalle piattaforme robotiche sul campo possono fornire una grande quantità di informazioni sullo stato di sementi, bestiame, colture, suolo, attrezzature agricole e sull'uso di acqua e fertilizzanti. La tecnologia sarebbe d'aiuto agli agricoltori per analizzare dati relativi a meteo, temperatura, umidità, prezzi e a fornire indicazioni su come ottimizzare la resa, migliorare la pianificazione, prendere decisioni più intelligenti per determinare come distribuire le risorse per ridurre al minimo gli sprechi e i costi.

Una possibile visione tecnologica a lungo termine del settore agricolo prevede una nuova generazione di sistemi robotici intelligenti, flessibili, multifunzionali e interconnessi, che lavorano autonomamente oppure in maniera collaborativa insieme all'uomo nelle aziende agricole e nell'industria alimentare. Robot agricoli, basati su piattaforme modulari con strumentazione e utensileria intercambiabile, attrezzati con nuove tecnologie di presa robotica morbida e sensori, guideranno le operazioni di produzione lungo tutta la catena alimentare. I futuri sistemi robotici utilizzeranno tecniche di intelligenza artificiale e deep learning per aumentare la propria produttività ed efficacia. Nel frattempo, lo studio di sistemi alternativi all'odierno sistema agroalimentare, comprese le innovazioni provenienti da settori come l'agricoltura verticale, contribuirà ulteriormente ad affrontare l'intensificazione dell'agricoltura, con un occhio di riguardo nei confronti dell'ambiente e del riscaldamento globale, della qualità degli alimenti e della salute. Per realizzare questa visione a lungo termine sarà quindi necessaria una transizione graduale a partire dalle pratiche agricole attuali e la maggior

parte degli agricoltori avrà bisogno di tecnologie che possano essere introdotte gradualmente, accanto e all'interno dei sistemi esistenti.

Queste nuove tecnologie devono possedere alcune caratteristiche fondamentali come affidabilità, efficienza nella gestione energetica (in modo tale da permettere ai dispositivi di lavorare per periodi prolungati), usabilità (le piattaforme devono essere facilmente utilizzate da utenti non esperti), manutenibilità (ad esempio implementando capacità di autodiagnosi o utilizzando architetture semplici e modulari facilmente riparabili), integrazione con sistemi di comunicazione mobile e adeguamento a nuovi e più rigidi standard di sicurezza.

Uno dei vantaggi dei sistemi robotici è la loro versatilità, che permette il loro utilizzo in numerose e varie attività. Questa multifunzionalità è dovuta anche alla possibilità di creare piattaforme di diversa dimensione, in base al compito che deve essere eseguito. Sarà possibile avere robot di grandi dimensioni, per eseguire operazioni pesanti, ma anche flotte di più robot leggeri e di dimensioni ridotte, che in maniera coordinata e interconnessa potrebbero eseguire le stesse lavorazioni per cui oggi si usano trattori di grossa taglia, che causano notevoli danni al suolo, compattando il terreno riducendone il grado di ossigenazione. Sarà possibile sviluppare diverse tipologie di piattaforme robotiche, a partire da quelle aeree (droni) a quelle terrestri, con diverse possibili architetture (cingolati, gommati, con arti meccanici, etc.). Sfruttando macchinari polivalenti, più leggeri e piccoli, si potrebbe avere un'inversione di tendenza per quanto riguarda l'andamento che oggi premia l'intensificazione e lo sfruttamento delle monocolture, oltre all'agricoltura di scala, tutti sistemi che favoriscono l'utilizzo di macchinari agricoli sempre più grandi, adatti a lavorare su superfici particolarmente estese.

Un ulteriore punto cardine che contribuirà alla rivoluzione del settore agricolo è costituito dal processo di elettrificazione, che sta già cominciando in altri settori, in particolare quello del trasporto e automotive. Per contrastare il cambiamento climatico, sta avvenendo una transizione nella motorizzazione dei veicoli, da una propulsione a combustibili fossili ad una propulsione elettrica.

I benefici non sono semplicemente correlati alla riduzione delle emissioni, ma si estendono su una grande varietà di applicazioni. Va anche detto, però, che è ancora necessario un grande sviluppo su queste tecnologie, in particolare sui sistemi di stoccaggio dell'energia elettrica e sulle batterie, che dovranno essere più densamente energetiche e durevoli nel tempo. L'elettrificazione permette una rivalutazione dell'architettura dei macchinari agricoli, che se oggi è basata su veicoli con motori diesel sovradimensionati, in futuro potrà essere costituita da piattaforme robotiche elettriche di dimensioni ridotte.

Figura 4: Esempi di piattaforme robotiche terrestri (Vitibot) e aeree.[8]

1.3 Robotica agricola

Lo sviluppo di robot ausiliari nel settore agricolo inizialmente sarà indirizzato su quelle applicazioni caratterizzate da un alto livello di ripetitività. I possibili campi di applicazione iniziano già dalla fase iniziale di gestione, trattamento e preparazione del terreno agricolo che precede la semina. La robotica di piccole dimensioni offre una soluzione alternativa alle lavorazioni tradizionali, evitando un'eccessiva compattazione del terreno ed eseguendo ad esempio operazioni di micro-aratura. Anche la fase di fertilizzazione, necessaria a fornire al terreno, e di conseguenza alla pianta, le sostanze nutritive adeguate, potrebbe essere eseguita in maniera localizzata, utilizzando un approccio di precisione e non di somministrazione in modo indiscriminato, che comporta sprechi di notevoli quantitativi di fertilizzanti. Il posizionamento e la mappatura dei semi potrebbero essere ulteriormente automatizzati per ottimizzare la densità, i requisiti di aria, luce, nutrienti e umidità del terreno delle singole piante coltivate.

Un'altra area di applicazione è quella della cura e gestione delle colture. Il monitoraggio dello stato di salute della pianta è utile alla prevenzione sulla diffusione di malattie e di parassiti. Le piattaforme robotiche sarebbero d'ausilio nell'individuare per tempo eventuali agenti patogeni, e nell'effettuare trattamenti con insetticidi e fungicidi mirati e localizzati solo sulla frazione della piantagione colpita, senza ricorrere all'uso indiscriminato di prodotti chimici. Il diserbo robotizzato è un'area molto promettente della ricerca attuale, che studia metodi alternativi per uccidere, rimuovere o ritardare la crescita delle piante indesiderate. La mappatura delle erbe infestanti consiste nel registrare la posizione e la tipologia di diverse specie di erbe, e la loro eliminazione deve essere effettuata senza danneggiare la coltura. utilizzando algoritmi di machine vision. I metodi automatizzati per il controllo delle erbe infestanti includono il diserbo meccanico guidato da algoritmi di machine vision, l'irrorazione selettiva di erbicidi e il diserbo laser. L'irrigazione è un altro settore in cui i robot avrebbero un ruolo importante: utilizzando un approccio di precisione ci sarebbe una riduzione dell'utilizzo di acqua, risorsa che con il cambiamento climatico sta diventando

sempre più preziosa e allo stesso tempo critica. Infine, l'acquisizione di dati tramite i sistemi sensoriali robotizzati sarebbe d'ausilio per la valutazione del momento ideale per il raccolto e per la previsione della resa dello stesso raccolto.

La robotica ha anche un possibile campo di applicazione nell'ultima fase del ciclo produttivo delle colture, ossia quello della raccolta. In particolare, ci sono grandi potenzialità nel settore della raccolta selettiva, che comprende numerosi ortaggi e frutti. Essa consiste nel raccogliere solo le parti del raccolto che soddisfano determinate soglie di qualità o quantità. Ad esempio, in una pianta di pomodoro i frutti non maturano tutti contemporaneamente, quindi la raccolta va effettuata in momenti successivi. Per eseguire al meglio questo compito, sono necessarie due caratteristiche: innanzitutto bisogna eseguire una classificazione dei prodotti, individuando quelli che soddisfano i criteri per la raccolta, e successivamente procedere con la fase di raccolta vera e propria, senza danneggiare il resto della pianta. La raccolta selettiva presenta diverse sfide per l'attuale tecnologia robotica, tra le quali l'esecuzione di un coordinamento sensorimotorio a partire da dati sensoriali rumorosi, in un ambiente agricolo complesso. Ciò richiederà adeguate operazioni di riconoscimento e di localizzazione spaziale, nonché una tecnologia robotica specializzata che sia precisa e in grado di manipolare prodotti morbidi e delicati come frutti e ortaggi.

Nelle applicazioni precedentemente descritte, che comprendono la quasi totalità delle fasi del ciclo produttivo, a partire dalla preparazione del terreno prima della semina, alla cura e gestione delle colture fino alla raccolta dei prodotti finali, si può osservare un fattore comune, ossia la presenza e l'equipaggiamento sulle piattaforme robotiche di una serie di tecnologie [8].

La prima di queste riguarda la sensoristica. L'integrazione di sistemi di misurazione all'interno dei sistemi robotici autonomi offre un potenziale significativo, in quanto la fase di percezione è il punto di partenza per poter effettuare delle decisioni e successivamente passare al compimento delle operazioni tramite gli opportuni attuatori. Le piattaforme robotiche aeree (droni) offrono

l'opportunità di misurazioni ottiche a distanza e del telerilevamento, mentre quelle terrestri permettono anche analisi di campionamento, come ad esempio analisi di composizione del suolo.

Un'altra tecnologia che sarà d'ausilio a numerose operazioni robotizzate è la machine vision, utile nella fase di monitoraggio delle colture, nella fenotipizzazione e identificazione delle diverse specie vegetali, nell'analisi della qualità, nel rilevamento dell'insorgenza di malattie. I sistemi di visione sono necessari anche per il rilevamento, la classificazione e il tracciamento di oggetti come frutta, piante, bestiame, persone, per la differenziazione delle colture rispetto alle erbe infestanti. La visione robotica in agricoltura richiede adattamento ai cambiamenti dell'illuminazione, delle condizioni atmosferiche, dello sfondo dell'immagine e dell'aspetto dell'oggetto, ad esempio durante la crescita delle piante o quando il frutto o la parte raccoglibile di un raccolto è coperta dal fogliame, garantendo allo stesso tempo una precisione sufficiente e prestazioni tali da garantire il processo decisionale e il controllo degli attuatori in tempo reale e in sicurezza. L'utilizzo della machine vision è perciò strettamente collegato allo sviluppo di ulteriori tecnologie, come le reti neurali, big data e machine learning, che permettono un progressivo apprendimento e adattamento automatico a partire da insiemi di dati provenienti dal mondo reale.

Infine, nell'ottica di sostituire la manodopera umana in tutte quelle attività di carattere manuale, i manipolatori robotici saranno equipaggiati su una moltitudine di veicoli autonomi. In campo agricolo ci sono numerose lavorazioni eseguite manualmente, come la raccolta di frutti e ortaggi. La manipolazione e la presa automatizzata di prodotti alimentari presenta una serie di sfide uniche rispetto ad altri settori. Tra queste, le significative variazioni di dimensione e forma tra esemplari dello stesso prodotto, il posizionamento eterogeneo dei prodotti e la loro natura fragile e delicata. Anche la pianificazione del movimento di presa è una sfida importante, in quanto il braccio robotico deve destreggiarsi tra i vari rami della pianta. In caso di presenza di un folto fogliame, ad esempio, la visione da sola fornirebbe dati limitati su un oggetto durante la presa e il

prelievo; gli operatori umani usano anche un feedback tattile per regolare la loro azione durante l'afferraggio di un prodotto, per assicurarsi che venga prelevato con successo. Un contributo importante in ambito di manipolazione sarà dato dai progressi in soft robotics, in cui si stanno già sviluppando pinze e dispositivi di afferraggio morbidi e deformabili.

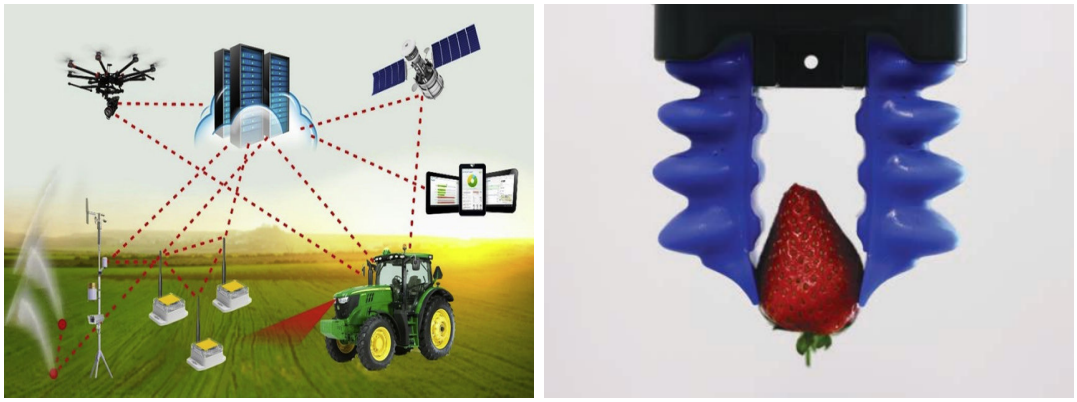

Figura 5: Schema di una possibile rete di collegamento tra sensori, satelliti, macchinari e stazioni di misurazione in un campo agricolo (sensor fusion). Esempio di un manipolatore morbido e deformabile, necessario per l'afferraggio di prodotti delicati come frutti e ortaggi. [8]

1.4 Guida autonoma

La guida autonoma è un requisito di fondamentale importanza per il funzionamento dei veicoli robotici, e sarà decisiva nel rendere competitivi i futuri macchinari agricoli rispetto a quelli attuali. La navigazione autonoma è un tema centrale della robotica in campo agricolo, in quanto le lavorazioni vengono effettuate in spazi estesi e complessi, diversamente ad esempio ad un ambiente industriale, in cui all'interno di una linea di produzione è possibile avere dispositivi robotici fissi che operano in un determinato spazio di lavoro. L'automazione delle funzioni di guida è un compito arduo, in quanto necessita un coordinamento di numerosi componenti e sensori, la pianificazione e l'ottimizzazione della traiettoria in ambienti non strutturati e l'attuazione di comandi in real time. Queste operazioni finora sono sempre state effettuate da operatori umani, quindi la difficoltà principale risiede nel dover trasferire l'intelligenza decisionale dall'uomo alla macchina. Inoltre il margine d'errore dev'essere pressochè nullo, dovendo rispettare certe condizioni di sicurezza dovute alla possibilità di impatto con persone, animali e oggetti presenti sui terreni agricoli. Pertanto è necessaria una certa affidabilità dal punto di vista della percezione, quindi della qualità dei dati provenienti da sensori (lidar, radar, sonar, telecamere, gps), che devono essere in grado di lavorare anche in condizioni metereologiche avverse, ma anche dal punto di vista della decisione, quindi della velocità, precisione e efficienza degli algoritmi che forniscono i comandi necessari alla navigazione.

Il primo compito di un algoritmo di guida autonoma consiste nel pianificare la traiettoria che il robot deve percorrere durante l'intera missione. In questa fase, denominata global path planning, viene calcolato il percorso ottimale che collega i punti in cui il veicolo deve passare, evitando ostacoli e rispettando alcune condizioni e vincoli, come ad esempio il raggio minimo di curvatura del veicolo. A partire da una mappa dell'ambiente in cui il robot deve muoversi, si elaborano, dal punto di inizio a quello di fine missione, tutti i punti intermedi attraverso i quali è necessario transitare.
Una volta creato il percorso, bisogna seguirlo. A tale scopo, bisogna sviluppare

un algoritmo di path following, che ricevendo in input la posizione attuale del veicolo e conoscendo la traiettoria precedentemente calcolata, restituisce in output i comandi di velocità da applicare alle ruote, in modo tale da raggiungere la posizione desiderata.

Le due fasi appena descritte sarebbero di per sè sufficienti a guidare un veicolo in uno spazio statico e ben definito. Questo però non è il caso di un terreno agricolo, in cui ci sono elementi in movimento, e in generale non si può escludere a priori la presenza di un ostacolo imprevisto. Diventa perciò d'obbligo l'implementazione di un algoritmo di obstacle avoidance, che effettua una riprogrammazione della traiettoria nel breve raggio, calcolando una deviazione volta a evitare tali ostacoli in maniera sicura ed efficiente.

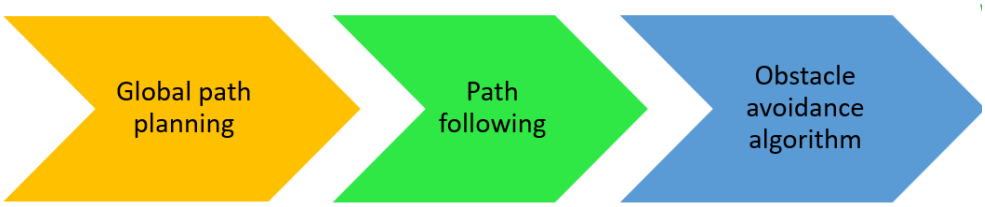

Figura 6: Schema della logica di guida autonoma.

2 Stato dell'arte

2.1 Mappatura

Prima di iniziare la trattazione vera e propria sugli algoritmi di guida autonoma, è necessario fare alcune considerazioni di carattere generale sul modello che verrà utilizzato. Il primo passaggio fondamentale, dovendo descrivere e rappresentare tutte le informazioni in formato digitale, in modo tale che siano comprensibili dal computer di bordo, è quello della mappatura dell'ambiente [9]. Per poter pianificare un percorso bisogna conoscere le regioni libere, distinguendole da quelle in cui sono presenti ostacoli. La pianificazione della traiettoria è un'attività strategica, che non può basarsi unicamente sulle informazioni di sensori e telecamere, che hanno un range operativo ridotto: è necessario perciò avere una certa conoscenza a priori della geometria del terreno su cui il veicolo deve spostarsi. Una mappa in formato digitale, oltre a includere la posizione e localizzazione di oggetti, è in grado di contenere una serie di informazioni aggiuntive utili alla navigazione e al processo decisionale, come ad esempio la morfologia, la pendenza, la tipologia di terreno, e molto altro.

In una mappa, definiamo lo spazio delle configurazioni come l'insieme di tutte le possibili configurazioni del robot. Esso quindi rappresenta anche tutte le trasformazioni che possono essere applicate al robot. Può essere suddiviso in spazio libero, ossia l'insieme di tutte le configurazioni che non collidono con gli ostacoli, e spazio degli ostacoli, ossia l'insieme delle configurazioni all'interno dello spazio di configurazione in cui il robot non può muoversi.

L'attività di mappatura, che solitamente è una procedura di una certa durata e complessità, ma che viene effettuata solo una volta, consiste nel raccogliere accuratamente le informazioni e caratteristiche dell'ambiente esterno e nel localizzare gli oggetti che lo costituiscono. Il livello di dettaglio e accuratezza è importante, ma deve essere tenuto in considerazione che può diventare complicata la memorizzazione di ambienti molto estesi in formato tridimensionale.

2.1.1 Occupancy grid map

Un approccio molto diffuso nella digitalizzazione delle mappe è quello della decomposizione a griglia di occupazione, in cui l'ambiente reale continuo viene discretizzato in una griglia dalla mesh fissa. Una trasformazione di questo tipo è mostrata in figura 7, che illustra la suddivisone dell'ambiente in celle discrete, a cui viene assegnato un valore nullo se la cella è completamente vuota e un valore non nullo se la cella è almeno parzialmente occupata da un ostacolo. Questo approccio è particolarmente utile quando un robot è dotato di sensori come lidar o radar, in quanto le misurazioni di distanza, combinate con la posizione assoluta del robot, possono essere utilizzate direttamente per aggiornare il valore di riempimento di ciascuna cella. Nella griglia di occupazione, ogni cella può avere un contatore, il cui valore nullo indica che la cella non è stata "colpita" da alcuna misurazione di distanza e, pertanto, è probabilmente libera. Con l'aumentare del numero di misurazioni, il valore della cella viene incrementato e, al di sopra di una certa soglia, la cella viene considerata un ostacolo. I valori delle celle vengono inoltre diminuiti quando un raggio di misurazione attraversa la cella, colpendone un'altra. In questo modo è possibile attenuare il problema dovuto a misurazioni errate, causate da letture rumorose e imprecise del sensore, ma è anche possibile rappresentare ostacoli transitori e ambienti dinamici.
L'utilizzo della griglia di occupazione presenta però alcuni svantaggi. In primo luogo, la natura discreta del metodo può portare ad alcune inesattezze, come l'occlusione di alcuni passaggi stretti se la mesh è troppo grossolana. Con un adeguato livello di raffinatezza della griglia, però, si ottengono risultati piuttosto soddisfacenti. D'altro canto, una mesh troppo raffinata aumenta notevolmente lo spazio di memoria richiesto. É necessario quindi trovare un giusto compromesso tra i due requisiti.

Una maniera per ridurre l'impatto sulla memoria e allo stesso tempo per semplificare le operazioni di navigazione è la bidimensionalizzazione della mappa. Da un punto di vista strettamente legato alla movimentazione di un veicolo terrestre, si può comprimere la mole di informazioni tipica di un ambiente

tridimensionale in una rappresentazione bidimensionale. L'operazione prevede di raffigurare gli elementi della mappa visti dall'alto, proiettando su un piano la nuvola di punti che definisce l'ambiente esterno tridimensionale. Così facendo si perde l'informazione relativa al posizionamento in quota degli ostacoli, e non è possibile distinguere se l'ostacolo è sospeso in aria o se è a contatto con il terreno. Per un veicolo terrestre in moto, però, non è di fondamentale importanza sapere a che altezza si trova l'ostacolo: è sufficiente sapere se in una determinata area è possibile transitare o meno. Tutti gli ostacoli che si trovano ad una quota inferiore o uguale all'altezza del veicolo sono ugualmente pericolosi. Facendo un esempio pratico, in un'operazione di taglio dell'erba, si eviterà il passaggio sia in una zona con una pietra a terra, sia in una in cui un albero ha un grosso ramo sospeso a mezza altezza, che urterebbe la parte superiore del tagliaerba.

Un'altra accortezza che spesso si utilizza è quella di assumere che il robot sia puntiforme. Questo approccio infatti evita di considerare l'ingombro del veicolo nella fase di pianificazione della traiettoria e in quella di movimentazione. Per evitare collisioni impreviste con altri ostacoli, dato che si è ridotto il robot a un punto, è necessario però "gonfiare" ogni ostacolo di un valore pari alla dimensione del raggio del robot. In questo modo si ingrandisce lo spazio degli ostacoli e si riduce lo spazio libero, ma risulta molto più agevole la computazione della traiettoria del robot da un punto di vista matematico e geometrico.

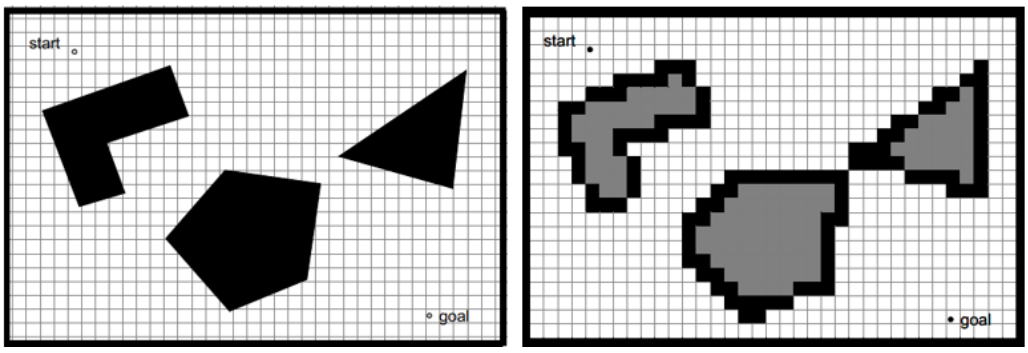

Figura 7: Esempio di una mappa a decomposizione a griglia fissa. Una mesh troppo grossolana può causare l'occlusione di passaggi stretti. [9]

2.1.2 Metodi di mappatura

L'operazione di mappatura può avvenire in diversi modi. Il più banale è la creazione di una mappa a mano, con un operatore che, data una griglia, seleziona le celle in cui sono presenti ostacoli. É un processo che può andare bene per ambienti di dimensioni ridotte e dalla geometria semplice. Richiede infatti la misurazione degli spazi e delle dimensioni degli ostacoli, in modo tale da trasferire adeguatamente e con le giuste proporzioni le distanze sulla mappa virtuale. Questo metodo può essere molto veloce, ma allo stesso tempo impreciso e soprattutto richiede un massiccio intervento da parte dell'uomo.

Un secondo metodo di mappatura consiste nello sfruttare immagini satellitari o aeree. In questo caso la geometria dell'ambiente si può ottenere facilmente disegnando linee e curve utilizzando l'immagine satellitare come traccia. Posizionando nella foto un marker con una lunghezza di riferimento nota, le dimensioni assolute degli oggetti sono ottenibili con una semplice proporzione. L'operazione di tracciatura delle geometria può essere sempre eseguita da un operatore (in questo caso rispetto al precedente l'operazione è molto più agevole e veloce) oppure da un software che riconosce i contorni degli oggetti e definisce le forme degli ostacoli, distinguendoli dalle aeree percorribili dal veicolo. Sfruttando l'intelligenza artificiale è possibile, oltre a definire posizione e forma degli oggetti, anche la loro tipologia e classificazione. Si crea per ogni oggetto un'etichetta semantica, che fornisce ulteriori informazioni utili alla navigazione. In questo modo si riesce a distinguere una zona di prato da una con erba alta, un albero da un edificio, etc.

Un ulteriore metodo di mappatura, completamente autonomo, è chiamato SLAM (Simultaneous localization and mapping). É un metodo con cui un veicolo robotico costruisce una mappa di un ambiente sconosciuto e contemporaneamente la utilizza per calcolare la propria posizione. Sia la traiettoria del veicolo che la posizione di tutti gli ostacoli è stimata in real time, senza la necessità di una conoscenza a priori dell'ambiente esterno. Il funzionamento alla base del metodo

prevede di far spostare il veicolo e di registrare la nuvola di punti misurata da un sensore di distanza. Integrando le informazioni provenienti da sensori odometrici (che misurano il numero di giri delle ruote), IMU (che misura l'accelerazione del veicolo) e GPS, le diverse nuvole di punti, ottenute in momenti successivi, vengono opportunamente allineate tra di loro, costruendo via via la mappa dell'intero sito di interesse. Gli errori e le imprecisioni dei sensori utilizzati rendono difficile l'ottenimento di una mappa precisa e accurata. Supponiamo ad esempio di utilizzare un encoder rotativo per misurare la velocità di rotazione delle ruote del robot e quindi stimarne la velocità di spostamento. Se anche solo una ruota slitta, la stima del movimento del robot sarà sbagliata, quindi le nuvole di punti saranno allineate erroneamente, dando vita ad una mappa distorta e inutilizzabile. É opportuno quindi correggere tali errori, in particolare se questi si propagano nel tempo. Una possibile soluzione è quella del "loop closure": si fa percorrere al robot una traiettoria ad anello, per cui quando si trova in un luogo già visitato, il robot riconosce l'ambiente circostante ed è in grado di correlare la posizione stimata con quella reale. In questo modo l'errore accumulato nel tempo viene corretto e si ha un riallineamento della mappa, che migliora se l'anello viene percorso più volte.

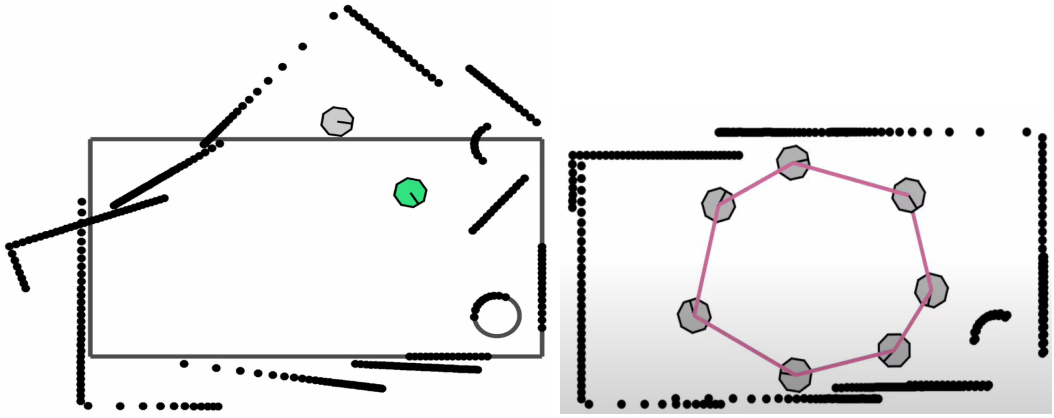

Figura 8: Esempio dell'utilizzo del metodo SLAM. A sinistra, a causa di errori nell'odometria, la posizione stimata (robot grigio) e quella reale (robot verde) sono diverse, e le nuvole di punti non sono allineate correttmente. A destra, con una procedura di loop closure, il sistema corregge l'errore accumulato e riallinea la mappa. [10]

2.2 Global path planning

Una volta ottenuta una mappa dell'ambiente in cui il robot deve operare, è necessario pianificare il percorso generale della missione che deve essere eseguita. Questa operazione è denominata global path planning, in quanto è una fase strategica che considera l'intera estensione della missione. Consiste nel trovare una traiettoria, all'interno della mappa dell'ambiente, a partire dalla posizione di partenza del veicolo fino a quella finale. L'obiettivo di un algoritmo di global path planning è quello di trovare un percorso percorribile e ottimale, considerando almeno un parametro di riferimento, come ad esempio il percorso più corto, quello più agevole o quello percorribile più velocemente. La traiettoria elaborata deve evitare gli ostacoli registrati sulla mappa di partenza, che non comprende eventuali oggetti imprevisti presenti sul terreno agricolo in una fase successiva alla mappatura. Una mappa, pur essendo precisa e dettagliata, infatti, è sempre e comunque una fonte di conoscenza parziale, che però può essere sfruttata per eseguire una prima pianificazione della traiettoria, la cui percorribilità verrà verificata in un momento successivo tramite l'ausilio di sensori e telecamere.
L'operazione di global path planning è normalmente eseguita "offline", ossia con il veicolo fermo prima dell'inizio della missione. É infatti un'operazione di pianificazione, che richiede un certo sforzo computazionale e quindi un certo tempo per essere elaborata. In campo agricolo, però, dovendo eseguire ciclicamente la maggior parte delle lavorazioni, è possibile memorizzare in un database, una volta calcolati, i diversi percorsi, che verranno selezionati opportunamente dall'agricoltore o da un sistema centralizzato autonomo.

2.2.1 Pianificatore geometrico

Un primo metodo che può essere d'ausilio alla risoluzione del problema posto dal global path planning è l'utilizzo di un pianificatore geometrico, che, una volta stabiliti il punto di partenza, di arrivo e le relative orientazioni nel piano del veicolo, trova la traiettoria tra i due punti con una combinazione di segmenti lineari e archi di circonferenza. Se si considera ad esempio un veicolo con un raggio minimo di curvatura e che può solo procedere in avanti a velocità costante (cosiddetto veicolo di Dubins), può essere dimostrato che il percorso di minima lunghezza è costituito da un massimo di tre curve [11].
Se si considera un veicolo uguale al precedente ma che è in grado anche di fare retromarcia (veicolo di Reeds-Sheep), si possono ottenere in generale percorsi più brevi, composti però da una combinazione di massimo cinque curve [12].

I pianificatori geometrici sono efficienti in termini di costo computazionale, in quanto la traiettoria è calcolata velocemente tramite metodi analitici e geometrici. Non sono però in grado di identificare gli ostacoli nella mappa e di creare percorsi che li evitino. Questo fatto limita il loro utilizzo ai soli casi in cui un operatore suddivide manualmente il percorso in waypoint intermedi. In tal modo l'operazione di tracciatura della traiettoria è semplificata ed avviene unicamente in zone libere della mappa.

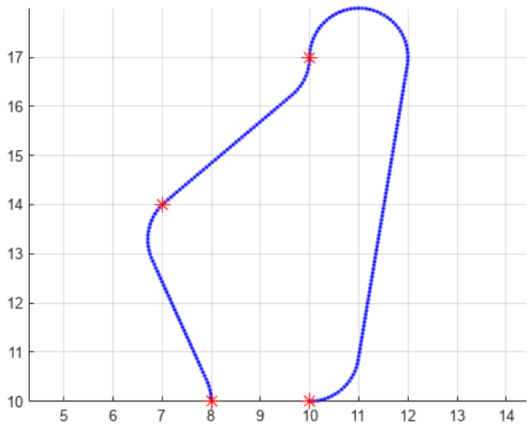

Figura 9: Esempio di un pianificatore geometrico per un veicolo di Dubins. Tra un waypoint e il successivo l'algoritmo trova la traiettoria più breve con una combinazione di massimo tre curve, tra cui segmenti lineari e archi di circonferenza con un raggio pari alla curvatura minima del veicolo.

2.2.2 Metodi search based e algoritmo A*

Un secondo approccio alla risoluzione del problema del global path planning è quello proposto dai metodi search based, che calcolano il percorso con un processo di ricerca all'interno di una mappa discretizzata del terreno. Essi prevedono di discretizzare la mappa, dividendola in punti o nodi, e di cercare il percorso più breve considerando solamente questi nodi. Un esempio è l'algoritmo A* [13], che associa a ciascun nodo un costo, dato da:

$$c(n) = f(n) + g(n)$$

Dove n è l'n-esimo nodo, $f(n)$ è la lunghezza del percorso dal nodo iniziale all'n-esimo nodo e $g(n)$ è una funzione euristica che misura la distanza euclidea (anche attraverso eventuali ostacoli) dal nodo considerato al nodo finale. Partendo dal nodo iniziale, l'algoritmo calcola il costo totale $c(n)$ di tutti i nodi adiacenti e seleziona quello con il valore minore di tutti. A parità di costo totale $c(n)$, viene scelto il nodo con il costo $g(n)$ minore, in quanto è più vicino al nodo finale. Si procede iterando questo procedimento, imponendo un costo totale infinito ai nodi in cui è presente un ostacolo, in maniera tale da escluderli ed evitare che vengano selezionati. Iterazione dopo iterazione, la mappa viene esplorata, fino al raggiungimento del nodo finale.

Facendo un esempio pratico, consideriamo la mappa di figura 10, con nodo di partenza A e di arrivo B colorati in azzurro, separati da un ostacolo colorato in nero. Si ipotizza una distanza di dieci unità tra i nodi nelle celle con un lato in comune e di quattordici unità tra quelli in celle con un vertice in comune. I costi $f(n)$ e $g(n)$ sono espressi in alto, mentre il costo totale $c(n)$ è posto al centro della rispettiva cella. Alla prima iterazione si calcolano i costi degli otto nodi adiacenti al punto A e si seleziona quello in alto a sinistra (evidenziato in rosso). Nelle successive iterazioni, la ricerca si espande verso sinistra, in quanto il costo $g(n)$ è inizialmente minore. Dato che l'ostacolo blocca l'espansione verso l'alto, la ricerca prosegue fin troppo a sinistra, fino a quando, ad un certo punto,

risulta più conveniente spostarsi a destra, superando l'ostacolo e raggiungendo facilmente il nodo di arrivo.

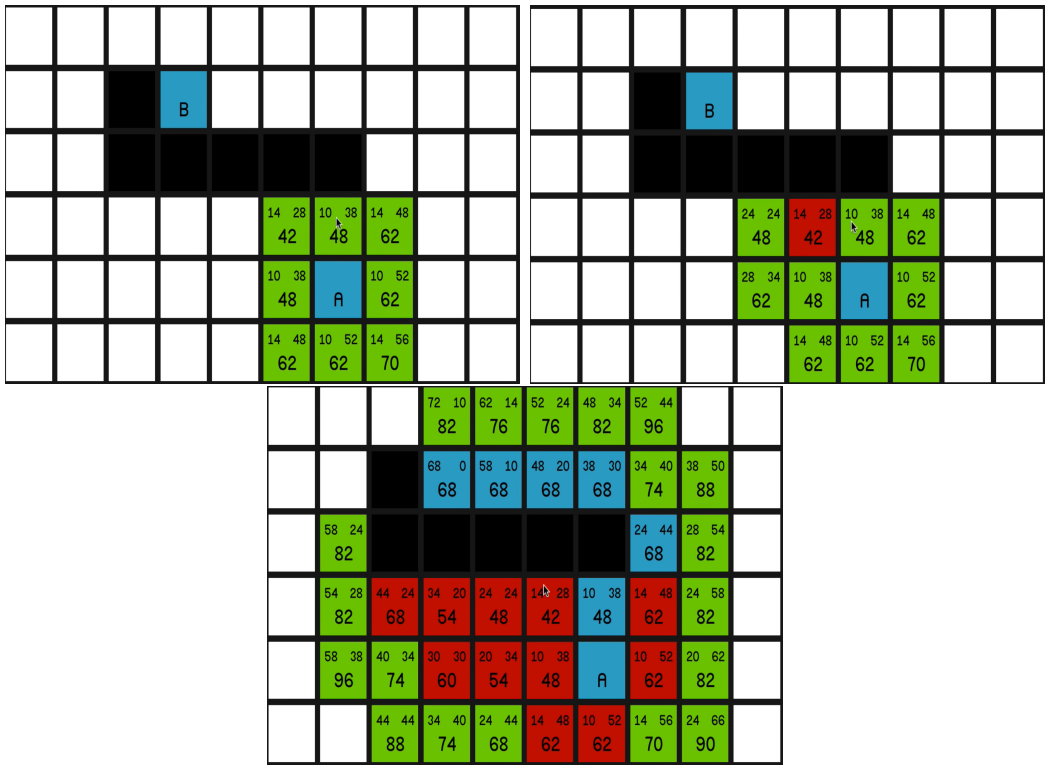

Figura 10: Esempio del funzionamento dell'algoritmo A*. Le celle in azzurro sono quelle di partenza (A) e di arrivo (B). Quelle nere appartengono all'ostacolo. Le rosse sono le celle selezionate ad ogi iterazione, e le verdi quelle adiacenti per cui è stato effettuato il calcolo del costo $c(n)$ [14].

L'algoritmo A* è efficace in quanto è in grado di trovare il percorso più breve (e senza collisioni con ostacoli), tra due punti all'interno di una mappa. Risulta però in alcuni casi eccessivamente lento: il suo costo computazionale infatti dipende fortemente dalla dimensione della griglia. Terreni agricoli di una determinata estensione richiedono griglie grandi e discretizzate finemente, in modo da descrivere con una certa precisione le forme degli ostacoli e le curve della traiettoria.

2.2.3 Metodi sampling based e algoritmo RRT*

I metodi sampling based costituiscono un'ulteriore classe di algoritmi di global path planning [15]. La generazione della traiettoria del robot viene eseguita selezionando in maniera random dei punti all'interno della mappa e connettendoli opportunamente. Tra questi risalta l'algoritmo RRT* (Rapidly-exploring Random Tree), che prevede la costruzione di una struttura ad albero, che si ramifica nello spazio libero della mappa, fino al raggiungimento del punto obiettivo.

L'algoritmo ad ogni iterazione seleziona randomicamente un punto nella mappa q_{rand}, e cerca il punto più vicino appartenente all'albero, denominato q_{near}. Alla prima iterazione si ha che $q_{near} = q_{start}$, in quanto l'albero è inizialmente costituito dal solo punto di partenza. Sul segmento dato da q_{rand} e q_{near}, viene individuato il punto q_{new}, che è il nuovo potenziale punto da collegare all'albero. La lunghezza della connessione tra l'albero e il nuovo punto è limitata da un fattore λ: se la lunghezza del segmento dato da q_{rand} e q_{near} è maggiore di λ, il nuovo punto è posizionato a una distanza pari a λ lungo lo stesso segmento. I punti selezionati casualmente stabiliscono quindi il controllo della direzione di crescita dell'albero. Il punto q_{new} viene aggiunto all'albero solamente se lungo il tratto dell'ipotetica connessione non sono interposti ostacoli.

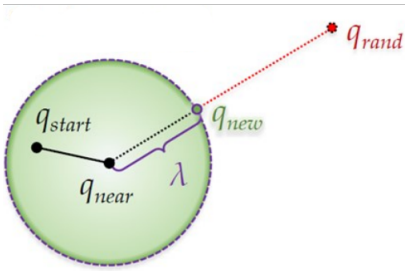

Figura 11: Grafico che rappresenta il meccanismo di individuazione di un nuovo punto da collegare alla struttura ad albero.

Un'ottimizzazione dell'algoritmo RRT* consiste nella congiunzione del punto q_{new} non necessariamente al punto più vicino q_{near}, bensì al nodo dell'albero che minimizza la distanza al punto di partenza. Il meccanismo della scelta del punto più adatto per la connessione all'albero è basato sull'utilizzo di una funzione

costo, che associa ad ogni nodo dell'albero un valore pari alla distanza percorsa attraverso le ramificazioni esistenti della struttura. Una volta individuato il potenziale nuovo punto q_{new}, vengono esaminati tutti i punti dell'albero lungo un certo raggio di azione attorno ad esso. Tra questi si sceglie il punto tramite il quale q_{new} ha il minimo costo. Inoltre, una volta eseguita la connessione si esegue una verifica sugli altri nodi nelle vicinanze di q_{new}: si controlla per tutti i nodi vicini se una connessione con il nuovo punto creato ne abbassi il costo, e in caso affermativo si procede con la creazione di una nuova ramificazione. In questo modo si ottimizza sempre il percorso di tutte le ramificazioni possibili che costituiscono la struttura ad albero.

Ad esempio, consideriamo il caso di figura 12, con q_{new} di colore giallo e q_{near} di colore verde. In questo caso il costo nell'unire questi due punti è molto alto, in quanto il percorso prevede di aggirare dall'alto l'ostacolo rosso. Si cercano quindi ulteriori possibili connessioni in un area di un determinato raggio, giungendo alla conclusione per cui risulta più conveniente unire q_{new} al punto di colore viola. Inoltre, successivamente alla connessione, si osserva come per raggiungere il punto verde convenga arrivare dal basso, per cui si modifica la precente ramificazione e se ne crea una nuova con il nuovo punto appena aggiunto.

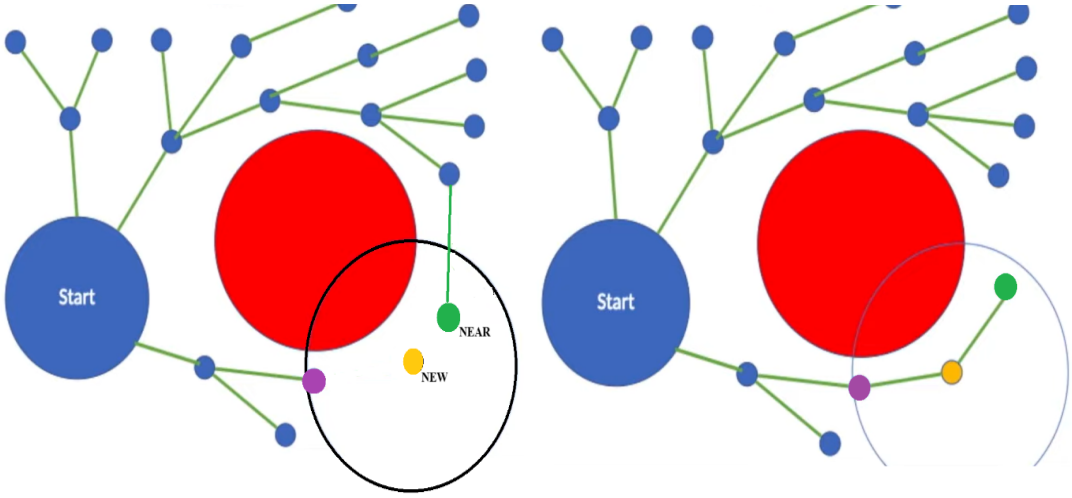

Figura 12: Meccanismo di collegamento di un nuovo punto all'albero e successiva modifica delle ramificazioni.

Il procedimento finora descritto continua fino a quando non si raggiunge un certo numero di iterazioni, che corrisponde al numero di punti dell'albero, o fino a quando non viene aggiunto un nodo in un una certa area circostante il punto di arrivo.

Come si osserva in figura 13, l'algoritmo RRT* ha la peculiarità di espandersi rapidamente verso regioni inesplorate. É in grado di trovare una soluzione, seppur non ottimale, con un basso numero di iterazioni. Con l'aumentare del numero di nodi dell'albero, lo spazio libero viene progressivamente occupato e la soluzione converge sempre più verso un percorso di lunghezza minima. Un numero elevato di punti richiede uno sforzo computazionale maggiore, ma la struttura ad albero ottenuta può essere sfruttata per calcolare il percorso in qualsiasi altro punto della mappa. Questo non è possibile con l'algoritmo A*, in quanto l'intero procedimento di ricerca del percorso deve essere eseguito ogni volta che varia il punto iniziale o finale, dato che è basato sul calcolo del costo nodale, il quale dipende dal posizionamento di tali punti.
Inoltre, un altro vantaggio dell'algoritmo RRT* rispetto a quello A* è rappresentato dal fatto che controllando il parametro λ, la distanza esplorata tra un'iterazione e quella successiva è maggiore, non essendo vincolata alla dimensione della cella con cui la mappa è stata discretizzata. In tale modo l'esplorazione della mappa avviene più velocemente.

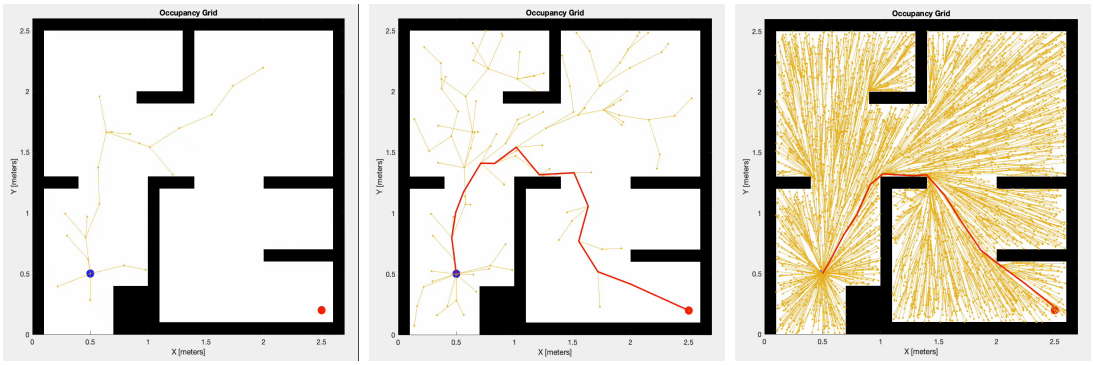

Figura 13: Esempio di path finding tramite algoritmo RRT*. L'albero di ricerca si espande velocemente verso aree inesplorate, a partire dal punto iniziale. Con pochi punti il percorso trovato non è ottimale. Aumentando il numero di punti di campionamento, il percorso tende a quello di distanza minima [16].

2.3 Path following

Una volta ottenuta la mappa dell'ambiente e calcolato il percorso che si desidera far percorrere al robot, bisogna fornire al veicolo le istruzioni e i comandi necessari alla sua movimentazione, affinchè la traiettoria sia seguita correttamente. Questa operazione è definita path following, e comprende l'insieme di regole e criteri utilizzati per guidare il veicolo nella direzione voluta. Innanzitutto bisogna capire dove si vuole che il robot si sposti, ossia identificare il punto e la direzione obiettivo, considerando la posizione attuale del veicolo e la traiettoria ottenuta nella fase di global path planning. Di fatto il veicolo deve orientarsi e localizzarsi all'interno della mappa, e stimare dove deve spostarsi, quindi determinare di quanto accelerare e di quanto sterzare.

La velocità lineare e quella di imbardata stimate nella fase di path following sono riferite al baricentro del telaio del veicolo. Il passaggio successivo consiste nel costruire un modello cinematico del rover, che permetta di ottenere velocità e verso di rotazione di ciascuna ruota, in modo tale che il telaio si muova nella direzione stabilita precedentemente. Questa fase, denominata anche motion planning, necessita di una profonda conoscenza della fisica del veicolo, in particolare della sua cinematica e dinamica. Un modello completo dovrebbe considerare il contatto ruota terreno e la relativa trasmissione di forze e moto, l'inerzia del veicolo, la configurazione e disposizione delle ruote, e alcuni vincoli cinematici come ad esempio il minimo raggio di curvatura. Saranno quindi fatte una serie di ipotesi e semplificazioni che renderanno più intuitivo il modello cinematico del rover.

In una fase successiva, non trattata in questa attività di tesi, le informazioni di tipo cinematico dovranno essere espresse in termini di corrente e tensione fornite ai motori elettrici: sarà quindi necessario implementare un controllore che regoli coppia e velocità di ciascun motore.

2.3.1 Controllore Pure Pursuit

Il controllore Pure Pursuit [17] è un metodo di sterzo automatico utilizzato per l'inseguimento di una traiettoria, molto diffuso nel campo della robotica mobile. Il suo scopo è quello di calcolare la curvatura necessaria a spostare il veicolo dalla sua posizione attuale a una posizione obiettivo. Il funzionamento dell'algoritmo prevede di individuare un punto di traguardo (goal point) che si trovi sul percorso da seguire ad una certa distanza dal veicolo, chiamata *lookahead distance*. Una volta trovato il goal point, si determina l'arco di circonferenza che lo unisce alla posizione attuale del veicolo: la curvatura di tale arco sarà successivamente applicata allo sterzo del veicolo, affinchè questo raggiunga la destinazione desiderata. Il meccanismo è molto simile al processo che un essere umano compie alla guida di un'automobile, che consiste nel guardare ad una certa distanza in avanti e nel dirigersi verso il punto della strada individuato.

Dal punto di vista geometrico, l'individuazione dell'arco di circonferenza che unisce il punto di posizione attuale con il goal point deve soddisfare una specifica condizione, dato che per soli due punti passano infinite circonferenze. Si sceglie perciò l'arco di circonferenza con il centro sulla retta passante per l'interasse delle ruote del veicolo, rappresentato in figura 14.

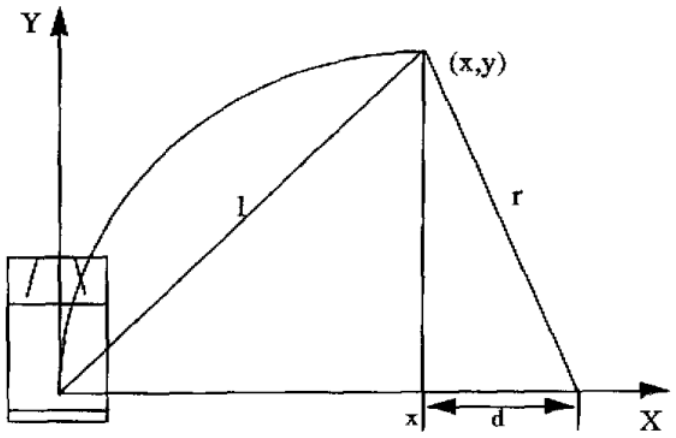

Figura 14: Schema geometrico dell'algoritmo pure pursuit. [17]

L'espressione della curvatura, date le coordinate del goal point (x, y) rispetto al robot, e fissata la distanza di lookahead distance l, si ottiene nel seguente modo:

$$x^2 + y^2 = l^2$$

$$x + d = r \Longrightarrow d = r - x$$

$$(r - x)^2 + y^2 = r^2 \Longrightarrow r^2 - 2rx + x^2 + y^2 = r^2 \Longrightarrow 2rx = l^2$$

$$r = \frac{l^2}{2x} \Longrightarrow \gamma = \frac{2x}{l^2}$$

L'implementazione dell'algoritmo è basata sui seguenti passaggi:

1. *Localizzazione del veicolo.* Sfruttando ad esempio sistemi di localizzazione come il GPS e il sensore IMU, si individuano le coordinate della posizione attuale del robot.

2. *Determinazione del waypoint più vicino al robot.* Per evitare che la ricerca del goal point cominci sempre dal punto iniziale del percorso, si individua il waypoint più vicino al robot. La ricerca del goal point inizierà da tale punto, in modo tale da escludere tutti i punti già oltrepassati e alleggerire, man mano che il veicolo avanza, lo sforzo computazionale.

3. *Individuazione del goal point.* Si trovano i punti sul percorso ad una distanza l dal robot e si sceglie quello più vicino. Quest'operazione è effettuata nel sistema di riferimento globale della mappa.

4. *Trasformazione del goal point in coordinate riferite al veicolo.* In modo tale da ottenere i parametri (x, y) rappresentati in figura 14.

5. *Calcolo della curvatura.* Utilizzando l'equazione precedentemente descritta, si ottiene la curvatura necessaria al veicolo per raggiungere il goal point, e si trasforma nell'equivalente angolo di sterzo da applicare al volante.

6. *Ripetizione del ciclo appena descritto.* Aggiornando di volta in volta la posizione del veicolo.

2.3.2 Modello cinematico del veicolo

La definizione del modello cinematico del veicolo ha lo scopo di ottenere le velocità angolari delle ruote in funzione delle velocità lineare e di imbardata espresse nel sistema di riferimento locale del robot [9].

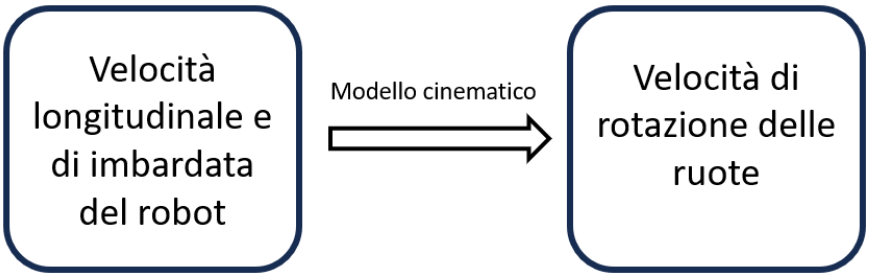

Figura 15: Schema del modello cinematico del veicolo.

Prima di considerare l'intero veicolo, è utile trattare la cinematica di una singola ruota fissa. Essa è una ruota che non ha un asse verticale di rotazione e perciò non può sterzare. La sua posizione rispetto al sistema di riferimento locale del robot (X_r, Y_r) è definita innanzitutto dalla distanza l dal punto P, che è il baricentro geometrico del veicolo. Il segmento di lunghezza l è orientato di un angolo α rispetto all'asse X_r. L'orientazione della ruota è poi definita dall'angolo β, che rimane costante essendo la ruota non sterzante. La ruota, di raggio r, ruota in funzione del tempo con legge $\omega(t)$.

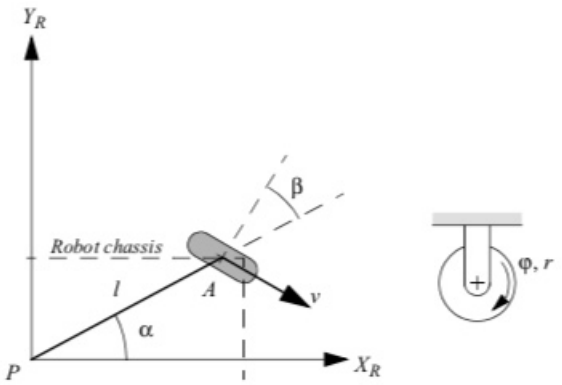

Figura 16: Definizione di una ruota fissa rispetto al sistema di riferimento locale solidale al telaio del veicolo [9].

Per lo studio cinematico di una ruota si considerano le seguenti ipotesi:

- Il piano della ruota rimane sempre verticale.
- Il punto di contatto tra la ruota e il terreno è unico.
- Consideriamo moto di puro rotolamento, ossia senza strisciamento.

Con queste ipotesi, si creano due condizioni di vincolo per il moto della ruota: la prima è quella di puro rotolamento della ruota, mentre la seconda riguarda la condizione di assenza di slittamento in direzione trasversale, per cui la ruota non è in grado di muoversi in direzione ortogonale al piano che la definisce.

Il vincolo di moto di puro rotolamento è collegato al fatto che al moto di traslazione in direzione longitudinale corrisponde una certa rotazione della ruota, in modo tale che nel punto di contatto con il terreno non ci sia strisciamento. Questo vincolo è espresso da:

$$\begin{bmatrix} sin(\alpha+\beta) & -cos(\alpha+\beta) & -lcos(\beta) \end{bmatrix} \cdot \begin{bmatrix} \dot{x_r} \\ \dot{y_r} \\ \dot{\theta} \end{bmatrix} - r\omega = 0 \tag{1}$$

Dove $\dot{x_r}, \dot{y_r}, \dot{\theta}$ sono le velocità del robot (punto P) espresse nel sistema di riferimento locale. Il primo termine rappresenta il moto totale lungo il piano della ruota. Questo termine dev'essere, in condizioni di puro rotolamento, esattamente uguale a quanto la ruota ha ruotato, ossia di $r\omega$.

Il vincolo di strisciamento laterale invece descrive il fatto che la velocità della ruota lungo l'asse ortogonale al piano che la contiene è nulla:

$$\begin{bmatrix} cos(\alpha+\beta) & sin(\alpha+\beta) & lsin(\beta) \end{bmatrix} \cdot \begin{bmatrix} \dot{x_r} \\ \dot{y_r} \\ \dot{\theta} \end{bmatrix} = 0 \tag{2}$$

Per il modello cinematico dell'intero veicolo, si consideri un robot con due sole ruote fisse, una destra e una sinistra. Si supponga di azionare i motori elettrici collegati alle ruote con un comando $u = [u_l, u_r]^T$ proporzionale alla loro velocità angolare $\omega = [\omega_l, \omega_r]^T$, e si considerino diverse situazioni. Ad esempio, se $u_l = u_r > 0$, il robot si muoverà in avanti nella direzione in cui esso è orientato. Nel sistema di riferimento locale solidale al robot, la velocità longitudinale $\dot{x_r}$ sarà proporzionale al raggio delle ruote r, mentre quella trasversale $\dot{y_r}$ è sempre nulla in quanto le ruote non possono avere un moto laterale. Se invece $u_l = -u_r > 0$, il robot ruoterà in senso orario, dato che le ruote stanno girando in verso opposto e con uguale velocità. Con una velocità di rotazione delle due ruote uguale in modulo e opposta in verso, quindi, non avviene una traslazione del punto P, posizionato al centro dell'asse che collega le due ruote. Si può concludere che la velocità lineare del punto P dipende dalla media delle velocità di rotazione delle due ruote.

La velocità angolare del robot nel piano, chiamata $\dot{\theta}$, invece, sarà proporzionale alla differenza tra le due velocità di rotazione e al raggio delle ruote, mentre sarà inversamente proporzionale alla lunghezza dell'interasse. Si consideri infatti il caso in cui una ruota sia ferma (ad esempio la sinistra), e l'altra sia messa in rotazione: dato che il punto P è a metà tra le due ruote, si muove con una velocità $\dot{x} = \frac{1}{2} r\omega_r$ e il punto di istantanea rotazione del robot corrisponde alla ruota ferma.

Mettendo insieme il tutto si ottiene il seguente modello cinematico del robot:

$$\begin{cases} \dot{x_r} = \frac{r(\omega_l + \omega_r)}{2} \\ \dot{\theta} = \frac{r(\omega_l - \omega_r)}{2l} \end{cases} \implies \begin{cases} \omega_r = \frac{\dot{x_r}}{r} - \frac{\dot{\theta} l}{r} \\ \omega_l = \frac{\dot{x_r}}{r} + \frac{\dot{\theta} l}{r} \end{cases}$$

In questo modo si ottengono le velocità di rotazione delle ruote del veicolo, in funzione della velocità longitudinale e di imbardata di quest'ultimo. A questo punto, è possibile implementare un controllore che comandi i motori elettrici e sposti il robot nella direzione voluta.

2.4 Obstacle avoidance

I temi trattati fino ad ora, ossia il path planning e il path following, sarebbero già di per sè sufficienti a movimentare un veicolo a guida autonoma: un veicolo dotato di tali tecnologie è infatti in grado di calcolare un percorso e di seguirlo. Queste competenze, però, possono essere applicate in sicurezza unicamente nei casi in cui si abbia una totale corrispondenza tra la mappa dell'ambiente e l'ambiente stesso. Questo può accadere se il veicolo si muove in un ambiente statico e controllato, che però è difficile da realizzare in ambito agricolo. Il mondo reale infatti è dinamico e mutevole: la probabilità che il veicolo incontri nel suo percorso un ostacolo imprevisto non è nulla. Per tale motivo, la capacità strategica di pianificazione, unita a quella di inseguimento della traiettoria, non è sufficiente. In un dispositivo robotico mobile è di grande importanza anche la capacità cognitiva e tattica, che consiste nell'adattamento agli eventi esterni. L'aspetto tattico permette al robot di evitare le collisioni con gli ostacoli, modulando la traiettoria in base alla percezione dell'ambiente circostante tramite le letture dei sensori.
La sfida della navigazione per un robot consiste nell'eseguire un piano per raggiungere la posizione di destinazione. Durante l'esecuzione, sia la competenza strategica che quella tattica hanno un ruolo fondamentale. Senza reagire agli eventi imprevisti, lo sforzo di pianificazione non sarà ripagato, perché il robot non raggiungerà mai fisicamente l'obiettivo. Senza pianificazione, lo sforzo di reazione non può guidare il comportamento complessivo del robot per raggiungere un obiettivo lontano: anche in questo caso, il robot non raggiungerà mai la meta. Per completare le funzioni di guida autonoma, arrivati a questo punto, è perciò necessaria l'implementazione di un algoritmo di obstacle avoidance, che permetta di evitare ostacoli imprevisti sulla traiettoria del veicolo. Esso viene anche definito algoritmo di local path planning, in quanto esegue una verifica in tempo reale di un'area di dimensioni ridotte nei pressi del veicolo, creando una mappa che si aggiorna continuamente e in maniera dinamica a partire dalle scansioni dei sensori. Elaborando tali informazioni viene scelta, tramite opportune regole e condizioni, la direzione da seguire per evitare tutti gli ostacoli vicini. Poiché il

robot non dispone di informazioni preliminari sulla presenza, forma e dimensione degli ostacoli, la natura dell'algoritmo è reattiva. Infatti, dato che le coordinate di un ostacolo possono cambiare in qualsiasi momento e non sono note a priori, l'algoritmo deve calcolare l'azione successiva ad ogni iterazione, sulla base del contesto e delle informazioni attualmente disponibili.

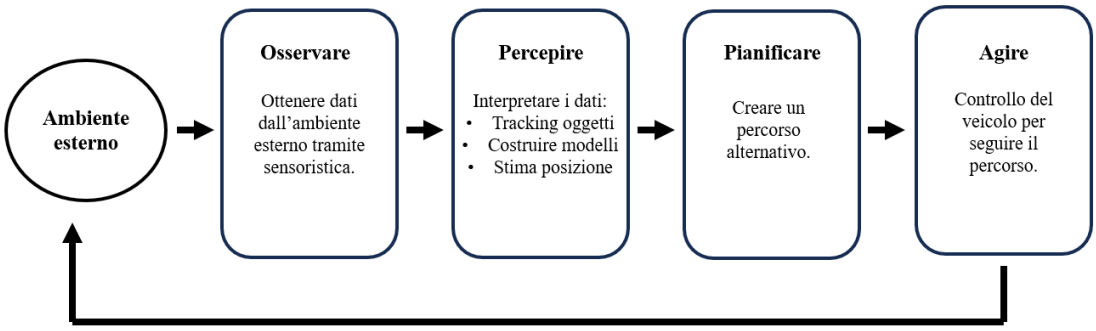

Figura 17: Schema delle operazioni principali che deve compiere un algoritmo di obstacle avoidance.

In questa sezione verranno analizzate, confrontando pregi e difetti, diverse possibili soluzioni al problema dell'obstacle avoidance, partendo dalle logiche più semplici fino a quelle più articolate e di maggior complessità.

2.4.1 Algoritmi base di obstacle avoidance

Uno dei più semplici algoritmi di obstacle avoidance che si possa immaginare è il Bug algorithm. L'idea alla sua base è quella di seguire il contorno di ogni ostacolo che si trova sulla strada del robot e quindi di circumnavigarlo. Il robot si dirige verso l'obiettivo finché non incontra un ostacolo. A questo punto inizia a seguire il perimetro dell'oggetto, ma si allontana da esso non appena è in grado di proseguire sulla retta che unisce il punto di partenza con il punto di arrivo. Il processo si ripete se sono presenti ulteriori ostacoli lungo la traiettoria, e termina quando viene raggiunto l'obiettivo.

In generale questo algoritmo è completo, in quanto è in grado di trovare una soluzione al problema dell'obstacle avoidance, ma presenta numerosi svantaggi che lo rendono inapplicabile in un contesto reale. Le traiettorie calcolate infatti sono spesso inefficienti, non ottimali e troppo lunghe. Inoltre, l'algoritmo non è applicabile nel caso di ostacoli in movimento, in quanto il robot è di per sè pericolosamente vicino agli ostacoli e non sarebbe in grado di reagire in tempo ad un loro movimento repentino.

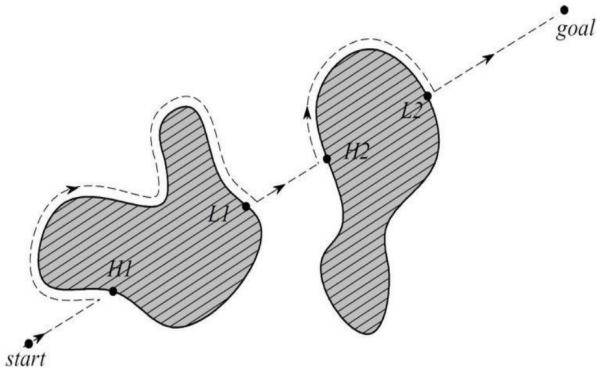

Figura 18: Esempio della traiettoria percorsa da un robot che implementa il Bug algorithm. [9]

Un algoritmo più articolato, che però presenta ancora una serie di inefficienze, è il potential field method. Esso prevede la definizione, tramite le misurazioni del sensore di distanza equipaggiato sul veicolo, di una mappa locale di dimensioni ridotte intorno alla posizione del robot. Successivamente, facendo una similitudine con una carica immersa in un campo elettrico, il robot viene

visto come un punto sottoposto a un campo di forze. Gli ostacoli causano una forza repulsiva nei confronti del robot, inversamente proporzionale alla distanza, mentre il goal point crea una forza attrattiva. La risultante delle forze, ottenuta sommando vettorialmente tutti i contributi, è utilizzata per stabilire la direzione e la velocità di spostamento del veicolo, in modo che esso eviti la collisione con gli ostacoli circostanti.

Il potential field method è un algoritmo più avanzato rispetto al Bug algorithm, ma presenta comunque delle inefficienze: in particolare rimane intrappolato in alcune situazioni, dette di minimo locale. Ad esempio, si osservi il caso rappresentato in figura 19, in cui due ostacoli della stessa dimensione sono posti in maniera simmetrica rispetto al segmento che unisce il robot al punto obiettivo. In questo caso, la risultante delle forze repulsive è uguale e opposta alla forza attrattiva esercitata dal goal point: la logica del potential field method suggerisce perciò che il veicolo debba fermarsi, mentre potrebbe tranquillamente procedere nello spazio compreso tra i due ostacoli. Un altro problema che si può verificare è quello delle oscillazioni in presenza di passaggi stretti. Immaginiamo ad esempio che il robot debba attraversare un corridoio: la traiettoria ideale passa per la linea mediana tra i due muri, ma se per qualsiasi motivo c'è una leggera deviazione da essa, il robot sentirà una forte repulsione da parte del muro vicino, e sarà spinto verso quello opposto. Considerando la dinamica e l'inerzia del veicolo, risulta difficile riallineare il veicolo sulla linea mediana, e il comportamento generale del robot sarà oscillatorio e instabile.

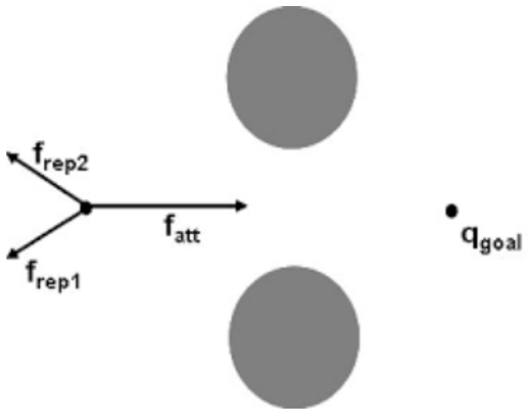

Figura 19: Esempio di una situazione di minimo locale, in cui il robot rimane intrappolato.

2.4.2 Follow the gap method

L'algortimo Follow the Gap [18],[19] è una logica di obstacle avoidance che si basa sulla creazione di un array di gap intorno al veicolo, e sul calcolo dell'angolo di svolta per dirigere il robot verso l'apertura più adeguata, garantendo il raggiungimento del punto di arrivo.

Supponiamo che, indipendentemente dalla loro forma, il robot e gli ostacoli siano considerati circolari (ad esempio identificando la circonferenza di raggio minimo che li circoscrive). Utilizziamo un sistema di riferimento cartesiano per definire il problema: la posizione del robot e il suo raggio sono dati da $(X_{rob}, Y_{rob}, r_{rob})$, mentre per gli ostacoli si avrà $(X_{obs}, Y_{obs}, r_{obs})$. Consideriamo inoltre le seguenti ipotesi:

- Il campo visivo (field of view) del sensore montato sul veicolo ha un'apertura limitata da due angoli limite (uno destro e uno sinistro) $\phi_{fov,l}$ e $\phi_{fov,r}$ ed è limitato ad una distanza massima di misurazione pari a d_{fov}.
- Il robot presenta un vincolo cinematico e ha un raggio minimo di sterzatura pari a r_{min}.
- Si adotta il cosidetto "point robot approach", che consiste nel considerare il robot puntiforme e nell'ingrandire gli ostacoli di una quantità pari a r_{rob}.

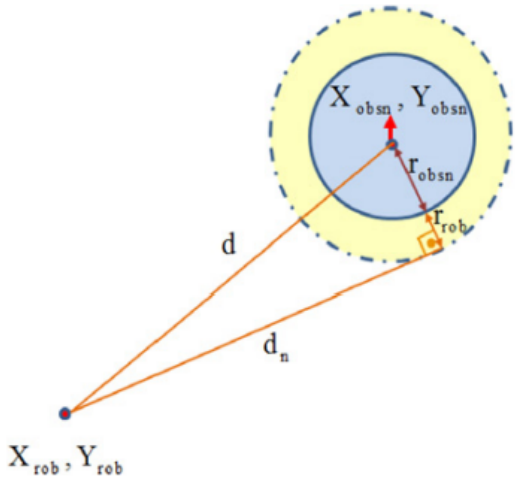

Figura 20: Raffigurazione del "point robot approach" [18].

Partendo da questi presupposti, l'obiettivo è di raggiungere le coordinate del punto obiettivo, evitando gli ostacoli mantenendo una distanza di sicurezza da questi e considerando i vincoli dovuti al campo visivo del sensore e al raggio minimo di sterzatura del veicolo.

Successivamente alla definizione dell'ambiente esterno, sfruttando le misurazioni del sensore di distanza, e dopo aver utilizzato il "point robot approach", ingrandendo gli ostacoli di un raggio pari a r_{rob}, si individuano tutte le aperture possibili in cui il veicolo potrebbe potenzialmente dirigersi. Un'apertura, o gap, è lo spazio libero presente tra due ostacoli. Per gli ostacoli alle estremità, però, ci può essere anche un limite dovuto al campo visivo ridotto o al raggio minimo di sterzatura del veicolo.
Ogni gap è definito dai due angoli che lo delimitano. Per i gap tra due ostacoli, si utilizzano gli angoli del bordo sinistro e destro dell'ostacolo, denominati $\phi_{obs,l}$ e $\phi_{obs,r}$. Per il primo e l'ultimo ostacolo, invece, è necessario stabilire se il limite è causato dal campo visivo ridotto (e quindi da ϕ_{fov}) o dal raggio minimo di sterzatura (e quindi da ϕ_{nhol}). La scelta è effettuata tramite la seguente regola:

$$\begin{cases} d_{nhol} < d_{fov} \Longrightarrow \phi_{lim} = \phi_{nhol} \\ d_{fov} < d_{nhol} \Longrightarrow \phi_{lim} = \phi_{fov} \end{cases}$$

Dove:

ϕ_{lim} è l'angolo di bordo del gap.

ϕ_{nhol} è l'angolo di bordo che deriva dal vincolo cinematico del raggio di sterzatura.

ϕ_{fov} è l'angolo di bordo legato all'apertura del campo visivo.

d_{nhol} è la distanza tra l'arco di raggio r_{min} e l'ostacolo all'estremità.

d_{fov} è la distanza tra la semiretta di apertura del campo visivo e l'ostacolo.

Considerando l'esempio di figura 21, notiamo che, oltre ai gap centrali tra gli ostacoli 1-2 e 2-3, si formano anche dei gap alle estremità: a sinistra, sull'ostacolo 1, il vincolo più stringente è quello dovuto al raggio minimo di sterzatura del veicolo (la cui circonferenza è di colore rosa), mentre a destra, sull'ostacolo 3, il vincolo più stringente è dovuto all'apertura limitata del campo visivo del sensore (la cui semiretta è colorata in verde).

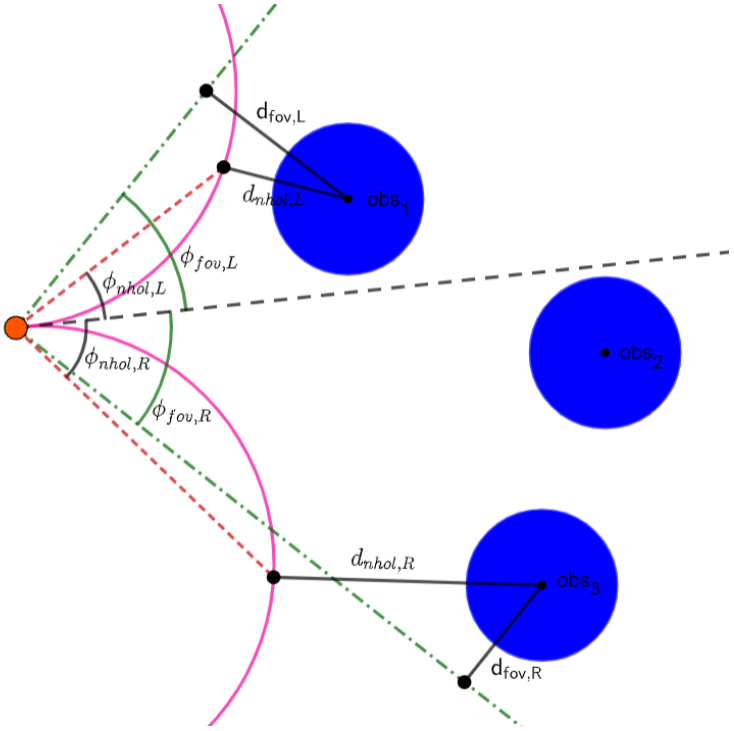

Figura 21: Esempio per la determinazione dei bordi dei gap, considerando il campo visivo del sensore e il raggio minimo di sterzatura del robot.

Si ottiene così un vettore, che per ogni gap contiene l'informazione relativa all'angolo limite sinistro e destro. É facile notare che se sono presenti N ostacoli, si ottengono $N + 1$ gap.

Una volta individuati tutti i gap, è essenziale trovare per ognuno di essi il gap center angle ($\phi_{gap,c}$), ossia l'angolo formato dal vettore che collega il robot al punto centrale del gap (definito come il punto medio del segmento tra i due ostacoli che formano il gap). Il gap center angle deve essere espresso in funzione delle grandezze note, che sono le distanze e gli angoli, misurati dal sensore, a cui

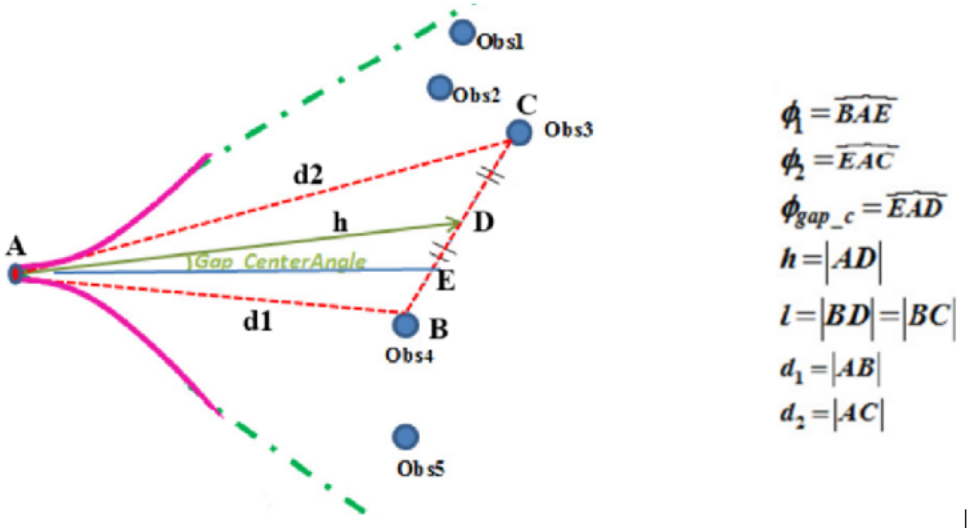

Figura 22: Grandezze geometriche utilizzate per il calcolo di $\phi_{gap,c}$ [18].

si trovano gli ostacoli che delimitano il gap. Esse sono rappresentate in figura 22, e sono d_1, d_2, ϕ_1, ϕ_2.

Un primo approccio per il calcolo di $\phi_{gap,c}$ è quello di fare la media tra gli angoli ϕ_1 e ϕ_2: in questo modo però si calcola l'angolo della bisettrice dell'angolo $C\hat{A}B$, facendo un'approssimazione che in alcuni casi può far avvicinare troppo il robot ad un ostacolo, invece di farlo dirigere verso il centro effettivo dell'apertura.
Il metodo rigoroso per il calcolo di $\phi_{gap,c}$, invece, si basa su una serie di relazioni geometriche. Per prima cosa si applica il teorema del coseno al triangolo ABC:

$$(2l)^2 = d_1^2 + d_2^2 - 2d_1d_2cos(\phi_1 + \phi_2) \Rightarrow l^2 = \frac{d_1^2 + d_2^2 - 2d_1d_2cos(\phi_1 + \phi_2)}{4} \quad (3)$$

Si applica successivamente il teorema di Apollonio al triangolo ABC e si sostituisce l^2 con Eq. (3):

$$d_1^2 + d_2^2 = 2l^2 + 2h^2 \Rightarrow h^2 = \frac{d_1^2 + d_2^2 + 2d_1d_2cos(\phi_1 + \phi_2)}{4} \quad (4)$$

A questo punto si applica il teorema del coseno al triangolo ABD:

$$l^2 = d_1^2 + h^2 - 2d_1hcos(\phi_1 + \phi_{gap,c}) \quad (5)$$

Infine, sostituendo in Eq. (5) le espressioni di l^2 e h^2 ottenute in Eq. (3) e (4) ed esplicitando $\phi_{gap,c}$ si arriva alla conclusione:

$$\phi_{gap,c} = arcos\left(\frac{d_1 + d_2 cos(\phi_1 + \phi_2)}{\sqrt{d_1^2 + d_2^2 + 2d_1 d_2 cos(\phi_1 + \phi_2)}}\right) - \phi_1 \tag{6}$$

L'ultimo step dell'algoritmo prevede di individuare, all'interno dell'array con tutti i gap, quello migliore, e di selezionare la direzione di svolta finale. La scelta del gap migliore è effettuata tramite una funzione che considera la dimensione del gap (d_{gap}) e di quanto esso sia orientato verso il goal point. Vengono premiati i gap più ampi (attraverso i quali si passa più in sicurezza) e quelli in direzione del goal point (tramite i quali si ottiene un percorso più veloce e diretto verso l'obiettivo). La funzione utilizzata per compiere la scelta è la seguente:

$$U_n = k_1 d_{gap,n} + k_2(\pi - \phi_{gap\_n,to\_goal})$$

Dove $\phi_{gap\_n,to\_goal} = |\phi_{gap,c,n} - \phi_{goal}|$ è la differenza tra il gap center angle dell'n-esimo gap rispetto all'angolo che il robot forma con il goal point (ϕ_{goal}). k_1 e k_2 sono due costanti che spostano il peso della decisione più verso l'ampiezza del gap piuttosto che sulla sua orientazione verso l'obiettivo.
Un accorgimento che si adotta per evitare che tra un'iterazione e l'altra ci sia un continuo cambio di scelta del gap, con conseguente comportamento "indeciso" e a zig-zag del robot, è quello di assegnare, unicamente al gap precedentemente scelto, una ricompensa R da sommare all'effettivo valore di U_n: in questo modo si premia il vecchio gap e si disincentiva un repentino cambio tra gap con valori di U_n simili. Per effettuare un cambiamento di gap è necessario quindi che quello vecchio diventi molto sconveniente, o che se ne trovi uno nuovo molto più conveniente.
La funzione U_n viene calcolata per tutti gli n gap e viene scelto quello con il valore maggiore. Una volta individuato il gap migliore, si calcola la direzione di svolta finale ϕ_{final}, facendo una media ponderata tra $\phi_{gap,c}$ e ϕ_{goal} in funzione della distanza dall'ostacolo più vicino al robot d_{min}. Se gli ostacoli sono vicini

al robot, quest'ultimo deve considerare prima la sicurezza, e dirigersi verso il centro del gap. Man mano che la distanza dagli ostacoli aumenta, si può virare leggermente verso il goal point, ottimizzando la traiettoria. L'espressione di ϕ_{goal} è la seguente:

$$\phi_{goal} = \frac{\frac{\alpha}{d_{min}}\phi_{gap,c} + \phi_{goal}}{\frac{\alpha}{d_{min}} + 1}$$

Dove il parametro α è un coefficiente che sposta la scelta più verso il centro del gap oppure verso il goal point.

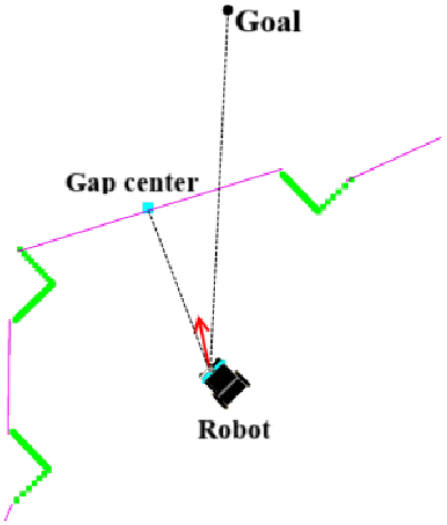

Figura 23: Rappresentazione del calcolo della direzione finale di svolta [19].

L'algoritmo follow the gap presenta diversi vantaggi, tra cui un numero ridotto di parametri di regolazione, che rende più facile e veloce il settaggio in campo del robot. Inoltre non presenta problemi di minimo locale, come ad esempio il potential field method. Un altro aspetto rilevante è che considera la cinematica del veicolo ed il suo raggio minimo di sterzatura, e mette in conto la possibilità che il cono visivo del sensore di distanza possa essere limitato. Infine, premiando i gap più ampi e scegliendo una direzione di svolta il più possibile vicina al centro del gap, si dimostra essere molto sicuro, massimizzando quando necessario la distanza dagli ostacoli circostanti.

2.4.3 Algoritmo VFH+ e VFH*

L'algoritmo VFH+ [20],[21] (Vector Field Histogram) è un metodo di obstacle avoidance che registra le informazioni dell'ambiente esterno su una griglia cartesiana, che viene aggiornata continuamente attraverso i dati provenienti dal sensore di distanza equipaggiato sul veicolo. Il metodo impiega poi un processo di riduzione dei dati in quattro fasi, al fine di calcolare i comandi di controllo per evitare la collisione con gli ostacoli circostanti. Nella prima fase, la griglia viene trasformata in un istogramma polare unidimensionale, costruito intorno alla posizione momentanea del robot. Ogni settore dell'istogramma contiene un valore che rappresenta la densità polare degli ostacoli in quella direzione. Nella seconda fase l'istogramma viene reso binario, ossia i settori vengono considerati vuoti o occupati. Nella terza fase, tramite alcune considerazioni sui vincoli cinematici del veicolo, si modifica l'istogramma polare, mascherando le direzioni irraggiungibili dal veicolo. Nell'ultima fase l'algoritmo seleziona, tra tutti i settori dell'istogramma polare, quello più adatto, e lo sterzo del veicolo viene allineato a quella direzione.

Il primo stadio dell'algoritmo prevede la rappresentazione dell'ambiente circostante il veicolo tramite l'aggiornamento in real time di una griglia cartesiana suddivisa in celle. All'interno di questa griglia si seleziona la cosiddetta active window, ossia una regione quadrata di dimensioni $w_s \times w_s$ e centrata nella posizione momentanea del veicolo, che si muove solidalmente ad esso. Similmente ad una occupancy grid map, ogni cella $C_{i,j}$ è caratterizzata da un valore $c_{i,j}$, a cui corrisponde la probabilità di trovare in quella posizione un ostacolo. Ad ogni lettura del sensore di distanza, la cella interessata viene incrementata di valore, aumentando la probabilità di trovare in essa un ostacolo.

Il passaggio successivo consiste nel creare un istogramma polare unidimensionale a partire dall'active window. Ogni cella $C_{i,j}$ appartenente all'active window viene

vista come un vettore, con direzione data dall'angolo:

$$\beta_{i,j} = arctan\frac{y_j - y_0}{x_i - x_0}$$

e modulo pari a:

$$m_{i,j} = c_{i,j}^2\dot{(}a - bd_{i,j}^2)$$

Dove:

a, b sono costanti positive

x_0, y_0 sono le coordinate attuali del centro del robot

x_i, y_j sono le coordinate della cella $C_{i,j}$

$d_{i,j}$ è la distanza tra la cella $C_{i,j}$ e il centro del robot

$\beta_{i,j}$ è l'angolo tra la cella $C_{i,j}$ e il centro del robot

Le costanti a e b sono ottenute in modo tale che $m_{i,j} = c_{i,j}^2$ nelle celle perimetrali della active window, in cui $d_{i,j} = \frac{w_s-1}{2}$. Quindi va rispettata la relazione:

$$a - b(\frac{w_s - 1}{2})^2 = 1$$

Il valore di $m_{i,j}$ decresce poi man mano che la cella si avvicina al robot: in questo modo gli ostacoli vicini sono pesati di più e sono considerati con una pericolosità maggiore. Inoltre, $m_{i,j}$ è proporzionale al quadrato di $c_{i,j}$, in modo tale da rinforzare il fatto che, ad un numero di misurazioni di distanza elevato, corrisponde una maggiore probabilità di trovare un ostacolo.

A questo punto l'active window viene suddivisa in un numero intero n di settori angolari, di ampiezza α, e per ogni settore k viene calcolata la densità polare degli ostacoli H_k. Questa non è che la somma di tutti i moduli $m_{i,j}$ delle celle appartenenti al k-esimo settore. Si nota come una finestra di forma circolare sarebbe geometricamente più appropriata, ma è computazionalmente più costosa da gestire rispetto a una finestra quadrata.

Inoltre, per tenere conto delle dimensioni del robot, ogni cella è ingrandita di un raggio pari a $r_{r+s} = r_r + d_s$, dove r_r è il raggio del robot e d_s è una distanza

di sicurezza. Così facendo, si considera il robot come un'entità puntiforme, e si "gonfiano" gli ostacoli circostanti. Il valore $m_{i,j}$ di una cella non solo viene aggiunto al settore a cui la cella appartiene, ma anche a tutti i settori adiacenti interessati dalla "bolla" gonfiata della cella stessa.
Per ogni cella si avrà che l'angolo di ingrandimento è pari a:

$$\gamma_{i,j} = arcsin \frac{r_{r+s}}{d_{i,j}}$$

Per ogni settore k, la densità polare degli ostacoli è poi calcolata con:

$$H_k = \sum_{i,j} m_{i,j} h_{i,j}$$

Dove:

$$\begin{cases} h_{i,j} = 1 & se \quad k\alpha \in [\beta_{i,j} - \gamma_{i,j}, \beta_{i,j} + \gamma_{i,j}] \\ h_{i,j} = 0 & altrimenti \end{cases}$$

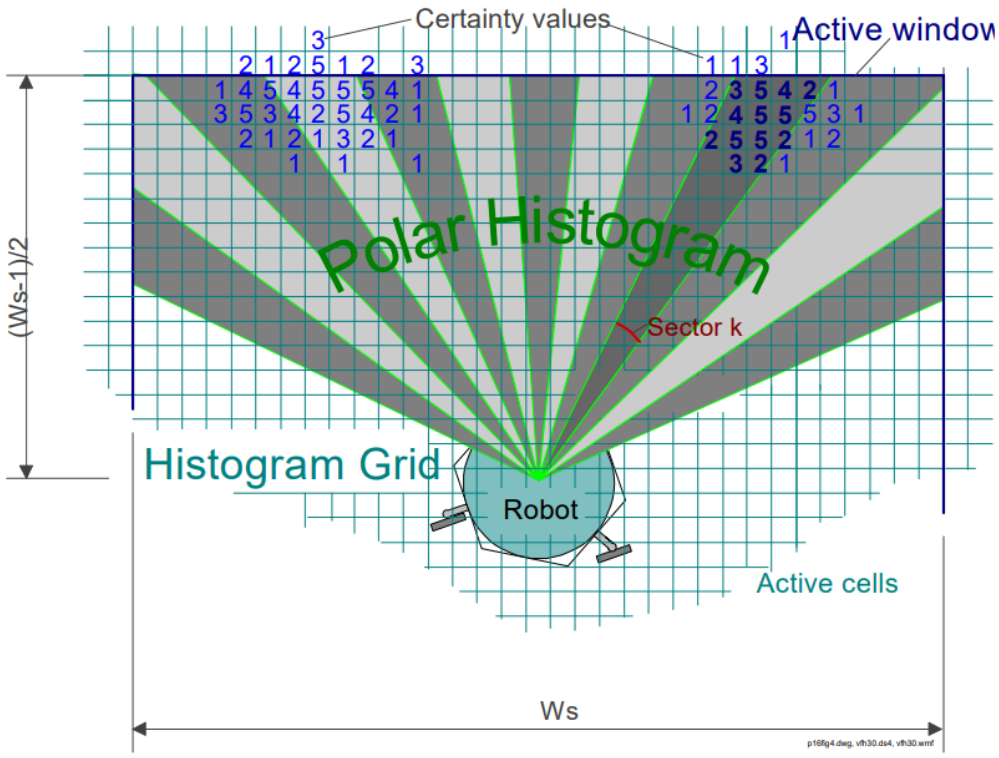

Figura 24: Rappresentazione dell'active window e del processo di suddivisioni in settori angolari per la creazione dell'istogramma polare [20].

Nell'istogramma polare ottenuto, ad ogni settore angolare corrisponde una colonna più o meno alta in base alla quantità e pericolosità degli ostacoli presenti. É utile, a questo punto, distinguere i settori liberi da quelli in cui sono presenti ostacoli. Per fare ciò è necessario scegliere un metodo per eseguire la classificazione: la scelta di un'unica soglia che distingue un settore libero da uno occupato potrebbe portare a problemi di instabilità, in cui un settore oscilla continuamente da uno stato libero a uno occupato e viceversa. Si individuano perciò due soglie, τ_{high}, al di sopra della quale un settore viene considerato occupato (1), e τ_{low}, al di sotto del quale un settore viene considerato vuoto (0). Se il valore di densità polare degli ostacoli è compreso tra le due soglie, il settore viene considerato uguale a come era nella precedente iterazione. Si ottiene così un istogramma polare binario.

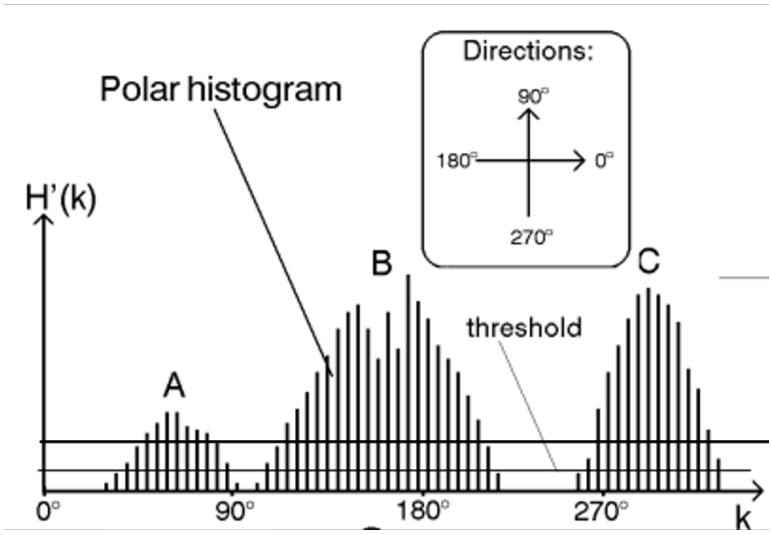

Figura 25: A partire dall'istogramma polare, i settori vengono considerati completamente liberi o occupati in base al loro valore di densità polare degli ostacoli rispetto alle due soglie τ_{high} e τ_{low} opportunamente scelte [20].

L'istogramma polare binario indica quali direzioni sono libere e quali ocuppate. Non considera però la dinamica e la cinematica del veicolo, presupponendo che esso sia in grado di cambiare istantaneamente la propria direzione di marcia. A meno che il robot non si fermi a ogni campionamento, questa ipotesi è chiaramente violata. La curvatura massima della traiettoria di un robot è spesso una funzione della sua velocità. Il raggio di sterzatura minimo può essere pari

a zero per un robot mobile a trazione differenziale, ma solo se la velocità di traslazione è nulla. Normalmente, più veloce è il robot, minore è la curvatura massima. L'algoritmo VFH+ considera il raggio minimo di sterzatura del veicolo ed esclude tutte le direzioni non raggiungibili dal veicolo, mascherando tutti i settori dell'istogramma polare che richiederebbero una traiettoria dinamicamente non attuabile. Considerando l'esempio di figura 26, l'ostacolo A blocca tutte le direzioni alla sua sinistra, in quanto la circonferenza di raggio minimo di sterzatura interseca lo spazio occupato dall'ostacolo A. Perciò tutte le direzioni oltre A, ossia oltre i 135°, vengono mascherate.

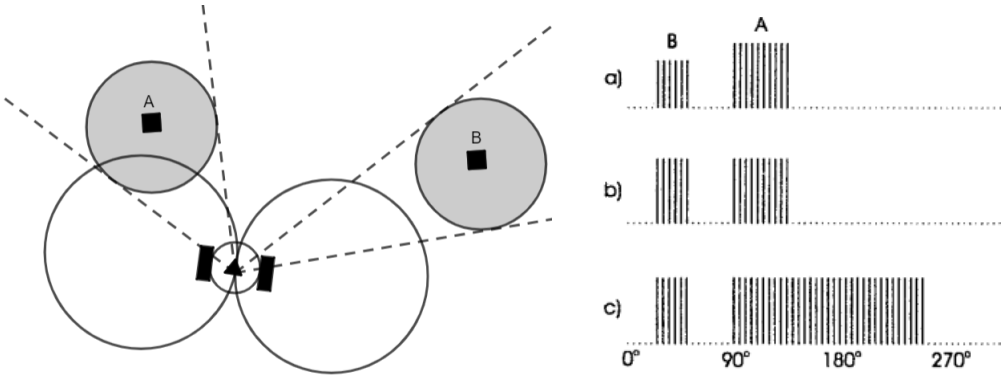

Figura 26: Esempio di costruzione dell'istogramma polare (a), binario (b) e mascherato (c).

Tra i settori liberi individuati nell'istogramma polare mascherato, ci sono alcune direzioni che sono migliori di altre. La fase finale dell'algoritmo prevede di individuare una serie di direzioni di svolta possibili e di selezionare quella più adatta tramite un'opportuna funzione di costo.

Innanzitutto si fa una distinzione tra aperture strette e larghe, in base al numero di settori angolari liberi consecutivi. Si calcolano per ogni apertura i settori di estremità destra k_r e sinistra k_l. Un'apertura è considerata larga se ha più di s_{max} settori consecutivi liberi e si ottiene quando sono presenti ampi spazi vuoti tra gli ostacoli o da situazioni in cui solo un ostacolo è vicino al veicolo. É considerata stretta, invece, se $(k_r - k_l) < s_{max}$, e si ottiene quando il veicolo si sposta in un'area densa di ostacoli.

All'interno di ogni apertura bisogna individuare il settore più adatto verso cui dirigersi, chiamato c (direzione candidata). Per un'apertura stretta, la priorità è

di stare alla larga dagli ostacoli e quindi si svolta verso il centro dell'apertura.

$$c = \frac{k_r + k_l}{2}$$

Per un'apertura larga, invece, ci sono diverse opzioni: si può far andare il robot verso l'estremità destra o sinistra dell'apertura, mantenendo l'ostacolo ad una certa distanza di sicurezza, oppure, se la direzione obiettivo del target k_t è piuttosto centrale, la si può direttamente seguire.

$$\begin{cases} c_r = k_r + \frac{s_{max}}{2} & \textit{verso il lato destro} \\ c_l = k_l - \frac{s_{max}}{2} & \textit{verso il lato sinistro} \\ c_t = k_t \qquad \textit{se} & k_t \in [c_r, c_l] \end{cases}$$

Successivamente, è necessario definire una funzione di costo, in modo che il robot selezioni la direzione candidata più appropriata (ossia con il costo minore) e la imposti come direzione di svolta ϕ_d. Una possibile funzione di costo è la seguente:

$$g(c) = \mu_1 \cdot \Delta(c, k_t) + \mu_2 \cdot \Delta(c, \frac{\theta_{current}}{\alpha}) + \mu_3 \cdot \Delta(c, k_{d,t-1})$$

Dove $\Delta(c_1, c_2)$ è una funzione che calcola il minimo numero di settori angolari compresi tra i due settori c_1 e c_2.

Il primo termine della funzione rappresenta il costo associato alla differenza tra una direzione candidata e la direzione di destinazione. Più grande è questa differenza, tanto più il robot si dirigerà lontano dalla direzione di destinazione, e quindi più grande sarà il costo. Il secondo termine rappresenta il costo associato alla differenza tra una direzione candidata e l'orientamento attuale del robot. Più grande è questa differenza, più il robot dovrà sterzare bruscamente. Il terzo termine rappresenta il costo associato alla differenza tra una direzione candidata e la direzione di movimento selezionata nell'iterazione precedente (al tempo $t-1$). Questo termine scoraggia grandi cambiamenti di direzione tra un'iterazione e la successiva, evitando problemi di instabilità e cambiamenti continui nello sterzo

del veicolo. Le costanti μ_1, μ_2, μ_3 sono scelte in modo tale da dare più peso ad un termine piuttosto che ad un altro: Più μ_1 è alto, più il comportamento del robot è orientato verso il target point. Più μ_2 è alto, tanto più il robot cerca di di eseguire un percorso fluido con un cambio minimo di direzione del movimento. Più è alto μ_3, più il robot cerca di dirigersi verso la direzione selezionata in precedenza.

L'algoritmo VFH+ è alquanto sofisticato e offre buone prestazioni in numerosi scenari, riducendo al minimo i problemi in casi di "minimo locale". Un robot controllato da VFH+ può raramente rimanere intrappolato in situazioni di vicolo cieco. Sebbene sia possibile elaborare una serie di regole euristiche che guidino il robot fuori dalle situazioni di trappola, il percorso risultante non è comunque ottimale. Quando viene segnalata una situazione di intrappolamento, un'opzione è di rallentare e fermare il robot, sospendendo temporaneamente l'algoritmo VFH+ e attivando un algoritmo di global path planning, per pianificare un nuovo percorso basandosi sulle informazioni disponibili.
Un'alternativa è quella di combinare contemporaneamente un modulo di global path planning con un modulo di obstacle avoidance. Un semplice modulo di obstacle avoidance, infatti, essendo di natura puramente locale, presenta intrinsecamente dei limiti. Osserviamo ad esempio il caso di figura 27, in cui è raffigurata una situazione in cui un robot percorre un corridoio e incontra due ostacoli sul suo percorso. Gli ostacoli sono indicati in nero, mentre in grigio è rappresentata la bolla di ingrandimento che permette di considerare il robot come un punto nello spazio. Il cerchio tratteggiato indica la distanza alla quale un ostacolo innesca una manovra di obstacle avoidance. Nella posizione mostrata nell'esempio, VFH+ rileva due aperture. Mentre l'apertura a sinistra dà luogo alla traiettoria indesiderata A, l'apertura a destra dà luogo alla traiettoria B, che al punto p si trasforma infine nella traiettoria desiderabile C. In situazioni problematiche come questa, VFH+ selezionerebbe la direzione B in media solo il 50% delle volte. È importante notare che questo non è un difetto solo dell'algoritmo VFH+, ma di tutti gli algoritmi di natura puramente locale. È anche importante notare che una distanza di innesco maggiore non eliminerebbe

il problema, ma lo anticiperebbe solo.

L'algoritmo VFH* [22] è in grado di superare situazioni problematiche come questa combinando VFH+ con l'algoritmo di ricerca A*. Esso, prima di eseguire la scelta finale della direzione svolta, analizza le conseguenze del dirigersi verso ogni direzione candidata, proiettando la traiettoria del robot diversi passi avanti e valutando possibili scenari futuri. Per ogni direzione candidata, VFH* calcola la nuova traiettoria applicando A* su una distanza totale pari a d_t, eseguendo step di lunghezza d_s. Vengono eseguite $n_g = \frac{d_t}{d_s}$ iterazioni, in cui ad ogni step viene riapplicato VFH+, basandosi sulle informazioni presenti sulla mappa. Al termine delle n_g iterazioni, si valuta la traiettoria migliore tramite una funzione di costo, similmente a come accadeva in VFH+.

Affinchè VFH* funzioni in real-time mentre il veicolo è in movimento, è necessaria una velocità di aggiornamento elevata: per tale motivo è fondamentale una scelta opportuna dei parametri d_s e n_g. Fissata l'ampiezza di uno step d_s, un numero troppo elevato di step n_g rallenterebbe troppo l'algoritmo, mentre un numero basso di n_g causerebbe una ricerca inefficace.

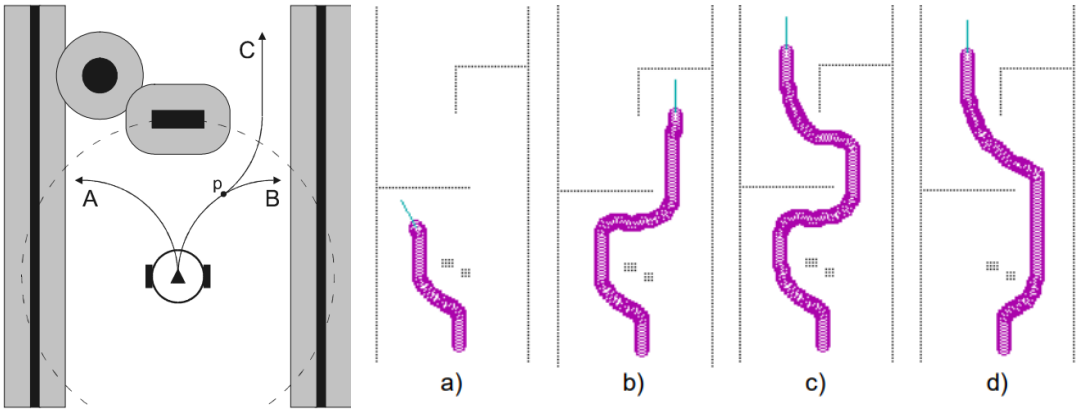

Figura 27: Esempio che dimostra l'efficacia dell'algoritmo VFH* rispetto a VFH+. A destra, risultati ottenuti con n_g pari a (a) 1, (b) 2, (c) 5, (d) 10. [22]

3 Progetto e algoritmo di guida autonoma

Il caso studio dell'attività di questa tesi è il rover elettrico a guida autonoma sviluppato dall'azienda Ecothea S.r.l. Si tratta di una piattaforma di medio-piccole dimensioni e di tipo modulare, che in futuro potrà essere equipaggiata con diverse tipologie di strumenti e a cui si potranno agganciare macchine operatrici di bassa potenza. Il veicolo, nella fase prototipale iniziale, sarà in grado di svolgere compiti basilari, come monitoraggio e trasporto di carichi lievi, ma l'architettura modulare permetterà in futuro di adattarlo a diverse altre funzioni, come ad esempio trattamenti di nebulizzazione, diserbo meccanico, o piantumazione di ortaggi. Si è quindi optato per la progettazione di un mezzo polivalente, che possa inizialmente sostituire la manodopera umana in una serie di operazioni basilari, che nel tempo aumenteranno la loro complessità. Il mezzo dovrà integrarsi in un ambiente agricolo tradizionale e non ancora completamente robotizzato, perciò saranno d'obbligo alcuni accorgimenti specifici, in particolare per quanto riguarda l'adattamento ad un'infrastruttura non specificamente pensata per un veicolo robotico. Il progetto ha anche un occhio di riguardo al tema della sostenibilità, in quanto il mezzo è a propulsione elettrica, il che prevede una motorizzazione e una alimentazione degli accessori di bordo totalmente elettrica.

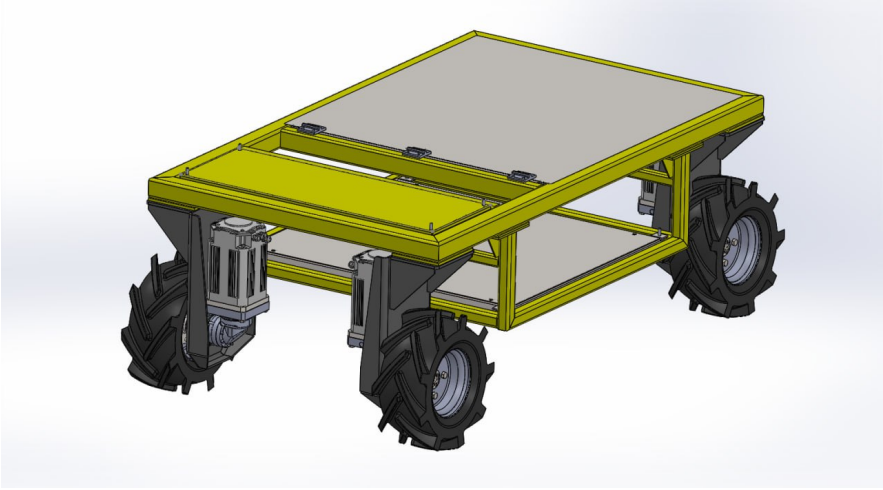

Figura 28: Rappresentazione CAD della piattaforma robotica in forma prototipale oggetto di studio.

3.1 Architettura del rover

Il rover è costituito da un telaio di forma rettangolare di dimensioni $1 \times 1.5m$. Per quanto riguarda le operazioni di guida autonoma, è importante considerare gli ingombri in lunghezza e larghezza della piattaforma, in quanto hanno un ruolo non trascurabile nelle manovre di movimentazione del veicolo. Risulta importante anche conoscere la sua altezza, così come le wheel e ground clearances, ossia le quote che determinano se il veicolo è in grado di superare un ostacolo dall'altezza ridotta.
Sul telaio sono montate in maniera simmetrica quattro ruote fisse, ciascuna delle quali è azionata da un singolo motore elettrico. Il veicolo è perciò a trazione differenziale: le ruote essendo fisse e non pivotanti, non sono in grado di ruotare intorno ad un asse verticale, e perciò le operazioni di sterzatura vengono eseguite tramite il cosiddetto torque vectoring. Quest'ultimo è un processo di gestione della coppia fornita ad ogni singolo motore, che permette di controllare il verso e la velocità di rotazione delle ruote. I vantaggi di una configurazione di questo tipo risiedono innanzitutto in un montaggio ed un controllo del moto più semplici, in quanto le ruote hanno un unico grado di libertà. Inoltre, un veicolo a trazione differenziale è in grado di ruotare su se stesso da fermo. Anche se il raggio minimo di sterzatura aumenta all'aumentare della velocità longitudinale del veicolo, per valori bassi di velocità esso si può considerare nullo. Il principale svantaggio di un veicolo a guida differenziale consiste invece nelle grandi forze di attrito e lo slittamento laterale degli pneumatici, che sono maggiormente sollecitati e soggetti ad usura.

Uno degli aspetti più importanti per il funzionamento di un sistema autonomo è l'acquisizione di informazioni dall'ambiente circostante. Esse sono fondamentali per avere una conoscenza dettagliata dello spazio in cui il robot si muove. Un veicolo a guida autonoma deve quindi possedere un elaborato sistema di sensori, che misurano diversi parametri e forniscono i dati necessari a prendere le dovute decisioni.

Innanzitutto è necessario conoscere la posizione del rover. Il problema della localizzazione è risolto tramite un sensore GPS (Global Positioning System). Il suo funzionamento è basato sulla trilaterazione dei segnali satellitari: ogni satellite emette ripetutamente un segnale radio contenente informazioni riguardanti l'orario di emissione e la posizione orbitale del satellite. Il ricevitore registra l'ora di arrivo del messaggio e per confronto calcola il tempo di volo del segnale. Stimando la velocità di propagazione dell'onda elettromagnetica, assunta uguale alla velocità della luce, è quindi possibile risalire alla distanza tra il ricevitore e il satellite da cui ha ricevuto il segnale. Per conoscere la posizione del ricevitore nello spazio, e quindi le sue coordinate x, y, z rispetto al sistema di riferimento solidale al globo terrestre, idealmente sarebbero sufficienti le informazioni provenienti da soli tre satelliti. La misurazione della distanza tra ricevitore e satellite è però soggetta a numerosi fonti di errore, tra cui:

- Inaccuratezza delle misurazioni temporali: nonostante i satelliti siano equipaggiati con orologi atomici, presentano comunque una leggera deriva nel tempo e devono essere sincronizzati. Inoltre, il ricevitore utilizza un semplice orologio al quarzo, molto meno preciso. per quanto la sua risoluzione sia bassa, essa va moltiplicata per la velocità della luce e produce errori non trascurabili.

- Inaccuratezza della posizione del satellite: mentre i dati relativi al clock interno del satellite vengono inviati con continuità, la posizione orbitale viene inviata con una frequenza minore, alterando quindi il risultato della triangolazione dei segnali.

- Influenza dell'atmosfera terrestre: l'onda radio è soggetta nel suo cammino ad una serie di fattori che ne rallentano o alterano la sua propagazione.

- Multipath: il ricevitore, a priori, non ha modo di stabilire se il segnale ricevuto proviene direttamente dal satellite o se è stato prima riflesso da altre superfici.

Per questo motivo, si sfrutta l'informazione proveniente da un quarto satellite per calcolare un fattore correttivo e ottenere così una stima della posizione più precisa. Nonostante questo, però, i valori misurati sono soggetti a errori dell'ordine del metro, che in campo agricolo e di precision farming non sono adeguati. Un metodo per ottenere precisione centimetrica è fornito dai sistemi GPS-RTK (Real Time Kinematic). Esso prevede l'utilizzo di due ricevitori: il primo, chiamato *base station*, viene posizionato in un punto di coordinate note e vi rimane lì fisso, mentre il secondo, chiamato *rover*, viene montato sul veicolo. Il metodo si basa sul fatto che i due ricevitori, se posti relativamente vicini, vedono un cielo con le stesse caratteristiche fisiche, quindi i segnali ricevuti conterranno le stesse tipologie di errore. Il rover, confrontando le informazioni provenienti dalla base fissa con i dati satellitari ricevuti, può compensare l'errore e ottenere in maniera molto precisa la sua posizione, relativamente alla base. Note poi le coordinate assolute della base, il rover può risalire alla propria posizione assoluta.

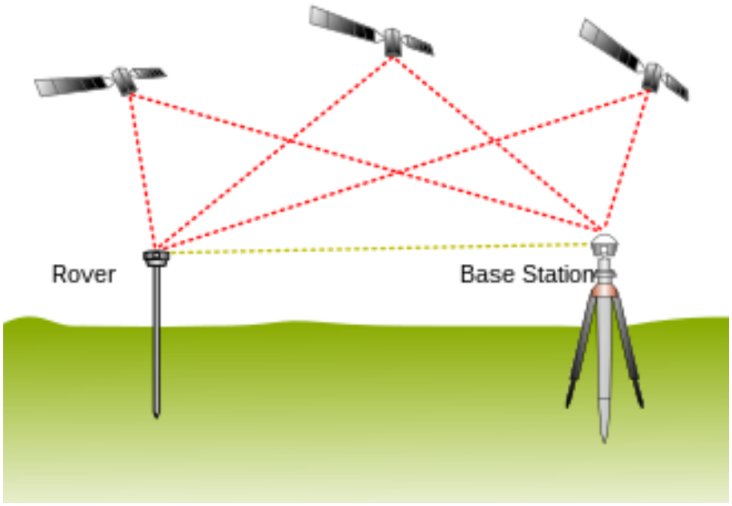

Figura 29: Schema del funzionamento di un sistema di localizzazione di tipo GPS-RTK [23].

Un'altra tipologia di sensori molto utili è quella dei sensori di rilevazione dell'inclinazione e orientazione del veicolo. Oltre a conoscere la posizione del rover, è importante anche avere un'informazione relativa alla sua orientazione nello spazio. A questo scopo si può utilizzare un sensore a effetto Hall, che come una bussola misura la direzione rispetto al campo magnetico terrestre. Un'altra

classe di sensori di questo tipo è data dalle unità di misura inerziali (IMU), costituite da un accelerometro e da un giroscopio. L'accelerometro è in grado di ricavare il valore istantaneo dell'accelerazione del veicolo, che può essere integrata nel tempo per ricavare una stima della posizione del veicolo, utile ad esempio quando il segnale GPS è troppo debole. Il giroscopio, invece, sfrutta le proprietà inerziali dovute al moto di precessione di un rotore, per ricavare l'informazione relativa all'orientazione del veicolo.

Infine, uno dei sensori più importanti nella robotica e nei sistemi a guida autonoma è il sensore di distanza, tramite il quale è possibile mappare l'ambiente circostante e stabilire la presenza di ostacoli. La percezione dello spazio esterno in prossimità del rover svolge un ruolo fondamentale per la fase di mappatura e in particolare per l'attività di obstacle avoidance. Una delle tecnologie più avanzate in questo campo è quella dei Lidar (Light Detection and Ranging) [24]. Un sensore Lidar è costituito da un trasmettitore che emette un fascio laser e da un ricevitore in grado di rilevare la componente riflessa dagli ostacoli. Il calcolo della distanza è ottenuto misurando il tempo impiegato dall'onda elettromagnetica per raggiungere il bersaglio e tornare indietro.

$$d = c \cdot t$$

Dove:

- d è distanza percorsa dal fascio laser.
- c è la velocità della luce.
- t è il tempo trascorso dall'emissione del laser alla ricezione (time of flight).

Un modo per misurare il time of flight del fascio luminoso è quello di utilizzare un laser a impulsi (modulato in ampiezza), ad esempio sfruttando un chip laser montato con condensatori attivati da un transistor MOSFET. Ad ogni apertura del gate, la carica elettrica accumulata nei condensatori viene scaricata nel chip, che emette l'impulso ottico in modo controllato. Un secondo metodo consiste

nell'utilizzare un'onda continua modulata in frequenza (FMCW) e nell'analizzare la differenza tra l'onda emessa e quella riflessa. Un altro metodo, ancora più semplice, consiste nel misurare la differenza di fase dell'onda riflessa.

Ripetendo le misurazioni nello spazio, facendo divergere il fascio laser a diverse angolazioni azimutali e verticali, si ottiene come output una nuvola di punti, a ciascuno dei quali è associato un valore di distanza. L'operazione di scansione dello spazio può avvenire in maniera meccanica o allo stato solido. Attualmente, la soluzione di scansione più diffusa è il sistema di rotazione meccanica, che dirige i fasci laser attraverso un meccanismo rotante (come uno specchio o un prisma) controllato da un motore. Ad esempio, per lo schema di rotazione meccanica illustrato in figura 30.a, un sistema di specchi inclinabili riflette il laser per generare un field of view (FoV) verticale. Il FoV orizzontale è ottenuto facendo ruotare la base del Lidar tramite un motorino elettrico. É possibile anche utilizzare più fasci laser per ridurre la complessità del meccanismo mobile. Gli svantaggi di questa configurazione risiedono nell'ingombro del meccanismo e nella fragilità in caso di vibrazioni.

Un'alternativa per la creazione di un ampio field of view è l'utilizzo di dispositivi MEMS (Microelectromechanical systems). Un sistema di questo tipo contiene ancora parti in movimento, ma si può definire quasi allo stato solido. É costituito da uno specchio incorporato in un chip (figura 30.b), che si inclina sotto l'effetto di due forze opposte: una forza elettromagnetica (forza di Lorentz) prodotta dal passaggio di corrente in una bobina avvolta attorno allo specchio, e una forza elastica causata da una barra di torsione, che funge da asse di rotazione. Controllando la corrente che attraversa la bobina, è possibile far seguire allo specchio una traiettoria di scansione programmata.

Un sistema di scansione completamente allo stato solido, che non ha parti in movimento, è infine illustrato in figura 30.c. Questo sistema prevede la diffusione di un laser attraverso un diffusore ottico, in modo tale da illuminare l'intera scena in una sola volta. L'immagine riflessa dal target viene infine catturata da una matrice 2D di fotodiodi, ed elaborata opportunamente per ottenere la nuvola

tridimensionale di punti. Uno svantaggio di questo metodo è il suo FoV limitato, in quanto non avendo parti rotanti non è in grado di scansionare l'ambiente a 360°.

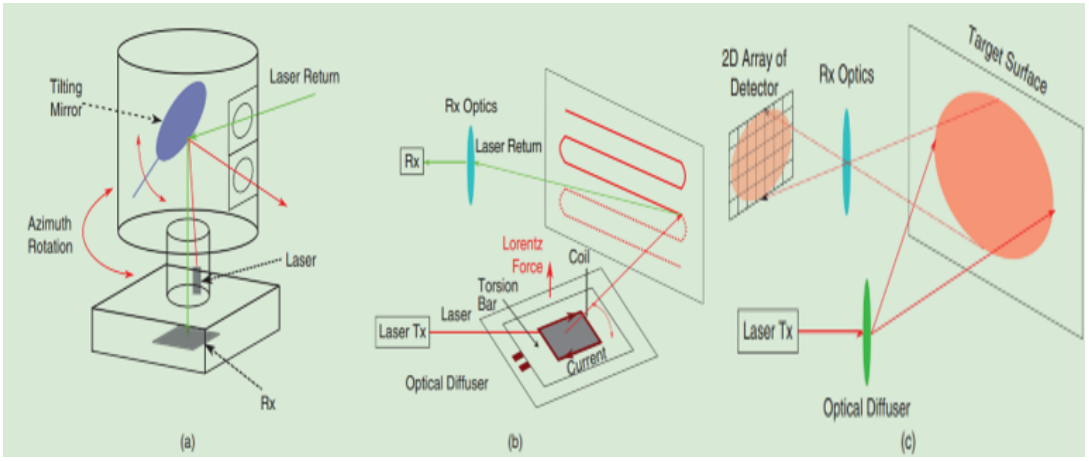

Figura 30: Schema dei principali meccanismi utilizzati nei Lidar per scansionare tridimensionalmente l'ambiente esterno [24]. (a) scansione meccanica, (b) scansione tramite dispositivo MEMS, (c) scansione allo stato solido.

Il fascio laser dei Lidar è caratterizzato da una certa lunghezza d'onda. Questa influisce sul costo del sensore ed è limitata da regolamenti sulla protezione dell'occhio umano. Le più comuni lunghezze d'onda dei dispositivi presenti sul mercato sono 905nm (near infrared o NIR) e 1550nm (short-wave infrared o SWIR). Con una lunghezza d'onda di 1550nm è possibile utilizzare potenze maggiori, per cui è possibile rilevare ostacoli più lontani. I sensori a 1550nm, però, sono più costosi.

Un altro aspetto da considerare nella scelta del Lidar è il suo campo visivo, determinato dall'apertura verticale e orizzontale con cui vengono eseguite le misurazioni. Un campo visivo ampio è desiderabile, in quanto permette di avere un quadro completo dell'ambiente circostante. D'altrocanto, ad un campo visivo ampio corrisponde un numero maggiore di punti acquisiti, che rende laboriosa la fase di elaborazione dei dati, e perciò si rende necessario l'utilizzo di un'elettronica di post-processing dei segnali più complessa e costosa. Il campo visivo è anche influenzato dal luogo di montaggio del sensore sul rover: esso non deve essere ostruito dal carico trasportato durante il funzionamento, e nemmeno da elementi

strutturali del telaio. Un'opzione è quella di montare un sensore a 360°sulla parte alta del veicolo. In alternativa, è possibile montare più sensori, dalle diverse caratteristiche e prestazioni, in diverse parti del rover: ad esempio uno sulla parte anteriore, utilizzato durante la marcia in avanti, e uno su quella posteriore, da utilizzare durante la retromarcia.

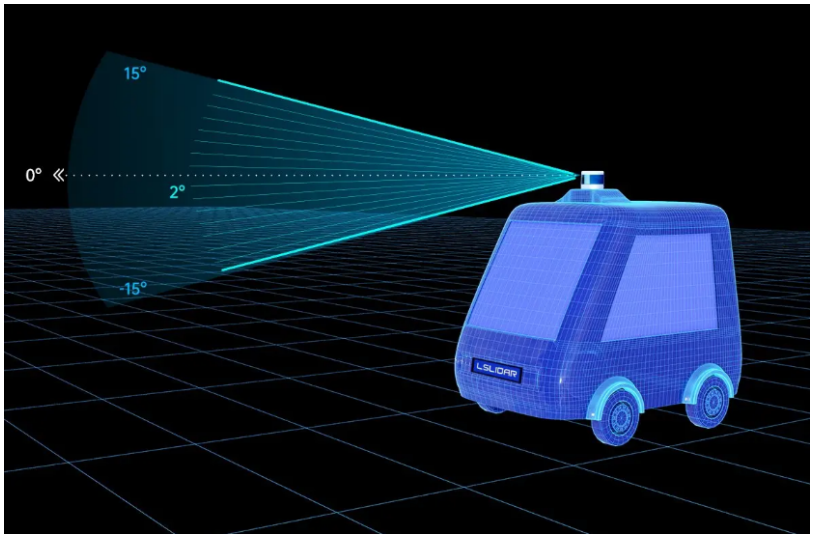

Figura 31: Esempio di montaggio e relativo field of view di un Lidar sul veicolo.[25]

3.2 Mappa dell'ambiente

Nella fase iniziale del progetto, sarà necessario far compiere al prototipo delle manovre basilari, in uno spazio ampio con pochi e semplici ostacoli. Per questo motivo non è di fondamentale importanza avere una mappa dettagliata dell'ambiente. Si è scelto perciò di creare un terreno agricolo virtuale in ambiente MATLAB®. La mappa è stata ottenuta sfruttando la classe `occupancyMap` implementata nel pacchetto NavigationToolBox®del software. La griglia della mappa è costituita da una matrice in cui gli elementi sono impostati a un valore unitario o nullo in base alla presenza o meno di un ostacolo. A titolo esemplificativo è stata creata una mappa che simula un terreno simile ad un vigneto. A partire da una matrice vuota di dimensioni 100×100 m, con 10 celle per metro, si definisce inizialmente un bordo perimetrale che simula una recinzione. Successivamente si creano dei filari equispaziati. Si posizionano inoltre una serie di ostacoli in maniera casuale tra i filari, a simulare possibili oggetti e ostacoli imprevisti che il rover dovrà evitare.

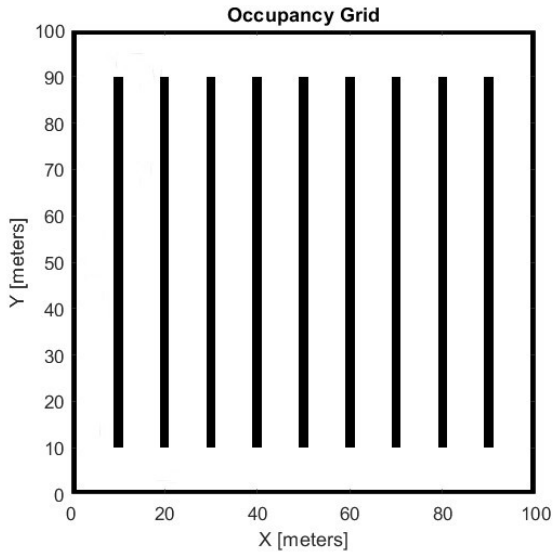

Figura 32: Mappa di esempio utilizzata per simulare l'algoritmo di guida autonoma in questa attività di tesi.

3.3 Global path planning

Nell'algoritmo di guida autonoma implementato sul rover è stato adottato un pianificatore geometrico di Dubins (vedi sezione 2.2.1). La scelta è ricaduta su un pianificatore di questo tipo in quanto, nel periodo iniziale di diffusione e introduzione della tecnologia, la fase di pianificazione del percorso sarà assistita da un operatore umano. In quest'ottica è possibile che l'operatore imposti (tramite tablet o dispositivo mobile) i punti intermedi di passaggio del veicolo, e che il pianificatore geometrico completi autonomamente l'operazione di tracciatura del percorso. Inoltre, in una mappa dalle forme semplici, standardizzate e ripetitive, come quella del vigneto che si è considerato, risulta particolarmente efficace l'utilizzo di un pianificatore geometrico, che sì richiede l'intervento di un operatore nella selezione dei punti intermedi, ma è in grado di calcolare una traiettoria precisa e "pulita" molto rapidamente e con uno sforzo computazionale ridotto.
Come si osserva in figura 33, le tipiche manovre da eseguire in un vigneto sono poche: normalmente le lavorazioni in vigna richiedono di attraversare longitudinalmente lo spazio tra i filari, di uscire da un interfilare e di rientrare in quello adiacente. Per tale motivo è sufficiente definire pochi waypoint e il pianificatore definirà tratti lineari tra i due filari, e archi di circonferenza per l'uscita/entrata tra due interfilari adiacenti.

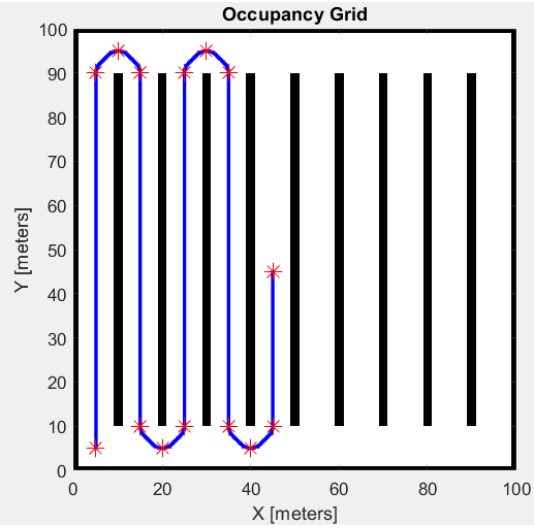

Figura 33: Path planning geometrico nella mappa oggetto di studio. In rosso i waypoint impostati dall'utente, in blu la traiettoria calcolata interpolando i punti con curve geometriche.

Un'altra possibile opzione, applicabile nel periodo iniziale di transizione dall'agricoltura tradizionale a quella automatizzata, è quella di registrare tramite gps la traccia percorsa dalla macchina agricola odierna, e di memorizzarla in modo tale che possa essere replicata dal veicolo a guida autonoma che la sostituirà. Questa strategia ha però il limite di poter essere unicamente utilizzata per operazioni precedentemente registrate, e non permette di calcolare deviazioni o modifiche rispetto alla traccia registrata.

In futuro, nel caso fosse necessario l'utilizzo di un pianificatore dalle migliori prestazioni, sarà consigliabile implementare un algoritmo di tipo RRT*. Tra le opzioni analizzate nello stato dell'arte attuale, è infatti l'algoritmo che fornisce risultati più soddisfacenti in termini di efficienza del percorso trovato. Esso riesce a pianificare un percorso anche in ambienti con un'alta densità di ostacoli. La qualità della traiettoria, però, dipende dal numero di punti con cui la mappa è stata esplorata: per avere curve regolari e non a zig zag, è necessario infittire molto la mappa, con un conseguente innalzamento dello sforzo computazionale e dell'utilizzo di memoria. La struttura ad albero creata, tuttavia, viene calcolata una volta sola e può essere sfruttata per trovare diversi percorsi all'interno della mappa, in base alle esigenze che richiede ciascuna tipologia di missione.

3.4 Algoritmo Pure Pursuit

Come algoritmo di path following si è deciso di implementare un controllore ispirato all'algoritmo Pure Pursuit, che, come visto in sezione 2.3.1, è un metodo sicuro ed efficiente per l'inseguimento di una traiettoria. L'algoritmo calcola la velocità necessaria al veicolo affinché esso segua il percorso precedentemente elaborato nella fase di global path planning, costituito da una serie di punti in un piano, chiamati waypoint e memorizzati in un omonimo vettore. La velocità longitudinale è assunta costante; l'obiettivo consiste quindi nell'individuazione dell'opportuna velocità di imbardata del veicolo.

Il funzionamento del metodo prevede il calcolo di un goal point sul percorso da seguire, ad una certa distanza, chiamata "lookahead distance", dalla posizione attuale del robot. Successivamente si calcola la velocità di imbardata necessaria per dirigersi verso il goal point appena individuato. A seguito dello spostamento del robot, viene trovato un nuovo goal point sul percorso e la velocità di imbardata viene aggiornata. In questo modo, punto dopo punto, il veicolo rintraccia il percorso e lo segue.

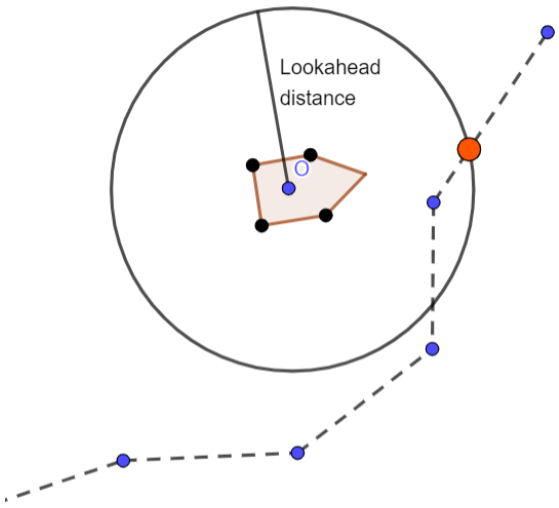

Figura 34: Rappresentazione del funzionamento del controllore Pure Pursuit: si traccia attorno al veicolo una circonferenza di raggio pari al parametro "lookahead distance" e si cerca l'intersezione con il percorso da seguire. Il punto trovato verrà utilizzato come goal point.

Per trovare un goal point, da un punto di vista matematico, si crea una circonferenza centrata nella posizione attuale del veicolo e di raggio "lookahead distance". Le intersezioni tra la circonferenza e i segmenti che creano il percorso da seguire costituiscono i potenziali goal point. Il percorso è memorizzato in un array 2D ed è rappresentato da punti nel piano (coordinata x e y). L'algoritmo prevede l'utilizzo di un ciclo `for` che attraversa ciascuna coppia di punti consecutivi, al fine di determinare se ci sono eventuali intersezioni all'interno di ogni segmento di linea. Se vengono trovati più goal point, si seleziona quello più appropriato. Questi passaggi verranno ripetuti fino al raggiungimento della fine del percorso.

Un parametro che influenza notevolmente le prestazioni del controllore è la "lookahead distance". Valori molto piccoli portano ad un inseguimento del percorso ricco di oscillazioni (instabilità), mentre valori molto grandi creano il problema dei "cutting corners", per cui si ha il rischio di tagliare le curve più strette. Pertanto la scelta di questo parametro è di fondamentale importanza, e dovrà essere supportata da prove sperimentali sul campo.

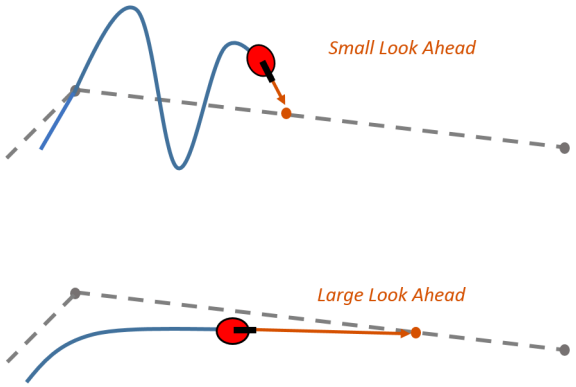

Figura 35: Effetto della scelta del parametro "lookahead distance": un valore troppo piccolo può causare instabilità, mentre uno troppo grande fa tagliare al veicolo le curve più strette.

Ci sono diverse modalità per trovare l'intersezione tra circonferenza e retta passante per due punti: si è adottata quella presentata in [26].
La particolarità di questo metodo è che la circonferenza è sempre centrata nell'origine del sistema di riferimento cartesiano. Nella nostra applicazione, però, la circonferenza è centrata nella posizione attuale del robot [$currentX$,

$currentY$]: pertanto è necessario eseguire una traslazione nell'origine, sottraendo le coordinate della posizione del robot ai punti che definiscono la retta ($pt_1 = [x_1, y_1]$ e $pt_2 = [x_2, y_2]$). Questa procedura consiste in un semplice cambiamento di sistema di riferimento, volto unicamente al funzionamento del metodo sopra introdotto: i risultati, una volta conclusa la procedura di ricerca delle intersezioni, verranno nuovamente riferiti al sistema di riferimento globale della mappa.

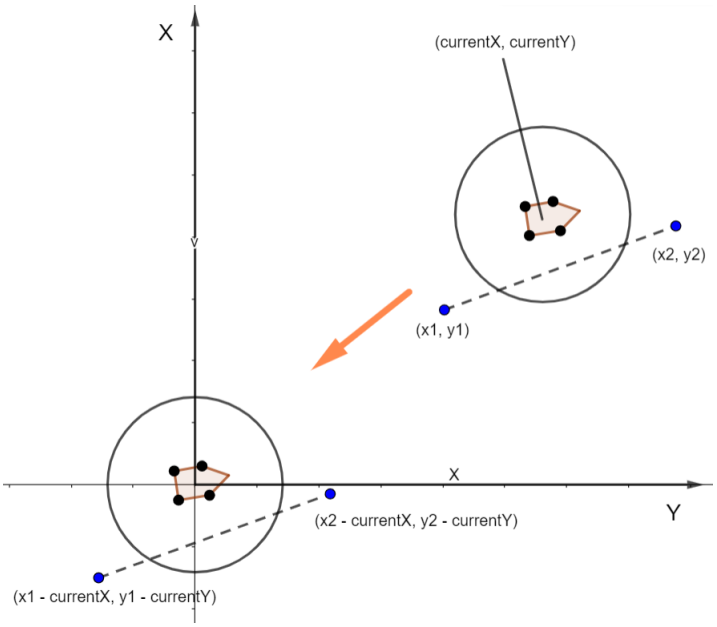

Figura 36: Per applicare il metodo descritto è necessario attuare una traslazione all'origine degli assi. Una volta trovate le soluzioni, il sistema sarà nuovamente traslato nel punto in cui si trova effettivamente il robot.

Un aspetto su cui bisogna prestare attenzione è il fatto che si è interessati a trovare l'intersezione tra la circonferenza e il segmento che unisce due waypoint consecutivi del percorso, e non l'intera retta passante per i due punti. In alcune condizioni, sebbene le intersezioni trovate siano sulla retta definita da pt_1 e pt_2, esse non sono esattamente all'interno dell'intervallo tra (x_1, y_1) e (x_2, y_2). Queste intersezioni vanno scartate in quanto i goal point non rientrerebbero nel percorso che il veicolo deve seguire: è necessario controllare per ogni intersezione trovata se la coordinata x ricade all'interno dell'intervallo $[min(x_1, x_2), max(x_1, x_2)]$ e se la coordinata y ricade all'interno dell'intervallo $[min(y_1, y_2), max(y_1, y_2)]$.

Tutti i possibili risultati delle intersezioni tra retta e circonferenza sono i seguenti:

- Nessuna intersezione. In tal caso, il veicolo si dirigerà verso l'ultimo waypoint registrato.
- Intersezioni multiple, ma non comprese tra (x_1, y_1) e (x_2, y_2). Questa situazione sarà da considerare come "nessuna intersezione".
- C'è una e una sola intersezione all'interno dell'intervallo compreso tra (x_1, y_1) e (x_2, y_2). Il robot seguirà l'intersezione trovata.
- Ci sono due intersezioni e entrambe sono comprese tra (x_1, y_1) e (x_2, y_2). In questa situazione, bisogna determinare quale punto è il migliore da seguire. Una soluzione è quella di selezionare il punto più vicino al secondo waypoint (x_2, y_2), quindi più vicino alla fine del percorso, calcolando la distanza tra le intersezioni e il punto (x_2, y_2).

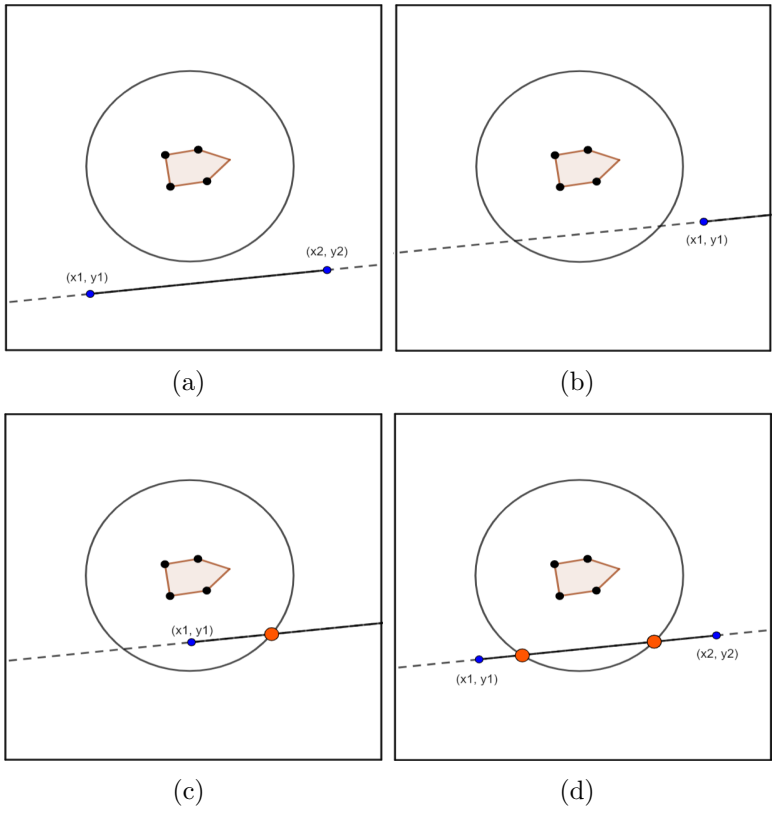

Figura 37: I quattro possibili casi di intersezione tra segmento e circonferenza nell'algoritmo Pure Pursuit.

Nella scelta del goal point è necessario adottare il seguente accorgimento, che riguarda un possibile caso particolare, in cui il robot viaggia attraverso curve strette, e potrebbero crearsi molteplici intersezioni con più segmenti di linea. In tali situazioni è consigliabile programmare l'algoritmo in modo che il robot segua il primo punto valido che trova. In questo modo, nei casi estremi in cui il percorso si sovrappone o esistono molteplici curve strette, il robot non salterà completamente una porzione del percorso, "tagliando" la curva.

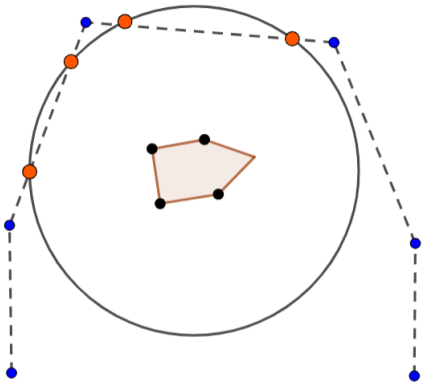

Figura 38: In alcuni casi è possibile che la circonferenza intersechi diversi segmenti. É necessario selezionare l'intersezione più adeguata.

Inoltre, un altro accorgimento, per impedire al robot di tornare indietro nel percorso, è quello di creare un parametro "lastFoundIndex" che memorizzi l'indice del waypoint appena superato. Ogni volta che il ciclo viene eseguito, verranno così facendo controllati solo i punti del percorso successivi al waypoint appena passato. In questo modo, i segmenti che il robot ha già attraversato non verranno sottoposti nuovamente alla ricerca di un'intersezione. Questo parametro è impiegato quando non viene trovata un'intersezione, ad esempio quando il robot devia dal percorso o quando parte non esattamente del punto di start: in questi casi il robot seguirà il punto a "lastFoundIndex". In pratica, questo è un modo con cui il robot, quando è lontano dal percorso e non sapendo dove andare, si dirige verso l'ultimo punto del percorso in cui è transitato, per poi riprendere da dove aveva lasciato.

Tramite il seguente esempio, rappresentato in figura 39, è possibile analizzare un ulteriore caso particolare: supponiamo che il robot entri per la prima volta nel percorso alla prima iterazione del ciclo, con $lastFoundIndex = 1$. La ricerca

dell'intersezione tra segmento e circonferenza parte dal primo waypoint del percorso: ipotizziamo venga trovata un'intersezione tra i primi due waypoint, che quindi viene selezionata come punto obiettivo per il robot da seguire. Alla fine della prima iterazione del ciclo, $lastFoundIndex$ viene aggiornato a 1 poiché il punto obiettivo è stato trovato tra $waypoint(1)$ e $waypoint(2)$. Quando inizia la seconda iterazione del ciclo, il robot si è avvicinato a $waypoint(2)$. Dato che $lastFoundIndex$ è ancora pari a 1, la ricerca di intersezioni parte nuovamente da $waypoint(1)$. Questa volta, ad esempio, si trovano due intersezioni con il percorso: una tra $waypoint(1)$ e $waypoint(2)$ e l'altra tra $waypoint(2)$ e $waypoint(3)$. Secondo la procedura spiegata finora, l'algoritmo sceglierebbe come goal point l'intersezione tra $waypoint(1)$ e $waypoint(2)$ e interromperebbe il ciclo di ricerca. Ma come si osserva in figura 39, questa non è una scelta ottimale, in quanto il robot tornerebbe indietro. Per evitare questo tipo di scelta di goal point, si aggiunge un'ulteriore condizione da rispettare: l'interruzione del ciclo `for` di ricerca può avvenire solo se la distanza dal goal point scelto e il successivo waypoint è minore della distanza tra la posizione attuale del robot e il successivo waypoint. Se questa condizione non è verificata, il ciclo di ricerca continua. Inoltre, si incremeta anche $lastFoundIndex$ nel caso in cui il robot non trovi intersezioni nel segmento successivo. In questo modo si evita che il robot torni indietro e $waypoint(lastFoundIndex)$ diventa temporaneamente il nuovo goal point.

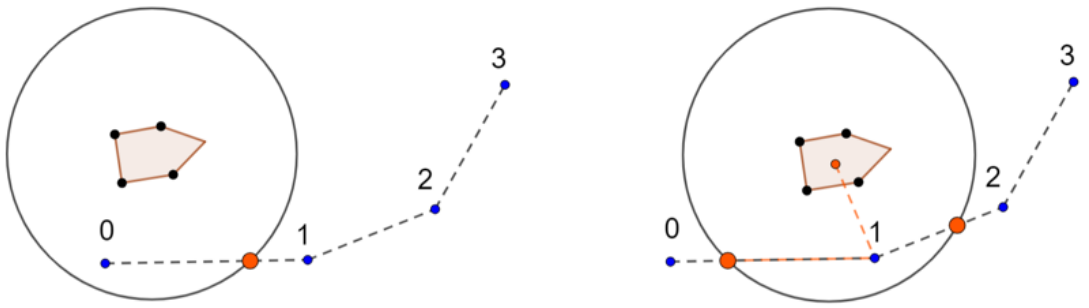

Figura 39: Esempio in cui sono rappresentate le prime due iterazioni dell'algoritmo Pure Pursuit per l'individuazione del goal point.

Una volta determinato un goal point, il passo successivo è far muovere il robot

verso quel punto. Si consideri la situazione illustrata in figura 40, in cui il robot si trova in $[currentX, currentY]$ e il goal point in $[targetX, targetY]$: Il robot dovrà eseguire due azioni contemporaneamente: andare avanti e svoltare verso il punto di destinazione. Per fare ciò, è necessario calcolare separatamente la velocità lineare e quella di imbardata e sommarle per ottenere la velocità finale. L'angolo di rotazione che il robot deve eseguire, denominato $turnError$, può essere calcolato sottraendo l'angolo di orientazione attuale del robot $currentHeading$ dall'angolo $absTargetHeading$, formato dal vettore $[targetX - currentX, targetY - currentY]$ con l'asse x del sistema di riferimento cartesiano globale. Quest'ultimo si può ottenere attraverso l'utilizzo della funzione *atan2*, che però restituisce come output valori compresi tra $-\pi$ e $+\pi$. Il sistema di riferimento adottato ha però angoli che variano però da 0 a 2π, quindi si aggiunge semplicemente 2π nel caso di angoli negativi.

Ci sono alcuni casi speciali che vale la pena menzionare. Se ad esempio $currentHeading = 1°$ mentre $absTargetAngle = 359°$, otterremmo $turnError = 359° - 1° = 358°$, il che è totalmente insensato, in quanto invece di girare in verso orario di 2° il robot gira in senso antiorario di 358°. Per rendere la funzione il più efficiente possibile, il robot dovrebbe semplicemente nel verso opposto se $turnError$ risulta superiore a 180°.

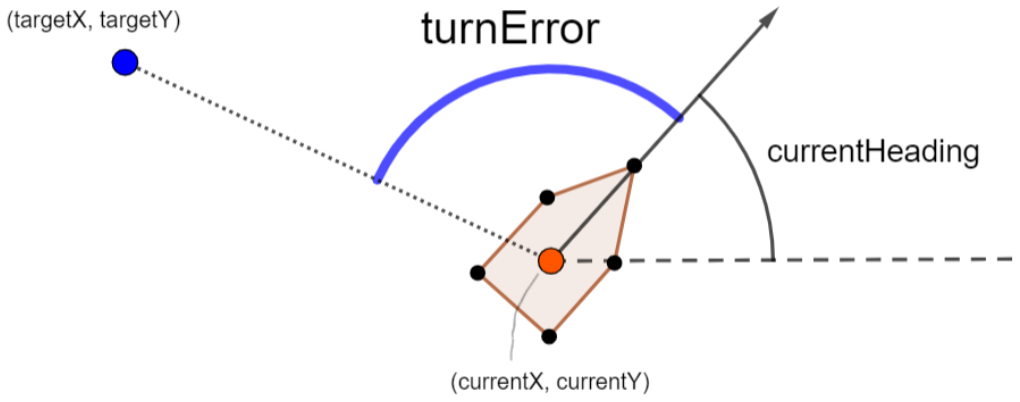

Figura 40: Una volta determinato il goal point, la velocità di imbardata sarà proporzionale all'angolo denominato $turnError$.

3.5 Modello cinematico del rover

L'ottenimento di un modello che descrive la cinematica dell'intero veicolo parte dalla comprensione del funzionamento di una singola ruota (sezione 2.3.2), che, contribuendo al moto del robot, causa l'imposizione di alcuni vincoli cinematici su di esso. La tipologia, la dimensione e il posizionamento rispetto al telaio di ciascuna ruota influiscono su tali vincoli, che devono essere combinati per ottenere il comportamento globale del robot. In questa trattazione si prende come esempio un robot a trazione differenziale con due coppie di ruote fisse (non pivottanti), con un montaggio simmetrico rispetto al baricentro geometrico del telaio del veicolo. Le ipotesi per ottenere un primo modello semplificato del robot sono le seguenti:

- Si ipotizza il robot come un corpo rigido (telaio) montato su ruote, che si muove su un piano.
- I gradi di libertà del telaio sono tre, due coordinate nel piano e una rotazione attorno all'asse perpendicolare al piano.
- Si ignorano tutti i gradi di libertà dovuti ad esempio alla flessibilità degli assali, dei giunti e degli altri componenti della trasmissione.

É utile scegliere un sistema di riferimento, rispetto al quale risulti conveniente definire le grandezze fisiche di rilievo per il problema trattato. Innanzitutto si stabilisce un sistema di riferimento globale inerziale, di coordinate X_i, Y_i, orientandolo in maniera opportuna rispetto alla mappa. É utile definire anche un sistema di riferimento locale, di coordinate X_r, Y_r, solidale al telaio del rover e centrato nel baricentro geometrico del robot P. La posizione del punto P nel sistema di riferimento globale è definita da x e y, mentre l'angolo tra i due sistemi di riferimento è dato da θ. Con queste tre grandezze lo stato del sistema è completamente definito, in quanto in una rappresentazione bidimensionale un corpo ha tre gradi di libertà. La configurazione di un robot è determinata quindi dalla sua posizione e dalla sua orientazione.

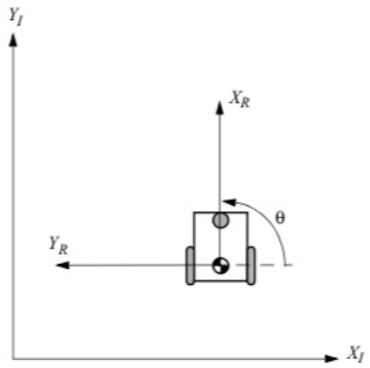

Figura 41: Rappresentazione del sistema di riferimento globale e locale del modello cinematico.[9]

Chiamiamo ξ il vettore che definisce lo stato del robot, con pedice i nel sistema di riferimento globale e pedice r in quello locale (ad esempio $\xi_i = [x, y, \theta]^T$).
Il collegamento tra il moto espresso nei due sistemi di riferimento si effettua tramite la matrice di rotazione $R(\theta)$. Per cui avremo:

$$\dot{\xi}_r = R(\theta) \cdot \dot{\xi}_i \Longrightarrow \dot{\xi}_i = R(\theta)^{-1} \cdot \dot{\xi}_r$$

$$R(\theta) = \begin{bmatrix} cos(\theta) & sin(\theta) & 0 \\ -sin(\theta) & cos(\theta) & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

Si procede quindi all'ottenimento del modello cinematico del rover considerando il contributo di ogni singola ruota. Si suppone che il sistema di riferimento locale, solidale al robot, sia allineato in modo tale che il robot si muova in avanti lungo $+X_r$. Si identificano quindi i valori di α e β per ciascuna coppia di ruote: per la ruote di destra si avrà $\alpha = -\frac{\pi}{2}, \beta = \pi$ mentre per quelle di sinistra $\alpha = \frac{\pi}{2}, \beta = 0$. Dato che le ruote sono fisse e montate parallelamente a due a due, l'equazione di moto di vincolo allo strisciamento laterale è uguale per ciascuna coppia di ruote. Combinando le equazioni (1) e (2) per entrambe le coppie, si ottiene:

$$\begin{bmatrix} \begin{bmatrix} 1 & 0 & l \\ 1 & 0 & -l \end{bmatrix} \\ \begin{bmatrix} 0 & 1 & 0 \end{bmatrix} \end{bmatrix} \cdot R(\theta) \cdot \dot{\xi}_i = \begin{bmatrix} \begin{bmatrix} r & 0 \\ 0 & r \end{bmatrix} \cdot \begin{bmatrix} \omega_r \\ \omega_l \end{bmatrix} \\ 0 \end{bmatrix}$$

Di conseguenza

$$\dot{\xi}_i = R(\theta)^{-1} \cdot \begin{bmatrix} 1 & 0 & l \\ 1 & 0 & -l \\ 0 & 1 & 0 \end{bmatrix}^{-1} \cdot \left[\begin{bmatrix} r & 0 \\ 0 & r \end{bmatrix} \cdot \begin{bmatrix} \omega_r \\ \omega_l \end{bmatrix} \right] = R(\theta)^{-1} \cdot \begin{bmatrix} \frac{r\omega_l}{2} + \frac{r\omega_r}{2} \\ 0 \\ \frac{r\omega_l}{2l} - \frac{r\omega_r}{2l} \end{bmatrix}$$

A questo punto, è possibile ottenere i valori di ω_l e ω_r da dare in input ai motori elettrici che azionano le ruote, a partire dai valori di $\dot{x_r}$ e $\dot{\theta}$, ossia la velocità lineare e angolare ottenuti tramite l'algoritmo di path following (Pure Pursuit).

$$\begin{cases} \dot{x_r} = \frac{r(\omega_l + \omega_r)}{2} \\ \dot{\theta} = \frac{r(\omega_l - \omega_r)}{2l} \end{cases} \implies \begin{cases} \omega_r = \frac{\dot{x_r}}{r} - \frac{\dot{\theta} l}{r} \\ \omega_l = \frac{\dot{x_r}}{r} + \frac{\dot{\theta} l}{r} \end{cases}$$

Per quanto riguarda la logica di controllo del robot, è possibile stabilire due variabili di controllo, $u_v = \frac{(u_r + u_l)}{2}$ e $u_\omega = \frac{(u_l - u_r)}{2}$ tramite le quali si esprimono le velocità del punto P, ossia del baricentro del veicolo. Si ha quindi una variabile di controllo sulla velocità lineare del robot e una sulla sua rotazione nel piano:

$$\begin{cases} \dot{x} = \frac{r}{2}(u_r + u_l)cos(\theta) = ru_v cos(\theta) \\ \dot{y} = \frac{r}{2}(u_r + u_l)sin(\theta) = ru_v sin(\theta) \\ \dot{\theta} = \frac{r}{2l}(u_l - u_r) = \frac{r}{l}u_\omega \end{cases}$$

Il modello cinematico del robot ottenuto si riferisce alle velocità, ed è quindi un modello differenziale del primo ordine. É utile in diversi casi, ma non considera tutti gli effetti dinamici del robot, come ad esempio il fatto che il robot ha una distanza di accelerazione o frenata non nulla. É possibile definire un modello differenziale del secondo ordine, che consideri anche l'accelerazione e ne definisca dei valori massimi e minimi entro cui il veicolo può lavorare. I modelli del primo ordine, inoltre, hanno il difetto per cui gli azionamenti devono subire variazioni istantanee. Per esempio, nel robot a guida differenziale tra uno step di calcolo

e quello successivo nell'algoritmo di path following Pure Pursuit, la velocità viene cambiata in modo discontinuo. Questo su un veicolo reale è difficilmente implementabile, in quanto richiede accelerazioni in certi casi troppo elevate. Un modello differenziale del secondo ordine può ad esempio essere creato definendo i parametri di controllo u_l e u_r proporzionali all'accelerazione e non alla velocità di rotazione dei motori. In questo modo si rende la velocità una funzione continua nel tempo. Si avrà quindi:

$$\begin{cases} \dot{x} = \frac{r}{2}(\omega_r + \omega_l)cos(\theta) \\ \dot{y} = \frac{r}{2}(\omega_r + \omega_l)sin(\theta) \\ \dot{\theta} = \frac{r}{2l}(\omega_l - \omega_r) \\ \dot{\omega_l} = u_l \\ \dot{\omega_r} = u_r \end{cases}$$

Questa strategia di controllo adottata, non prevede nessun feedback sulla posizione effettiva del robot. Infatti, si segue una traiettoria impostata definendo a priori i valori di velocità con cui azionare le ruote, senza effettivamente misurare la posizione del robot e considerare correzioni da fare nel caso di eventuali errori sul posizionamento. Questa strategia ha però degli svantaggi:

- Non è sempre facile calcolare a priori la traiettoria del robot se sono presenti vincoli da rispettare sui valori di velocità e accelerazione.
- Il veicolo non è in grado di adattarsi o correggere la traiettoria in caso di eventuali variazioni dinamiche imposte dall'ambiente esterno (ad esempio una folata di vento che sposta il robot, o il bloccaggio di una ruota o suo improvviso strisciamento).

3.6 Obstacle avoidance

L'algoritmo di obstacle avoidance viene attivato quando il veicolo si avvicina troppo ad un ostacolo, e di conseguenza i comandi del global path planner vengono disattivati, in quanto l'inseguimento della traiettoria potrebbe portare alla collisione con un ostacolo imprevisto. La soglia di attivazione non deve essere troppo bassa, in quanto il sistema non avrebbe tempo di reagire all'ostacolo, ma nemmeno troppo elevata, altrimenti le manovre di evitamento dell'ostacolo inizierebbero fin troppo in anticipo. Un buon compromesso si è trovato con una soglia pari a $3.5m$. Quando si individua un ostacolo al di sotto di tale soglia, la priorità diventa quella di evitare il contatto con qualsiasi elemento presente nelle vicinanze, a costo di deviare dalla traiettoria precedentemente calcolata. Da quest'ultima si acquisisce solamente il punto obiettivo verso cui è preferibile dirigersi, che diventerà il goal point dell'algoritmo di obstacle avoidance. La posizione degli ostacoli non è nota a priori e potrebbe variare nel tempo se questi sono in movimento: per tale motivo è necessario reagire prontamente, adattandosi ad ogni istante ai cambiamenti esterni.

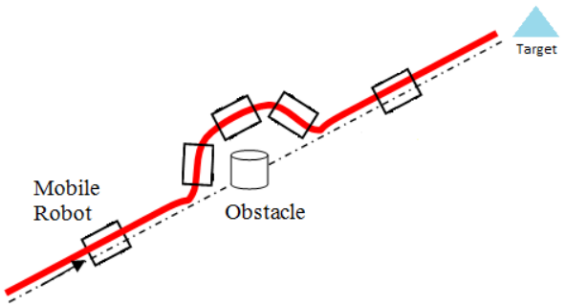

Figura 42: Raffigurazione della traiettoria calcolata da un generico algoritmo di obstacle avoidance.

L'algoritmo implementato si basa sul follow the gap method e prevede la creazione di una serie di gap a partire dai dati campionati dal sensore di distanza, l'individuazione della miglior direzione di svolta per ognuno di essi e la selezione del gap migliore. Una volta eseguiti tutti questi step, si procede con il calcolo dell'angolo di imbardata finale, da applicare al veicolo al fine di evitare gli ostacoli presenti sulla traiettoria.

Il funzionamento dell'algoritmo parte dall'acquisizione dei dati del sensore di distanza, in modo tale da ottenere la mappa in real time dell'ambiente circostante il veicolo. Per tale scopo si può utilizzare un sensore ad ultrasuoni, radar, o il più moderno e preciso lidar. Nell'ambiente virtuale simulato in Matlab, si è sfruttato l'oggetto `rangeSensor`, che fornisce misurazioni angolari e di distanza a partire dalla posizione attuale del sensore e dall'occupancy map del terreno. Le proprietà principali dell'oggetto sono la distanza minima e massima acquisibile, l'apertura angolare del campo visivo, e la risoluzione angolare orizzontale (proporzionale al numero di punti letti). L'oggetto restituisce in output, per ogni angolo (dal valore angolare minore del campo visivo a quello maggiore, incrementando di uno step pari alla risoluzione orizzontale) il corrispettivo valore di distanza misurato. A titolo di esempio, si è scelto un campo visivo pari a 180° (da -90° a +90°), per cui i primi valori di distanza sono quelli misurati a destra del veicolo, mentre gli ultimi a sinistra, considerando sempre l'orientazione del sensore, che risulta essere solidale al veicolo. É da notare il fatto che l'oggetto rangeSensor, purchè abbia un principio di funzionamento simile ai lidar reali, non è soggetto a tutti quei fenomeni di interferenza che portano ad avere in alcuni casi qualche lettura errata. In ottica futura, dovendo implementare questo algoritmo su un sistema che utilizza un sensore reale, sarà necessario elaborare ulteriormente i dati provenienti da quest'ultimo, individuando e filtrando opportunamente le letture non precise.

Una volta acquisiti i dati dal sensore, una prima elaborazione viene eseguita saturando i valori superiori alla distanza massima di acquisizione (restituiti come NaN, ossia "not a number"). Essi vengono saturati ad una soglia pari ad un metro in più rispetto alla distanza massima, in modo che si possa distinguere chiaramente un valore leggermente inferiore alla distanza massima di lettura rispetto ai valori superiori ad essa, che potenzialmente possono avere anche un valore infinito. Il sensore infatti non "vede" oltre la distanza massima e quindi non è possibile apprezzare le differenze di distanza tra valori oltre il campo visivo.
Successivamente si trova il minimo (o i minimi), ossia l'ostacolo più vicino al veicolo, e si applica una "bolla" di sicurezza, bloccando tutti i settori angolari nei

pressi del minimo, tramite l'imposizione di un valore di distanza nulla. Questo è un approccio conservativo, che ha l'obiettivo di non fare avvicinare mai il veicolo all'ostacolo più vicino, escludendo tutte le direzioni di svolta che lo porterebbero a collidere con quest'ultimo. La dimensione di questa bolla è pari a quella del robot: si adotta il cosiddetto point robot approach, secondo il quale il robot viene considerato puntiforme e gli ostacoli vengono gonfiati di un fattore pari al raggio del veicolo. In questa maniera si semplifica il problema dovuto all'ingombro del veicolo. Il numero di settori angolari che costituiscono la bolla dipende dalla distanza tra il veicolo e la bolla stessa, ed è pari a:

$$N = \frac{R}{d_{min} * \Delta\alpha}$$

Dove:

- R è il raggio del robot.
- d_{min} è la distanza dal punto più vicino al veicolo.
- $\Delta\alpha$ è la risoluzione angolare orizzontale del sensore.

Individuato il settore angolare con il valore di distanza minore, pari a d_{min}, si bloccano quindi N settori alla sua sinistra e altrettanti N settori alla sua destra. Si nota come l'angolo totale che viene bloccato dalla presenza della bolla di sicurezza cresca quando l'ostacolo più vicino si trova ad una distanza minore.

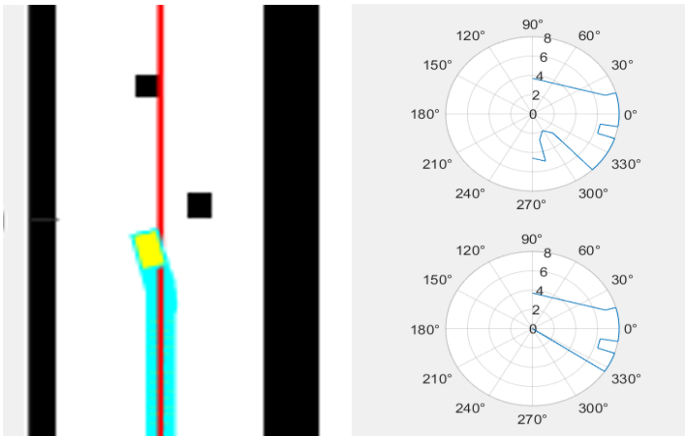

Figura 43: Esempio di elaborazione della "bolla" di sicurezza a partire dai acquisiti dal lidar. I settori angolari corrispondenti all'ostacolo più vicino al rover vengono bloccati.

Una volta processati i dati del lidar e creata la bolla di sicurezza, si procede con la suddivisione del campo visivo del sensore in una serie di settori, chiamati gap. Lo scopo di questo passaggio è di distinguere i settori angolari in cui è presente un ostacolo, da quelli che si interpongono tra due ostacoli, che quindi sono potenzialmente delle aperture in cui il veicolo può transitare. Il metodo utilizzato per distinguere un gap dal successivo consiste nell'individuare due misurazioni consecutive del sensore con una certa discontinuità nel valore di distanza. In questo modo si riescono ad identificare tutti i bordi degli oggetti e a creare i gap. É necessario però scegliere una soglia, un valore oltre il quale si stabilisce la demarcazione tra un gap e il successivo. Questo valore non deve essere troppo elevato, altrimenti si rischia di avere pochi gap, ma nemmeno troppo basso, altrimenti si rischia di averne un numero eccessivo, frammentando troppo il campo visivo. Un possibile problema si può presentare con ostacoli dalla geometria particolare: ad esempio un ostacolo con forma ad "L", visto dal sensore da una specifica angolazione, potrebbe essere suddiviso in due diversi gap. Seppur questa sia un'inefficienza dell'algoritmo, non causa grandi problemi in quanto non provoca la frammentazione dei gap di apertura tra un ostacolo e l'altro, che alla fine dei conti sono quelli di maggiore importanza, in quanto il veicolo deve muoversi tra di questi.

Lo step successivo prevede di calcolare, per ogni gap, la migliore direzione di svolta. L'approccio più semplice e conservativo è quello di dirigersi verso il punto medio di ogni gap. Così facendo si massimizza la distanza dalle estremità del gap, in prossimità delle quali sono presenti ostacoli. In alcuni casi, ad esempio in presenza di aperture molto ampie, questa strategia non risulta la più efficiente, in quanto può portare ad evitare l'ostacolo mantenendo una distanza eccessiva da esso e facendo percorre al veicolo una traiettoria molto più lunga del necessario. Si è implementata quindi, similmente al metodo follow the gap originale, una funzione che, per quanto riguarda il calcolo della direzione di svolta del singolo gap, pesa sia l'angolo verso il suo punto medio che l'angolo verso il goal point

(punto obiettivo appartenente al percorso creato nella fase di path planning).

$$\phi_{gap} = \frac{\frac{\alpha}{d_{min}} * \phi_{centergap} + \phi_{goal}}{\frac{\alpha}{d_{min}} + 1}$$

Dove:

- α è una costante opportunamente scelta.
- d_{min} è la distanza dall'ostacolo più vicino.

Si nota che, quando gli ostacoli sono ad una grande distanza, la direzione tende più al goal point, mentre al diminuire del valore di d_{min} essa tende al centro del gap, massimizzando la distanza di sicurezza dagli ostacoli. Per quanto riguarda la scelta del parametro α, un valore nullo fa sì che la direzione di svolta coincida con l'angolo del goal point (scelta da evitare in quanto lungo questa direzione potrebbe esserci un ostacolo). Aumentando il suo valore, invece, si dà sempre più peso alla direzione verso il centro del gap, aumentando di conseguenza anche la distanza di sicurezza.

L'ultimo passaggio dell'algoritmo è quello di individuare la migliore direzione di svolta finale, tra tutte quelle precedentemente individuate per i singoli gap. É necessario individuare una serie di parametri da tenere in considerazione per effettuare questa scelta: da un lato è auspicabile che il veicolo si diriga verso un'apertura sufficientemente ampia e che ne permetta il suo passaggio, dall'altro che scelga un gap il più possibile vicino alla traiettoria da seguire, evitando deviazioni troppo lunghe e inefficienti. A questo scopo si definisce una funzione che pesa entrambi questi fattori ed assegna un punteggio ad ogni gap: la scelta finale ricadrà sul gap con il maggior punteggio. Il punteggio per l'i-esimo gap è così definito:

$$U_i = k_1 * L_{gap} * d_{gap} + k_2 * (\pi - \phi_{\Delta gap/goal})$$

Dove:

- k_1 e k_2 sono due costanti opportunamente scelte.

- L_{gap} è l'ampiezza del gap.
- d_{gap} è la distanza media delle misurazioni che costituiscono il gap.
- $\phi_{\Delta gap/goal}$ è la differenza tra il center gap angle e l'angolo del goal point.

Il punteggio è costituito da due termini. Il primo tiene in considerazione l'ampiezza e la distanza media del gap: a parità di ampiezza si premia il gap con misurazioni più grandi e quindi ostacoli più lontani. Il secondo, invece, considera quanto la direzione del gap si discosta dal punto obiettivo, premiando i gap che si aprono in direzione del goal point. I coefficienti k_1 e k_2 spostano il peso del punteggio verso un termine o l'altro: se $k_2 > k_1$, più la scelta finale tenderà a ricadere sul gap che punta verso il goal point, mentre se $k_1 > k_2$ verrà selezionato un gap con ampiezza maggiore. Per quanto riguarda L_{gap}, ossia l'ampiezza del gap, essa è calcolata sfruttando il teorema del coseno, essendo noti gli angoli e le distanze dei valori all'estremità del gap. La distanza media del gap d_{gap}, invece, è ottenuta facendo la media aritmentica delle misurazioni di distanza, dal primo all'ultimo valore del gap, e poi adimensionalizzando il tutto rispetto alla massima distanza di lettura del lidar. In tal modo si associa un valore pari al 100% ai gap di distanza massima, e una percentuale decrescente ai gap più vicini al veicolo. Per ottenere $\phi_{\Delta gap/goal}$, invece, si esegue una differenza tra l'angolo del gap e l'angolo del goal point.

$$\phi_{\Delta gap/goal} = |\phi_{gap} - \phi_{goal}|$$

Più ϕ_{gap} e ϕ_{goal} sono simili, più la loro differenza è minore, quindi il secondo termine della funzione punteggio è più alto. In questo modo si premiano maggiormente i gap in direzione del goal point.

Un'ulteriore ottimizzazione dell'algoritmo viene fatta per risolvere possibili problemi di instabilità che si possono verificare in alcuni casi come quello in figura 44. Quando il punteggio tra due gap è molto simile, quindi ad esempio quando hanno simile ampiezza e sono simmetrici rispetto alla direzione del goal point, è possibile che tra un'interazione e le successive avvenga un continuo e repentino cambiamento di scelta del gap. In questi casi si ha una traiettoria a zig-zag, quasi

come se il robot fosse "indeciso". Questo crea un possibile fenomeno di instabilità, che può essere risolto assegnando al gap scelto nell'iterazione precedente un "premio", ossia un punteggio aggiuntivo che lo favorisca leggermente rispetto agli altri. In questo modo, scelto un gap, a meno di drastici aumenti di punteggio negli altri, si tende a favorire la decisione già presa e a mantenerla. Quindi, una volta imboccata un'apertura, per poterla cambiare è necessario che le condizioni diventino sufficientemente sfavorevoli in modo da superare anche il "premio" di punteggio aggiuntivo assegnato.

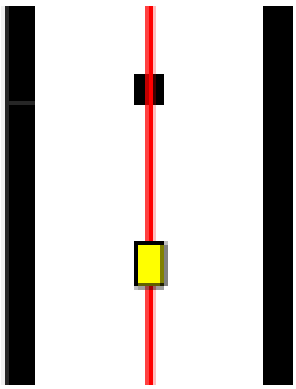

Figura 44: Esempio di un caso in cui si possono verificare problemi di instabilità nell'algoritmo di obstacle avoidance implementato.

3.7 Risultati

L'algoritmo sviluppato, una volta eseguita la mappatura dell'ambiente di lavoro, è in grado di definire una traiettoria tramite un pianificatore geometrico e di seguirla tramite una logica di path following (Pure Pursuit) e motion planning (adoperando il modello cinematico del veicolo). Inoltre, l'implementazione di un algoritmo di obstacle avoidance permette di evitare possibili ostacoli imprevisti presenti sulla traiettoria, portando il rover in sicurezza dal punto iniziale a quello finale della missione. Come si osserva in figura 45, in cui in una porzione della mappa sono stati inseriti ostacoli in maniera casuale, il rover è in grado di aggirarli, deviando dal percorso inizialmente calcolato ma sempre proseguendo verso il punto obiettivo. Questo anche in situazioni complicate, come ad esempio in prossimità di una curva e in aree occupate da molteplici ostacoli.

In assenza di ostacoli, invece, il rover segue in maniera precisa la traiettoria desiderata grazie al controllore Pure Pursuit, con l'accortezza di rallentare quando sta percorrendo una curva. Il controllore Pure Pursuit permette anche di rientrare nel percorso una volta terminata la deviazione per l'evitamento dell'ostacolo, oppure se la posizione iniziale del rover non corrisponde al punto di inizio del percorso calcolato.

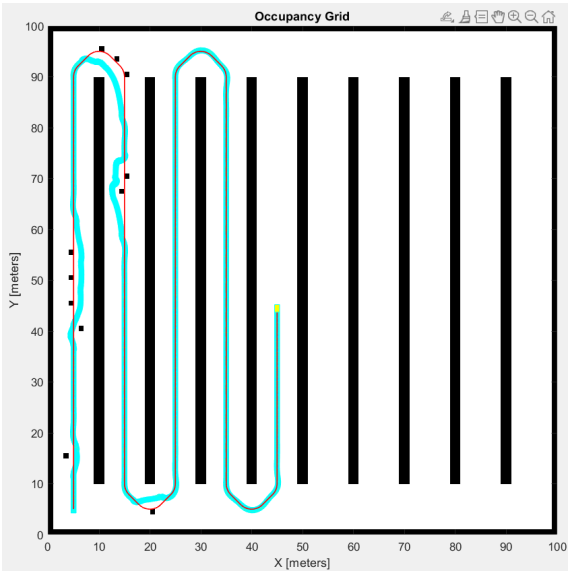

Figura 45: Traccia del percorso effettuato dal rover nell'ambiente simulato, in presenza di ostacoli imprevisti.

L'algoritmo di obstacle avoidance è efficace anche in presenza di ostacoli dinamici. É infatti in grado di cambiare decisione in seguito alle variazioni che avvengono nell'ambiente circostante. Questo grazie alla natura reattiva dell'algoritmo, che prende una decisione ad ogni iterazione, valutando ad ogni istante la situazione esterna, a partire dai dati acquisiti tramite il sensore di distanza. Questa capacità si può osservare nel caso illustrato in figura 46, in cui un ostacolo attraversa da sinistra a destra il campo visivo del rover. Inizialmente, l'ostacolo si trova a sinistra del veicolo (a), per cui l'algoritmo di obstacle avoidance seleziona l'apertura che si crea a destra e inizia a dirigersi verso di essa. Successivamente, l'ostacolo inizia a muoversi verso destra (b), tagliando la strada al rover e bloccando sempre di più l'apertura alla sua destra. L'algoritmo reagisce a tale situazione, invertendo la direzione di svolta (c) e privilegiando la nuova apertura che si crea a sinistra, divenuta più favorevole. Così facendo, il rover è in grado di superare l'ostacolo in sicurezza (d) e di proseguire lungo il percorso originale.

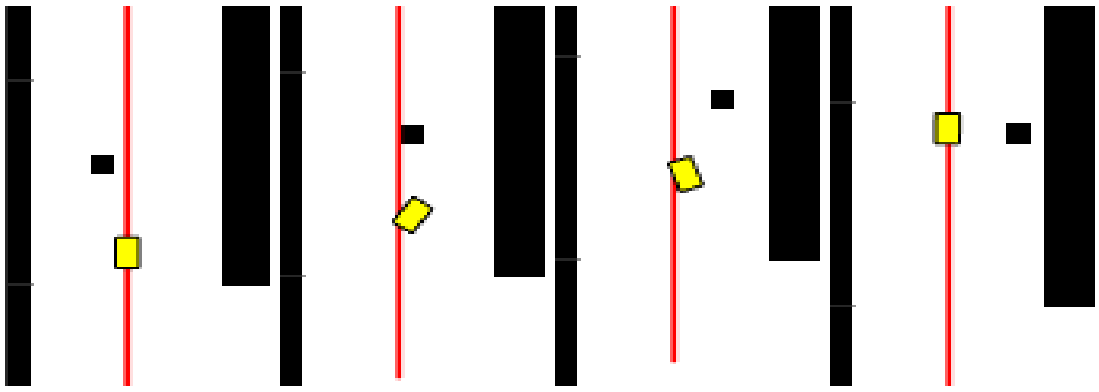

Figura 46: Illustrazione sequenziale della procedura di evitamento di un ostacolo in movimento.

4 Conclusioni e sviluppi futuri

In questa attività di tesi, in seguito ad una ricerca iniziale sulle potenzialità dei macchinari agricoli a guida autonoma, si è focalizzata l'attenzione sul tema dell'autonomous driving. Oltre ad un'analisi sullo stato dell'arte attuale e ad una documentazione sui principali e più diffusi algoritmi presenti in letteratura, l'argomento primario di quest'attività è stato lo sviluppo di un algoritmo di guida autonoma in ambiente MATLAB®. La prima parte dell'algoritmo ha previsto la creazione di una mappa, tramite la definizione di un terreno agricolo virtuale, utilizzato poi per simulare e testare la logica di guida autonoma. É stata successivamente eseguita la pianificazione del percorso da far compiere al rover, all'interno della mappa dell'ambiente calcolata. Questa fase, di tipo strategico, è denominata global path planning. Tramite la definizione del punto di partenza, di quello di arrivo, e di una serie di waypoint intermedi in cui si vuole che il rover transiti, un pianificatore di tipo geometrico elabora una traiettoria costituita da tratti lineari e archi di circonferenza. É stata quindi implementata una logica di path following, più precisamente un controllore Pure Pursuit. Quest'ultimo determina le velocità che il rover deve possedere affinchè segua la traiettoria precedentemente calcolata. Tramite la definizione di un modello cinematico del veicolo, il moto del rover viene espresso sotto forma di velocità angolari da trasmettere alle ruote, in modo da poter opportunamente azionare i motori elettrici ad esse collegate. Infine, è stato implementato un algoritmo di obstacle avoidance, che elabora una deviazione del percorso nel caso in cui un ostacolo imprevisto si trovi lungo la traiettoria del rover. Il suo scopo è di evitare la collisione con qualsiasi elemento esterno, in maniera sicura ma allo stesso tempo efficiente, rispettando il più possibile la traiettoria originale.

I risultati della simulazione virtuale dell'algoritmo di guida autonoma sono stati soddisfacenti, in quanto il rover è risultato in grado di seguire precisamente la traccia in assenza di ostacoli, e di evitarli efficacemente nel caso in cui questi fossero presenti. In ottica futura, sarà necessario testare l'algoritmo sviluppato

su un veicolo, sia esso il rover originale o un suo modello in scala. Con una simulazione in un ambiente reale, infatti, sarebbe possibile verificare l'efficacia della logica di guida autonoma ed eventualmente eseguire le dovute correzioni o migliorie nel caso nascessero delle criticità. Il passaggio da veicolo virtuale a veicolo reale presenta infatti una serie di ulteriori questioni da attenzionare, tra cui ad esempio imprecisioni nelle letture dei sensori e considerazioni relative alla dinamica di un veicolo in moto.

Riferimenti bibliografici

[1] FAO, "World food and agriculture - statistical yearbook 2021," 2021. https://doi.org/10.4060/cb4477en.

[2] United Nations, "World urbanization prospects: The 2018 revision," *Econ. Soc. Aff.*, 2018.

[3] D. K. Ray, N. D. Mueller, P. C. West, and J. A. Foley, "Yield trends are insufficient to double global crop production by 2050," *PloS one*, vol. 8, no. 6, 2013.

[4] H. Ritchie and M. Roser, "Sector by sector: where do global greenhouse gas emissions come from?," *Our World in data*, 2023.

[5] FAO, "Employment indicators 2000–2021." https://doi.org/10.4060/cc6627en.

[6] A. King, "Technology: The future of agriculture," *Nature*, vol. 544, no. 7651, pp. S21–S23, 2017.

[7] L. F. Oliveira, A. P. Moreira, and M. F. Silva, "Advances in agriculture robotics: A state of the art review and challenges ahead," *Robotics*, vol. 10, no. 2, p. 52, 2021.

[8] T. Duckett, S. Pearson, S. Blackmore, B. Grieve, *et al.*, "Agricultural robotics: the future of robotic agriculture," *UK-RAS network robotics and autonomous systems*, 2018.

[9] R. Siegwart, I. R. Nourbakhsh, and D. Scaramuzza, *Introduction to autonomous mobile robots*. MIT press, 2011.

[10] Matlab YouTube channel, "Understanding slam using pose graph optimization — autonomous navigation, part 3." https://youtu.be/saVZtgPyyJQ?si=JzoE-AX2CJ88cas4.

[11] L. E. Dubins, "On curves of minimal length with a constraint on average curvature, and with prescribed initial and terminal positions and tangents," *American Journal of mathematics*, vol. 79, no. 3, pp. 497–516, 1957.

[12] J. Reeds and L. Shepp, "Optimal paths for a car that goes both forwards and backwards," *Pacific journal of mathematics*, vol. 145, no. 2, pp. 367–393, 1990.

[13] P. E. Hart, N. J. Nilsson, and B. Raphael, "A formal basis for the heuristic determination of minimum cost paths," *IEEE transactions on Systems Science and Cybernetics*, vol. 4, no. 2, pp. 100–107, 1968.

[14] S. Lague, "A* pathfinding (e01: algorithm explanation)." `https://youtu.be/-L-WgKMFuhE?si=4VK-JJsKFsv1YXly`.

[15] S. M. LaValle, *Planning algorithms*. Cambridge university press, 2006.

[16] Matlab YouTube channel, "Path planning with a* and rrt — autonomous navigation, part 4." `https://youtu.be/QR3U1dgc5RE?si=d0xoRx4MUZ1RwkJn`.

[17] R. C. Conlter, "Implementation of the pure pursuit path tracking algorithm," *Camegie Mellon University*, 1992.

[18] V. Sezer and M. Gokasan, "A novel obstacle avoidance algorithm:"follow the gap method"," *Robotics and Autonomous Systems*, vol. 60, no. 9, pp. 1123–1134, 2012.

[19] M. Demir and V. Sezer, "Improved follow the gap method for obstacle avoidance," in *2017 IEEE International Conference on Advanced Intelligent Mechatronics (AIM)*, pp. 1435–1440, IEEE, 2017.

[20] J. Borenstein, Y. Koren, *et al.*, "The vector field histogram-fast obstacle avoidance for mobile robots," *IEEE journal of robotics and automation*, vol. 7, no. 3, pp. 278–288, 1991.

[21] I. Ulrich and J. Borenstein, "Vfh+: Reliable obstacle avoidance for fast mobile robots," in *Proceedings. 1998 IEEE international conference on robotics and automation*, vol. 2, pp. 1572–1577, IEEE, 1998.

[22] I. Ulrich and J. Borenstein, "Vfh*: local obstacle avoidance with look-ahead verification," in *Proceedings. 2000 IEEE International Conference on Robotics and Automation.*, vol. 3, pp. 2505–2511, IEEE, 2000.

[23] Wikipedia, "Real-time kinematic positioning." `https://en.wikipedia.org/wiki/Real-time_kinematic_positioning`.

[24] Y. Li and J. Ibanez-Guzman, "Lidar for autonomous driving: The principles, challenges, and trends for automotive lidar and perception systems," *IEEE Signal Processing Magazine*, vol. 37, no. 4, pp. 50–61, 2020.

[25] Leishen Intelligent System, "Lslidar mechanical lidar c16." `https://www.lslidar.com/product/c16-mechanical-lidar/`.

[26] Wolfram MathWorld, "Circle-line intersection." `https://mathworld.wolfram.com/Circle-LineIntersection.html`.