POLITECNICO DI MILANO

Corso di Laurea Magistrale in Ingegneria dell'Automazione
Dipartimento di Elettronica e Informazione

STUDIO E APPLICAZIONE DI TECNICHE DI CONTROLLO PREDITTIVO PER IL POSIZIONAMENTO E IL COORDINAMENTO DI AGENTI MOBILI

Relatore: *Prof. Marcello Farina*

Correlatore: *Prof. Riccardo Scattolini*

Tesi di Laurea di

Luca Giulioni

Matricola 768823

Anno Accademico 2011-2012

A coloro che credono in me da sempre.

Indice

Elenco delle figure

Sommario

Il *model predictive control* ha assunto negli anni una notevole importanza in molti ambiti applicativi per la capacità intrinseca di gestire problemi di controllo in presenza di vincoli sul valore assunto dalle variabili di stato o di ingresso durante l'evoluzione. Particolarmente interessante è la possibilità di sfruttare la predizione in avanti lungo l'orizzonte di tempo prescelto, prodotta nella fase di ottimizzazione, per realizzare un'implementazione dell'algoritmo di tipo distribuito. Tali informazioni, unite ad una struttura di controllo di tipo decentralizzato e ad una rete di comunicazione tra i regolatori locali, permettono di migliorare le prestazioni ottenute fino a quelle ideali di un controllo centralizzato; il vantaggio è rappresentato dalla possibilità di poter distribuire in questo modo il carico computazionale sui vari agenti nei quali può essere suddiviso il sistema.

In quest'ottica l'idea del presente lavoro è quella di sfruttare un implementazione *MPC* distribuita per coordinare il movimento di una flotta di robot mobili. Al fine di garantire l'assenza di collisioni si fa riferimento al concetto di potenziale artificiale presente in letteratura e lo si integra nel problema di ottimizzazione *MPC.* L'algoritmo utilizzato è basato inoltre sul concetto di *MPC* robusto di tipo *tube based* che è in grado di assicurare, in presenza di un disturbo limitato, il mantenimento della traiettoria del sistema reale controllato in un intorno della traiettoria ideale in assenza di disturbo.

L'algoritmo di controllo proposto è stato validato in simulazione e testato direttamente per il controllo di una coppia di robot reali in movimento all'interno di un piano di lavoro. Alcuni dei risultati più significativi sono riportati al termine della trattazione. Sono infine evidenziati i punti deboli dell'approccio adottato e le possibili direzioni di miglioramento individuate.

Abstract

In the last decades *Model predictive control* has gained a great role in a wide range of practical applications. This success is motivated by the possibility to easily deal with constraints on both state and control variables during the evolution of the system under control.

MPC has also shown to be particularly useful for developing distributed multi-agent coordination algorithms, in view of the fact that predictions of the state trajectory are available at each time step and can be transmitted. The advantage of distributed control with respect to centralized implementations lies on the fact that the computational load can be distributed among the involved agents. The use of information transmitted according to a communication network between local regulators, enables distributed control implementations with similar performance to centralized implementations.

From this point of view, the idea beyond this work is to use a distributed controller to coordinate a fleet of moving robots in the same environment. To guarantee the absence of collisions between them we use the concept of artificial potential field, widely discussed in the literature. This approach can be naturally used for defining a proper cost function in the *MPC* framework. The proposed algorithm is based on the concept of robust MPC control, and in particular on the *tube based* strategy. Using this approach we are able to guarantee that the real controlled system, on which a bounded disturbance is supposed to act, remains constrained in a neighborhood of the trajectory of the ideal system, which is defined by neglecting the effect of uncertainties.

The proposed control algorithm has been validated using a simulation environment and then used to control a pair of real robots that move on the same working plane. Some of the most relevant results are presented in the last chapter. In the end the weakness of our approach and some possible future research directions are underlined.

1 – CONTROLLO DISTRIBUITO PER ROBOT MOBILI

1.1 – Introduzione

Classicamente per la gestione di sistemi caratterizzati da molteplici ingressi ed uscite, a partire dalle piccole fino alle grandi dimensioni, la teoria del controllo è stata sviluppata in base all'idea di fare ricorso ad un unico controllore centralizzato che abbia a disposizione il modello dell'intero sistema e tutte le informazioni raccolte su di esso ad ogni istante.

A tal proposito una particolare architettura di controllo molto diffusa in ambito industriale è quella detta di "*Model Predictive Control*", o *MPC*, alla quale sono stati dedicati, negli ultimi anni, numerosi studi volti a determinare condizioni che garantiscano proprietà di stabilità e robustezza alle incertezze. L'idea di fondo di tale tecnica è in realtà quella di trasformare il classico problema di controllo in un problema di ottimizzazione matematica, in modo da poter inserire in modo semplice vincoli e limitazioni presenti sul sistema reale.

In particolare il modello dell'intero sistema viene utilizzato per predire, in anello aperto e su di un orizzonte finito, l'evoluzione delle variabili di stato a partire del valore che esso assume all'istante attuale ed in funzione della sequenza di ingressi di controllo futuri. In questo modo, essendo tale sequenza di ingressi un grado di libertà nel problema considerato, la predizione può essere utilizzata all'interno di una cifra di merito per far sì che lo stato segua adeguatamente un determinato obiettivo, producendo in uscita esattamente il valore ottimo degli ingressi che consente di raggiungere le prestazioni desiderate. In accordo con la tecnica detta di "*Receding Horizon*", viene poi applicato al sistema solamente il primo degli ingressi calcolati dall'ottimizzatore, e la procedura viene ripetuta nuovamente sulla base della misura aggiornata dello stato attuale; tale approccio consente in realtà di formulare il problema di controllo aggiungendo facilmente vincoli statici e dinamici sul valore assunto dalle variabili di controllo o dalle variabili di stato lungo tutto l'orizzonte predittivo; non solo, i vincoli che è possibile inserire nell'ottimizzazione sono sia di tipo *hard*, ovvero il cui rispetto deve essere garantito, sia di tipo *soft*, implementati come termine di penalizzazione all'interno della cifra di merito.

Il controllo centralizzato, utilizzato in svariati campi applicativi, trova difficile applicazione al problema di gestione di sistemi su larga scala come ad esempio grandi impianti di produzione, oppure reti di trasporto come quella per il traffico stradale, per

la distrubuzione di energia elettrica, per la fornitura idrica, e così via. Tali sistemi infatti sono spesso caratterizzati, oltre che dalle grandi dimensioni, da una struttura composta di numerosi sottosistemi interagenti, ciascuno dei quali dotato di svariati sensori ed attuatori, con dinamiche complessive anche piuttosto complicate; d'altra parte essi rivestono una notevole importanza economica dunque è importante lo studio di soluzioni alternative che consentano di raggiungere obiettivi quali evitare la congestione dei canali, massimizzare il *throughput* e mantenere il più possibile limitati i costi legati alle variabili di ingresso.

Per questi motivi utilizzare un singolo controllore centralizzato con un elevato livello di intelligenza, per quanto idealmente consentirebbe di raggiungere le migliori performance, risulta spesso non essere praticamente fattibile a causa di elevate richieste computazionali oppure di ritardi nella comunicazione dell'enorme numero di informazioni necessarie quando sono coinvolti svariati nodi. Ancora, se ci si limita all'analisi in ambito commerciale, potrebbe accadere che alcune parti di una rete di distribuzione siano in carico a gestori differenti che non comunicano all'esterno sufficienti dettagli sulla sottorete in questione per consentire al sistema di controllo principale di svolgere al meglio il suo compito. Infine è bene tenere presente che una logica centralizzata riduce la robustezza e la *reliability* dell'architettura complessiva, rendendola più facilmente esposta a guasti e malfunzionamenti indesiderati e nello stesso tempo più difficile da manutenere.

Un'alternativa nel controllo di sistemi di grandi dimensioni o reti di trasporto è dunque quella di tipo distribuito o multi-agente. In particolare l'idea è quella di vedere l'intero sistema come composto da svariati sottosistemi più piccoli, ognuno dotato dei propri sensori ed attuatori, e lasciare che ognuno di questi sia controllato localmente da un regolatore con informazioni limitate e capacità di calcolo e di azione ridotte; per ottenere poi prestazioni simili a quelle di un controllore centralizzato si sfrutta l'interconnessione dei singoli *"agenti"* e lo scambio di dati volto al raggiungimento di una sorta di obiettivo comune.

A questa classe di architetture di controllo appartengono gli schemi di regolazione decentralizzata e distribuita. Ad esempio, nel più semplice schema di controllo decentralizzato, si suppone che sia possibile scomporre i vettori di variabili di ingresso ed uscita in dei set disgiunti che possono essere gestiti localmente mediante appositi regolatori; trascurando le possibili connessioni presenti tra i sottosistemi ottenuti si possono così implementare regolatori del tutto indipendenti, con l'obiettivo di raggiungere le prestazioni desiderate per il set di variabili di uscita di competenza per ciascuno di essi. Ovviamente il progetto di tali controllori è semplice ed efficace fintanto che le interazioni presenti sul sistema reale sono limitate o poco influenti, mentre la presenza di interazioni più marcate può compromettere il raggiungimento della stabilità o ridurre le prestazioni, soprattutto quando gli obiettivi dei vari sottosistemi sono in un qualche modo contrastanti. In ogni caso per sistemi di grandi dimensioni con accoppiamenti ridotti, la soluzione decentralizzata è una buona alternativa al problema

di controllo centralizzato poiché, con un piccolo deterioramento delle prestazioni, consente di ridurre notevolmente il peso computazionale, che viene distribuito sui singoli regolatori.

In realtà, tuttavia, si può pensare di migliorare le prestazioni dello schema decentralizzato appena descritto ed estenderne il campo di applicazione sfruttando lo scambio di informazioni tra i singoli controllori, in modo che ciascuno possa compiere il proprio lavoro essendo a conoscenza, anche solo parzialmente, del comportamento degli altri sottosistemi, così da poter predire ed in qualche modo compensare l'effetto delle interazioni che altrimenti verrebbero trascurate; si parla in tal caso di soluzioni di controllo distribuito.

Nel caso di implementazione distribuita del controllo predittivo *MPC*, un singolo agente ha a disposizione ad ogni passo di campionamento la sequenza delle sue azioni future, o ancor meglio la traiettoria predetta, in funzione di queste, all'interno dell'orizzonte considerato e che pertanto può comunicare a tutti gli altri agenti. In questo modo, supponendo che la sequenza effettiva non si discosti troppo da quella predetta, l'informazione scambiata può essere utilizzata facilmente da ciascuno degli altri controllori nella soluzione del proprio problema di ottimizzazione per tenere conto degli effetti di interazione, ottenendo complessivamente prestazioni simili a quelle che si otterrebbero nel caso centralizzato.

Gli algoritmi di controllo *MPC* distribuito possono essere classificati sulla base della loro architettura e della struttura utilizzata per lo scambio di informazioni. In particolare si hanno algoritmi cosiddetti *"fully connected"*, o completamente connessi, quando le informazioni sono scambiate, ad ogni istante di campionamento, tra tutti gli agenti della rete; al contrario si può pensare di realizzare un sistema solo parzialmente connesso, limitando lo scambio di dati solamente tra sottogruppi di regolatori per i quali l'interazione è più forte, così da ridurre in parte la dimensione del flusso di comunicazione e tutte le problematiche da esso derivanti come i ritardi, l'occupazione dei canali, etc.. Quest'ultima struttura distribuita è particolarmente vantaggiosa quando si hanno sistemi di grandi dimensioni con un elevato numero di sottosistemi parzialmente interagenti e risulta notevolmente interessante in tutti i casi pratici in cui le interazioni sono limitate solamente tra agenti "vicini"; chiaramente deve inoltre essere adottata per motivi pratici se si ricorre ad una struttura di comunicazione wireless di tipo punto punto, a causa delle limitazioni nel *range* dei dispositivi.

Una seconda classificazione può essere fatta sulla base del protocollo con cui le informazioni vengono scambiate ed utilizzate nella soluzione del problema di controllo; in particolare l'idea più semplice è quella di realizzare algoritmi "non iterativi" in cui l'informazione è trasmessa ed utilizzata una sola volta all'interno di un passo di campionamento; chiaramente però, se si pensa al concetto ideale del controllo distribuito di raggiungimento da parte dei singoli agenti di una forma di consenso globale che permetta di raggiungere prestazioni ottimali, è evidente che la soluzione non iterativa risulta in un certo senso limitante. Al contrario allora esistono algoritmi di

tipo iterativo in cui, all'interno di un singolo passo di campionamento, l'informazione tra i vari agenti viene scambiata più volte fino alla soluzione del problema, corrispondente ad un accordo sulle azioni che ciascuno deve intraprendere; di nuovo questa soluzione permette, come si può facilmente intuire, di raggiungere prestazioni più simili ad un controllo centralizzato, ma richiede che l'ammontare di dati trasmessi cresca notevolmente.

Infine l'ultima importante distinzione tra classi di controllori distribuiti basati su *MPC*, può essere fatta sulla base della cifra di merito utilizzata nell'ottimizzazione e sugli obiettivi dell'ottimizzazione stessa. In particolare si può pensare ad algoritmi di tipo "indipendente", in cui ciascun regolatore persegue un proprio obiettivo minimizzando, anche in presenza delle informazioni sulle interazioni, un indice di prestazione che potremmo definire locale. In contrapposizione abbiamo algoritmi di tipo "cooperativo" in cui la funzione di costo utilizzata ha un orizzonte di tipo globale, ed è realizzata con l'idea di rendere ottima l'uscita complessiva dell'intero sistema.

1.2 – Teoria dei giochi.

L'ultima distinzione apre un parallelo intressante tra il concetto di controllo distribuito e la teoria dei giochi. In questo ambito, infatti, vengono studiati analiticamente i comportamenti e le dinamiche, cooperanti e non, di sistemi composti da un certo numero di agenti interagenti tra loro; in particolare la teoria di interesse è quella dei giochi dinamici. A tal proposito un gioco viene detto "dinamico", o differenzale, se risulta rilevante l'ordine con cui le decisioni vengono prese, o in altre parole se la decisione presa ad un determinato istante di tempo t da uno degli agenti può dipendere dallo stato del sistema, che a sua volta dipenderà dalle decisioni prese negli istanti precedenti da tutti gli altri agenti in competizione; è immediato verificare che questo è esattamente il caso di interesse del controllo distribuito.

Esattamente come nel caso del controllo distribuito, si distinguono inoltre giochi non cooperativi quando ogni giocatore persegue un proprio interesse; chiaramente questa situazione può portare, a meno che gli interessi tra più giocatori siano coincidenti, ad obiettivi contrastanti poiché ciascun giocatore deve prendere decisioni basate sulla propria utilità; il conflitto nasce allora per la presenza di altri partecipanti, ciascuno dei quali deve prendere una decisione, che potremmo dire personale, in presenza degli altri.

Nella teoria dei giochi, accanto al concetto di singola azione, esiste quello di strategia, inteso come set di regole decisionali mediante le quali le azioni stesse vengono scelte sulla base delle condizioni dell'ambiente circostante; tale strategia, che rappresenta il punto centrale dello studio, può essere semplicemente prefissata, oppure dipendere da informazioni sullo stato del sistema, o infine avere un connotato di tipo probabilistico. Nel caso in esame diventa allora interessante parlare di strategia ottima, intendendo dal

punto di vista controllistico esattamente la scelta della miglior sequenza di ingressi in grado di produrre il risultato desiderato. Nel contesto del singolo giocatore, il concetto di ottimalità di una strategia è immediato da verificare, e corrisponde alla massimizzazione di una certa funzione di utilità, costruita sulla base dell'ambiente in cui esso opera. Al contrario nel caso multigiocatore tale significato risulta ambiguo, poiché è necessario specificare in che senso l'uscita complessiva del sistema, inteso come realtà complessa costituita da ciascun singolo sottosistema con i propri ingressi e le proprie uscite, ovvero da ciascun singolo giocatore, risulti migliore o peggiore di un altro possibile *output*.

A tal proposito, per far fronte all'ambiguità discussa ed avere un mezzo per la valutazione del comportamento dei singoli agenti nel contesto multigiocatore, si è soliti ricorrere a ben definiti concetti della teoria dei giochi che vanno sotto il nome di "*Pareto ottimalità*", *"Nash equilibrium"* e *"MaxMin strategy"*.

In particolare una strategia si definisce *Pareto-ottimale* se non esiste alcuna mossa in grado di migliorare la condizione di uno qualsiasi dei giocatori lasciando inalterata quella degli altri; ciò significa in altri termini che la strategia adottata sta massimizzando la funzione di utilità di ciascun singolo giocatore, assunta come criterio di ottimalità.

Nel caso del *Nash equilibrium* si valuta invece l'ottimalità dal punto di vista del singolo giocatore, con riferimento allo stato in cui si trovano tutti gli altri; in particolare il sistema si trova all'equilibrio se il giocatore considerato non è incentivato a cambiare la propria strategia fintanto che gli altri mantengono inalterata la propria, ovvero se la strategia che ha scelto di assumere è quella che massimizza, fissate le strategie altrui, la propria funzione di costo. Tale definizione è interessante poiché è possibile dimostrare che ciascun gioco possiede almeno un punto di equilibrio di questo tipo.

La strategia *maxmin*, infine, è una strategia, non necessariamente unica, basata sull'idea di rendere massima la funzione di costo del singolo giocatore nel suo caso peggiore, ovvero quando essa è resa minima dalle strategie degli altri giocatori; con questo approccio particolare si ottiene un metodo di scelta della propria strategia in grado di massimizzare l'utile atteso senza dover fare particolari assunzioni sulle strategie adottate dagli altri giocatori.

1.3 – Parallelo tra teoria dei giochi e controllo distribuito.

A questo punto è possibile vedere facilmente come tali definizioni possano essere riutilizzate in un contesto di controllo distribuito basato su *MPC*, come quello di cui si è discusso in precedenza, per classificare ulteriormente le varie architetture.

In particolare il problema di controllo distribuito può essere interpretato come un gioco dinamico multigiocatore infinito, in cui cioè ciascun singolo agente può intraprendere un numero illimitato di azioni. La strategia è allora, sotto quest'ottica, la sequenza di valori di ingresso futuri rispetto alla quale il problema di ottimizzazione deve essere risolto, mentre la cifra di merito è fatta di un insieme di funzioni di utilità continue nei loro argomenti, e genericamente quadratiche nella variabile di controllo. In questo modo un algoritmo classificato come indipendente, cioè basato sull'utilizzo di una cifra di merito locale a ciascun singolo regolatore, ed iterativo, ovvero fondato sulla ricerca, mediante un numero non definito di cicli di scambio di informazioni, di un consenso comune, rappresenta di fatto un gioco dinamico in cui ciascun agente persegue la ricerca di un *Nash equilibrium*. Al contrario un algoritmo distribuito iterativo e basato sulla cooperazione, dunque l'utilizzo di una funzione di ottimo comune all'intero sistema, non è altro che un gioco dinamico con l'obiettivo di raggiungere una soluzione *Pareto ottimale*, esattamente come farebbe un unico controllore centralizzato in possesso del modello dell'intero sistema e di tutte le informazioni necessarie. Infine, una strategia di tipo *MaxMin*, è sostanzialmente quella utilizzata nella soluzione di un problema di controllo distribuito di tipo non cooperativo e, a differenza dei precedenti, di tipo non iterativo; in tal caso infatti, ad ogni periodo di campionamento, si ricorre ad un solo ciclo di trasmissione delle informazioni tra agenti vicini, dopodiché ciascun agente calcola la propria strategia ottima minimizzando una cifra di merito locale nella quale sono inserite le informazioni sulle traiettorie predette ricevute, nell'ipotesi che queste siano effettivamente inseguite, entro certi limiti, da ciascun agente; in questo modo, allora, qualsiasi eventuale errore nell'inseguimento viene a configurarsi come un disturbo limitato, e compensato dall'utilizzo di un approccio di controllo di tipo robusto.

Ricapitolando, dunque, il controllo distribuito, sia esso implementato in forma cooperativa o in forma non cooperativa, iterativa o non iterativa, consente tramite lo scambio di informazioni tra i singoli controllori di raggiungere prestazioni paragonabili a quelle che si otterrebbero idealmente con un unico controllo centralizzato in grado di conoscere l'intero sistema. Il comportamento che si ottiene quando tale approccio è sviluppato secondo la filosofia *MPC*, che permette ad ogni passo di avere a disposizione non solo il valore da applicare alla variabile di controllo ma anche la predizione lungo un determinato orizzonte della traiettoria assunta dalle variabili di stato, rappresenta un tipico panorama dinamico studiato nell'ambito della teoria dei giochi. Tale interessante analogia consente allora di prevedere l'evoluzione delle strategie di ciascun singolo agente verso un consenso o un punto di equilibrio comune, e di capire intuitivamente l'effetto delle interazioni.

1.4 – Il controllo MPC nel coordinamento.

Lo studio di metodologie di controllo distribuito come quelle descritte, ad esempio basate su tecniche di tipo predittivo, riveste dunque una notevole importanza dal punto di vista economico poiché consente di migliorare le prestazioni di una soluzione puramente decentralizzata nel controllo di impianti su larga scala, di reti di trasporto, di reti di gestione del traffico terrestre ed aereo, per le quali il ricorso ad un controllo centralizzato risulterebbe impossibile o difficile da implementare.

A proposito di quest'ultimo ambito un interesse particolare si ha nel campo della robotica mobile riguardante lo studio del comportamento cooperativo di gruppi di robot, a partire dal movimento in formazione, allo svolgimento di *task* complessi in collaborazione o al completamento di *task* multipli nel medesimo ambiente con il minimo intralcio. Tutti questi aspetti si rivelano infatti utili in applicazioni di tipo industriale, in applicazioni di sorveglianza, nell'esplorazione di ambienti ignoti o pericolosi o in applicazioni militari.

Per questo motivo lo scopo del progetto presentato è quello di studiare una tecnica di controllo predittivo distribuito in grado di coordinare il movimento sul piano di una flotta di robot, in assenza di un sistema di supervisione. Una volta costruita tale legge, ed assicurati contemporaneamente il suo funzionamento ed il rispetto di proprietà di stabilità e robustezza, l'idea è quella di utilizzarla per analizzare i possibili comportamenti che si ottengono in risposta a diversi meccanismi di consenso per la gestione del traffico. In particolare, per raggiungere tale obiettivo, i passi da svolgere saranno la comprensione del funzionamento dei dispositivi a disposizione, del metodo di programmazione e di comunicazione, la realizzazione di un sistema di rilevamento della posizione in coordinate assolute, lo studio della letteratura alla ricerca di soluzioni per la stabilizzazione del movimento ed il superamento degli ostacoli ed infine la realizzazione di un sistema distribuito per la cooperazione.

1.5 – Struttura dei capitoli.

La trattazione è strutturata al seguente modo. Il Capitolo 2 descrive l'apparato sperimentale a cui si fa ricorso nel seguito, i dispositivi utilizzati e gli algoritmi che ne consentono il funzionamento. Il Capitolo 3 è una panoramica delle tecniche di controllo per robot mobili, presenti in letteratura, ritenute più interessanti ed adatte al caso in esame; dopo una breve presentazione ciascuna di esse viene valutata sul sistema reale con l'obiettivo di trovarne una di riferimento. Il Capitolo 4 affronta il problema dell'*obstacle avoidance* anch'esso al fine di scegliere una tecnica che consenta di implementare un meccanismo di movimento senza collisioni tra i robot. Il Capitolo 5

introduce il controllo *MPC* e presenta una sua applicazione al controllo del movimento di un singolo robot; in tale ambito viene anche testata la metodologia scelta per l'*obstacle avoidance.* Il Capitolo 6 è basato sull'applicazione del controllo robusto di tipo *tube based* al problema di coordinamento in esame; in particolare viene presentata la sua versione distribuita. Infine il Capitolo 7 riassume i risultati trovati con la tecnica descritta in alcune situazioni tipiche di gestione del traffico.

2 – L' APPARATO SPERIMENTALE

2.1– Introduzione.

Come sintetizzato nel precedente capitolo l'idea fondamentale del progetto è quella di poter studiare l'effetto di algoritmi di controllo distribuito sul coordinamento ed in particolare in un contesto multiagente ispirato al problema di gestione del traffico.

A tal proposito si è scelto di fare ricorso ad una piccola flotta di robot mobili che permettesse, in un primo momento, di risolvere il problema di posizionamento di ciascuno di essi all'interno del medesimo ambiente, e dunque con possibilità di interazione.

Al momento della scelta dei dispositivi i requisiti fondamentali sono stati individuati nelle seguenti caratteristiche:

- **Dimensioni ridotte**, che consentano di gestire più facilmente la flotta, anche in vista di un possibile aumento del numero di agenti coinvolti.
- **Elevate capacità di movimento**, per poter compiere manovre di aggiramento degli ostacoli.
- **Comunicazione wireless**, necessaria per implementare una filosofia di controllo distribuita, in cui ciascun robot, dotato del proprio regolatore, deve poter scambiare informazioni con gli agenti circostanti.
- **Discrete capacità di calcolo**, che consentano di svolgere le operazioni fondamentali necessarie come gestire la comunicazione, applicare i comandi desiderati agli attuatori ed eventualmente ospitare una parte dell'algoritmo di controllo. Per non complicare il problema, infatti, data la complessità computazionale di un metodo come *MPC* anche in presenza di una filosofia distribuita, si è scelto di fare ricorso ad un calcolatore che simuli i diversi controllori di bordo, mantenendoli tra loro virtualmente separati.
- **Semplicità di programmazione**
- **Costo limitato**

Inoltre si è dovuto tener conto anche della problematica di posizionamento, ovvero delle possibilità di integrazione tra i robot scelti ed un sistema di localizzazione assoluto, descritto più in dettaglio nel seguito.

2.2 – Il robot epuck.

Tenendo presenti queste linee guida ed analizzando le possibilità presenti sul mercato, si è scelto di utilizzare un robot chiamato *E-Puck* sviluppato dall'Ecole Polytechnique Fédérale de Lausanne *(EPFL)* per scopi di ricerca nel campo della *swarm robotics*, ovvero dello studio del comportamento collettivo. Questo rispetta infatti tutti i requisiti sopra menzionati ed inoltre è realizzato secondo una filosofia aperta, che rende disponibili sulla rete un gran numero di informazioni riguardanti l'utilizzo pratico della piattaforma. Nel seguito sarà descritto brevemente l'hardware a disposizione, per informazioni dettagliate si rimanda alla pubblicazione degli ideatori in [1].

Il robot in questione è controllato da un microcontrollore *dsPic30F6014A* a 16 bit con una frequenza di clock di $64\,MHz$, che può essere programmato in linguaggio *C* attraverso l'ambiente di sviluppo *MPlab* della casa produttrice. Il firmware scritto può essere caricato direttamente sfruttando un *bootloader* che consente la programmazione tramite porta seriale. La limitazione principale deriva dalla memoria a disposizione che è solamente di $8\,kB$ per la *RAM* e di $144\,Kb$ per la memoria *flash*.

La connessione wireless è realizzata tramite un modulo *Bluetooth* che consente di gestire la comunicazione verso un *PC* o verso altri robot dello stesso tipo fino ad un numero di sette; tale dispositivo permette anche di creare un servizio di porta seriale, che facilita la trasmissione di dati e consente di programmare il microcontrollore attraverso un canale wireless.

L'attuazione del movimento è lasciata a due motori passo passo con una risoluzione di $1000\,steps/giro$ ed una velocità massima di un giro al secondo, che possono essere controllati direttamente imponendo la velocità desiderata in numero di passi al secondo. Questi sono disposti in modo da avere due ruote indipendenti sul medesimo asse centrale del robot nella configurazione ad uniciclo che nel seguito sarà descritta più dettagliatamente.

Il resto dell'*hardware* è composto da un insieme di sensori e piccoli attuatori che si è scelto di non utilizzare in questo progetto; abbiamo in particolare a disposizione dei sensori di prossimità ad infrarosso, disposti intorno al robot, un accelerometro, un microfono ed una piccola videocamera *VGA* frontale. Abbiamo inoltre come uscita una serie di led ed un altoparlante che possono essere sfruttati semplicemente come meccanismo di segnalazione. In sintesi le funzionalità del robot possono essere riassunte nello schema di Figura 2.2.1

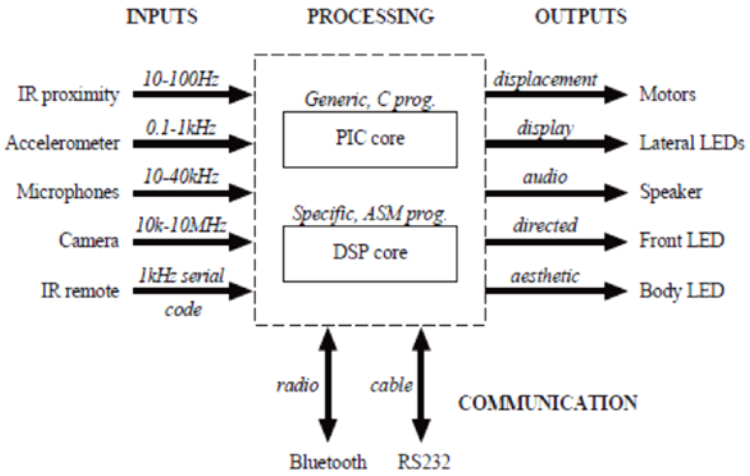

Fig. 2.2.1 *– Hardware Epuck.*

Per quanto riguarda la parte *software* sono disponibili in rete numerose librerie scritte in linguaggio *C,* oppure in *Assembler* per le componenti più delicate*,* che consentono di gestire direttamente i dispositivi elencati senza doversi preoccupare delle problematiche di basso livello. In particolare, queste sono state utilizzate nella scrittura di una routine in grado di predisporre ogni singolo robot ad essere pilotato direttamente dal *PC,* ricevendo ed interpretando comandi attraverso il canale di comunicazione wireless.

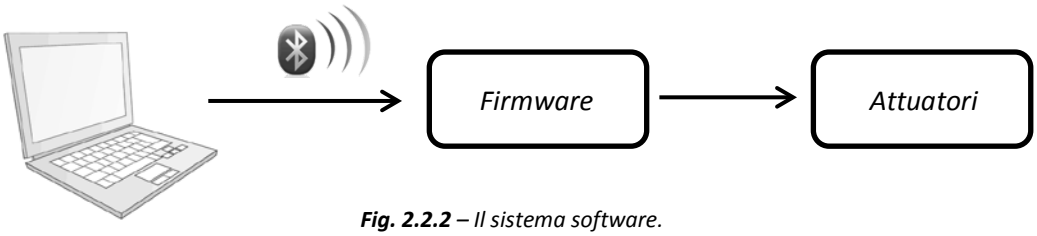

Fig. 2.2.2 *– Il sistema software.*

A tal proposito si è scelto, in una prima fase del progetto, di gestire tutto tramite *MATLAB* e di implementare sul microcontrollore solamente una procedura di sincronizzazione, il canale di comunicazione e la routine di interpretazione ed applicazione dei comandi ai motori, con una particolare attenzione alla saturazione degli ingressi di controllo; come già detto si è sfruttato per questo la possibilità di vedere la connessione wireless come porta seriale. Per il codice utilizzato si rimanda all'appendice.

La scelta dell'ambiente *MATLAB* per la gestione lato calcolatore dell'intero progetto è stata fatta per consentire un più agevole ricorso alle tecniche moderne di ottimizzazione che saranno necessarie per la soluzione del problema *MPC.* In realtà per alcuni aspetti come la localizzazione tramite sistema di visione di cui si discuterà nel seguito, oppure la comunicazione *wireless* su porta seriale, sarebbe maggiormente efficiente

un'implementazione attraverso linguaggi di più basso livello, come ad esempio il *C* e le librerie grafiche *openCV.* Tuttavia, data la particolare attenzione che questo aspetto richiederebbe, e che esula dagli scopi della presente trattazione, si è scelto di lasciare tale tipo di approfondimento ad eventuali sviluppi futuri.

2.3 – Il modello cinematico.

A questo punto è bene considerare anche la struttura fisica del robot. In particolare esso è costituito da una struttura circolare alla quale sono connesse due ruote coassiali indipendenti sull'asse diametrale; il terzo punto d'appoggio è realizzato semplicemente mediante la struttura stessa. Le dimensioni sono di $52\ mm$ per l'asse tra le ruote e $21\ mm$ per il diametro delle ruote stesse. In questo modo la massima velocità di spostamento in linea retta, corrispondente alla massima velocità di rotazione dei motori di un giro al secondo, è di $12.9\ cm/s$. Lo schema con il quale si può modellare il sistema è il seguente:

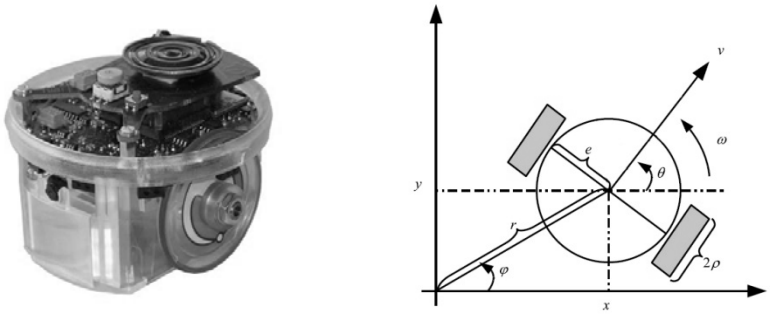

Fig. 2.3.1 *– Il robot Epuck ed il suo modello cinematico.*

In particolare si può supporre che il punto di appoggio sia neutrale ai fini del moto, e pertanto ininfluente sulla dinamica finale, e di conseguenza modellare il robot come se fosse costituito dalle sole due ruote attuate.

La struttura in questione può allora essere descritta matematicamente secondo un modello, molto diffuso in letteratura, denominato *unicycle model* oppure, in italiano, modello ad uniciclo. Supponiamo in particolare di considerare il robot come un corpo rigido e di descrivere il suo comportamento attraverso la posizione del suo centro geometrico e la sua orientazione, valutata ad esempio come angolo tra l'asse delle ascisse di un sistema di riferimento cartesiano e l'asse principale di movimento, che per ipotesi sarà ortogonale all'asse delle ruote.

Inoltre teniamo presente che, sebbene nel nostro caso le variabili di ingresso a disposizione siano le due velocità di rotazione applicate a ciascuna delle ruote, indicate

con ω_R ed ω_L, è possibile, conoscendo la struttura del robot, darne una rappresentazione del tutto equivalente nei termini della velocità lineare del centro e della velocità angolare dell'intero corpo, indicate con v ed ω. In dettaglio conosciamo il raggio delle ruote R e la lunghezza del loro asse $2E$ pertanto, tramite semplici relazioni geometriche, possiamo porre:

$$\begin{cases} v_R = \omega_R R \\ v_L = \omega_L R \end{cases} \qquad v = \frac{v_R + v_L}{2} \qquad \omega = \frac{v_R - v_L}{E}$$

ottenendo due nuove e più intuitive variabili di controllo per il sistema.

Utilizzando queste due nuove variabili, la rappresentazione, in coordinate cartesiane, delle equazioni cinematiche che governano il moto del robot è semplicemente la seguente:

$$\begin{cases} \dot{x} = v\, cos\theta \\ \dot{y} = v\, sin\theta \\ \dot{\theta} = \omega \end{cases}$$

e coincide, nel caso generale, con il modello ad uniciclo presente in letteratura, al quale ci si rifarà nel seguito per il progetto delle leggi di controllo. Del resto il passaggio dalle nuove alle vecchie variabili di controllo è dettato da una relazione puramente algebrica e quindi invertibile senza alcun problema ed alcun carico computazionale.

Notiamo ora che, in generale, un corpo rigido in moto planare è caratterizzato da tre gradi di libertà, che possiamo far coincidere come nel caso considerato con la posizione di un punto appartenente al corpo, ad esempio il centro, e l'orientazione complessiva. Nel caso considerato, tuttavia, il robot è vincolato ad avanzare solamente in direzione perpendicolare all'asse delle ruote mentre la rotazione del corpo è lasciata alla rotazione differenziale delle due ruote fino al limite del movimento sul posto. Tale comportamento è legato alla presenza, implicita nel modello scritto, di un vincolo anolonomo che possiamo scrivere nella forma seguente:

$$\dot{y}\, cos\theta - \dot{x}\, sin\theta = 0$$

Questo costringe a prestare alcune attenzioni in fase di controllo dal momento che limita le possibilità di movimento.

Quando si parla di controllo di sistemi non lineari, ed in particolare sistemi soggetti ad un vincolo di tipo anolonomo, come nel caso considerato dell'uniciclo, un risultato da tenere presente è il cosiddetto teorema di *Brockett* presente in [2]. Questo affronta il problema di stabilizzabilità, tramite leggi in *feedback,* di un generico stato di equilibrio e

permette di dimostrare che un sistema come quello considerato non può essere stabilizzato asintoticamente utilizzando una legge di controllo in retroazione continua e tempo invariante. Sebbene l'enunciato e la dimostrazione del teorema saranno tralasciati, rimandando per i dettagli alla pubblicazione dell'autore, l'indicazione principale che ne deriva è interessante e dice che, fissato un punto di equilibrio z^*, sistemi che possono essere ricondotti alla struttura seguente *(chained form)*:

$$\dot{z} = \sum_{i=1}^{m} f_i(z)u_i \qquad z \in \mathbb{R}^n \quad u_i \in \mathbb{R}^m$$

con un numero n di variabili di stato strettamente superiore al numero m delle variabili di ingresso, non possono essere stabilizzati con una legge della forma citata se in un intorno compatto dell'equilibrio i vettori $f_i(z)$ sono tra loro indipendenti e continuamente differenziabili.

A tal proposito è semplice vedere che il modello del robot descritto in coordinate cartesiane ricade esattamente in questa categoria e, pertanto, che le possibili leggi di controllo in *feedback* che troveremo in letteratura saranno tutte di tipo discontinuo oppure tempo variante. Al contrario vedremo nel seguito che questo non vale più se si fa ricorso ad una rappresentazione in coordinate polari, per la quale è quindi possibile progettare leggi di tipo continuo.

2.4 – Rilevamento e stima della posizione.

Dopo aver descritto le caratteristiche dei robot scelti per il progetto, ed aver visto che dal punto di vista cinematico essi possono essere trattati come degli unicicli, è necessario occuparsi della tematica del rilevamento di posizione, ovvero, in termini più precisi, della misura del vettore di stato (x, y, θ).

A tal proposito, durante la fase di scelta dell'*hardware*, è stato fatto il punto della situazione sulle possibili tecnologie a disposizione per la misura *indoor* della posizione e, se possibile, dell'orientamento di ciascun robot. Il principale requisito è la possibilità di poter distinguere l'identità di ogni agente all'interno dell'area di lavoro e di avere una frequenza di aggiornamento delle informazioni sufficientemente elevata.

La prima idea per il rilevamento di posizione ed orientamento di un robot del tipo considerato può essere quella di far ricorso all'odometria, ovvero all'integrazione numerica delle velocità campionate alle due ruote. Il vantaggio viene in realtà dal fatto che tale soluzione è genericamente a costo zero, poiché le grandezze in questione vengono già di per sé misurate per chiudere l'anello di controllo più interno che consente al motore di inseguire il riferimento di velocità, esattamente come nel caso in questione. L'unico passo da fare è allora quello di implementare uno stimatore, che nel

caso più semplice di integrazione può essere descritto ad esempio dalle seguenti espressioni:

$$v = \frac{v_L + v_R}{2} \qquad \omega = \frac{v_R - v_L}{E} \qquad \begin{cases} x_{new} = x_{old} + \Delta T\ v\ cos(\theta_{old}) \\ y_{new} = y_{old} + \Delta T\ v\ sin(\theta_{old}) \\ \theta_{new} = \theta_{old} + \Delta T\ \omega \end{cases}$$

indicando con ΔT il periodo di campionamento scelto. Chiaramente tale soluzione è di tipo relativo, ovvero è strettamente connessa alla conoscenza precisa della posizione e dell'orientazione iniziale del robot; inoltre essa è soggetta, come tutti i metodi basati sull'integrazione, al problema del *drift* che si verifica al passare del tempo, rendendo il dato accumulato scorretto; tuttavia la tecnica permette di produrre dei dati ad una frequenza potenzialmente elevata. Per questo motivo si ritiene una soluzione valida solamente allo scopo di fornire dati all'interno dell'intervallo di campionamento di una tecnica più lenta ma più precisa, magari di tipo assoluto; nel nostro caso, comunque, tale aspetto viene trascurato.

Un'idea per il rilevamento assoluto della sola posizione può essere invece quella di far ricorso ad un sistema di triangolazione basato sulla misura, a bordo dei robot, di segnali provenienti da stazioni in posizione prefissata, o viceversa nella misura effettuata dalle stazioni di segnali prodotti dai robot. Un possibile esempio di segnale adatto allo scopo è quello radio, con sequenze di trasmissione diverse a seconda di ciascun robot; questo, rispetto ad uno stesso sistema basato su segnali luminosi, ha il vantaggio di non soffrire del problema della schermatura che potrebbe essere causata dal movimento degli altri robot. Un altro possibile segnale con le stesse caratteristiche è quello magnetico, che tuttavia richiede un notevole dispendio energetico nella fase di trasmissione, che pertanto non può essere implementata a bordo dei robot, complicando poi il problema del riconoscimento. In questi casi sarebbe comunque necessario equipaggiare i robot con opportuni sistemi di trasmissione o sensori ed algoritmi di rilevamento, e predisporre in modo complementare nell'area di lavoro degli opportuni punti di ricezione o trasmissione. Sicuramente il costo complessivo di una tecnica di questo tipo risulta elevato. Queste soluzioni sono allora interessanti, dato l'elevato *range* raggiungibile dai segnali citati, quando l'area da coprire è notevole, come ad esempio quella di una stanza, mentre risultano svantaggiose quando gli spazi sono piccoli, dal momento che richiedono notevole *hardware* aggiuntivo.

In tal caso, invece, ovvero se l'area da considerare è limitata, si può pensare di fare ricorso ad un sistema di rilevamento visivo posto al di sopra del piano di lavoro, in grado di scattare un'immagine della zona interessata e distinguere all'interno di essa i vari robot. A differenza della soluzione precedente, questo richiederebbe di utilizzare solamente una telecamera esterna, e non necessiterebbe di alcun componente attivo a bordo dei robot, quanto al massimo delle soluzioni di segnalazione del tutto passive. In questo modo, dunque, non è necessario aggiungere ai microcontrollori un ulteriore carico di lavoro oppure aumentare il dispendio energetico dei robot, rendendo più pratico il loro utilizzo. Oltretutto, l'utilizzo di opportune soluzioni per la marcatura dei

robot, può consentire di rilevare, oltre alla posizione, anche l'orientamento, che con la tecnica precedente sarebbe difficile da ottenere.

D'altra parte, se si considerano le reali richieste del progetto e si suppone di implementare il tutto in un'area delle dimensioni comparabili a quelle di una scrivania, si può vedere che anche telecamere di livello commerciale sono del tutto adatte allo scopo, e che ad esempio si può fare ricorso ad una semplice *webcam*, disponibile ad un costo veramente limitato.

Quest'ultima possibilità, in particolare, si è rivelata essere interessante per una prima implementazione del progetto e pertanto altre soluzioni di rilevamento sono state accantonate. Più in dettaglio si è scelto di realizzare il sistema di misura facendo ricorso ad una comune *webcam* interfacciata al *PC* e controllata anch'essa via *MATLAB*; in questo ambiente si è progettato un semplice algoritmo di analisi delle immagini in grado di ricavare i parametri desiderati di posizione ed orientamento.

2.5 – Implementazione su webcam.

L'idea utilizzata è quella di distinguere i vari robot attraverso dei *marker* di diverso colore e di dare a questi una forma particolare che consenta di misurare in modo univoco l'orientamento sul piano. In particolare si è scelto di ricorrere a due zone colorate di forma circolare per ciascun robot, poste diametralmente una rispetto all'altra:

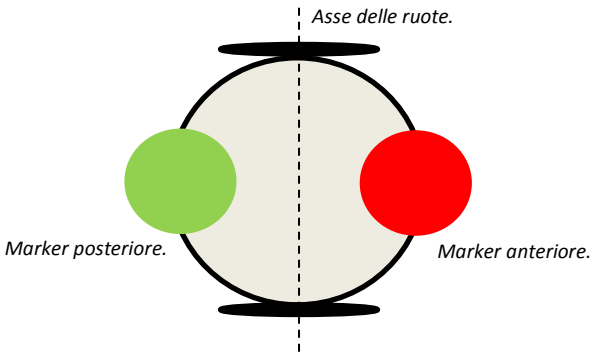

Fig. 2.5.1 *– Disposizione dei marker per il rilevamento di posizione ed orientazione.*

e di ricavare la posizione del centro geometrico a partire dalla posizione di queste due. Per calcolare la posizione di un marcatore di questo tipo, infatti, si possono utilizzare delle routine già disponibili che analizzano un'immagine alla ricerca di zone contigue di punti omogenei e permettono di estrarre, a partire da queste, delle particolari informazioni, come ad esempio la posizione del centro o dei parametri geometrici di vario genere.

Per questo motivo l'intero algoritmo è stato progettato nel modo seguente. Innanzitutto viene acquisita, ad ogni richiesta, un'immagine a colori dalla videocamera e, per ognuno dei due marcatori, questa viene filtrata opportunamente al fine di selezionare il solo colore corrispondente. Fatto questo l'immagine risultante viene convertita in bianco e nero evidenziando così nettamente la zona circolare corrispondente; infine di tale zona viene calcolata la posizione del centro. Ottenute in questo modo le posizioni dei centri dei due *marker* non resta che applicare delle semplici considerazioni geometriche e ricavare sia la posizione che l'orientazione del robot desiderate, avendo così a disposizione l'intero vettore di stato.

In realtà, nella definizione delle coordinate utilizzate per indicare la posizione del robot, è necessario scegliere se fare ricorso ad un sistema di riferimento reale, oppure direttamente al sistema di riferimento intrinseco all'immagine che si ha in uscita dall'algoritmo descritto. In entrambi i casi è allora necessario identificare una opportuna trasformazione che consenta di passare tra i due spazi; questo può essere fatto attraverso una procedura di calibrazione che produce una mappa dal mondo reale al mondo dei *pixel*, detta comunemente *camera matrix*. Senza scendere nei dettagli, per i quali si rimanda all'ampia trattazione disponibile in rete, si è scelto di fare ricorso ad una semplice procedura iterativa per la determinazione dei parametri propri della telecamera che consentono di generare la mappatura desiderata e di rappresentare in seguito tutte le grandezze utilizzando un sistema di riferimento reale.

Ricapitolando, dunque, il ricorso al sistema di visione e l'utilizzo della tecnica, descritta finora a grandi linee, per la misura della posizione e dell'orientamento di ciascun robot, consente di considerare il vettore di stato che descrive il sistema ad uniciclo come direttamente accessibile per il sistema di controllo che vedremo nel seguito.

3- TECNICHE DI CONTROLLO PER ROBOT MOBILI

3.1 – Introduzione.

La maggior parte dei robot mobili dotati di due sole ruote, per le quali è possibile imporre indipendentemente le velocità di rotazione, è modellata come un sistema meccanico caratterizzato da vincoli non olonomi. Il problema di controllo di questa categoria di robot risulta in ogni caso possibile sebbene sia necessario tener presente di alcune limitazioni.

Abbiamo già visto che un sistema modellabile come uniciclo è caratterizzato dai tre gradi di libertà, tipici di un corpo rigido che si muove sul piano, e due ingressi di controllo disponibili, che corrispondono alla velocità lineare v, diretta lungo un asse perpendicolare alle ruote, ed alla velocità angolare ω. Il sistema, sebbene risulti controllabile in anello aperto, si dimostra incapace di essere stabilizzato mediante l'utilizzo di un feedback continuo e tempo invariante, ovvero in altre parole non esiste una legge di controllo algebrica in grado di stabilizzare semplicemente il sistema in un qualsiasi equilibrio.

Gli ingressi a disposizione consentono comunque di controllare il sistema portandolo in una posizione qualsiasi e con l'orientazione desiderata, ovvero in altre parole che questo risulta completamente controllabile, esattamente come desiderato. Il problema di navigazione che ci poniamo è dunque ben posto. Dopo aver descritto il sistema mediante le equazioni viste in precedenza, possiamo prendere in considerazione il problema di controllo vero e proprio. A tal proposito, analizzando la vasta letteratura riguardante il problema di controllo del moto di robot mobili, si possono distinguere sostanzialmente tre diverse metodologie per la ricerca di una soluzione.

Un primo approccio è quello del controllo di tipo *sensor based,* incentrato sul problema di pianificazione interattiva del movimento all'interno di un ambiente di lavoro di tipo dinamico. Con questa definizione si intende, in particolare, un ambiente la cui struttura può variare nel tempo, ad esempio per la presenza di altri robot in movimento o per l'interazione di operatori umani. A causa di tale variabilità imprevista è necessario fare ricorso a dei sensori, tipicamente a bordo del robot, che consentano di interagire con l'ambiente stesso; gli schemi di controllo solitamente utilizzati sono di tipo intelligente, come ad esempio quelli basati su logiche fuzzy oppure sull'apprendimento tramite reti neurali.

La seconda categoria suddivide il problema di navigazione in una fase di pianificazione del percorso e, solo successivamente, in una fase di esecuzione; in questo caso viene innanzitutto generato un percorso privo di collisioni ricorrendo ad una mappa, nota a priori, dello spazio di lavoro, dopodiché un controllore di tipo *path-following* si occupa di garantire che il robot segua il percorso progettato. In questa fase di esecuzione possono essere implementati opportuni algoritmi di ottimizzazione basati su criteri come il minimo tempo di viaggio, la minima distanza percorsa, oppure la minimizzazione dell'energia richiesta.

Infine un terzo approccio al problema di navigazione si ottiene facendo uso direttamente di una filosofia di tipo *motion control*, il cui obiettivo è in altre parole rappresentato semplicemente dall'inseguimento, il più possibile accurato, di un segnale di riferimento desiderato caratterizzato da ben specificati requisiti temporali.

L'idea proposta nel nostro lavoro è di fatto costituita da un insieme di caratteristiche prese dalle categorie presentate ma d'altra parte non può essere ridotta a nessuna di esse. In particolare l'obiettivo è quello di sfruttare una strategia di controllo predittivo di alto livello per risolvere *online,* ed in tempo reale, il problema di navigazione e coordinamento, senza dover definire in anticipo una traiettoria da seguire, ovvero senza una fase esplicita di *path planning*. Tale navigazione deve inoltre poter avvenire in presenza di ostacoli fissi noti e di ostacoli mobili rappresentati dagli altri robot; di questi si suppone di poter conoscere, grazie all'approccio predittivo, la traiettoria futura, e pertanto il controllo del movimento avviene all'interno di un ambiente solo parzialmente incerto. In realtà la tecnica predittiva che si pensa di utilizzare agisce da pianificatore, ma solo limitatamente all'orizzonte di predizione che si considera, senza necessariamente definire l'intero percorso dalla posizione attuale fino all'obiettivo; inoltre non si tiene in considerazione, per il momento, il fattore temporale, supponendo che non vi siano requisiti particolari sul tempo di viaggio. Prima di poter applicare il controllo di alto livello appena descritto, tuttavia, sarà necessario predisporre un'opportuna legge in grado di controllare, parallelamente, il moto vero e proprio del robot; l'obiettivo di queste pagine è dunque quello di trovare una soluzione a tale problema.

Nella fase di progettazione è infine altrettanto necessario tenere presente che un robot mobile, come quello a disposizione ad esempio, è caratterizzato da limitazioni fisiche sulla velocità di movimento. Tali limitazioni possono essere formalizzate nella pratica come valori massimi sulle velocità di rotazione applicabili alle due ruote, e dunque conseguentemente rappresentano dei limiti ai segnali di controllo a disposizione, che dovranno prevedere opportune saturazioni.

3.2 – Path following e trajectory tracking.

Tralasciando d'ora in poi, in questo capitolo, il problema di come le traiettorie o il percorso di riferimento siano generati, nella vasta letteratura sull'argomento si riscontrano due diversi punti di vista riguardanti il problema di controllo del moto di un robot. Più nel dettaglio se è necessario predisporre un movimento con requisiti temporali ben specificati, il problema da risolvere è categorizzato come inseguimento di una traiettoria nelle variabili di stato, intesa come segnale di riferimento vero e proprio. Tuttavia molto più spesso, nelle situazioni pratiche, il tempo può essere entro certi limiti messo da parte così che il problema vero e proprio resta quello di inseguire un percorso puramente geometrico. In letteratura si identifica solitamente il primo caso con il termine *trajectory tracking*, mentre il secondo viene denominato *path following*.

Si considerino le coordinate che rappresentano lo stato del robot uniciclo $q(t)$ e le coordinate desiderate $q_r(r)$. Si definisce genericamente l'errore $e_q(t) = q(t) - q_r(r)$. Supponendo di conoscere l'intera traiettoria la tipologia di *tracking* può essere progettata a priori.

Sotto quest'ottica l'inseguimento può essere codificato identificando il parametro r con l'istante di tempo attuale, ovvero ponendo nella parametrizzazione $r(t) = t$. Questo approccio può essere esteso considerando per il parametro r una generica funzione del tempo; tale funzione si rifletterà direttamente sulla scrittura dell'errore di inseguimento. Si nota che possiamo in questo modo andare ad alterare la traiettoria applicando una scalatura temporale appropriata tramite $r(t)$, ad esempio scegliendo genericamente $r = a\,t$.

L'alternativa di puro *path following* è basata su di una relazione stabilita tra lo stato attuale del sistema $q(t)$ e l'intero percorso memorizzato, e non solamente con uno dei suoi punti. Tale relazione deve essere una proiezione in grado di restituire istante per istante il punto della traiettoria da inseguire, $q_d(r)$, ovvero il punto a cui far convergere il sistema. Il parametro descrittivo r deve venire interpretato come funzione della posizione attuale del robot e del percorso da seguire, ovvero assumere una forma $r = \pi(q)$, con π una proiezione di qualsiasi tipo della posizione attuale sul *path*. Il sistema di controllo deve far sì che il sistema reale insegua tale punto. L'unica distinzione rispetto ad un metodo di *tracking* risiede nel fatto che con questo approccio, ovvero slegando completamente il parametro descrittivo r dal tempo, non è in alcun modo garantito che il sistema raggiunga l'obiettivo sulla traiettoria desiderata entro un intervallo di tempo ben definito.

Supponiamo che sul sistema reale agisca una perturbazione che lo costringe a restare fermo nella posizione di partenza; se il problema è impostato come *trajectory tracking* il punto obiettivo continuerà a muoversi indipendentemente dal comportamento del sistema e ciò significa che l'errore di inseguimento crescerà fino ad un valore sufficientemente elevato da rischiare di provocare dei comportamenti instabili nel sistema controllato. Nel caso contrario invece, con un approccio *path following*, il punto

da inseguire resta fermo in risposta alla perturbazione, poiché legato direttamente allo stato del robot che resta inalterato. In questo modo è possibile imporre un limite all'errore ed evitare il raggiungimento di una condizione di instabilità. Combinare entrambi gli approcci attraverso un parametro di peso ha lo scopo di ottenere i benefici del *path following* quando gli errori sono grandi ma di preservare i vincoli temporali nel completamento del compito; se una qualsiasi perturbazione fa sì che gli errori diventino ampi, il punto desiderato tende ad essere scelto secondo una filosofia *path following*, così da non avanzare lungo il percorso; quando invece il robot riesce a recuperare la distanza ed avvicinarsi al *path*, il punto desiderato viene selezionato secondo la filosofia *trajectory tracking*. In sintesi è sufficiente nella pratica fare in modo che il parametro r possa tendere all'equivalente temporale t, fino al caso limite ideale di inseguimento classico.

In una tecnica *trajectory tracking* pura la variazione di r ha una dinamica rigidamente imposta ad $\dot{r} = 1$ oppure, se si considera un'estensione più generale, ad una funzione di tipo $\dot{r} = f(t)$. Per rilassare tale vincolo si può fare in modo di progettare la dinamica come un'opportuna funzione dell'errore di inseguimento corrente, ovvero scegliere $\dot{r} = g(e_q)$.

Una funzione che comprenda entrambe le caratteristiche deve essere progettata tenendo conto di alcune proprietà fondamentali. Innanzitutto se l'errore e_q è piccolo, essa deve tendere all'unità, così che l'inseguimento ritorni equivalente al caso deterministico, ovvero far sì che robot reale e riferimento possano avanzare contemporaneamente. Al contrario, se l'errore assume un valore grande, ovvero se veicolo ed obiettivo sono ad elevata distanza, il punto di riferimento deve potersi fermare ed attendere, cioè la funzione $g(e_q)$ deve essere mantenersi piccola fino a che non venga raggiunto un intorno valido dell'obiettivo corrente. Infatti, quando gli errori sono grandi nel *path following,* ci aspettiamo che il punto proiettato sul percorso desiderato vari lentamente, ovvero che la sua dinamica tenda a zero.

3.3 – Il problema di navigazione.

Abbiamo visto dunque che il generico problema di movimento può essere interpretato sotto due ottiche distinte, ovvero separato nel problema di inseguimento di traiettoria, oppure nel problema di inseguimento di un percorso. Innanzitutto concentriamo l'attenzione sul primo dei due problemi, apparentemente più complesso. Supponiamo a tal proposito di avere prodotto esternamente una traiettoria di riferimento (x_r, y_r, θ_r) che soddisfa i vincoli, ovvero per la quale ad ogni istante di tempo vale:

$$\begin{cases} \dot{x_r} = v_r \cos\theta_r \\ \dot{y_r} = v_r \sin\theta_r \\ \dot{\theta_r} = \omega_r \end{cases}$$

Lungo questa traiettoria la velocità lineare desiderata $v_r = \sqrt{\dot{x_r}^2 + \dot{y_r}^2}$ e quella angolare ω_r siano inoltre tali che risulti:

$$0 \leq \sup_{t \geq 0} v_r(t) < v_{max} \qquad\qquad \sup_{t \geq 0} |\omega_r| < \omega_{max}$$

ovvero, che siano rispettati i limiti dettati dalla saturazione degli ingressi di controllo $v(t)$ ed $\omega(t)$. Supponiamo, inoltre, di avere libero accesso allo stato del sistema, cioè di conoscere lungo tutta la traiettoria percorsa dal robot il vettore (x, y, θ); in questa situazione avremo evidentemente tracking perfetto se risulta ad ogni istante di tempo $x = x_r \ , \ y = y_r \ , \ \theta = \theta_r$.

A questo punto, per poter procedere con la soluzione del problema di controllo, risulta vantaggioso definire l'errore di posizionamento del robot (intendendo con tale termine sia la posizione effettiva sia l'orientazione) in un sistema di riferimento locale al robot stesso, ovvero in moto con esso. In particolare, ricordando che nella descrizione originale (x, y) sono le coordinate cartesiane del centro del robot e θ è l'angolo tra la direzione di viaggio, lungo la quale è diretta la velocità del robot, e l'asse x, possiamo costruire facilmente la trasformazione di coordinate che produce le variabili desiderate:

$$\begin{bmatrix} x_e \\ y_e \\ \theta_e \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x - x_r \\ y - y_r \\ \theta - \theta_r \end{bmatrix}$$

Nelle nuove variabili, che descrivono l'evoluzione dell'errore, otteniamo un sistema dinamico del tutto equivalente a quello di partenza descritto mediante le equazioni seguenti:

$$\begin{cases} \dot{x_e} = \ \omega\, y_e - v + v_r \cos\theta_e \\ \dot{y_e} = -\omega\, x_e + v_r \sin\theta_e \\ \dot{\theta_e} = \omega_r - \omega \end{cases}$$

Il vero vantaggio, sfruttando la nuova formulazione, risulta evidente notando che l'inseguimento perfetto della traiettoria nel problema di partenza corrisponde ora ad

avere $x_e = 0,\ y_e = 0,\ \theta_e = 0$. Ciò significa, in altre parole, che il problema originale di inseguimento della traiettoria può essere studiato come problema di stabilizzazione dell'origine del nuovo sistema descritto, semplificando notevolmente le cose.

A partire da questo momento si riportano alcune delle soluzioni più interessanti, presenti in letteratura, per la scrittura di un controllore non lineare in grado di consentire al robot uniciclo di inseguire una traiettoria prefissata.

3.4 - Saturation feedback controller.

Tenendo presente quanto detto finora, una prima soluzione di controllo che possiamo prendere in considerazione è quella denominata dagli autori *"saturation feedback controller"* e presentata in [1]. La legge proposta è basata su di un'azione di controllo tempo variante sviluppata facendo particolare attenzione alle limitazioni sulle velocità di rotazione e di spostamento del robot modellato.

In particolare il problema di inseguimento con saturazione, indicato dalla sigla *TPSC*, è definito a partire dal modello dell'uniciclo, descritto sopra, come la ricerca delle leggi di controllo per gli ingressi v ed ω che consentano al robot di inseguire correttamente una traiettoria di riferimento (x_r, y_r, θ_r) sotto l'ipotesi $v < v_{max}$ ed $\omega < \omega_{max}$. In altre parole l'obiettivo, comune a tutte le leggi che vedremo nel seguito, è far si che risultino valide le relazioni:

$$\lim_{t\to\infty} |x(t) - x_r(t)| = 0 \qquad \lim_{t\to\infty} |y(t) - y_r(t)| = 0 \qquad \lim_{t\to\infty} |\theta(t) - \theta_r(t)| = 0$$

Innanzitutto, una volta definita la traiettoria desiderata e riscritto il modello dell'uniciclo secondo il modello ad errore descritto in precedenza, applichiamo un ulteriore cambio di coordinate:

$$\begin{bmatrix} x_0 \\ x_1 \\ x_2 \end{bmatrix} = \begin{bmatrix} \theta_e \\ y_e \\ -x_e \end{bmatrix} \qquad\qquad \begin{bmatrix} u_o \\ u_1 \end{bmatrix} = \begin{bmatrix} \omega_r - \omega \\ v - v_r \cos x_0 \end{bmatrix}$$

rielaborando le equazioni otteniamo facilmente un sistema dinamico, ancora una volta del tutto equivalente a quello originale, che può essere riassunto come:

$$\begin{cases} \dot{x_0} = \ u_0 \\ \dot{x_1} = (\omega_r - u_0)\, x_2 + v_r \sin x_0 \\ \dot{x_2} = -(\omega_r - u_0)\, x_1 + u_1 \end{cases}$$

A partire da questo è facile vedere che, grazie al passaggio per le variabili di errore, il problema di inseguimento di partenza è diventato nel nuovo sistema, caratterizzato dal vettore di stato (x_0, x_1, x_2), un semplice problema di stabilizzazione di un punto di equilibrio. Dal momento che tutte le trasformazioni effettuate risultano invertibili, e che

l'origine del sistema coincide nel problema originale esattamente con l'obiettivo desiderato $x = x_r$, $y = y_r$, $\theta = \theta_r$, il *TPSC* è risolto a patto di ottenere la convergenza a zero delle nuove variabili; tale situazione diviene allora il nuovo obiettivo di controllo. Ancora si può notare che tale formulazione comprende, di fatto, due casi di interesse poiché si riduce ad un problema di regolazione, se $v_r(t) = 0$ ed $\omega_r(t) = 0$, oppure di tracking se $\lim_{t\to\infty} v_r^2(t) + \omega_r^2(t) \neq 0$. Mentre per un percorso lineare oppure circolare vale la seconda delle ipotesi, esistono casi intermedi, come il citato problema del parcheggio, in cui entrambe le problematiche devono poter essere risolte contemporaneamente dal medesimo controllore. A tal proposito si nota che la chiave per studiare la stabilità nel primo caso risiede nell'utilizzo di una variabile di ingresso u_0 persistentemente eccitante ($\lim_{t\to\infty} u_o(t) \to \infty$), mentre nel secondo caso è legata alla persistente eccitazione di $\omega_r(t)$ oppure di $v_r(t)$. Una possibile soluzione ad entrambi i problemi potrebbe dunque essere cercata nella costruzione di un ingresso opportuno tale che il segnale $u_0 - \omega_r$ risulti in qualche modo persistentemente eccitante.

Per poter tenere in conto opportunamente dei vincoli sulle variabili di controllo, gli autori si appoggiano sull'utilizzo della funzione di saturazione; prima di procedere è dunque bene darne una definizione univoca. Preso un $\delta > 0$, in particolare, essa è definita come segue:

$$sat_\delta(x) = \begin{cases} x & \forall\, |x| \leq \delta \\ \delta\, sgn(x) & \forall\, |x| > \delta \end{cases}$$

3.4.1 – La legge di controllo TPSC.

Le leggi di controllo proposte sono le seguenti:

$$u_1 = -sat_a(k_o x_2) \qquad k_0 > 0$$

$$u_0 = -\frac{\beta(x_0, x_1, x_2, t)}{\alpha(x_1, x_2, t)} - sat_b(k_1\, \bar{x}_0)$$

Il valore di saturazione a è scelto in modo che risulti

$$0 < a < v_{max} - \sup_{t\geq 0} |v_r(t)|$$

Tale scelta, infatti, permette di avere soddisfatto il vincolo di saturazione sulla velocità lineare poiché ne deriva di conseguenza la relazione

$$|v| \leq |u_1| + |v_r \cos x_0| < v_{max}$$

Nell'espressione di u_0 sono presenti le due funzioni α, β e la variabile ausiliaria $\bar{x}_0$ definiti in seguito. In particolare per $\bar{x}_0$ abbiamo l'espressione

$$\bar{x}_0 = x_0 + \frac{\varepsilon h(t) x_1}{1 + \sqrt{V_1}}$$

dove i parametri ε e γ devono soddisfare le relazioni

$$0 < \gamma < 1 \qquad\qquad 0 < \varepsilon < \frac{1}{1+\gamma}$$

mentre le funzioni $h(t)$ e $V_1(x_1, x_2)$ sono definite come segue:

$$h(t) = 1 + \gamma \cos\mu t > 0$$

$$V_1(x_1, x_2) = x_1^2 + x_2^2$$

Inoltre la scelta del parametro μ dipende strettamente dalla tipologia di traiettoria che il robot deve seguire; in particolare esso viene preso pari ad uno se per la traiettoria considerata risulta soddisfatta una delle condizioni seguenti:

$$\int_0^\infty v_r(t)dt = \infty \qquad\qquad \lim_{t\to\infty} \omega_r(t) = 0 \qquad\qquad \lim_{t\to\infty} \inf|\omega_r(t)| > 0$$

oppure viene imposto semplicemente $\mu = 0$ in caso contrario. In realtà è facile verificare che almeno uno dei tre vincoli sussiste sempre per le traiettorie di interesse pratico (ad esempio la prima condizione è soddisfatta per una traiettoria lineare o circolare, la seconda è soddisfatta nel problema di parcheggio parallelo, etc ...) per cui possiamo supporre genericamente che valga $\mu = 1$.

Le funzioni $\alpha(\cdot)$ e $\beta(\cdot)$ sono invece definite come segue:

$$\alpha = 1 - \frac{\varepsilon\, h(t) x_2}{1 + \sqrt{V_1}}$$

$$\frac{\beta}{\varepsilon} = \frac{\dot{h}(t)\, x_1 + h(t)\, \omega_r\, x_2 + h\, v_r\, \sin x_0}{1 + \sqrt{V_1}} - \frac{h(t)\, x_1}{\left(1 + \sqrt{V_1}\right)^2 \sqrt{V_1}} (x_1\, v_r\, \sin x_0 - sat_a(k_0\, x_2)\, x_2)$$

Si può notare, prima di tutto, che il primo termine soddisfa la relazione

$$0 < 1 - \varepsilon(1+\gamma) \le \alpha(x_1, x_2, t) < 2$$

e pertanto risulta limitato. Inoltre si può notare che:

$$\lim_{\varepsilon\to 0} \frac{\beta(x_0, x_1, x_2, t)}{\alpha(x_1, x_2, t)} = 0$$

Pertanto, per continuità, è sempre possibile determinare un valore di ε sufficientemente piccolo ed un opportuno valore di b tale che siano garantite:

$$\sup_{t\geq 0} u_0 < \omega_{max} - \sup_{t}|\omega_r| \qquad \rightarrow \qquad |\omega(t)| \leq |u_o| + |\omega_r| < \omega_{max}$$

Ciò significa che la legge di controllo scelta permette di rispettare le limitazioni ai valori degli ingressi imposta dalla saturazione degli attuatori.

3.4.2 – Note sulla legge di controllo scelta.

In questo paragrafo verrà fornita una giustificazione della legge di controllo prescelta; per maggiori dettagli si veda [3].

Si noti, prima di tutto, che la variabile ausiliaria $\bar{x}_0$, in base alla definizione data

$$\bar{x}_0 = x_0 + \frac{\varepsilon h(t) x_1}{1 + \sqrt{V_1}}$$

evolve secondo la seguente equazione dinamica

$$\dot{\bar{x}}_0 = \alpha(x_1, x_2, t)u_0 + \beta(x_0, x_1, x_2, t) \qquad \textit{(ii)}$$

Inoltre si noti che la convergenza a zero degli stati $(\bar{x}_0, x_1, x_2)$ implica la convergenza a zero delle variabili (x_0, x_1, x_2) che, a sua volta, implica la convergenza delle variabili $x(t), y(t)$ e $\theta(t)$ a quelle di riferimento, rispettivamente $x_r(t), y_r(t)$ e $\theta_r(t)$.

La scelta di u_0 è stata quindi compiuta al solo fine di rendere convergente a zero lo stato $\bar{x}_0$, si veda la sua equazione dinamica. Se infatti si sostituisce in questa l'espressione di u_0 si ottiene:

$$\dot{\bar{x}}_0 = -\alpha(x_1, x_2, t) sat_b(k_1 \bar{x}_0)$$

Dato che $\alpha(x_1, x_2, t)$ è positivo e limitato è facile provare che la legge prescelta garantisce che $\bar{x}_0(t) \rightarrow 0$ per $t \rightarrow \infty$.

Per quanto riguarda la legge di controllo per la variabile di ingresso u_1, si considera la funzione quadratica e definita positiva V_1 definita in precedenza. La sua derivata risulta

$$\dot{V}_1 = 2(x_2 u_1 + x_1 v_r \sin(x_0))$$

Sostituendo nell'ultima espressione la legge di controllo si ottiene che

$$\dot{V}_1 = -2x_2 sat_a(k_0 x_2) + 2x_1 v_r \sin(x_0)$$

Ricordando inoltre la forma dell'espressione di $\bar{x}_0$ si ottiene che

$$\dot{V}_1 = -2x_2 sat_a(k_0 x_2) + 2x_1 v_r \sin\left(\bar{x}_0 - \frac{\varepsilon h(t) x_1}{1 + \sqrt{V_1}}\right)$$

che, considerando che $\bar{x}_0 \to 0$ dallo studio effettuato in precedenza sulla sua equazione dinamica, garantisce che $\dot{V}_1$ sia, almeno per t sufficientemente alto, definita negativa. Questo garantisce la convergenza a zero delle variabili x_1 ed x_2.

Si possono notare alcuni dettagli riguardanti il ragionamento seguito nella definizione delle due leggi di controllo, che sono di particolare interesse; innanzitutto la funzione $h(t)$ che compare nella definizione della variabile $\bar{x}_0$ non è univoca e dipende dalle proprietà del segnale di riferimento $\omega_r(t)$. Inoltre, ricordando il significato della funzione $V_1(t)$, costruita sull'errore di posizionamento del robot, è immediato verificare che il movimento verso l'obiettivo desiderato è strettamente connesso ad avere $V_1(t) \to 0$; in tale situazione, se gli errori iniziali sono nulli, ovvero il robot si trova inizialmente sulla traiettoria da inseguire, allora anche tutti gli errori di inseguimento sono zero nel tempo, e dunque si ha *tracking* perfetto del percorso desiderato. Se infine i valori di soglia delle funzioni di saturazione a e b tendono all'infinito, l'intero problema si riconduce semplicemente all'inseguimento della traiettoria in assenza di saturazioni dei motori.

3.4.3 – Implementazione della legge di controllo TPSC.

Tenendo presente i risultati raggiunti finora, ovvero la struttura delle due leggi di controllo per gli ingressi a disposizione, possiamo applicare praticamente l'approccio *TPSC* al controllo del robot. Prima di procedere, tuttavia, è necessario definire opportunamente i valori di soglia delle saturazioni utilizzando i vincoli noti sulle velocità delle due ruote, sinistra e destra, indicate rispettivamente come v_L e v_R (derivate a loro volta dalle velocità di rotazione ω_L e ω_R). In particolare, ricordando le relazioni cinematiche definite inizialmente per ricondurre il robot al modello uniciclo presente in letteratura, si può scrivere $v_R = v + \omega\, E/2$ e $v_L = v - \omega\, E/2$ e dunque in generale si avrà la seguente disuguaglianza:

$$\sup_{t\geq 0} \max(v_L, v_R) \;\leq\; \sup_{t\geq 0} v + \frac{E}{2} \sup_{t\geq 0} \omega \;\leq\; v_{max} + \frac{E}{2}\, \omega_{max}$$

da cui possiamo direttamente ricavare una relazione per la scelta di v_{max} ed ω_{max} ed infine per a e b. La scelta che ne deriva è tuttavia in generale maggiormente conservativa rispetto ai risultati che si potrebbero ottenere tarando opportunamente i parametri in simulazione.

Per implementare il controllo, una volta eseguite le verifiche appena descritte, è necessario innanzitutto creare un modulo che, a partire dal valore corrente delle variabili di stato del robot (x, y, θ) e dal valore delle grandezze di riferimento (x_r, y_r, θ_r), passi attraverso la definizione delle variabili di errore fino alle nuove variabili (x_0, x_1, x_2). Tali segnali sono poi quelli che vanno ad alimentare la legge di controllo

assieme alle grandezze v_r ed ω_r, prodotte esternamente da un modulo di pianificazione della traiettoria, per calcolare i due ingressi u_0 ed u_1. Infine il valore degli ingressi ricavato dal controllore deve essere riconvertito al caso generale della coppia velocità lineare e di rotazione del modello ad uniciclo, v ed ω, e per ultimo al caso in questione di velocità delle due ruote, v_L e v_R. Ricordando che la trasformazione di variabili effettuata è sempre invertibile, il tutto si ottiene direttamente attraverso le equazioni seguenti:

$$\begin{bmatrix} v \\ \omega \end{bmatrix} = \begin{bmatrix} u_1 + v_r \cos x_0 \\ \omega_r - u_0 \end{bmatrix} \qquad v_R = v + \frac{E}{2}\omega \qquad v_L = v + \frac{E}{2}\omega$$

L'attuazione vera e propria dei comandi alle due ruote avviene poi, in modo trasparente, appoggiandosi ad un anello di controllo locale per l'applicazione ai motori dei valori di corrente e tensione necessari ad ottenere e garantire le velocità effettive desiderate.

Riassumendo, il sistema in anello chiuso ottenuto applicando in sequenza i vari blocchi avrà l'aspetto dello schema seguente:

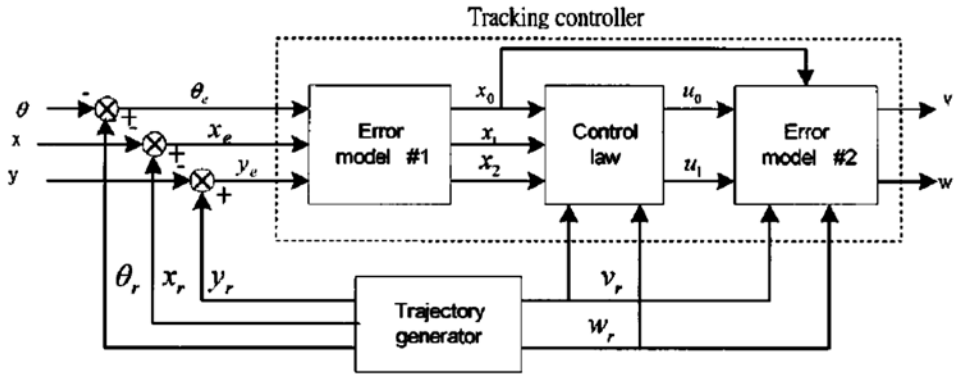

Fig. 3.4.3.1– *Schema del controllo saturation feedback.*

Per tarare opportunamente l'algoritmo, una volta definiti i valori di saturazione dai limiti fisici del sistema reale, si hanno a disposizione i parametri $(k_0, k_1, \varepsilon, \gamma)$ la cui entità viene scelta sulla base delle prestazioni richieste all'inseguimento della traiettoria, ed in generale del tipo di traiettoria richiesta. In generale comunque si rileva che grandi valori di k_0 e di ε corrispondono alla richiesta di una convergenza veloce, mentre i due parametri k_1 e γ influiscono di fatto in maniera minore sul risultato; il tutto deve sicuramente essere ottimizzato mediante prove specifiche.

Un'ultima nota sull'approccio seguito, e su quello che in realtà verrà adottato anche nel seguito della trattazione, va fatta a proposito del ricorso ad un modello puramente cinematico del robot mobile. L'approssimazione che si ottiene trascurando tutti gli aspetti dinamici, infatti, risulta generalmente accettabile se ci si limita a considerare basse velocità di movimento lungo la traiettoria, come all'atto pratico avviene in gran parte dei problemi di controllo di questo tipo. Del resto, tuttavia, la presenza di smorzamento e di un'inerzia non trascurabile del sistema meccanico dovrebbero

contribuire solamente a rendere più regolare l'azione di controllo portando a risultati addirittura migliori di quelli ottenibili in simulazione.

3.4.4 – Simulazione e prove sperimentali.

Sulla base di quanto descritto finora si è poi proceduto all'implementazione vera e propria della legge di controllo, alla sua simulazione ed infine al test sul sistema reale.

In particolare il primo passo è stato quello di calcolare, a partire dai valori limite dati alla velocità di rotazione delle due ruote di $1000\ steps/s$, il valore di v_{max} ed ω_{max} e di conseguenza, secondo l'espressione vista, il valore da assegnare alle soglie a e b della funzione di saturazione. Per quanto riguarda gli altri parametri è stata effettuata semplicemente una taratura mediante prove ripetute, tenendo presenti solamente le linee guida di cui si è discusso nei paragrafi precedenti; si è notato che al variare del percorso geometrico scelto anche tali parametri hanno bisogno di essere rimessi a punto per ottenere buone prestazioni.

La prima prova effettuata è quella di inseguimento di una traiettoria rettilinea, percorsa a velocità costante, prodotta da un modulo di pianificazione; la posizione iniziale del robot è diversa da quella del riferimento iniziale. In tal caso otteniamo dalla simulazione effettuata sul modello del sistema reale visto nel Capitolo 2 il comportamento di Figura 3.4.4.1.

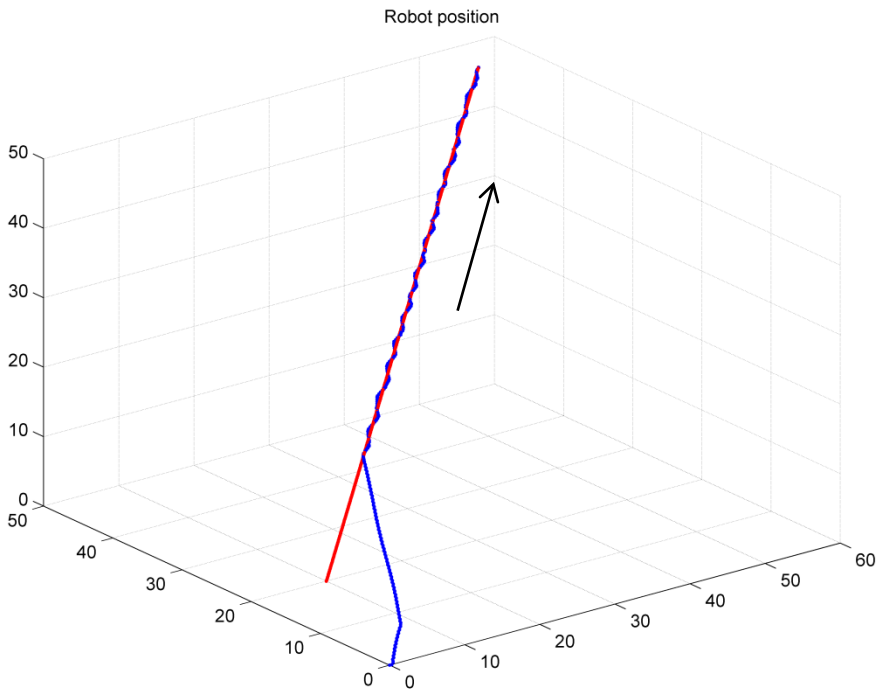

Fig. 3.4.4.1*– Simulazione di una traiettoria rettilinea – TPSC.*

Il risultato ottenuto è soddisfacente ed il modello insegue correttamente il riferimento imposto mantenendo il valore delle velocità alle ruote, ovvero degli ingressi di controllo effettivi, al di sotto del valore limite dettato dalla saturazione. Quello che si può notare dal grafico è una leggera oscillazione nell'intorno del percorso ideale che non si è riusciti ad eliminare tramite taratura dei parametri; del resto lo scopo di questa sezione è solamente quello di verificare la validità delle leggi presentate e sceglierne una adatta a proseguire con lo studio su *MPC.*

Utilizzando gli stessi parametri ed il setup sperimentale descritto nel Capitolo 2, ovvero il *software* per il calcolo della posizione attraverso la *webcam* e per l'invio dei comandi tramite canale *Bluetooth*, si è poi utilizzata la legge di controllo per guidare il robot lungo la traiettoria prescelta. In parallelo è stato fatta andare la simulazione a partire dalle medesime condizioni iniziali. Il risultato è quello che si vede in Figura 3.4.4.2 in cui abbiamo in verde il robot reale ed in blu il modello simulato.

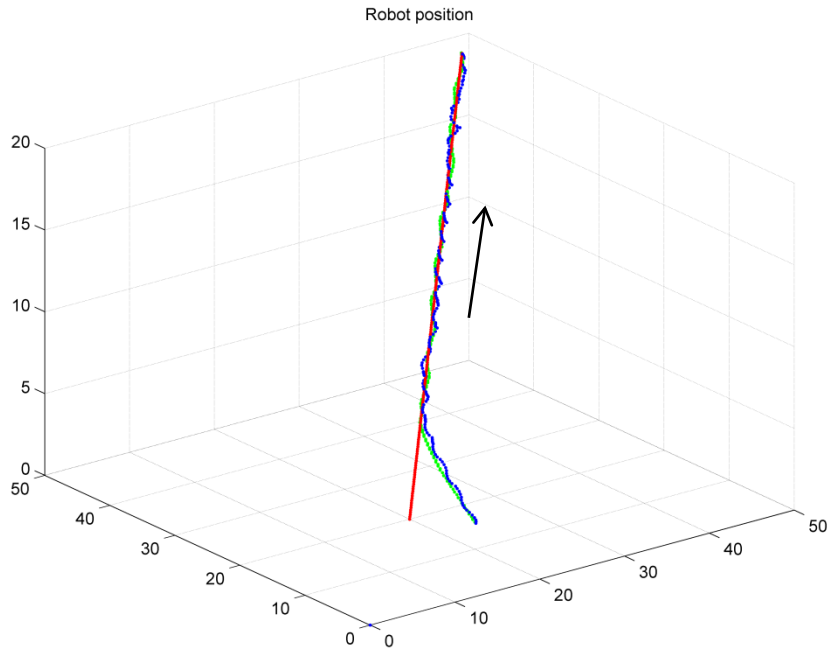

Fig. 3.4.4.2*– Simulazione e test di una traiettoria rettilinea – TPSC.*

Quello che si può notare è che anche il robot reale si comporta bene nel completamento del *task* ed anzi le oscillazioni che si vedono nel comportamento simulato si riducono per effetto della dinamica trascurata nella modellazione. La legge di controllo presentata è dunque a tutti gli effetti validata anche nel caso in questione; tuttavia, come vedremo meglio nel seguito, sarà accantonata a favore di una diversa soluzione.

3.5 - Backstepping controller.

Il secondo approccio, tratto dalla letteratura, che si è scelto di riportare in questo riepilogo è descritto in [4]. L'idea per risolvere il problema di controllo del sistema ad uniciclo è, in questo caso, basata sull'applicazione diretta del metodo di Lyapunov con l'obiettivo di ottenere nell'inseguimento risultati di stabilità semi-globale o globale della traiettoria, almeno per determinati percorsi dalle caratteristiche ben definite. Tale tecnica riconduce di fatto all'utilizzo di controllori di tipo *"backstepping"*. Il indica un processo di realizzazione della legge di controllo ottenuto in modo ricorsivo mediante la scomposizione del problema in sottoproblemi per ognuno dei quali si specifica una legge stabilizzante basata su quelle sottostanti.

Esattamente come fatto nel caso precedente, viene innanzitutto definito l'errore di inseguimento (x_e, y_e, θ_e) facendo ricorso ad un sistema di riferimento locale; a tal proposito riportiamo brevemente la trasformazione utilizzata e le equazioni dinamiche già viste:

$$\begin{bmatrix} x_e \\ y_e \\ \theta_e \end{bmatrix} = \begin{bmatrix} \cos\theta & \sin\theta & 0 \\ -\sin\theta & \cos\theta & 0 \\ 0 & 0 & 1 \end{bmatrix} \begin{bmatrix} x - x_r \\ y - y_r \\ \theta - \theta_r \end{bmatrix} \qquad \rightarrow \qquad \begin{cases} \dot{x}_e = \ \omega\, y_e - v + v_r\, \cos\theta_e \\ \dot{y}_e = -\omega\, x_e + v_r\, \sin\theta_e \\ \dot{\theta}_e = \omega_r - \omega \end{cases}$$

Il vantaggio è ancora una volta quello di aver trasformato il problema di inseguimento della traiettoria nelle variabili originali in un più semplice problema di stabilizzazione dell'origine, intesa come punto di equilibrio, attraverso dei passaggi del tutto invertibili.

Nella trattazione viene poi presentato un set di funzioni limitate, continue e con derivata limitata da utilizzare nelle leggi di controllo; in particolare si definisce, fissato un valore $0 < \varepsilon < \pi$, un set di funzioni $\mathcal{L}_\varepsilon^\infty$ tale che:

$$\mathcal{L}_\varepsilon^\infty = \{\, \varphi : R \rightarrow (-\pi + \varepsilon, \pi - \varepsilon) : \varphi \in C^\infty\,, \varphi(0) = 0\,, \varphi(z)z < 0\ \forall z \neq 0\ \ e\ \ \varphi' limitato\ \}$$

3.5.1 – La legge di controllo backstepping.

Le leggi di controllo effettive tratte da [4] per essere testate sono in realtà due.

La prima di queste è una legge la cui validità è di tipo locale, con un significato che sarà specificato più in dettaglio nel paragrafo seguente, ed è riassunta dalle equazioni dei due ingressi di controllo:

$$v = v_r\, cos\theta_e + c_1\, x_e$$

$$\omega = (1 + \varphi'(y_e v_r)\, x_e v_r)^{-1}(\gamma\, \eta(t)\, y_e\, v_r + \ \omega_r + \ \varphi'(y_e v_r)(v_r^2\, sin\theta_e + y_e\, \dot{v}_r) + c_2\, \gamma\, \bar{\theta}_e)$$

in cui la variabile ausiliaria $\bar{\theta}_e$ è descritta a partire dalla variabile di errore θ_e come:

$$\bar{\theta}_e = \theta_e + \varphi(y_e v_r)$$

con $\varphi(\cdot)$ ad indicare una delle funzioni del set $\mathcal{L}_\varepsilon^\infty$ introdotto nel paragrafo precedente.

Inoltre v_r ed ω_r indicano rispettivamente la velocità lineare ed angolare del riferimento che si sposta lungo la traiettoria e, allo stesso modo, $\dot{v}_r$ è la sua accelerazione lineare. Le costanti c_1, c_2 e γ sono opportuni parametri da cui dipenderanno le prestazioni effettive dell'algoritmo nell'inseguimento ed infine la $\eta(t)$ può essere calcolata esplicitamente a partire dall'espressione

$$\eta(t) = \bar{\theta}_e \int_0^1 \cos\ (-\varphi(y_e v_r) + s\ \bar{\theta}_e)\ ds$$

La seconda delle leggi testate è derivata dalla prima per avere un carattere globale; la distinzione più precisa sarà fatta nel seguito della trattazione. Tale legge è descritta dalla equazioni:

$$v = v_1 - y_e \omega + c_4\ \bar{x}_e$$

$$\omega = \omega_r + \gamma\ y_e\ v_r \int_0^1 \cos(s\ \theta_e)\ ds + c_5\ \gamma\ \theta_e$$

con il termine $v_1$che riassume per comodità l'espressione:

$$v_1 = \omega y_e + v_r \cos\theta_e - c_3 \dot{\omega}\ y_e + c_3(\omega\ x_e - v_r \sin\theta_e)$$

In queste $\bar{x}_e$ è costruito a partire dall'errore di posizionamento lungo l'asse x come

$$\bar{x}_e = x_e + c_3\ \omega\ y_e$$

Le costanti c_3, c_4 e c_5 sono di nuovo dei parametri di peso da cui dipendono le prestazioni dell'algoritmo nella fase di inseguimento. Infine v_r ed ω_r assumono lo stesso significato di velocità di riferimento già visto mentre $\dot{\omega}$ rappresenta l'accelerazione angolare attuale del robot.

3.5.2 – Note sulla legge di controllo a validità locale.

In questo paragrafo verrà fornita una giustificazione delle leggi di controllo prescelte; per dettagli ulteriori si veda direttamente [4].

A partire dal modello ad errore costruito si nota che presa una qualsiasi funzione φ all'interno del set $\mathcal{L}_\varepsilon^\infty$ si può osservare che, ponendo $x_e = 0$ e $\theta_e = -\varphi(y_e\ v_r)$, si

ottiene per la seconda variabile di stato l'equazione dinamica $\dot{y}_e = -v_r \sin(y_e v_r)$, per cui l'equilibrio $y_e = 0$ è uniformemente stabile.

Si definisce una nuova variabile:

$$\bar{\theta}_e = \theta_e + \varphi(y_e v_r)$$

il sistema viene di conseguenza trasformato e la sua ultima equazione diventa:

$$\dot{\bar{\theta}}_e = \omega_r - \omega + \varphi'(y_e v_r)\,(-\omega x_e v_r + v_r^2 \sin\theta_e + y_e \dot{v}_r)$$

Si consideri una funzione di Lyapunov quadratica che pesi i due errori nel posizionamento, definiti nel sistema originale, e la nuova variabile di stato; prendiamo con $\gamma > 0$:

$$V_1(t, x_e, y_e, \bar{\theta}_e) = \frac{1}{2}x_e^2 + \frac{1}{2}y_e^2 + \frac{1}{2\gamma}\,\bar{\theta}_e^2$$

Questa è definita positiva e radialmente illimitata; la sua derivata rispetto al tempo risulta:

$$\begin{aligned}\dot{V}_1 = {} & x_e(\omega y_e - v + v_r \cos\theta_e) + y_e(-\omega x_e + v_r \sin(-\varphi(y_e v_r) + \bar{\theta}_e)) \\ & + \frac{1}{\gamma}\,\bar{\theta}_e\big(\omega_r - \omega + \varphi'(y_e v_r)(-\omega x_e v_r + v_r^2 \sin\theta_e + y_e \dot{v}_r)\big)\end{aligned}$$

È possibile ricorrere ad una equivalenza, per la quale si rimanda ai dettagli presenti in [2], che consente di riscrivere uno dei termini come:

$$\begin{aligned}&\sin(-\varphi(y_e v_r) + \bar{\theta}_e) \\ &\qquad = \sin\big(-\varphi(y_e v_r)\big) \\ &\qquad + \bar{\theta}_e \int_0^1 \cos(-\varphi(y_e v_r) + s\bar{\theta}_e)\, ds = \sin\big(-\varphi(y_e v_r)\big) + \eta(t)\end{aligned}$$

si può allora riorganizzare la derivata della funzione V_1 come:

$$\begin{aligned}\dot{V}_1 = {} & x_e(-v + v_r \cos\theta_e) - y_e v_r \sin\varphi(y_e v_r) \\ & + \frac{1}{\gamma}\bar{\theta}_e\big(\gamma\eta y_e v_r + \omega_r - (1 + \varphi'(y_e v_r)x_e v_r)\omega \\ & + \varphi'(y_e v_r)(v_r^2 \sin\theta_e + y_e \dot{v}_r)\big)\end{aligned}$$

In questa formulazione compaiono entrambe le variabili di controllo, v ed ω, dunque è sufficiente scegliere due leggi opportune in grado di rendere la funzione V_1 decrescente e di conseguenza stabilizzare il sistema in anello chiuso. A tal proposito la scelta, per la quale si rimanda di nuovo a [2], ricade sulle funzioni già descritte.

Si nota che la legge di controllo introdotta per la variabile ω può risultare, a causa della sua particolare struttura, non definita in ogni istante di tempo t; tuttavia è possibile provare che per ogni condizione iniziale contenuta in un intorno sufficientemente

piccolo dell'origine tale situazione non è mai verificata. La soluzione trovata ha cioè un carattere locale, ovvero ritornando al problema originale, è valida a patto che il robot sia inizialmente non troppo distante dalla traiettoria desiderata.

Si verifica infatti che esiste effettivamente un intorno dell'origine $\Omega \subseteq \mathbb{R}^3$ tale per cui qualsiasi condizione iniziale in esso contenuta porti ad una legge di controllo per la variabile $\omega(t)$ ben definita, almeno in un intervallo di tempo $[0, T)$.

A tal proposito iniziamo con il definire, per ogni coppia di valori r_1 ed r_2 non negativi, un insieme $B(r_1, r_2)$:

$$B(r_1, r_2) = \{(x_e, y_e, \theta_e) \in \mathbb{R}^3 : r_1 r_2 |x_e| < 1\}$$

Si può verificare che vale la proprietà $B(0, r_2) = B(r_1, 0) = \mathbb{R}^3$. Fatto questo si definisce l'insieme Ω come un secondo set descritto da:

$$\Omega = \{(x_e, y_e, \theta_e) \in R^3 : V_1(t, x_e, y_e, \theta_e) < c^* \; \forall t \geq 0\}$$

in cui $c^* > 0$ è la più grande costante tale per cui risulta valida la relazione

$$\{(x_e, y_e, \theta_e) \in \mathbb{R}^3 : V_1(t, x_e, y_e, \theta_e) < c^* \; \forall t \geq 0\} \subset B(\|v_r\|_\infty , \|\varphi'\|_\infty)$$

Da tali definizioni deriva che, data la forma assunta da $\dot{V}_1$, se lo stato (x_e, y_e, θ_e) si trova nell'insieme Ω è destinato a rimanervi confinato, e di conseguenza l'ingresso progettato $\omega(t)$ è come desiderato ben posto. Oltretutto, ricordando che per costruzione V_1è una funzione non crescente lungo le soluzioni del sistema in anello chiuso, segue immediatamente che le traiettorie ottenute restano limitate nel tempo. Ciò permette di concludere direttamente che l'origine ottenuta è un equilibrio uniformemente stabile.

Si nota una cosa interessante: si osserva che è teoricamente possibile aumentare la regione Ω a patto di scegliere una funzione φ opportuna il cui gradiente φ' sia sufficientemente piccolo; in tal senso potremmo quindi dire che la soluzione trovata è semi-globalmente stabile.

Scegliendo delle costanti $c_1 > 0$ e $c_2 > 0$, la derivata della funzione V_1 diventa:

$$\dot{V}_1(t, x_e, y_e, \bar{\theta}_e) = -c_1 x_e^2 - y_e v_r \sin \varphi(y_e v_r) - c_2 \bar{\theta}_e^2$$

Osserviamo che la forma assunta in anello chiuso dalla derivata garantisce che i segnali x_e^2, $(y_e v_r \sin \varphi(y_e v_r))$ e $\overline{\theta_e^2}$ siano continui con la loro derivata; avendo inoltre derivate limitate si ottiene che, secondo un risultato noto con il nome di *Barbalat's lemma* e per il quale si rimanda alla fonte [2], i rispettivi segnali devono necessariamente convergere a zero al crescere del tempo. Ricordando allora la definizione della variabile $\bar{\theta}_e$, tutto ciò implica direttamente la convergenza di $\theta(t)$. Dato che $V_1(t, x_e, y_e(t), \theta_e(t))$ è decrescente e definita positiva per costruzione, quindi limitata inferiormente a zero, tenderà sicuramente ad un valore costante non negativo. Questo implica che anche il limite di $|y_e(t)|$ deve esistere ed essere come per gli altri segnali un numero reale finito

l_y. D'altra parte, se tale valore non fosse nullo, esisterebbe allora una sequenza di istanti di tempo crescenti $\{t_i\}_{i=1}^{\infty}$ con $t_i \to \infty$ tale per cui entrambi i limiti di $v_r(t_i)$ ed $|y_e(t_i)v_r(t_i)|$ risulterebbero diversi da zero; ma questo è impossibile poiché abbiamo provato che $|y_e v_r|$ tende a zero al crescere del tempo.

In sintesi il risultato trovato permette di dire che, data una traiettoria ammissibile con v_r, $\dot{v}_r$ ed ω_r limitati, esiste sempre una funzione φ contenuta nel set $\mathcal{L}_{\varepsilon}^{\infty}$ tale per cui il punto di equilibrio, descritto da $x_e = 0\,, y_e = 0,\ \theta_e = 0$, risulta in anello chiuso uniformemente stabile, sulla base delle scelte fatte per le leggi di controllo. A questo risultato si può inoltre aggiungere che, se $v_r(t)$ non converge a zero, allora per piccole condizioni iniziali la corrispondente soluzione (x_e, y_e, θ_e) converge a zero, cioè risulta:

$$\lim_{t\to\infty}\left(|x_e(t)| + |y_e(t)| + |\theta_e(t)| \right) = 0$$

e quindi facendo riferimento al problema originale, la traiettoria è inseguita correttamente.

3.5.3 – Note sulla legge di controllo a validità globale.

Le leggi viste possono essere generalizzate in modo da rendere la stabilità ottenuta di tipo globale. In particolare, per poter estendere il controllore al caso di *tracking* globale, si nota che le due scelte $x_e = c_3\omega\, y_e$ e $\theta_e = 0$ rappresentano delle funzioni stabilizzanti per l'espressione di y_e ricavata dal modello. Viene scelta allora una nuova variabile:

$$\bar{x}_e = x_e + c_3\omega y_e \qquad\qquad c_3 > 0$$

l'equazione dinamica di x_e risulta dunque sostituita dalla seguente:

$$\dot{\bar{x}}_e = \omega y_e - v + v_r\cos\theta_e - c_3\dot{\omega}y_e - c_3\omega(-\omega x_e + v_r\sin\theta_e)$$

Per semplicità di notazione si definisce inoltre l'espressione

$$v_1(t) = \omega y_e + v_r\cos\theta_e - c_3\dot{\omega}y_e + c_3(\omega x_e - v_r\sin\theta_e)$$

Anche in questo caso si sceglie poi una funzione di Lyapunov che pesi le tre variabili di stato, compresa quella appena definita:

$$V_2(t, \bar{x}_e, y_e, \theta_e) = \frac{1}{2}\bar{x}_e^2 + \frac{1}{2}y_e^2 + \frac{1}{2\gamma}\theta_e^2$$

scegliendo ancora una volta una costante $\gamma > 0$. Per la derivata possiamo scrivere quindi, utilizzando direttamente l'identità trigonometrica già vista:

$$\dot{V}_2 = -c_3\omega^2 y_e^2 + \bar{x}_e(-y_e\omega + v_1 - v) + \frac{1}{\gamma}\theta_e(\gamma y_e v_r\int_0^1 \cos(s\theta_e)\,\mathrm{ds} + \omega_\mathrm{r} - \omega)$$

La presenza di entrambe le variabili di controllo può essere sfruttata per rendere tale espressione equivalente alla seguente:

$$\dot{V}_2 = -c_3\omega^2 y_e^2 - c_4\bar{x}_e^2 - c_5\theta_e^2$$

ottenendo la forma vista nel paragrafo precedente per le leggi di controllo.

Questa volta, supponendo ancora che il riferimento sia caratterizzato da segnali v_r, $\dot{v}_r$, ω_r ed $\dot{\omega}_r$ limitati, avremo che tutte le traiettorie sono globalmente ed uniformemente limitate; inoltre se $v_r(t)$ non converge a zero, oppure $v_r(t)$ tende a zero ma con $\omega_r(t)$ che non converge a zero allora la soluzione in closed loop rispetta:

$$\lim_{t\to\infty}(|x_e(t)| + |y_e(t)| + |\theta_e(t)|) = 0$$

ovvero si ha un inseguimento accurato della traiettoria come desiderato. Di nuovo la prova di tale risultato si fonda su svariati risultati notevoli e pertanto è solo accennata; per una trattazione completa si rimanda alla fonte.

Notiamo innanzitutto che, essendo la funzione V_2 definita positiva e radialmente illimitata, allora anche le traiettorie originali $x_e(t), y_e(t), \theta_e(t)$ sono necessariamente uniformemente limitate e definite per ogni $t \geq 0$. Inoltre possiamo dire come fatto in precedenza che, data l'espressione di $\dot{V}_2$ i segnali $\omega(t)^2 y_e(t)^2$, $\bar{x}_e\,(t)^2$ e $\theta_e(t)^2$ si trovano in L_1, le loro derivate sono limitate, e pertanto tendono a zero al crescere del tempo. Allora, ricordando la definizione data alla variabile di stato $\bar{x}_e$, anche $x_e(t)$ tende necessariamente a zero. Resta dunque da provare solamente che $y_e(t)$ tende a zero sotto la prima delle ipotesi. Se si pone $\eta_1 = \int_0^1 \cos s\theta_e \, ds$ si osserva che questo tende ad uno al crescere del tempo poiché l'errore tende ad azzerarsi. Se $v_r(t)$ non si annulla, dal *closed loop* abbiamo l'espressione:

$$\dot{\theta_e} = -c_5\gamma\theta_e - \gamma y_e(t) v_r(t)\eta_1(t)$$

da cui, per la limitatezza del primo termine, deriva necessariamente che il secondo deve tendere a zero. Accanto all'ipotesi fatta allora, esattamente come avevamo visto nel caso locale, la $y_e(t)$ non può che convergere confermando il risultato.

3.5.4 – Implementazione della legge di controllo.

In entrambi i casi presentati il controllore che si vuole implementare dovrà contenere un blocco di conversione delle variabili di stato che consenta di passare nel sistema di coordinate ad errore ed un blocco che costruisca le variabili di controllo vere e proprie da inviare al robot. Di nuovo queste saranno espresse nella forma (v, ω) riferita al modello ad uniciclo ideale e dovranno da ultimo essere riconvertite nelle velocità di rotazione da applicare ai due motori ω_R ed ω_L. Il tutto dovrà essere accompagnato dalle informazioni provenienti da un pianificatore esterno della traiettoria, in grado di fornire istante per istante la posizione del riferimento, la sua velocità lineare ed angolare e, quando necessario, anche le accelerazioni relative.

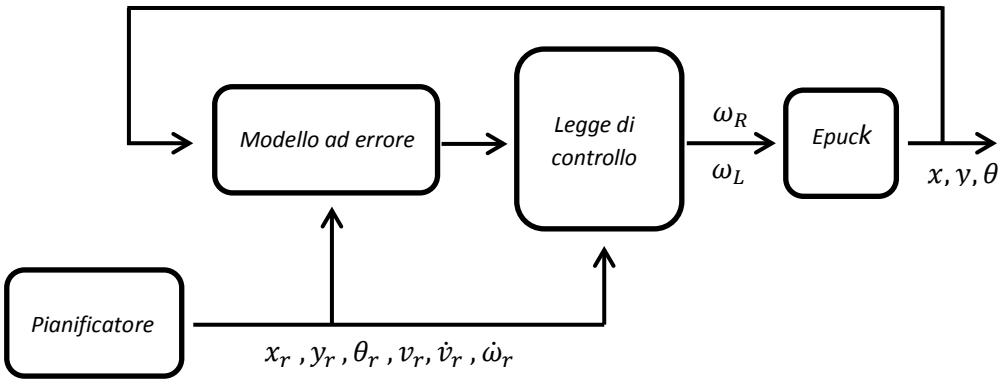

Fig. 3.5.4.1*– Schema del controllo*

Per quanto riguarda la scelta della $\varphi(\cdot)$ all'interno del set $\mathcal{L}_\varepsilon^\infty$, di cui si è discusso, si è scelto di testare il comportamento delle due funzioni proposte come esempio dagli autori, ovvero:

$$\varphi(z) = \frac{\sigma\, z}{1+z^2} \quad con \quad 0 < \sigma < 2(\pi - \varepsilon) \qquad \varphi(z) = \sigma_1 \operatorname{atan}(\sigma_2 z) \quad con \begin{cases} 0 < \sigma_1 < 2(\pi-\varepsilon)/\pi \\ \sigma_2 > 0 \end{cases}$$

calcolandone esplicitamente la derivata ove necessario. Allo stesso modo per gli integrali che compaiono nelle espressioni viste si sono calcolate le soluzioni in forma chiusa ed utilizzato direttamente il risultato nella costruzione delle leggi di controllo.

Questa volta, a differenza della soluzione precedente, non c'è modo di specificare a priori il valore di saturazione a cui saranno soggette le variabili di controllo reali per cui è stato necessario tarare i parametri mediante prove successive, garantendo il rispetto dei vincoli e nello stesso tempo buone prestazioni nel completamento dei *task* di inseguimento richiesti.

Per far questo si è fatto riferimento alle linee guida contenute in [4] ed al significato, riassunto nei paragrafi precedenti, dei vari termini utilizzati.

3.5.5 – Simulazione e prove sperimentali.

La prima delle leggi *backstepping* che si è scelto di implementare è quella con validità locale, con il significato discusso sopra; l'ipotesi è quella di scegliere una condizione di iniziale per il robot non eccessivamente distante dal primo elemento della sequenza di riferimenti.

Per uniformità con quanto fatto nel validare la legge *TPSC* si fa riferimento ad un pianificatore di traiettoria in grado di produrre un percorso rettilineo a velocità costante. Applicando la legge di controllo descritta in 3.5.2 al modello del sistema reale presentato nel Capitolo 2 si ottiene il risultato di Figura 3.5.5.1.

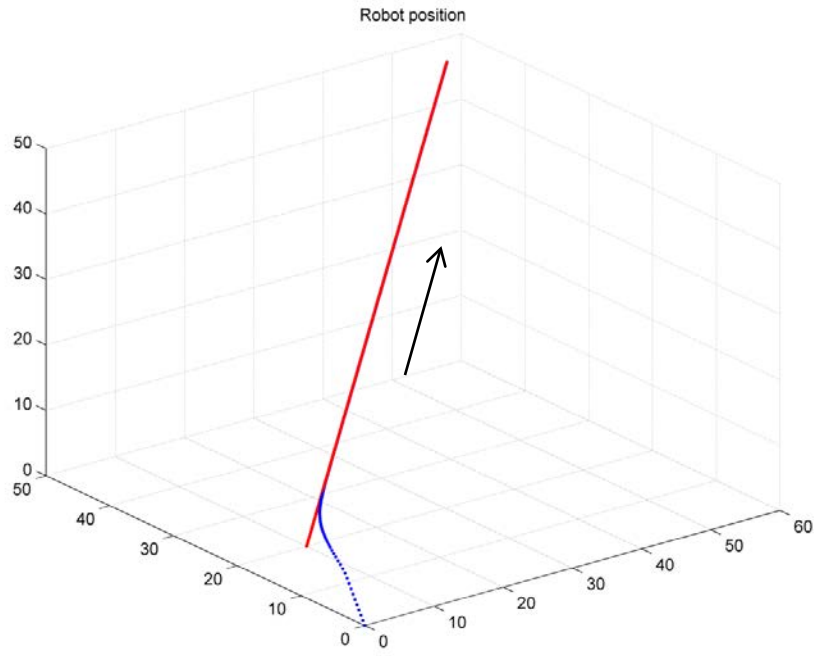

Fig. 3.5.5.1*– Simulazione di una traiettoria rettilinea – backstepping locale.*

Tale risultato si può considerare soddisfacente poiché la traiettoria è inseguita perfettamente con una fase breve nell'annullamento del disturbo iniziale. Questo disturbo deve per ipotesi essere limitato ad un set che, abbiamo visto, dipende dalla funzione $\varphi(\cdot)$ scelta; tuttavia nella serie di prove effettuate per validare la legge di controllo presentata non si è trovata all'interno dell'area di lavoro disponibile una

posizione di partenza che non soddisfacesse i requisiti e pertanto la soluzione risulta a tutti gli effetti valida per i nostri scopi. In realtà vedremo che anche questa sarà scartata a fronte di una legge più semplice ed altrettanto efficace.

Ad ulteriore conferma si è poi proceduto con il test della legge di controllo sul robot reale, ricorrendo alla stessa struttura di cui si è discusso nel paragrafo 3.4.4. Di nuovo si è scelto di far girare in parallelo una simulazione effettuata sul modello a partire dalle medesime condizioni iniziali.

Il risultato è quello che si può vedere in Figura 3.5.5.2 in cui abbiamo in verde il sistema reale ed in blu la simulazione.

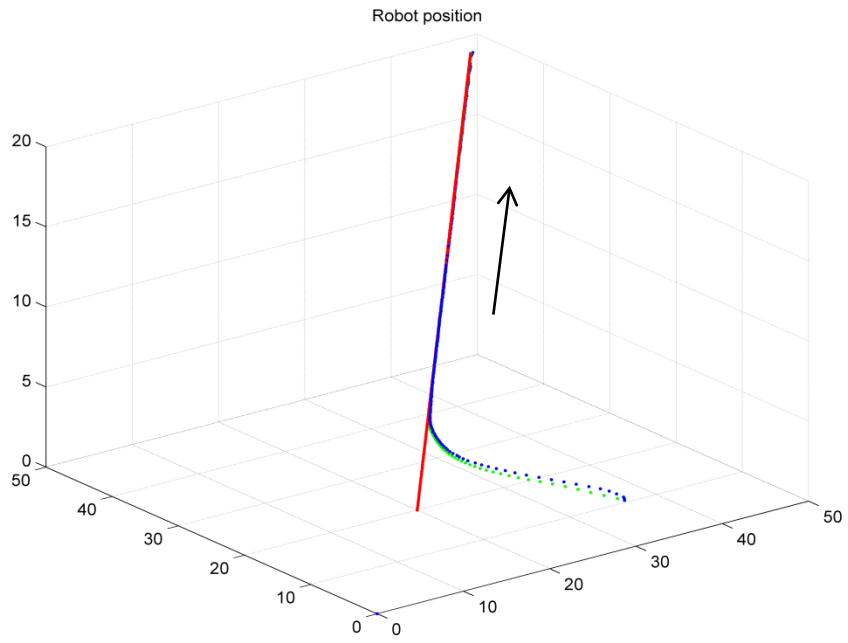

Fig. 3.5.5.2*– Simulazione e test di una traiettoria rettilinea – backstepping locale.*

Allo stesso modo si è scelto di verificare il funzionamento della legge di controllo *backstepping* con validità globale. Di nuovo la simulazione lungo una traiettoria rettilinea porta al risultato di Figura 3.5.5.3.

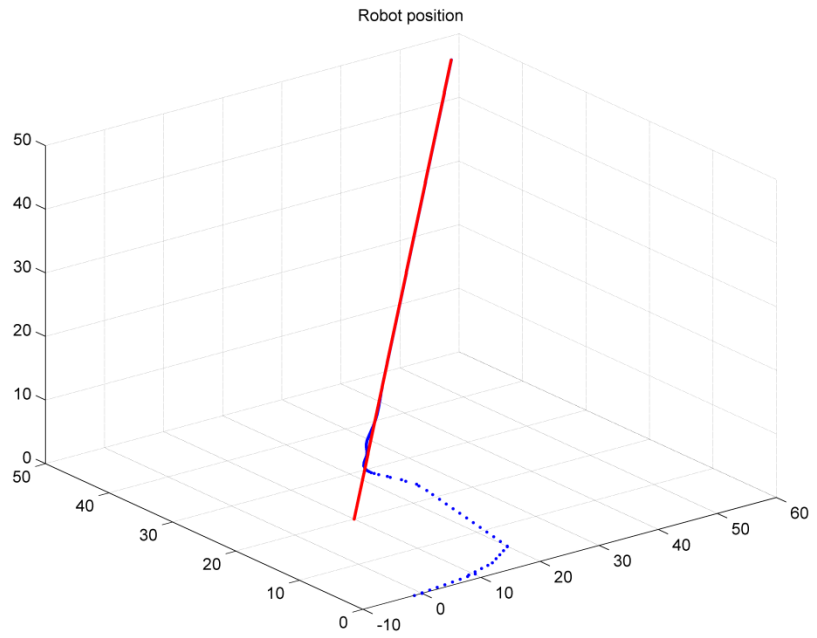

Fig. 3.5.5.3*– Simulazione di una traiettoria rettilinea – backstepping globale.*

Come per il caso precedente la legge consente di raggiungere il risultato desiderato e l'inseguimento della traiettoria prodotta dal pianificatore è ottenuto con prestazioni soddisfacenti.

Per concludere la stessa legge è stata testata sul sistema reale ed in parallelo simulata a partire dalla medesima condizione iniziale. Il risultato è quello di Figura 3.5.5.4 in cui abbiamo in verde il sistema reale ed in blu quello simulato.

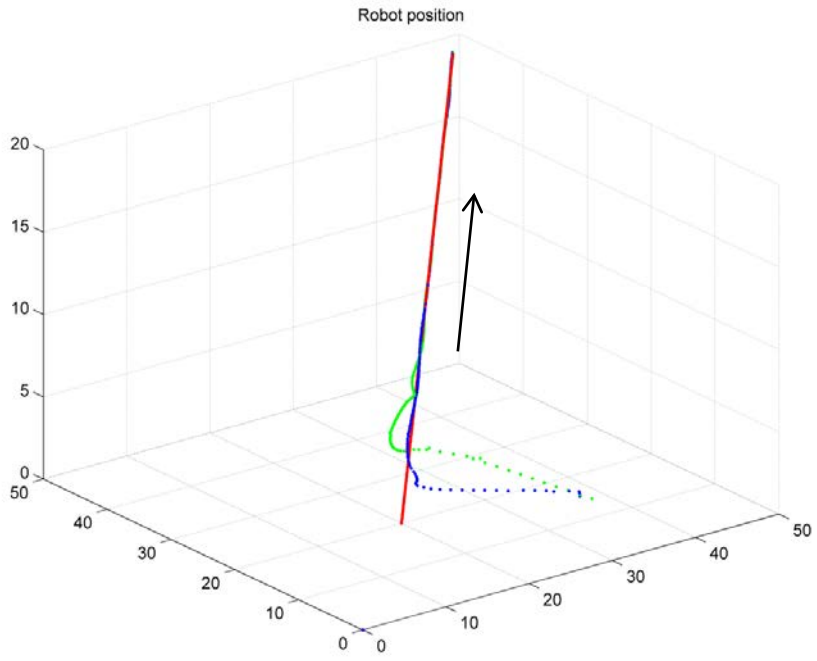

Fig. 3.5.5.4*– Simulazione e test di una traiettoria rettilinea – backstepping*

Sebbene anche questa tecnica consenta di risolvere il problema di inseguimento proposto il disallineamento tra la traiettoria simulata e quella reale dimostra che insorgono alcuni comportamenti fortemente non lineari che non rientrano nella modellazione. Tale risultato, visibile anche ad occhio nudo sull'apparato sperimentale, è legato al fatto che la legge considerata provoca un comportamento maggiormente aggressivo del robot che è soggetto a manovre più repentine durante il moto.

Per questo motivo questa seconda implementazione del controllore *backstepping* viene scartata a priori a favore della soluzione definita locale.

3.6 - Error adaptive controller.

La soluzione che si è scelto di riportare ora, presentata in [6] con il nome di *error adaptive controller*, è interessante poiché, al contrario di quelle viste finora, affronta il problema di navigazione anche dal punto di vista del *path following*, anzi permette di costruire una legge in grado di mediare tra l'inseguimento di un percorso puramente geometrico ed il rispetto di requisiti temporali ben precisi.

In generale ricordiamo brevemente quanto già detto ovvero che, supposto di avere un modulo di pianificazione esterno del percorso nel tempo, vi possono essere diverse soluzioni per correlare traiettoria e stato attuale del robot.

La prima di queste possibilità, discussa finora sotto il nome di approccio *trajectory tracking,* riguarda il caso in cui, per la soluzione del problema di inseguimento, sono previsti requisiti temporali ben determinati. Il riferimento da inseguire è allora un segnale tempo variante, o, in altre parole, una posizione che si sposta continuamente nel tempo lungo un percorso assegnato.

Al contrario è possibile decidere di trascurare l'aspetto temporale e supporre che in ogni istante il punto da inseguire sia uno qualsiasi di un percorso predefinito; tale soluzione è frequentemente definita con il termine *path following* ed è la più utilizzata nei casi applicativi.

Dopo la descrizione separata di queste due categorie, la trattazione fatta in [6] ha l'obiettivo di sviluppare un approccio che unisca i vantaggi di entrambe. In particolare si fa in modo che il moto del riferimento da inseguire lungo la traiettoria sia legato non solo all'istante di tempo attuale, ma anche all'errore nel posizionamento del robot, ovvero l'errore di inseguimento. Tale accorgimento fa sì che si possano inglobare nella legge di controllo i vantaggi derivanti da un approccio al problema puramente *path following.*

Data la natura non lineare del problema, ampiamente discussa in precedenza, la soluzione adottata è nuovamente basata sulla teoria di Lyapunov.

Definito il percorso geometrico da percorrere, si può descrivere il riferimento desiderato mediante un solo parametro r nelle coordinate dello spazio di stato. Il progredire dell'inseguimento può essere identificato con l'evoluzione del parametro r lungo il percorso predefinito. Questo, a sua volta, può coincidere semplicemente con il tempo, se stiamo affrontando un problema di *tracking*, oppure con l'ascissa curvilinea se stiamo risolvendo un problema di puro *path following*. Possiamo allora classificare il tipo di inseguimento del percorso in accordo al modo in cui imponiamo il progresso del parametro r.

3.6.1 – La legge di controllo error adaptive.

Abbiamo visto finora che si può mettere in gioco nella descrizione della traiettoria un parametro che agisca da peso, così da cercare di preservare parte delle necessità temporali, dato che il *path following* non ha nessuna garanzia di tipo deterministico, e nello stesso tempo essere in grado di metterle in sospeso quando l'errore nell'inseguimento risulta elevato; all'atto pratico tutto ciò porta ad ottenere una sorta di *trajectory tracking* rilassato. In realtà questa tematica presente in [6] viene messa in secondo piano e riportata solo per gli spunti di riflessione a cui conduce, che potranno essere utili in un secondo momento; la legge di controllo a cui si fa riferimento è dunque presa nel suo aspetto stabilizzante, di maggior interesse per la trattazione che stiamo facendo.

In particolare le leggi implementate saranno le seguenti:

$$v = e_v + v_r \cos e_\phi$$

$$\omega = e_\omega + \omega_r + \frac{1}{A_\phi^2}\, e_x\, v_r \sin e_\phi - \frac{1}{A_\phi^2}\, e_y v_r \cos e_\phi$$

In queste i due termini e_v ed e_ω rappresentano una sorta di termine correttivo in *feedback* costruito sulla base del vettore di errore e_q come

$$e_v = -\frac{1}{\tau_1} a_1(e_q) \qquad\qquad e_\omega = -\frac{1}{\tau_2}\frac{1}{A_\phi^2}\, a_2(e_q)$$

I parametri τ_1, τ_2 rappresentano delle costanti di tempo per la dinamica del termine correttivo e dunque rivestono un ruolo importante nelle prestazioni. Allo stesso modo importante per l'aspetto della traiettoria ottenuta è la costante A_ϕ^2 che dovrà variare a seconda della forma del percorso prescelto. Le due funzioni $a_1(e_q)$ ed $a_2(e_q)$ sono invece definite in accordo alle equazioni seguenti:

$$a_1(e_q) = e_x \cos e_\phi + e_y \sin e_\phi \qquad\qquad a_2(e_q) = A_\phi^2 \sin e_\phi$$

I termini v_r ed ω_r che compaiono nelle espressioni precedenti sono i riferimenti di velocità lineare ed angolare prodotti dal modulo di pianificazione esterno mentre e_x, e_y ed $e_\phi$sono gli errori di posizionamento ed orientamento opportunamente ruotati nel sistema di riferimento dell'obiettivo.

3.6.2 – Note sulla legge di controllo scelta.

La trattazione prende in considerazione il modello visto finora di veicolo mobile ad uniciclo, descritto nello spazio di stato dalle coordinate $q = (x, y, \phi)$, che si considerano direttamente accessibili. Si suppone ancora che siano $u = (v, \omega)$ le due variabili di controllo che rappresentano come già visto la velocità lineare del robot e la sua velocità angolare.

Prima di continuare si richiama brevemente anche il modello cinematico, sintetizzato mediante equazioni non lineari in q ma lineari nell'ingresso u come:

$$\dot{q} = B(q)\,u \qquad B = \begin{vmatrix} \cos\phi & 0 \\ \sin\phi & 0 \\ 0 & 1 \end{vmatrix} \qquad u = \begin{vmatrix} u \\ \omega \end{vmatrix} \qquad q = \begin{vmatrix} x \\ y \\ \phi \end{vmatrix}$$

Il riferimento può essere generalmente descritto in funzione del parametro r ed espresso come vettore $q_r(r) = \big(x_r(r), y_r(r), \phi_r(r)\big)$.

Dopo aver definito un sistema di coordinate connesse alla sequenza di riferimenti si pongono come (e_x, e_y) gli errori di posizionamento del robot, in coordinate cartesiane e relativamente a tale sistema di assi, ed e_ϕ l'errore di orientamento. Si scrivono inoltre come $u_r(r) = \big(v_r(r), \omega_r(r)\big)$ gli ingressi desiderati espressi anch'essi in funzione del parametro r.

E' evidente che applicando la legge di composizione all'ultima delle equazioni del modello otteniamo:

$$\omega_r(t) = \frac{d\phi_r}{dt} = \frac{d\phi_r}{dr}\frac{dr}{dt} = \phi_r'(r)\,\dot{r}$$

ed allo stesso modo una analoga relazione può essere scritta anche per v_r se consideriamo la lunghezza s_r del percorso, ottenendo $v_r(t) = \dot{s}_r$. In generale dunque a seguito della scelta fatta per esprimere la traiettoria si può definire il legame delle variabili di ingresso al tempo come $u_r(t) = u_r(r)\,\dot{r}$.

Si costruisce poi il vettore di errore nel sistema di riferimento locale prescelto, così come fatto nei paragrafi precedenti per le altre leggi di controllo:

$$e_q = R(\phi_r)(q - q_r) \qquad R(\phi_r) = \begin{vmatrix} \cos\phi_r & \sin\phi_r & 0 \\ -\sin\phi_r & \cos\phi_r & 0 \\ 0 & 0 & 1 \end{vmatrix}$$

Utilizzando le coordinate relative appena definite e ricordando la definizione delle grandezze desiderate lungo il percorso, si scrivono le equazioni dinamiche legate al sistema di errore

$$\begin{vmatrix} \dot{e}_x \\ \dot{e}_y \\ \dot{e}_\phi \end{vmatrix} = \begin{vmatrix} -v_r \\ 0 \\ -\omega_r \end{vmatrix} + \begin{vmatrix} 0 & \omega_r & 0 \\ -\omega_r & 0 & 0 \\ 0 & 0 & 0 \end{vmatrix} \begin{vmatrix} e_x \\ e_y \\ e_\phi \end{vmatrix} + \begin{vmatrix} \cos e_\phi & 0 \\ \sin e_\phi & 0 \\ 0 & 1 \end{vmatrix} \begin{vmatrix} v \\ \omega \end{vmatrix}$$

in cui si vede come intuitivamente l'errore aumenta all'avanzare dell'obiettivo.

Per il controllo del robot viene proposta, come nei casi precedenti, una legge costruita sulla base del teorema di *Lyapunov*; per i dettagli, trascurati nella trattazione corrente, si rimanda direttamente a [6]. In sintesi, considerando la struttura ad uniciclo del modello, la funzione adottata è la seguente:

$$V = \frac{1}{2}\left(e_x^2 + e_y^2\right)A_\phi^2\left(1 - \cos\left(e_\phi\right)\right)$$

con l'obiettivo di rendere il punto $e_q = \left(e_x, e_y, e_\phi\right) = 0$ un equilibrio asintoticamente stabile.

Ricordando che risulta $u \to u_r(t) = \dot{r}\, u_r(r)$ se $u_r \neq 0$ con la dinamica imposta $\dot{r} = g\left(e_q\right)$ vengono scelte come leggi di controllo quelle derivate dalle espressioni seguenti:

$$e_v = -\frac{1}{\tau_1}a_1\left(e_q\right) \quad \tau_1 > 0 \qquad e_\omega = -\frac{1}{\tau_2}\frac{1}{A_\phi^2}a_2\left(e_q\right) \quad \tau_2 > 0$$

$$a_1\left(e_q\right) = e_x\cos e_\phi + e_y \sin e_\phi \qquad a_2\left(e_q\right) = A_\phi^2 \sin e_\phi$$

da cui infine i due input desiderati risultano:

$$v = e_v + v_r(t)\cos\left(e_\phi\right)$$

$$\omega = e_\omega + \omega_r(t) + \frac{1}{A_{phi}^2}e_x v_r(t)\sin e_\phi - \frac{1}{A_\phi^2}e_y v_r(t)\cos e_\phi$$

Si nota che, i termini v ed ω, sono composti da un elemento in *feedforward* a cui si sovrappongono i termini di correzione e_v ed e_ω.

Il raggiungimento dell'asintotica stabilità dell'origine del sistema errore, che corrisponde per quanto visto all'obiettivo di inseguimento desiderato, può essere provato valutando direttamente la derivata della funzione di Lyapunov V lungo le equazioni di stato. Per i

dettagli e la dimostrazione del risultato si rimanda ancora una volta a [6]; si riporta solamente l'espressione della derivata della funzione di Lyapunov:

$$\dot{V} = v\left(e_y \sin e_\phi + e_x \cos\ e_\phi\right) + \omega\left(A_\phi^2 \sin e_\phi\right) + v_{des}(-e_x) + \omega_{des}\left(-A_\phi^2 * \sin e_\phi\right)$$

Da cui, sostituendo v ed ω, si ottiene l'espressione

$$\dot{V} = e_v a_1(e_q) + e_\omega a_2(e_q)$$

ed applicando la legge di controllo, infine, la forma definita negativa

$$\dot{V} = -\frac{1}{\tau_1} a_1(e_q)^2 - \frac{1}{\tau_2}\frac{1}{A_\phi^2} a_2(e_q)^2 \leq 0$$

3.6.3 – Soluzioni per il trajectory tracking e il path following.

Si considera ora il problema di scegliere una funzione $g(t, e_q)$ mediante la quale imporre la dinamica di evoluzione dell'obiettivo riassunta nella variabile r, cioè:

$$\dot{r} = g(t, e_q)$$

Si parte innanzitutto dal caso più semplice di puro path following, in cui cioè la funzione g in questione non dipende dal tempo ma solo dall'errore nel posizionamento. Si è visto che per errori piccoli la $g(e_q)$ deve tendere ad uno mentre deve assumere valori ridotti quando l'errore cresce; da tali condizioni una possibile funzione limitata tra zero ed uno viene proposta in [6] come

$$g(e_q) = \exp(-|e_q|)$$

In questo modo, quando l'errore cresce, g tende a zero come desiderato. Tale elementare funzione non garantisce le prestazioni desiderate per sistemi soggetti a vincoli di tipo anolonomo; per questo tipo di robot si può vedere infatti che se si prende una funzione di Lyapunov V, definita come la somma di funzioni relative agli errori e_q, allora la sua derivata rispetto al tempo non può essere definita negativa ma solamente semidefinita negativa. Questo aspetto è una conseguenza diretta del vincolo non olonomo; infatti se e_x ed e_ϕ sono entrambi nulli ad un certo istante di tempo, allora anche $\dot{e}_y$ deve essere nulla a causa della struttura del sistema; dunque e_y non può decrescere e pertanto nemmeno V, impedendo alla $\dot{V}$ di essere definita negativa. In altre parole se il robot è eccessivamente lontano dal *path* ed e_x, e_ϕ si annullano in un certo istante, allora la funzione $g(e_q)$ presa nella forma precedente tende a zero; di conseguenza in tale caso limite una legge di controllo *Lyapunov based* per il veicolo non

può essere trovata poiché ogni movimento del robot soddisfacente i vincoli reali anolonomi porta ad incrementare e_x oppure e_ϕ mentre e_y rimane costante.

In questa situazione si osserva comunque che quando $\dot{r}$ si riduce, la convergenza all'obiettivo diventa più lenta; questo aspetto in particolare è dovuto al fatto che l'input v definito non può mai superare il valore $v_r(t) = \dot{r}\, v_r(r)$, in modo da evitare un aumento di e_x.

Riassumendo il tutto infine, si può affermare che quando $\dot{V}$ si annulla il decremento della funzione V risulta legato all'incremento della variabile $\dot{r}$ e di conseguenza viene alterata la velocità di convergenza dell'algoritmo. Dunque, per ottenere prestazioni ottimali, se $\dot{V}$ è nullo allora $\dot{r}$ deve risultare il più possibile grande per avere una convergenza sufficientemente veloce.

Situazione ideale sarebbe quella di progettare la legge di controllo così da avere una funzione di *Lyapunov* la cui derivata $\dot{V}$ è il più possibile simile ad un termine $-KV$, con K maggiore di zero, in modo da avere una convergenza alla traiettoria desiderata di tipo esponenziale, sebbene a causa del vincolo tale obiettivo non possa essere raggiunto. Tenendo presente tutte le osservazioni fatte finora una funzione plausibile dell'errore risulta $g(e_q) = \exp(K_V \dot{V})$ e sarà quella testata.

Si prende infine in considerazione il problema, ampiamente presentato, di unire *path following* e *trajectory tracking*, ovvero di costruire la funzione dinamica $g(t, e_q)$, aggiungendo oltre all'errore nel posizionamento anche il contributo del parametro temporale. Nel complesso questo permette di ottenere un *tracking* deterministico rilassato, nel senso che il determinismo non è richiesto quando l'errore di posizionamento è ampio. Tale aspetto viene riportato solamente per completezza poiché si è scelto di non testarlo sul sistema reale, fermandoci ad analizzare la soluzione *path following*.

In sintesi, la dipendenza di g dall'errore può essere la stessa vista in precedenza poiché restano valide le medesime idee già discusse. L'intenzione è quella di introdurre la dipendenza dal tempo in modo da permettere che r resti a riposo quando il robot è distante dall'obiettivo; quando il robot recupera riavvicinandosi al *path* desiderato deve essere imposta una riduzione della differenza $(r - t)$. Per questi motivi la funzione progettata $g(t, e_q)$ proposta in [6] non può essere limitata ad uno poiché, nel secondo caso, deve consentire ad $\dot{r}$ di avvicinarsi al tempo t raggiungendolo. Inoltre nell'origine essendo $t - r = 0$ ed $e_q = 0$ necessariamente deve essere $g = 1$. La scelta proposta, per la quale non entreremo nel dettaglio, è la seguente:

$$g(t, e_q) = \exp(\dot{V})\,(1 + K_{tr} \operatorname{atan}(t - r))$$

in cui $K_{tr} > 0$ è un fattore di scala che indica quanto la convergenza di r a t risulta veloce. In tal caso $g(t, e_q)$è limitata superiormente da $1 + K_{tr}\frac{\pi}{2}$ ed inferiormente da $1 - K_{tr}\frac{\pi}{2}$.

Si può notare in questo modo che il secondo fattore di $g(t, e_q)$ è solo una legge di controllo per il parametro r. Allora l'intera funzione può essere interpretata come un nuovo grado di libertà del sistema dal momento che si sta progettando il comportamento di r come se fosse una ulteriore variabile di stato. In questa legge l'evoluzione desiderata per il parametro è $r = t$ quindi il termine dovuto al peso unitario è semplicemente un termine in *feedforward*. Dunque, quando la differenza $t - r$ diventa piccola, la componente $K_{tr}\text{atan}(t - r)$ agisce da legge di controllo proporzionale; in questa la funzione arcotangente è utilizzata semplicemente come funzione limitata.

3.6.4 – Implementazione della legge di controllo.

Per quanto riguarda la legge di controllo appena descritta si è scelto di trascurare l'approccio misto al *trajectory tracking* ed al *path following* e di implementare solamente la prima delle soluzioni, supponendo che l'inseguimento di cui vogliamo testare le prestazioni non abbia alcun requisito temporale deterministico. Tale scelta è motivata dall'obiettivo finale dello studio di questa soluzione, come delle altre proposte, che è volto solo alla ricerca di una legge opportuna per il controllo del movimento del robot, al quale sarà poi affiancato un controllore di tipo *MPC* per la gestione della tematica di pianificazione e raggiungimento dell'obiettivo.

In tal caso è necessario solamente calcolare l'errore di posizionamento in coordinate cartesiane, a partire dalla misura dello stato che si suppone disponibile grazie alla telecamera, ruotarlo in un sistema di riferimento locale all'obiettivo, ed utilizzarlo per costruire la legge di controllo vera e propria in grado di produrre gli ingressi (v, ω) del sistema ad uniciclo ideale; infine, come sempre, è necessario riconvertire questi nella velocità di rotazione delle due ruote da inviare al robot reale. Lo schema che ne deriva è quindi quello di Figura 3.6.4.1.

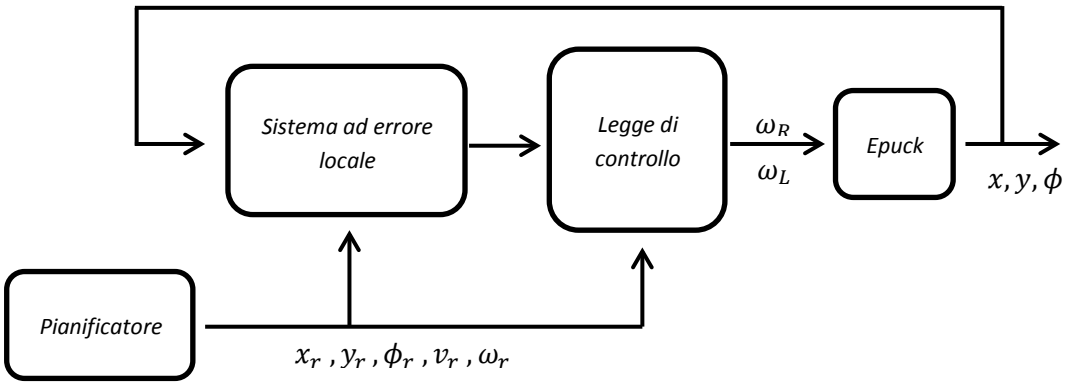

Fig. 3.6.4.1– *Schema del controllo error adaptive.*

I parametri che compaiono nella legge di controllo influiscono sulle prestazioni e sull'aspetto della traiettoria ottenuta, in particolare sulla sua curvatura e sulla rapidità delle manovre che il robot può compiere; non essendo esplicitamente presente il concetto di saturazione delle variabili di controllo, il tutto è lasciato alla taratura mediante prove pratiche.

3.6.5 – Simulazioni e prove sperimentali.

Come sottolineato nei paragrafi precedenti l'intera trattazione svolta a proposito della legge *error adaptive* è lasciata come panoramica importante sulle possibili interpretazioni del concetto di inseguimento. La legge effettivamente implementata è quella sviluppata per il *path following* e descritta in 3.6.2.

Innanzitutto, esattamente come per le altre soluzioni di controllo viste, si è scelto di testare la legge in simulazione con l'obiettivo di inseguire una traiettoria rettilinea. Il risultato è quello che si vede in Figura 3.6.5.1.

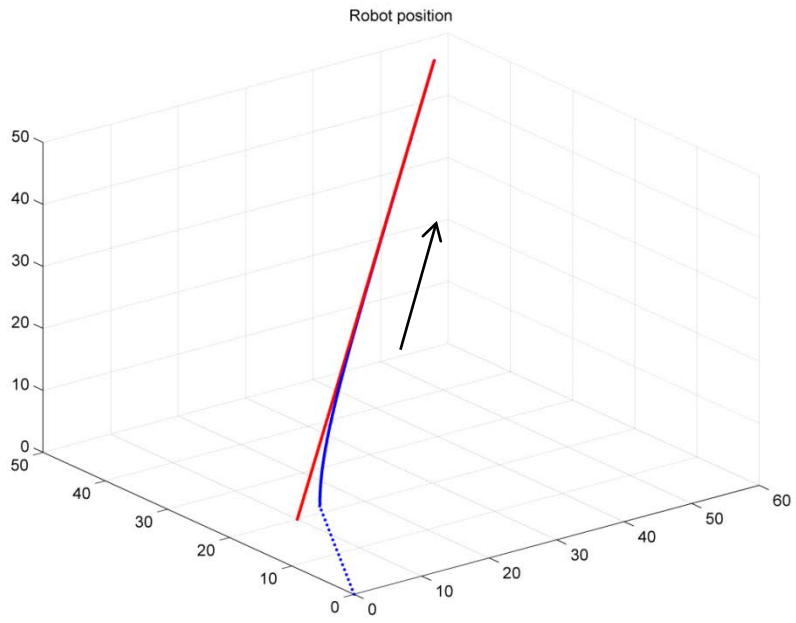

Fig. 3.6.5.1– *Simulazione di una traiettoria rettilinea – error adaptive*

Come atteso la legge di controllo consente di inseguire correttamente la traiettoria con brevi tempi di assestamento; in corrispondenza una opportuna taratura dei parametri fa si che gli ingressi di controllo necessari al completamento del compito non raggiungano mai i loro valori di saturazione.

Lo stesso algoritmo è stato poi valutato sul sistema reale sfruttando, come visto in tutti i casi precedenti, l'apparato sperimentale descritto nel Capitolo 2. Per validare il tutto a fianco del robot reale è stata lanciata anche una simulazione effettuata sul modello a partire dalla stessa posizione di partenza.

Il risultato che si ottiene è quello di Figura 3.6.5.2. In verde si vede la traiettoria tenuta dal robot mentre in blu quella del modello.

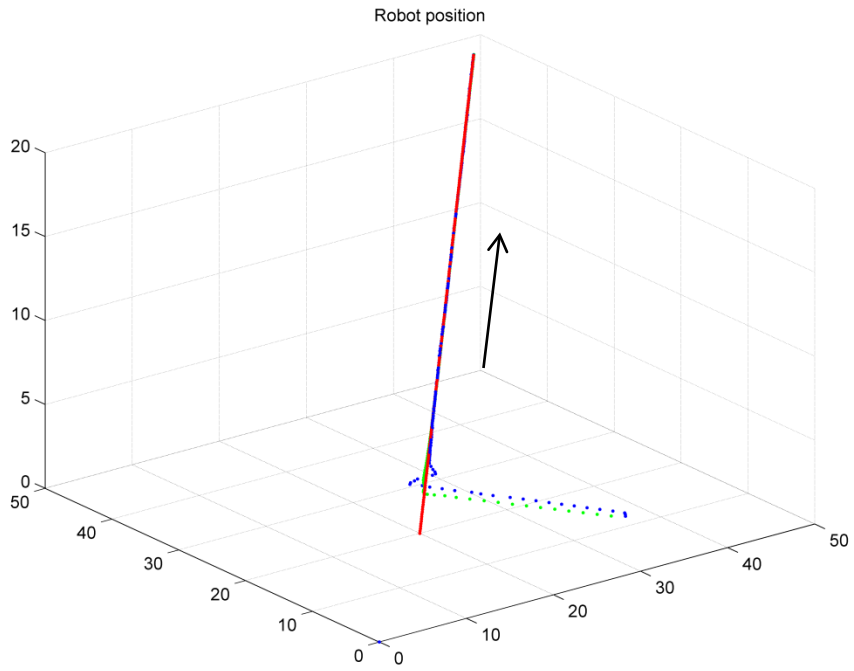

***Fig. 3.6.5.2**– Simulazione e test di una traiettoria rettilinea – error adaptive controller.*

Il risultato ottenuto permette quindi di validare la tecnica presentata nei paragrafi precedenti. Tuttavia anche questa sarà accantonata a favore di un metodo che vedremo a fine capitolo.

3.7 - Closed loop steering law.

Un'altra interessante soluzione che si è scelto di presentare è riportata in [5] e denominata *closed loop steering law*. Questa è sviluppata esplicitamente per il problema di parcheggio dell'uniciclo ed è risultata essere, in un primo momento, particolarmente adatta al caso in esame dal momento che non si parla di traiettoria e pertanto non si fanno ipotesi particolari su di essa.

Si considera un veicolo posizionato ad una distanza non nulla dall'obiettivo; ricordiamo che, nelle ipotesi fatte, il suo moto, governato dall'azione combinata di una velocità angolare ω ed una velocità lineare v sempre diretta lungo il suo asse di movimento, può essere modellato nello spazio cartesiano come:

$$\begin{cases} \dot{x} = v\,cos\phi \\ \dot{y} = v\,sin\phi \\ \dot{\phi} = \omega \end{cases}$$

In questa rappresentazione, a differenza di quanto fatto nei casi precedenti, si suppone che le coordinate (x, y, ϕ) utilizzate siano calcolate rispetto all'obiettivo.

Si ricorre poi alla rappresentazione di queste in coordinate polari definendo:

$$e = \sqrt{x^2 + y^2} \;\rightarrow\; \dot{e} \;=\; \frac{1}{2}\sqrt{(x^2 + y^2)}\,(2x\dot{x} + 2y\dot{y}) = \frac{x\dot{x} + y\dot{y}}{e}$$

sostituendo in tale espressione il valore delle dinamiche $\dot{x}$ ed $\dot{y}$ si ottiene:

$$\dot{e} = \frac{xv\cos\phi + yv\sin\phi}{e} = \frac{v}{e}(x\cos\phi + y\sin\phi) = \frac{v}{e}(e\cos\theta\cos\phi + e\sin\theta\sin\phi)$$
$$= -v\cos(\theta - \phi)$$

Graficamente abbiamo la situazione presentata in Figura 3.7.1.

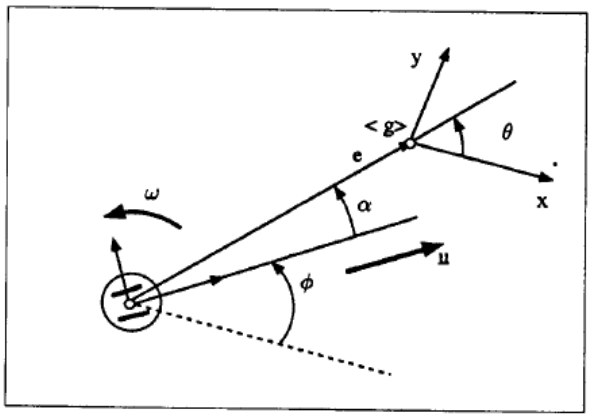

Fig. 3.7.1– *Rappresentazione in coordinate polari del sistema ad errore.*

Si definisce inoltre un nuovo angolo $\alpha = \theta - \phi$ come angolo tra l'asse principale del veicolo ed il vettore distanza dall'obiettivo, e, appena calcolato. Si nota che deve valere

$$\dot{\theta}\, e = v \sin\alpha \qquad\qquad \dot{\alpha} = \dot{\theta} - \dot{\phi}$$

pertanto complessivamente si può riscrivere il sistema nelle nuove coordinate come:

$$\begin{cases} \dot{e} = -v\, cos\alpha \\ \dot{\alpha} = -\omega + v\dfrac{sin\alpha}{e} \\ \dot{\theta} = v\, sin\alpha \end{cases}$$

Questa rappresentazione è del tutto equivalente al sistema di partenza e può essere ricavata semplicemente mediante trasformazioni geometriche opportune.

Essa inoltre rappresenta, in un certo senso, il sistema di visione associato a qualsiasi reale esperienza di guida. L'unica cosa che è bene notare è che, dal momento che tutte le equazioni scritte sono basate sull'uso di coordinate polari, esse sono valide solamente per valori non nulli assunti dall'errore di inseguimento e, altrimenti entrambi gli angoli θ ed α risultano indefiniti, perdendo la corrispondenza con il sistema di partenza.

Consideriamo il problema originale di determinare opportune strategie che consentano al robot di aggiungere l'obiettivo; in particolare determinare delle strategie di guida che portino asintoticamente il veicolo alla posizione ed orientamento desiderati. Ricordiamo che il teorema di Brockett, citato nel capitolo precedente, secondo il quale non è possibile trovare una legge di controllo in *feedback* continua e tempo invariante per il sistema in questione, vale solamente se si fa ricorso ad una rappresentazione in coordinate cartesiane, mentre la limitazione cade nel caso corrente di coordinate polari [5].

Per la soluzione del problema del parcheggio si suppone di poter misurare direttamente il vettore di stato (e, α, θ) in corrispondenza ad un qualsiasi valore della distanza $e > 0$, altrimenti, come già detto, si perde l'univocità nella rappresentazione dei due angoli. L'obiettivo è quello di trovare una legge di controllo dipendente dallo stato del tipo $(v, \omega) = g(e, \alpha, \theta)$ che garantisca il raggiungimento in tempo finito dell'origine, intesa, per le limitazioni sulla variabile e, come punto limite.

Come nei casi precedenti si fa ricorso al metodo di Lyapunov per il progetto vero e proprio di un controllore stabilizzante.

3.7.1 – La legge di controllo steering law.

Utilizzando la rappresentazione descritta nel paragrafo precedente si scelgono le leggi di controllo seguenti:

$v = \gamma\, cos\alpha\; e \qquad\qquad \gamma > 0$

$\omega = k\,\alpha + \gamma \dfrac{cos\alpha\; sin\alpha}{\alpha}(\alpha + h\theta) \qquad\qquad k, h > 0$

in queste le costanti γ, k ed h rappresentano i parametri da tarare per ottenere un buon inseguimento delle traiettorie mentre la coppia (e, θ) identifica l'errore di posizionamento scritto in coordinate polari ed infine α è, come abbiamo visto, l'angolo tra la direzione di moto e la congiungente tracciata dal robot all'obiettivo, ovvero una sorta di errore angolare di inseguimento.

3.7.2 – Note sulla legge di controllo scelta.

L'idea di base è quella di mappare il moto del robot su di una funzione scalare di cui garantire la decrescenza mediante la scelta accurata delle variabili di controllo. A tal proposito la scelta fatta in [5], che riportiamo senza dimostrazione, è la seguente:

$$V = V_1 + V_2 = \frac{1}{2}\lambda e^2 + \frac{1}{2}(\alpha^2 + h\theta^2) \qquad \lambda, h > 0$$

Tale funzione assume un significato pratico se si interpretano i due termini V_1 , V_2 come norme pesate dei due errori nel posizionamento, e, e nell'allineamento, (α, θ), del veicolo. La derivata rispetto al tempo di tale funzione, lungo le traiettorie definite dal modello del sistema, risulta:

$$\dot{V} = \dot{V}_1 + \dot{V}_2 = \; \lambda e\dot{e} + \; \alpha\dot{\alpha} + h\theta\dot{\theta} = -\lambda e v \cos\alpha + \alpha\left(-\omega + v\frac{\sin\alpha}{\alpha}\frac{(\alpha + h\theta)}{e}\right)$$

A partire da questo risultato si fa in modo di rendere non positivo il primo termine, corrispondente a $\dot{V}_1$, scegliendo per la velocità lineare v la forma continua riportata nel paragrafo precedente

$$v = \gamma \cos\alpha\, e \qquad \gamma > 0 \qquad\qquad \rightarrow \qquad\qquad \dot{V}_1 = -\lambda \cos^2\alpha\, e^2 \leq 0$$

In questo modo la prima componente della funzione V è sempre non crescente nel tempo e pertanto, dal momento che è limitata inferiormente dallo zero, converge asintoticamente ad un limite finito e non negativo. Inoltre, il fatto che la sua derivata sia semplicemente proporzionale al quadrato della variabile scalare positiva *e*, implica che *e* stessa deve essere monotona e non crescente nel tempo; il suo valore nullo non può

essere raggiunto in tempo finito e pertanto è assicurata di fatto la validità della rappresentazione polare per tutta la durata della manovra di parcheggio.

Tenendo presente la scelta precedente e sostituendo il valore progettato per l'ingresso v si può riscrivere il secondo termine della derivata come

$$\dot{V}_2 = \alpha\left(-\omega + \gamma\frac{\cos\alpha\,\sin\alpha}{\alpha}(\alpha + h\theta)\right)$$

Con lo stesso criterio è possibile trovare, anche per questo termine, una variabile di controllo per la velocità angolare ω, di forma continua, che consenta di rendere l'espressione non crescente. La scelta fatta in [5] ricade in particolare sulla legge

$$\omega = k\alpha + \gamma\frac{cos\alpha\ sin\alpha}{\alpha}(\alpha + h\theta) \quad k > 0 \qquad \rightarrow \qquad \dot{V}_2 = -k\alpha^2 \leq 0$$

che porta complessivamente alla non positività della funzione $\dot{V}$. La dimostrazione dettagliata del risultato viene tralasciata e nel seguito si riportano solo i tratti più importanti; una analisi completa si trova in [5].

Da quanto scritto deriva che, la funzione V scelta per descrivere il moto del sistema, non può crescere nel tempo e di conseguenza, essendo anche limitata inferiormente, deve convergere asintoticamente ad un limite finito e non negativo. Tale fatto, unito con la natura radialmente illimitata della forma quadratica stessa, garantisce la limitatezza della traiettoria dello stato in corrispondenza di qualsiasi condizione iniziale limitata; in più, a causa di tale limitatezza, segue direttamente la continuità uniforme nel tempo della $\dot{V}$. Infine, come conseguenza di tale continuità nel tempo, e a seguito dell'esistenza di un limite di convergenza per $\dot{V}_1$, ne deriva necessariamente, secondo il già citato *Barbalat's lemma*, per il quale si rimanda in dettaglio a [5], che la $\dot{V}$ deve convergere a zero al crescere del tempo.

Questo implica la convergenza della traiettoria dello stato in un sottoinsieme della linea $(e, \alpha, \theta) = (0, 0, \theta)$ per la quale risulta nulla tale derivata; non resta quindi che dimostrare che l'unico possibile punto di convergenza è dato dall'origine e che tale situazione accade in corrispondenza a derivate nulle per le variabili di stato.

Si prendono le equazioni in anello chiuso del sistema nelle coordinate equivalenti definite:

$$\begin{cases} \dot{e} = -\gamma\cos^2\alpha\, e \\ \dot{\alpha} = -k\alpha - \gamma h\dfrac{\cos\alpha\,\sin\alpha}{\alpha} \\ \dot{\theta} = \gamma\cos\alpha\,\sin\alpha \end{cases}$$

A causa della convergenza a zero di entrambe le variabili e ed α segue immediatamente dalla prima e dalla terza equazione che entrambe le derivate $\dot{e}$ e $\dot{\theta}$ convergono a zero; inoltre, a causa della limitatezza delle traiettorie di stato, la convergenza di $\dot{\theta}$ implica

anche che θ tende ad un limite finito $\bar{\theta}$ al crescere del tempo. Se dunque si prende la seconda equazione anche $\dot{\alpha}$ deve necessariamente tendere ad un limite finito e pari a $\gamma h\bar{\theta}$. A questo punto, tenendo conto ancora della convergenza a zero di α, e notando inoltre che $\dot{\alpha}$ è una funzione uniformemente continua, come implicato dalla limitatezza della traiettoria di stato, segue che anche $\dot{\alpha}$ converge a zero e pertanto, ricordando la sua espressione, anche $\bar{\theta}$ deve necessariamente essere nullo, ovvero θ deve tendere a zero, confermando la validità dello schema di controllo utilizzato.

3.7.3 – Implementazione della legge di controllo.

La legge di controllo in *feedback,* ottenuta con questo approccio, risulta una forma notevolmente semplice e può essere facilmente implementata anche su microcontrollori *onboard* come si vorrebbe fare nel caso dei robot epuck in questione. Inoltre i comportamenti ottenuti dal sistema in anello chiuso sono molto simili ai comportamenti naturali che ognuno di noi terrebbe nella guida di un veicolo, proprio per la particolare scelta delle variabili di stato.

Per l'implementazione effettiva è necessario in tal caso costruire un blocco che, a partire dalle variabili di errore calcolate mediante la misura dello stato e la posizione del riferimento da inseguire, riporti alle nuove variabili scritte in coordinate polari

$$e = \sqrt{(x - x_{rif})^2 + (y - y_{rif})^2} \qquad \theta = \operatorname{atan}\left(-\frac{y - y_{rif}}{x - x_{rif}}\right) \qquad \alpha = \theta - \left(\phi - \phi_{rif}\right)$$

L'unica accortezza riguarda il caso in cui il robot raggiunge il suo obiettivo, ovvero l'errore di posizionamento diventa nullo e pertanto la trasformazione in coordinate polari perde di univocità; sarà quindi necessario stabilire una determinata soglia ed interrompere l'azione di controllo quando tale condizione è verificata.

Nuovamente in uscita dalla legge di controllo presentata un opportuno blocco algebrico deve convertire le variabili di ingresso ideali (v, ω) nelle due velocità di rotazione da inviare alle ruote del robot (ω_R, ω_L).

Nel complesso allora il sistema proposto, compreso di un sistema esterno per la pianificazione del percorso, avrà l'aspetto seguente

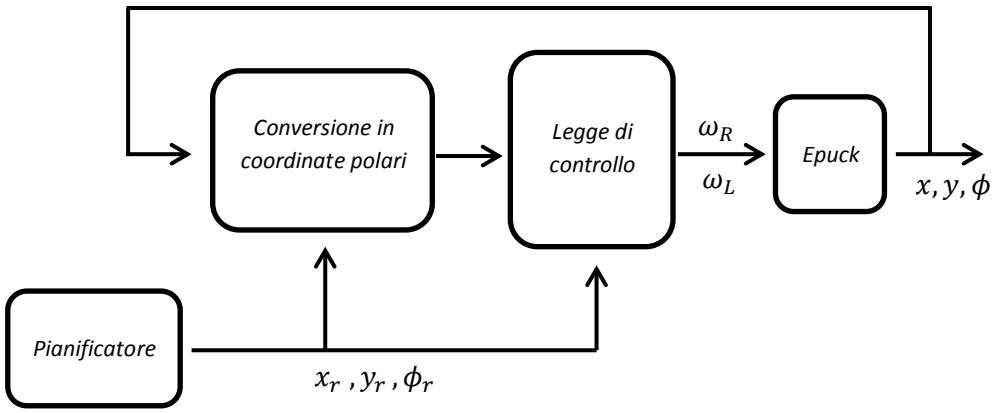

Fig. 3.7.3.1– *Schema del controllo steering law.*

Si noti che in tal caso non abbiamo bisogno di conoscere velocità ed accelerazioni del sistema di riferimento da inseguire, ma solo la sua posizione; tale aspetto rende, come abbiamo già detto, tale soluzione la più adatta al problema che si vuole affrontare.

I parametri h, γ, k che compaiono nella legge di controllo descritta nei paragrafi precedenti influiscono direttamente sulle prestazioni di inseguimento e sulla curvatura del percorso risultante dal movimento.

3.7.4 – Simulazioni e prove sperimentali.

La legge di controllo presentata nei paragrafi precedenti viene innanzitutto simulata facendo attenzione a scegliere dei parametri che consentano di mantenere limitate le variabili di controllo a disposizione e nello stesso tempo di ottenere buone prestazioni nell'esecuzione del compito. In particolare, esattamente come per tutti i casi visti finora, si è scelto di fare ricorso ad una traiettoria di riferimento rettilinea, con una posizione di partenza diversa da quella del sistema controllato. Il risultato della simulazione, ottenuta secondo il modello presentato nel Capitolo 2, è riportato in Figura 3.7.4.1.

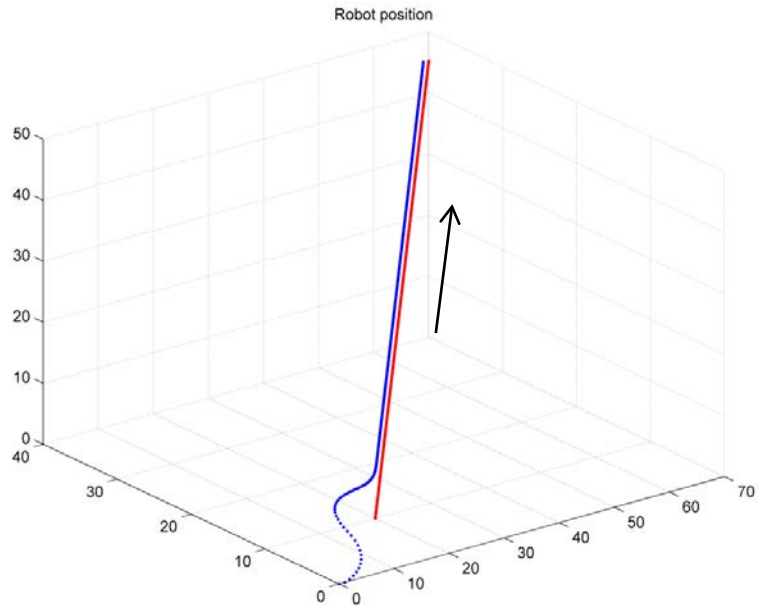

Fig. 3.7.4.1*– Simulazione di una traiettoria rettilinea – steering law.*

Apparentemente si nota un piccolo errore di regime nel posizionamento del sistema rispetto alla traiettoria; tale comportamento si ritiene essere dovuto alla scelta dei parametri il cui piccolo valore, scelto per impedire di raggiungere le saturazioni degli attuatori, impedisce al controllo di essere efficace quando la velocità è ridotta. A tal proposito non si sono fatte ulteriori tarature poiché, come già detto in precedenza, l'obiettivo della sezione corrente è quello di presentare e validare delle tecniche di controllo del moto presenti in letteratura al fine di costruire un panorama che sia in grado di motivare le scelte fatte nel seguito della trattazione. In ogni caso si può considerare soddisfacente il risultato ottenuto e dunque valida la legge di controllo.

Come ulteriore prova la stessa viene poi implementata sul robot reale e, come fatto in tutti gli altri casi, in parallelo viene lanciata la simulazione sul modello.

Il risultato ottenuto è quello di Figura 3.7.4.2. In verde si osserva il comportamento del sistema reale ed in blu il comportamento del sistema simulato.

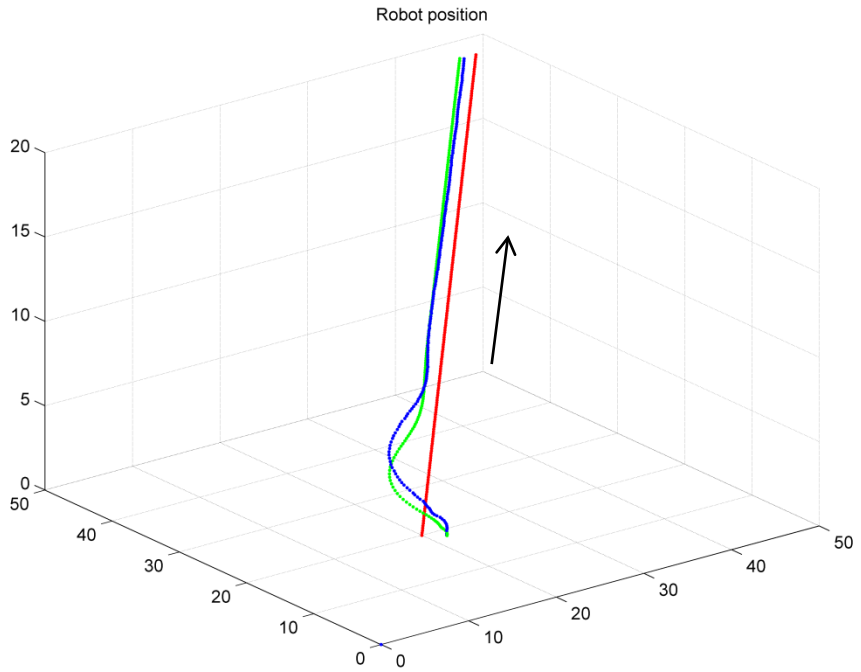

Fig. 3.7.4.2*– Simulazione e test di una traiettoria rettilinea – steering law.*

Come atteso otteniamo anche in questo caso un risultato soddisfacente, sempre tenendo presente quanto detto sull'errore di regime. Il piccolo disallineamento tra i due sistemi è dovuto semplicemente alla fase di partenza del robot che presenta una forte non linearità, che tuttavia non ha alcun effetto sul comportamento successivo durante l'inseguimento.

Tale legge di controllo è sembrata in un primo momento la più valida per la stabilizzazione del movimento del robot ad uniciclo in funzione dell'obiettivo successivo di realizzazione di un sistema *MPC*. L'applicazione dell'anello così realizzato al sistema vero porta ad ottenere una dinamica che è ancora non lineare, e dunque poco adatta ad una implementazione online di una tecnica predittiva. Si è inoltre visto che una sua linearizzazione dinamica, in accordo a quanto detto nell'introduzione del capitolo, non consente di controllare completamente il comportamento del sistema e pertanto deve essere scartata. Per questi motivi anche quest'ultima tecnica presentata dalla letteratura viene messa da parte a favore di quella che vedremo nel paragrafo 3.8.

3.7.5 – Approfondimenti ed interpretazione fisica.

Utilizzando la legge vista in precedenza, gli autori proseguono prendendo in considerazione il problema di navigazione. Dal momento che l'idea del nostro progetto è quella di affrontare tale particolare aspetto tramite *MPC*, si riporta, per completezza, solamente una traccia dei risultati per i quali si rimanda direttamente a [5].

In particolare, supponendo che il percorso sia parametrizzato nell'ascissa curvilinea s, si considera il sistema di riferimento relativo $p(s)$ in moto continuo, a partire da una posizione iniziale p_0, con una velocità $\dot{s}$ assegnata. Considerando il *frame* $p(s)$, come obiettivo locale del veicolo, è possibile verificare che le equazioni cinematiche originali si modificano, in funzione della variabile $\dot{s}$, come

$$\begin{cases} \dot{e} = -v\cos\alpha + \dot{s}\cos\theta \\ \dot{\alpha} = -\omega + v\dfrac{\sin\alpha}{\alpha} - \dot{s}\dfrac{\sin\theta}{e} \\ \dot{\theta} = v\dfrac{\sin\alpha}{\alpha} - \dot{s}\dfrac{\sin\theta}{e} - \dfrac{\dot{s}}{R(s)} \end{cases}$$

indicando con $R(s)$ il raggio di curvatura corrente della traiettoria, con segno positivo o negativo a seconda della posizione rispetto alla traiettoria stessa del *frame*.

In tale ottica tutti i termini aggiuntivi, dipendenti da $\dot{s}$, possono essere considerati come un disturbo dovuto al fatto che l'obiettivo è in movimento. Se a tale nuovo sistema si applicano le equazioni di controllo viste in precedenza si ottiene l'anello chiuso:

$$\begin{cases} \dot{e} = -\gamma\cos^2\alpha\, e + \dot{s}\cos\theta \\ \dot{\alpha} = -k\alpha + \gamma h\dfrac{\cos\alpha\sin\alpha}{\alpha}\theta - \dot{s}\dfrac{\sin\theta}{e} \\ \dot{\theta} = \gamma\sin\alpha\cos\alpha - \dot{s}\dfrac{\sin\theta}{e} - \dfrac{\dot{s}}{R} \end{cases}$$

La variabile $\dot{s}$ rappresenta ora una nuova azione di controllo che si ha a disposizione per il completamento del compito desiderato.

A questo scopo si riporta senza dimostrazione la scelta fatta, supponendo per semplicità che sia $h > 1$, per la velocità di movimento dell'obiettivo $\dot{s} \in (0, v)$

$$\dot{s} = \begin{cases} 0 & V = \lambda e^2 + (\alpha^2 + h\theta^2) > \varepsilon \\ f(e,\alpha,\theta) & V = \lambda e^2 + (\alpha^2 + h\theta^2) \leq \varepsilon \end{cases} \qquad 0 < \varepsilon < \frac{\pi^2}{4}$$

con $f(e,\alpha,\theta)$ funzione radialmente continua e centrata nel dominio ellissoidale $\lambda e^2 + (\alpha^2 + h\theta^2) = \varepsilon$, che raggiunge il valore massimo v_{max} solo in corrispondenza dell'origine ed un valore nullo agli estremi dell'ellisse stesso.

Il motivo di tale scelta è interessante e viene riportato. Innanzitutto, quando lo stato del veicolo è al di fuori del dominio ellissoidale, come ad esempio può essere all'inizio del movimento quando è necessario raggiungere il punto di partenza, l'obiettivo è

mantenuto in una posizione assoluta fissa specificata da $\dot{s} = 0$ così che il problema di controllo è quello statico già affrontato con le due leggi di controllo continue che guidano il sistema mantenendo $e > 0$.

A causa di questo comportamento lo stato del veicolo rientra in tempo finito sulla superficie dell'ellissoide lungo la quale la velocità di evoluzione della traiettoria è ancora nulla per la definizione della funzione $f(e, \alpha, \theta)$; nell'istante di tempo immediatamente successivo si ha il veicolo interno al dominio in questione con $e > 0$ a garantire che $\dot{s}$ avrà un valore diverso da zero. Sotto tali condizioni, un contributo positivo è aggiunto ad $\dot{e}$, come stabilito dalla prima delle equazioni del sistema. Dunque, durante il moto all'interno dell'ellisse, la traiettoria dello stato non può mai raggiungere il punto $e = 0$ in un tempo finito, garantendo in questo modo la continuità della legge definita anche all'interno dell'ellissoide stesso.

Inoltre la funzione di Lyapunov V all'interno dell'ellissoide, a causa della presenza del valore $\dot{s}$, non ha più la garanzia di avere una derivata semidefinita negativa, questo induce la possibilità che dall'interno di tale dominio in un certo intervallo di tempo la norma dello stato possa crescere in modo da consentire alla traiettoria di raggiungere nuovamente la superficie dell'ellissoide, ma questa volta dall'interno. Se tuttavia questo accade, in corrispondenza ad un certo istante di tempo, si può vedere di nuovo che, sulla base delle precedenti considerazioni, deve risultare $e > 0$. Di conseguenza l'applicazione dello stesso ragionamento porta a vedere che ci sarà ancora un istante di tempo successivo in cui lo stato si ritroverà nuovamente confinato all'interno dell'ellissoide; tutto ciò porta a concludere che una volta che la traiettoria del veicolo risulta catturata per la prima volta all'interno dell'ellissoide vi rimarrà confinata continuamente in tutti gli istanti futuri, mantenendo sempre un errore di posizionamento non nullo.

Dato che questo comportamento porta ad avere una velocità $\dot{s}$ nulla solo sul confine dell'ellissoide, e che inoltre questa è una quantità non negativa, si ha la garanzia che l'ascissa curvilinea s cresca in modo monotono nel tempo, così come il punto obiettivo si muoverà monotonicamente lungo tutto il percorso fino al suo completamento. Inoltre, durante la sua evoluzione lungo il percorso, il *frame* obiettivo $p(s)$ manterrà il veicolo sempre indietro con un errore sulla distanza ed un disallineamento le cui norme al quadrato sono sempre contenute nell'intervallo dettato da ε proprio per la scelta della funzione di Lyapunov V. Questa volta nel problema di *path following* un ruolo importante è giocato dal parametro λ che compare nella funzione V, poiché assegnando un valore positivo siamo in grado di determinare il limite superiore ε/λ per e, ovvero il massimo errore di posizionamento ammissibile nell'inseguimento del percorso assegnato.

Come già detto l'approccio seguito è interessante poiché rientra in un certo senso nell'esperienza quotidiana di guida. Infatti, in presenza di un buon allineamento con il percorso stradale, siamo naturalmente portati a puntare progressivamente su obiettivi più lontani lungo la strada stessa, supposto che essi rimangano sufficientemente allineati con la nostra auto; ciò ovviamente induce ad un aumento della velocità. Al

contrario, quando il disallineamento inizia a crescere, come accade ad esempio avvicinandosi ad una curva della strada, siamo portati a ridurre la nostra velocità e puntare gli occhi su obiettivi progressivamente più vicini, così da ridurre l'errore sulla distanza pur di mantenere il disallineamento in un *range* accettabile. Infine, quando tale disallineamento si riduce nuovamente alla fine della curva, possiamo di nuovo riprendere ad aumentare la velocità.

Da ultimo, in [5], viene riportata l'estensione ad un problema di navigazione attraverso dei punti di passaggio assegnati da un pianificatore esterno, tralasciando la soluzione di *path following*. Quello che si propone di fare è semplicemente applicare la tecnica descritta per l'inseguimento immaginando che i vari punti siano connessi uno all'altro mediante segmenti lineari, ovvero di fatto applicare la più semplice e computazionalmente meno dispendiosa forma di interpolazione.

Agendo in questo modo si può notare una differenza rispetto al caso continuo visto in precedenza; in particolare dal momento che si hanno dei salti finiti della variabile di stato θ in corrispondenza dei punti angolari sui punti di passaggio. Si può verificare, e per questo si rimanda a [5], che l'effetto introdotto dalla presenza di tali salti fa si solamente che la traiettoria seguita dallo stato sia generalmente discontinua nella variabile θ in corrispondenza delle transizioni dell'ascissa curvilinea s da un segmento al successivo. Quando l'angolo misurato tra due segmenti successivi è motlo ampio il salto corrispondente di θ può anche essere così ampio da portare istantaneamente lo stato al di fuori del dominio ellissoidale dove, una volta che la velocità $\dot{s}$ risulta istantaneamente azzerata, la legge di controllo tende a riportare indietro lo stato stesso sulla superficie ellissoidale e di conseguenza al suo interno dove rimarrà fino all'eventuale salto successivo. Ovviamente in corrispondenza di questi comportamenti si potrà notare una deviazione ampia dal segmento prefissato come percorso; per far fronte a questo viene allora proposto, in corrispondenza di ampi salti della variabile angolare θ, di ridurre il valore del parametro λ così da ridurre indirettamente la dimensione dell'ellissoide e confinare maggiormente il valore dell'errore e, facendo uscire lo stato dall'ellissoide prima ancora di variare il sistema di riferimento da inseguire, così da azzerare $\dot{s}$ e mantenere limitata l'escursione dalla traiettoria originale.

3.8 – Feedback linearization.

Tutte le leggi di controllo viste, da quelle *trajectory tracking* a quelle di tipo *path following*, consentono di risolvere il problema del parcheggio per un robot di tipo uniciclo. Tuttavia il sistema ottenuto in anello chiuso è ancora un sistema non lineare e pertanto difficile da gestire con un approccio di tipo *MPC*, come quello che si desidera implementare.

In particolare, ricorrendo all'ultima tecnica vista per la stabilizzazione dell'origine e linearizzando il modello in anello chiuso nell'intorno della posizione attuale del robot, oppure nell'intorno del riferimento desiderato, otteniamo, come già accennato, in entrambi i casi un sistema non completamente controllabile. Una soluzione alternativa consiste invece nell'applicare direttamente al robot una legge *feedback* linearizzante che semplifichi esternamente il problema della stabilizzazione. Ricordando ancora una volta il teorema di Brockett [2] per sistemi come quello considerato, si sottolinea come il meccanismo di retroazione desiderato debba necessariamente essere di tipo dinamico e non semplicemente di tipo algebrico.

A tal proposito, a partire dal modello matematico generale del sistema di cui si è discusso nel Capitolo 2

$$\begin{cases} \dot{x} = v\,cos\phi \\ \dot{y} = v\,sin\phi \\ \dot{\phi} = \omega \end{cases}$$

è possibile allargare lo stato imponendo una dinamica alla velocità v, ovvero aggiungendo la nuova equazione

$$\dot{v} = a$$

La variabile a, che rappresenta l'accelerazione del robot, sarà dunque, per il momento, una nuova variabile di controllo assieme alla velocità di rotazione ω.

Fatto questo, con una opportuna trasformazione di coordinate, si può riscrivere il modello nelle nuove variabili di stato; in particolare supponiamo di porre:

$$z_1 = x \qquad z_2 = y \qquad z_3 = \dot{x} \qquad z_4 = \dot{y}$$

ne risultano di conseguenza le equazioni dinamiche:

$$\begin{cases} \dot{z}_1 = z_3 \\ \dot{z}_2 = z_4 \\ \dot{z}_3 = a\,cos\phi - v\omega\,sin\phi = \boxed{a_x} \\ \dot{z}_4 = a\,sin\phi + v\omega\,cos\phi = \boxed{a_y} \end{cases} \quad \textit{Variabili di controllo}$$

a partire dalle quali possiamo definire i nuovi ingressi di controllo (a_x, a_y). E' immediato verificare che il sistema ottenuto è composto, in realtà, da una coppia di

integratori doppi tra loro completamente indipendenti e, pertanto, risulta come desiderato un sistema lineare.

Il passaggio di coordinate effettuato, invece, risulta sempre invertibile purché sia $v \neq 0$, ovvero a patto che il robot non sia fermo:

$$z = \psi(\xi) \qquad \rightarrow \qquad \xi = \psi^{-1}(z) \quad per\ v \neq 0$$

Tale condizione è del resto sempre verificata nelle prove che si intende fare e, pertanto, la trasformazione può essere utilizzata senza particolari attenzioni.

Per l'applicazione pratica di tale controllore linearizzante non resta che invertire la relazione scritta per ricavare gli ingressi ω ed a, e di conseguenza v, da utilizzare:

$$\begin{bmatrix} \omega \\ a \end{bmatrix} = \begin{bmatrix} -v\ sin\phi & cos\phi \\ v\ cos\phi & sin\phi \end{bmatrix}^{-1} \begin{bmatrix} a_x \\ a_y \end{bmatrix} \quad v \neq 0$$

di nuovo si può notare come per $v = 0$ la matrice risulti singolare, ed il problema non risolubile.

Riordinando gli stati del sistema ottenuto è possibile distinguere facilmente i due integratori disaccoppiati:

$$\begin{cases} \dot{z}_1 = z_3 \\ \dot{z}_3 = a_x \end{cases} \qquad\qquad \begin{cases} \dot{z}_2 = z_4 \\ \dot{z}_4 = a_y \end{cases}$$

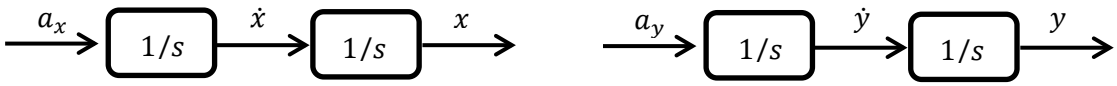

Fig. 3.8.1*– Doppi integratori in uscita all'anello di controllo*

per i quali il progetto di una legge di controllo stabilizzante esterna risulta notevolmente semplificato.

In sintesi, dunque, la legge in *feedback* vista consente di realizzare un anello di controllo interno che rende esternamente lineare il sistema ad uniciclo.

3.8.1 – Legge di controllo.

A questo punto, dunque, possiamo occuparci di risolvere il problema del parcheggio vero e proprio considerando come variabili di controllo le due accelerazioni lungo le

coordinate cartesiane a_x ed a_y. A tal proposito è possibile, innanzitutto, imporre al sistema l'inseguimento di una coppia di riferimenti di posizione (x_r, y_r) semplicemente adottando una legge lineare di tipo proporzionale; ad esempio:

$$a_x = -a_1(z_1 - x_{ref}) - a_2 z_3$$

$$a_y = -a_3(z_2 - y_{ref}) - a_4 z_4$$

e scegliendo opportunamente il valore dei coefficienti $a_1 \dots a_4$ affinché il sistema ottenuto sia stabile e con una dinamica sufficientemente veloce. In tal caso, infatti, dalla scelta dei parametri che compaiono nelle due leggi dipende la posizione degli autovalori dell'anello chiuso, dal momento che stiamo implementando direttamente un *feedback* sullo stato.

$$\begin{cases} \dot{z}_1 = z_3 \\ \dot{z}_3 = -a_1 z_1 - a_2 z_2 + a_1 x_r \\ \dot{z}_2 = z_4 \\ \dot{z}_4 = -a_3 z_2 - a_4 z_4 + a_3 y_r \end{cases} \qquad \dot{z} = \begin{bmatrix} 0 & 1 & \begin{bmatrix} 0 & 0 \end{bmatrix} \\ -a_1 & -a_2 & \\ \begin{bmatrix} 0 & 0 \end{bmatrix} & & \begin{matrix} 0 & 1 \\ -a_3 & -a_4 \end{matrix} \end{bmatrix} z + \begin{bmatrix} a_1 \\ a_3 \end{bmatrix} u$$

la matrice di stato ottenuta è, come previsto, diagonale a blocchi e pertanto possiamo evidenziare i due sottosistemi disgiunti e progettare ciascun controllore separatamente. Dal momento che, inoltre, i due sottoblocchi sono del tutto identici potremmo utilizzare la stessa legge per entrambi, e quindi concentrarci per il progetto solamente su di un sistema ridotto.

Sottolineiamo nuovamente che, come è possibile vedere direttamente, la linearizzazione utilizzata è di tipo dinamico, ovvero che dipende strettamente dal valore assunto dalle variabili di stato. Al contrario una soluzione di tipo statico non è applicabile a causa della presenza del vincolo anolonomo mentre la linearizzazione algebrica porterebbe ad un sistema non completamente controllabile.

Quanto visto finora è valido sia nel caso a tempo continuo sia, con qualche piccola modifica, nel caso a tempo discreto; in particolare quest'ultimo è quello di interesse pratico per l'implementazione sul sistema reale, e pertanto merita di essere visto in dettaglio.

A tal proposito supponiamo di aver scelto un passo di campionamento τ. Utilizzando il metodo di Eulero in avanti una prima semplice discretizzazione per il sistema ridotto è la seguente:

$$\begin{cases} \dot{z}_1 = z_3 \\ \dot{z}_3 = a_x \end{cases} \quad \rightarrow \quad \begin{cases} z_1^+ = z_1 + \tau\, z_3 \\ z_3^+ = z_3 + \tau\, a_x \end{cases}$$

Se al sistema ottenuto si applica la legge algebrica definita in precedenza si ottiene la legge che governa l'evoluzione del sistema in anello chiuso:

$$\begin{bmatrix} z_1^+ \\ z_3^+ \end{bmatrix} = \begin{bmatrix} 1 & \tau \\ -\tau a_1 & 1-\tau a_3 \end{bmatrix} \begin{bmatrix} z_1 \\ z_3 \end{bmatrix} = \left(\begin{bmatrix} 1 & 0 \\ 0 & 1 \end{bmatrix} + \underbrace{\begin{bmatrix} 0 & 1 \\ -a_1 & -a_3 \end{bmatrix}}_{I} \underbrace{\tau}_{A_0} \right) \begin{bmatrix} z_1 \\ z_3 \end{bmatrix}$$

Per scegliere i valori opportuni di a_1 ed a_3 è bene calcolare il determinante della matrice $(\lambda I - A)$:

$$p(\lambda) = \det\big(\lambda I - (I + A_0\tau)\big) = \det\big(\tilde{\lambda} I - A_0\big)\tau \qquad \tilde{\lambda} = \frac{\lambda - 1}{\tau}$$

da cui si ottiene:

$$\det \begin{bmatrix} \tilde{\lambda} & -1 \\ a_1 & \tilde{\lambda} + a_3 \end{bmatrix} = \tilde{\lambda}^2 + a_3\tilde{\lambda} + a_1$$

Se si sceglie, ad esempio, che il polinomio ottenuto sia nella forma $\left(\tilde{\lambda} + \tilde{\eta}\right)^2$ una possibile soluzione è quella di imporre $a_1 = \eta$ ed $a_3 = 2|\eta|$; questa porta direttamente alla condizione di stabilità:

$$\tilde{\lambda} = -\eta \quad \rightarrow \quad \frac{\lambda - 1}{\tau} = -|\eta| \quad \rightarrow \quad \lambda = 1 - |\eta|\tau \quad \rightarrow \quad |\eta|\tau < 1$$

si può vedere, cioè, come la scelta dei parametri dipenda dal tempo di campionamento scelto per il controllo.

Abbiamo quindi visto, in quest'ultima sezione, come sia possibile risolvere il problema del parcheggio evitando tutte le tecniche descritte nelle precedenti e senza bisogno di far ricorso a particolari assunzioni sul percorso fatto dal robot. In ogni caso quanto riportato è stato utile ad analizzare più da vicino e comprendere le problematiche legate alla robotica mobile ed i possibili approcci risolutivi; l'implementazione effettiva sui dispositivi scelti per il progetto ha permesso, inoltre, di verificare accuratamente il loro comportamento nelle diverse situazioni e messo in luce aspetti pratici rilevanti per le fasi successive dello sviluppo.

La tecnica di *feedback linearization* proposta ha inoltre l'indubbio vantaggio di rendere disponibile all'esterno un sistema completamente lineare con un comportamento dinamico dei più semplici; questo ha consentito, successivamente, di affrontare il problema di controllo *MPC* evitando il ricorso alla teoria non lineare, e pertanto è risultato l'approccio effettivamente adottato nelle implementazioni.

3.8.2 – Implementazione.

L'implementazione della legge descritta finora risulta relativamente semplice e poco dispendiosa dal punto di vista computazionale. A tal proposito si ricorda brevemente che i passi da effettuare per controllare il sistema consistono innanzitutto nel calcolo dei nuovi ingressi di controllo a_x ed a_y secondo una qualsiasi legge stabilizzante, dopodiché si ha il ricorso alla trasformazione inversa che consente di ricavare gli ingressi intermedi a ed ω ed infine l'accelerazione trovata deve essere integrata per calcolare la velocità v, *input* effettivo del sistema ad uniciclo visto. Sarà in particolare quest'ultimo passo a necessitare di maggior attenzione.

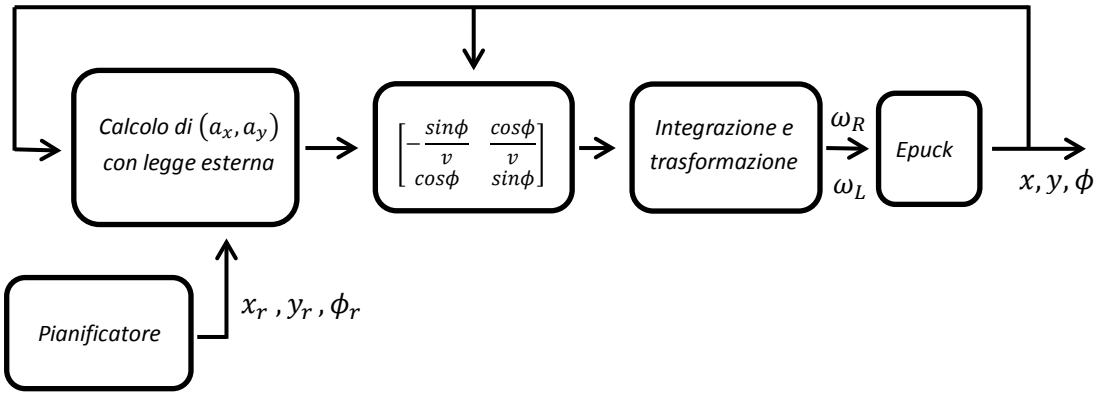

Fig. 3.8.2.1*– Schema del controllo con feedback*

Nel sistema aumentato presentato in 3.8.1 l'ultima equazione $\dot{v} = a$ è puramente fittizia dato che, in realtà, la variabile di controllo a cui possiamo fare riferimento è solamente la velocità lineare v e non l'accelerazione a utilizzata per definire il resto dell'algoritmo di linearizzazione. Sebbene questo non presenti alcuna difficoltà nel caso a tempo continuo, quando si passa a considerare una implementazione a tempo discreto è necessario sostituire l'espressione vista con una sua approssimazione.

Per quanto riguarda tale fase di integrazione numerica a tempo discreto si è pensato in un primo momento di fare ricorso alla soluzione più immediata, ovvero di porre

$$v^+ = v + a\,\tau$$

incrementando ad ogni passo di campionamento il valore cumulato della velocità. Tale approccio si è rivelato valido per l'esecuzione delle simulazioni e dei primi test sulla legge di controllo stabilizzante e sulla legge *MPC* per il singolo robot*,* ovvero fintanto che, a causa delle ridotte richieste computazionali, i tempi di campionamento utilizzati sono stati mantenuti limitati. Alcuni problemi hanno iniziato a manifestarsi quando si è passati a simulare le dinamiche di coordinamento poiché, a seguito dei maggiori tempi computazionali richiesti dall'algoritmo, è stato necessario rallentare l'intero sistema.

Per far fronte a tale situazione si è allora deciso di implementare l'anello di linearizzazione interna ad una frequenza superiore rispetto al controllo *MPC* esterno, che vedremo nei capitoli successivi, e di ricorrere contemporaneamente ad un algoritmo di integrazione numerica più complesso come *ode23t*. In particolare l'idea è stata quella di far evolvere in avanti il modello sistema, con gli ingressi a_x ed a_y calcolati, all'interno dell'intervallo, ricavare il valore finale della velocità raggiunta a partire dalle condizioni iniziali attuali, e poi ricostruire l'effettiva velocità da applicare mediando tra quella corrente e quella appena trovata.

Tale tecnica consente così di far ricorso agli algoritmi per l'integrazione dell'ambiente *MATLAB* che si sono dimostrati efficienti ed in grado di risolvere il problema che si era presentato; sebbene le simulazioni del presente e del successivo capitolo siano state eseguite con la semplice integrazione proposta inizialmente, il resto della trattazione sarà invece affrontato con la tecnica appena descritta.

3.8.3 – Simulazioni e prove.

Esattamente come fatto per tutte le leggi di controllo presentate nei paragrafi precedenti la validazione della procedura di linearizzazione mediante *feedback* dinamico descritta viene effettuata innanzitutto simulando l'inseguimento di una traiettoria rettilinea a velocità costante. Per ottenere tale risultato il controllore esterno che produce le accelerazioni a_x ed a_y è stato sviluppato secondo la semplice legge lineare vista. Il risultato è quello presentato in Figura 3.8.3.1.

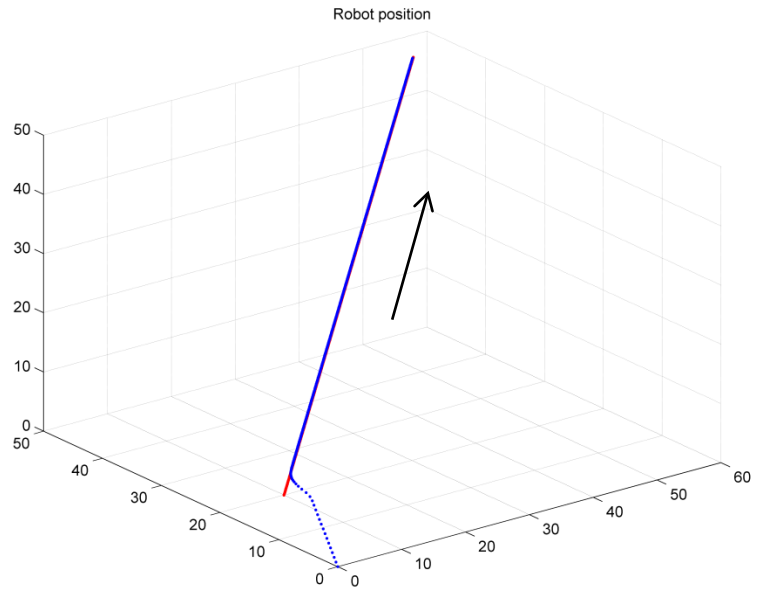

Fig. 3.8.3.1*– Simulazione di una traiettoria rettilinea – feedback dinamico e legge lineare.*

Si nota che, anche partendo da una posizione iniziale diversa dal riferimento, il modello controllato insegue correttamente la traiettoria richiesta con tempi di rientro sufficientemente brevi. In contemporanea i valori delle costanti che compaiono nella legge di controllo esterna sono scelti in modo da assicurare il rispetto dei vincoli sulla velocità di movimento dovuti agli attuatori che non entrano mai in saturazione.

Dal momento che tale legge sarà quella poi utilizzata nel seguito della trattazione, si è scelto di effettuare anche prove di inseguimento ulteriori, che non sono state riportate per i casi precedenti. Situazioni interessanti sono state identificate nell'inseguimento di un percorso circolare, di un percorso rettangolare, ed infine di un percorso generico realizzato interpolando dei punti di passaggio tramite *spline*.

In particolare il movimento lungo una traiettoria circolare ha portato al risultato di Figura 3.8.3.2.

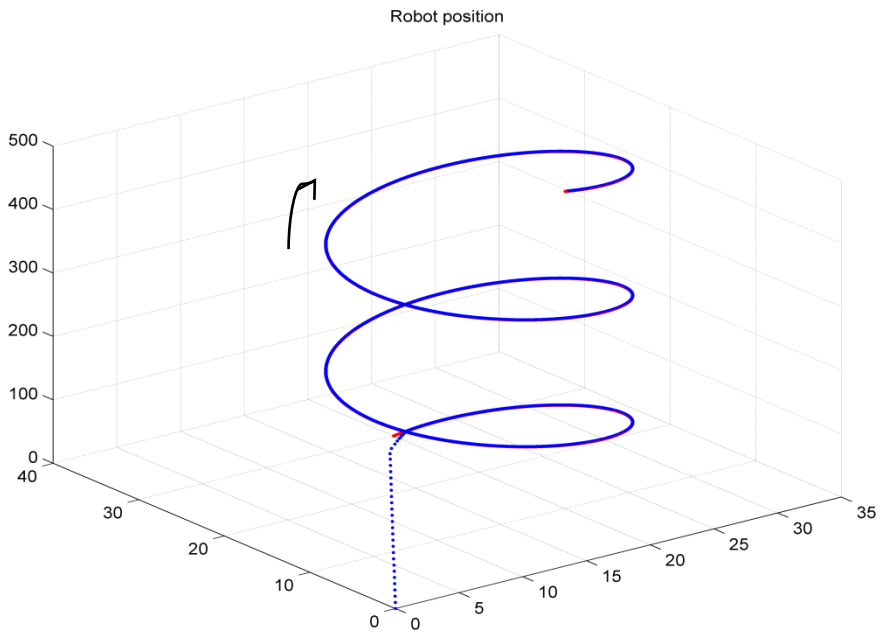

Fig. 3.8.3.2– *Simulazione di una traiettoria circolare – feedback dinamico e legge lineare.*

Di nuovo si nota che la legge proposta è in grado di stabilizzare il modello del robot lungo il percorso prescelto permettendogli di completare il *task*.

Tenendo presente i risultati ottenuti i due esperimenti precedenti sono stati ripetuti sul sistema reale facendo girare in parallelo la simulazione a partire dalle stesse condizioni iniziali. Il risultato dell'inseguimento di una traiettoria rettilinea è visibile in Figura 3.8.3.3. In questa, come nei grafici successivi, la linea verde rappresenta la traiettoria tenuta dal sistema vero mentre la linea blu rappresenta quella del modello di simulazione.

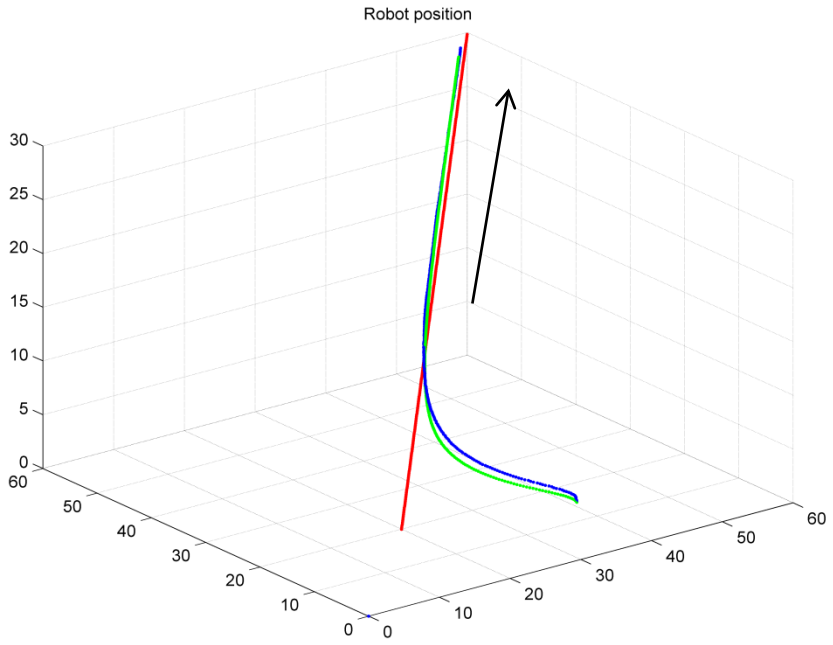

Fig. 3.8.3.3*– Simulazione e test di una traiettoria rettilinea – feedback dinamico e legge lineare.*

L'inseguimento di una traiettoria circolare porta invece al risultato di Figura 3.8.3.4.

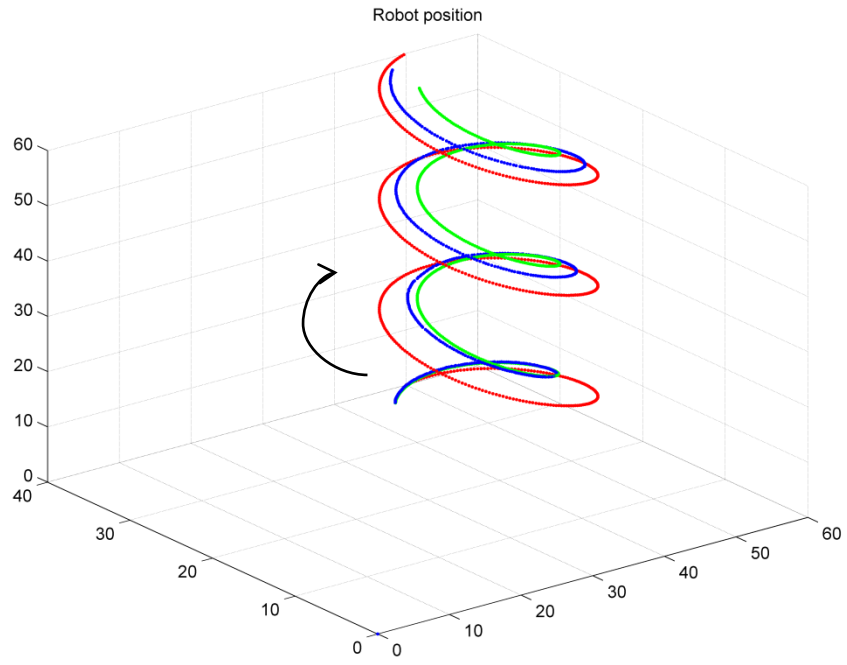

Fig. 3.8.3.4*– Simulazione e test di una traiettoria circolare – feedback dinamico e legge*

Per le due prove successive si omette il risultato ottenuto dalla sola simulazione e si mostra il grafico complessivo ottenuto dal parallelo tra modello e sistema reale. Il comportamento lungo la traiettoria rettangolare si vede in Figura 3.8.3.5.

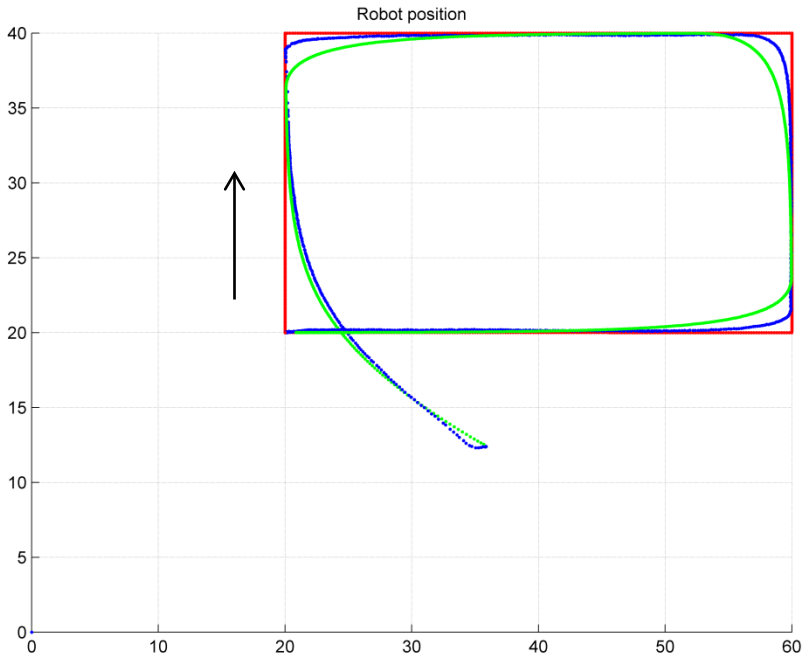

Fig. 3.8.3.5*– Simulazione e test di una traiettoria rettangolare – feedback dinamico e legge lineare.*

Anche in questo caso la legge di controllo si dimostra efficace e consente al modello ed al robot di completare il compito richiesto muovendosi lungo un percorso rettangolare. Il disallineamento nel punto di curvatura è dovuto nuovamente a dei comportamenti non lineari dovuti al punto di appoggio, trascurati nel modello, ed alla dinamica reale del robot; tuttavia questo non risulta rilevante per la trattazione svolta nel seguito.

Infine l'inseguimento di una traiettoria di tipo *spline* ottenuta interpolando alcuni punti di passaggio porta al risultato di Figura 3.8.3.6.

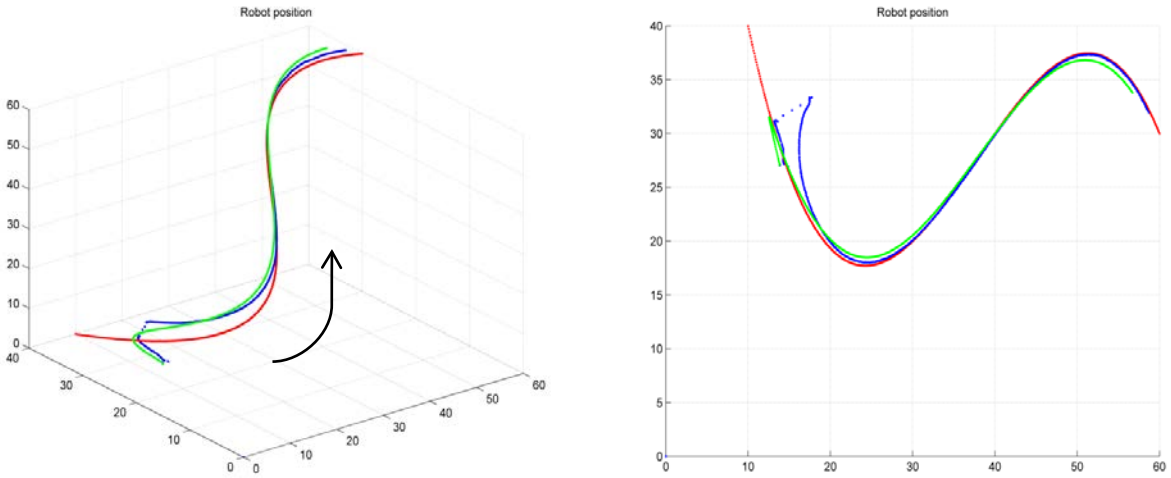

Fig. 3.8.3.6*– Simulazione e test di una traiettoria spline – feedback dinamico e legge lineare.*

Le buone prestazioni ottenute confermano ancora una volta la validità della soluzione proposta. Per questo motivo, per la semplicità della legge utilizzata e per la forma lineare ottenuta in uscita all'anello chiuso, tale soluzione sarà adottata per la scrittura del controllore stabilizzante interno al robot ad uniciclo.

4- OBSTACLE AVOIDANCE PER ROBOT MOBILI

4.1 – Introduzione.

In generale il problema di navigazione di un robot mobile verso un obiettivo, cioè con una posizione finale da raggiungere all'interno di un ambiente solo parzialmente noto a priori, consiste nella definizione di un percorso lungo il quale il robot è in grado di muoversi in sicurezza sfruttando le conoscenze a disposizione, ed eventualmente le informazioni provenienti dai propri sensori di bordo. Più in particolare, dunque, il classico problema della navigazione autonoma necessita sia di una fase statica di definizione del percorso da seguire, che consenta al robot di completare il proprio *task*, sia di una componente dinamica in grado di far fronte alle incertezze presenti nell'ambiente circostante, come ad esempio la presenza di ostacoli fissi o mobili rilevati durante il movimento lungo il percorso pianificato. Per questo motivo si parla solitamente di due sottoproblemi principali che vanno sotto il nome di *"path planning"* ed *"obstacle avoidance"*.

L'obiettivo che ci si è prefissati nel presente lavoro è invece quello di costruire una tecnica di controllo in grado di affrontare contemporaneamente ed *online* i due problemi, o meglio di guidare il robot alla posizione desiderata senza dover passare da una fase di pianificazione dell'intero percorso. Nello specifico l'idea è quella di sfruttare un opportuno anello di controllo locale per gestire il movimento del robot verso un preciso riferimento ed un anello di controllo esterno, basato sul tecniche predittive, in grado di generare il riferimento migliore per procedere verso il completamento del *task*, restando a distanza dagli ostacoli. A tal proposito il ricorso ad un approccio di tipo predittivo per l'ottimizzazione del comportamento del robot ripropone, limitatamente all'orizzonte considerato, i problemi di obstacle avoidance tipici della fase di *path planning* di un qualsiasi problema di navigazione autonoma in senso generico.

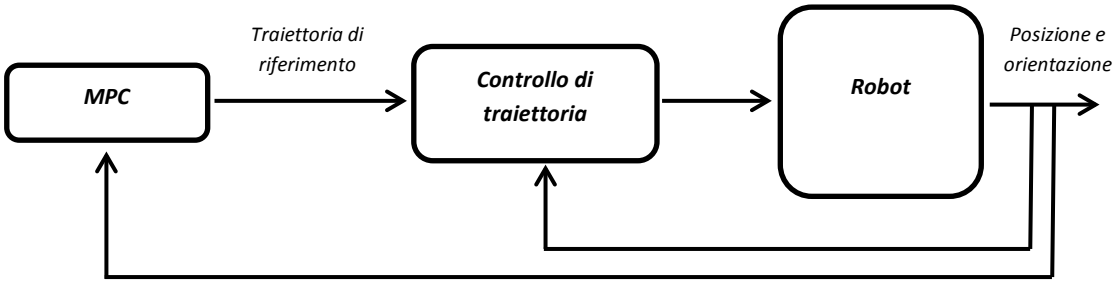

***Fig. 4.1.1** – Schema del controllo in cascata MPC e legge stabilizzante.*

Ciò porta ad analizzare innanzitutto le soluzioni già presentate in letteratura prima di procedere alla ricerca di quella maggiormente adeguata al problema in esame. Data infatti l'importanza che ha assunto negli ambiti più svariati il problema della navigazione autonoma di robot mobili, la letteratura è ricca di soluzioni riguardanti sia la tematica del *path planning* che quella dell'*obstacle avoidance*. Un'attenzione particolare è inoltre dedicata ai sistemi soggetti a vincoli non olonomi, che rappresentano il modello per la maggior parte di dispositivi mobili presenti nella realtà.

Scendendo maggiormente nel dettaglio, supponiamo di avere a disposizione una mappa dell'ambiente di navigazione e di definire una determinata posizione come obiettivo; in tal caso il *path planning* implica la ricerca di un percorso geometrico a partire dall'attuale posizione del robot fino alla posizione desiderata. Risulta evidente che le prestazioni nell'esecuzione di tale procedura di pianificazione dipendono strettamente dal tipo di problema; ad esempio è sufficiente considerare che spesso i robot mobili operano in ambienti non strutturati in cui la conoscenza a priori dell'ambiente stesso è assente o comunque parziale. Ancora, può accadere che l'ambiente, seppur noto, non si mantenga statico nel tempo, ad esempio a causa del moto di altri robot oppure per la presenza di esseri umani nell'area di lavoro. Infine, anche dopo aver pianificato il percorso più opportuno, va tenuto presente che l'esecuzione stessa del task risulta affetta da incertezza, e che pertanto si dovranno utilizzare meccanismi di controllo in feedback.

A causa dell'incertezza e della mancata conoscenza di parte della struttura dell'ambiente possiamo dunque vedere che, per generare un movimento privo di collisioni fino all'obiettivo, è assolutamente necessario che al *path planning* globale venga associato un sistema locale di gestione della navigazione, in grado di far fronte alla presenza di eventuali ostacoli, compreso il loro rilevamento e la definizione di opportune manovre per evitarli.

Tenendo presente questi aspetti allora, si può definire il concetto di *obstacle avoidance* come insieme di metodologie con le quali è possibile deformare il percorso pianificato di un robot per evitare possibili ostacoli. Di conseguenza il moto complessivo di un dispositivo dipenderà non solo dal percorso pianificato, ma, in relazione alla posizione attuale, anche dalle letture dei sensori sull'ambiente circostante nell'istante considerato.

A tal proposito esiste una grande varietà di algoritmi per l'*avoidance* che vanno dal semplice *re-planning* della traiettoria a variazioni reattive della strategia di controllo. Questi si differenziano a seconda dell'uso fatto delle informazioni provenienti dai sensori e delle strategie vere e proprie applicate per aggirare gli ostacoli incontrati. Tali caratteristiche sono quindi quelle che dovremo tener presenti per valutare il meccanismo più adatto al caso in esame, trascurando tutti gli aspetti più specificatamente riferiti al concetto di pianificazione del percorso.

4.2 – Gli algoritmi di avoidance in sintesi.

L'approccio più semplice e dal comportamento facilmente intuibile è quello che compare in letteratura sotto il nome di "*bug's algorithm"*. L'idea utilizzata è infatti quella di muoversi in linea retta verso l'obiettivo finché non viene incontrato un ostacolo ed in tal caso circumnavigarlo fintanto che il moto verso l'obiettivo non è nuovamente possibile. Le informazioni dei sensori necessarie all'implementazione sono date solamente dalle letture più recenti, che indicano all'istante attuale la presenza di un ostacolo in prossimità del robot.

Una soluzione molto diffusa, e con una struttura leggermente più complessa, è quella basata sulla definizione di un "potenziale artificiale"; ciò che ne deriva è di fatto un particolare *path planning* contenente al suo interno anche la garanzia di *obstacle avoidance*. Più precisamente il robot mobile è considerato come una particella che si muove immersa in un campo potenziale generato dall'obiettivo ed influenzato dagli ostacoli presenti nell'ambiente circostante; per la definizione di tale campo è sufficiente pensare che l'obiettivo generi un potenziale attrattivo opportunamente ampio mentre ciascun ostacolo stia generando un potenziale repulsivo localizzato. Con questo principio gli ostacoli possono essere a loro volta noti a priori, ed in tal caso il potenziale repulsivo può essere calcolato offline come quello attrattivo legato all'obiettivo, oppure possono essere rilevati online, tramite i sensori a bordo del robot, ed il relativo potenziale valutato in tempo reale per determinarne l'effetto sull'ambiente. Quest'ultimo aspetto è interessante per quanto riguarda il problema di coordinamento che stiamo considerando. Sulla base della definizione data si nota come, accanto alla capacità di *obstacle avoidance*, l'approccio alla pianificazione tramite potenziale artificiale comprenda anche una strategia di controllo del movimento del robot; in particolare esso ne definisce naturalmente il vettore di velocità, utilizzato per guidarlo attraverso gli ostacoli fino all'obiettivo.

Altra particolare tecnica di *obstacle avoidance* presente in letteratura è quella denominata "*Vector Field Histogram*". L'idea di base è quella di generare un istogramma basato sulla rappresentazione polare dell'occupazione spaziale dovuta agli ostacoli nelle immediate vicinanze del robot. Tale istogramma, costruito direttamente attorno all'attuale posizione del robot sfruttando i dati cumulativi dei sensori, viene poi valutato per scegliere il settore angolare più consono tra tutti quelli considerati, ovvero quello con la densità di ostacoli più bassa, ed allineare con questo la direzione di sterzo del robot.

Tra le varie soluzioni di letteratura esiste anche il concetto di "*elastic band*" che combina invece un problema di *path planning* globale con un controllo del robot basato sui sensori ed applicato in tempo reale per garantire un movimento privo di collisione fino all'obiettivo. A tal proposito l'aggettivo *elastic* indica un percorso deformabile e libero da ostacoli. L'idea è in particolare quella di definire inizialmente l'*elastic band* identificandolo con il semplice percorso generato da un pianificatore ad alto livello; fatto

questo, durante l'esecuzione del *task*, ogni volta che un ostacolo viene individuato, tale percorso è deformato in accordo ad una forza artificiale con l'obiettivo di ottenere un *path* dall'aspetto continuo e nello stesso tempo garantire il mantenimento di una certa distanza dagli ostacoli. In questo modo, sfruttando il fatto che il percorso elastico si deforma nel momento in cui cambiamenti nell'ambiente previsto sono rilevati dai sensori, si è in grado di consentire al robot di sopperire alle incertezze e di evitare ostacoli di cui non era nota l'esistenza oppure in movimento.

Infine nelle tecniche di base si registra un approccio dinamico cosiddetto "*behavior-based*" che prevede di modellare il comportamento di un robot mobile di fronte ad un ostacolo con un sistema non lineare dinamico. Nel fare questo si cerca di rendere l'obiettivo come un equilibrio globalmente stabile per il sistema, mentre si interpretano gli ostacoli come equilibri instabili; in questo modo la combinazione dei due guida automaticamente il robot al punto desiderato evitando collisioni.

4.3 - Bug's algorithm.

In letteratura gli algoritmi "ad insetto" sono presenti sostanzialmente in due versioni, diversificate sulla base della strategia adottata in prossimità dell'ostacolo. In entrambi i casi rappresentano un metodo semplice per far fronte ad ostacoli inaspettati durante il moto del robot a partire da un punto iniziale verso un obiettivo. In generale allora, in quanto parte delle tecniche per l'*obstacle avoidance*, lo scopo dell'algoritmo è quello di produrre un percorso privo di collisioni tra il robot mobile e gli ostacoli. Il principio di fondo con cui tale risultato è ottenuto è quello di aggirare lungo il perimetro gli ostacoli rilevati, alternando tra una legge di avanzamento di tipo *go to goal* ed una legge *border following*. Le due differenti versioni sono distinte dal tipo di condizione sotto cui il comportamento di inseguimento del bordo, per la gestione dell'ostacolo, viene poi sostituito nuovamente con una legge di movimento verso l'obiettivo per il completamento del *task*. In particolare si è fatto riferimento a [8].

A tal proposito si consideri innanzitutto un generico robot in moto in uno spazio a due dimensioni come un semplice punto in uno spazio piano, in movimento da una posizione iniziale S ad un obiettivo finale G; tale robot sia inoltre dotato di un sensore di contatto, o più genericamente di prossimità, in grado di rilevare gli ostacoli una volta raggiunti (*range* ridotto). Nella prima versione dell'algoritmo non appena l'ostacolo *i-esimo* è rilevato il robot inizia a percorrere un giro completo lungo il suo perimetro a partire dal punto di collisione H; tale moto attorno all'ostacolo ha come scopo quello di rilevare il punto a minima distanza dall'obiettivo L che sarà poi utilizzato per riprendere la navigazione verso il completamento del *task*. Terminato il primo giro, dunque, il robot ripercorre nuovamente parte del contorno fino a raggiungere il punto di uscita L, identificato in precedenza, ed infine ripartire in linea retta verso l'obiettivo. Tale

approccio è chiaramente inefficiente poiché il robot impiega molto tempo nella ricerca del punto a distanza minima da cui ripartire, tuttavia garantisce che qualsiasi obiettivo raggiungibile possa effettivamente essere raggiunto indipendentemente dalla conformazione degli ostacoli.

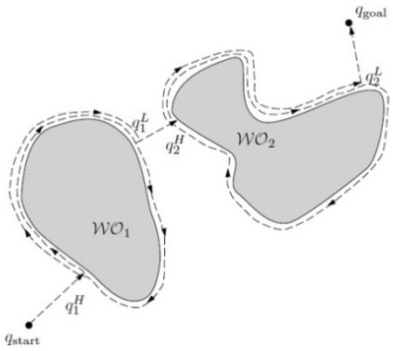

Fig. 4.3.1 *– Bug's algorithm prima versione.*

Nella seconda versione dell'algoritmo la circumnavigazione dell'ostacolo inizia, come nel caso precedente, dal punto di collisione H, ma termina questa volta quando il robot attraversa nuovamente la linea che stava seguendo inizialmente diretto al target, ponendo fine al comportamento *boundary-following* e ritornando alla navigazione verso l'obiettivo. La procedura è poi ripetuta ogni volta che si incontrano ostacoli. In generale questa soluzione porta ad avere un tempo di viaggio più breve rispetto alla versione presentata in precedenza ed è maggiormente efficiente soprattutto in spazi aperti; tuttavia questa volta vi possono essere situazioni in cui il robot non è in grado di completare il *task*, come ad esempio in presenza di strutture a labirinto, in cui esso può rimanere incastrato.

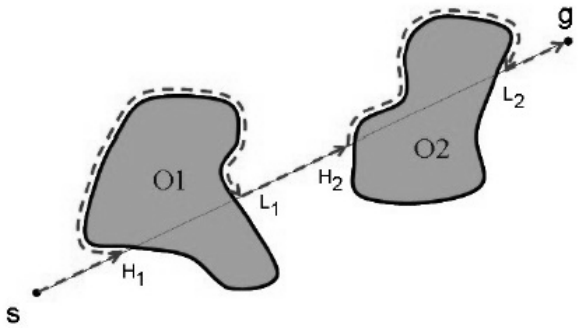

Fig. 4.3.2 *– Bug's algorithm seconda versione.*

Seppur molto semplici ed immediati, gli algoritmi di navigazione "ad insetto" hanno ovviamente una serie di lati negativi. Innanzitutto non si tiene in alcun modo conto del modello cinematico del robot, che viene interpretato come punto in movimento, e ciò

porta a gravi limitazioni, soprattutto quando si considerano robot anolonomi le cui capacità di movimento sono vincolate; inoltre dal momento che solo l'informazione più recente proveniente dai sensori viene tenuta in conto, eventuali disturbi possono avere degli effetti gravi sulla prestazione complessiva della tecnica.

In realtà questa tecnica non risulta nel nostro caso di alcuna utilità pratica, poiché l'idea di fondo è quella di pianificare online i movimenti del robot senza predisporre alcun percorso preferenziale verso l'obiettivo e pertanto senza avere a disposizione la legge definita come *go to* goal; inoltre l'utilizzo della webcam come sistema di localizzazione permette in teoria di ricorrere a soluzioni più complesse.

4.4 - Potenziali artificiali.

L'utilizzo di un campo potenziale artificiale per la pianificazione del percorso di un robot mobile è basato su un semplice e potente principio proposto per la prima volta da Khatib nel 1980 per gestire il movimento di un manipolatore [9]. In particolare nel caso in questione il robot è considerato come una particella in grado di muoversi all'interno di un campo caratterizzato dal potenziale generato dall'obiettivo e dagli ostacoli presenti nell'ambiente circostante. L'obiettivo genera un potenziale attrattivo, mentre ciascun ostacolo genera un potenziale repulsivo; se si interpreta tale definizione in senso energetico è facile vedere come il gradiente del campo vettoriale così costruito possa essere visto in ciascun punto come una forza. Se si suppone che il robot sia sensibile a tale potenziale, avremo allora che, una volta immerso nel campo, risulterà soggetto all'azione di una forza che lo guida verso l'obiettivo per effetto della componente attrattiva generata dal target, mentre lo spinge lontano dagli ostacoli per effetto della componente repulsiva dovuta al campo da essi generato. Intuitivamente il comportamento che si ottiene è simile a quello di una particella carica elettricamente che si muove tra altre cariche dello stesso segno e di segno opposto.

Distribuzioni di potenziali come quelle definite possono inoltre essere viste come un terreno con montagne, generate dagli ostacoli, alternate a valli, il cui punto più basso rappresenta l'obiettivo finale che il robot deve raggiungere. Questo, nel suo moto, seguirà naturalmente un percorso lungo il gradiente negativo della funzione potenziale, ovvero, in altre parole, viaggerà verso il punto più basso della vallata. A partire da tale analogia però è facile rendersi conto di come in particolari situazioni la presenza di minimi locali, diversi dall'obiettivo, ma comunque caratterizzati dall'avere un gradiente nullo, possano far si che il robot resti intrappolato, senza poter completare il compito assegnato.

Più nel dettaglio supponiamo che il robot in questione sia un punto in movimento in uno spazio di dimensione n e che la sua posizione possa essere riassunta nel vettore q; per

semplicità e senza perdere di generalità ipotizziamo poi che il problema sia applicato ad un robot in moto su di un piano, così che risulti $n = 2$ e che la posizione del robot sia definita completamente dalla coppia $q = (x, y)$, come nella maggior parte dei problemi di interesse pratico.

Il campo potenziale all'interno del quale si muove il robot sarà allora una funzione scalare $U(q)$ da $\mathbb{R}^2$ in $\mathbb{R}$ generata dalla sovrapposizione di potenziali repulsivi ed attrattivi, dovuti rispettivamente agli ostacoli ed all'obiettivo:

$$U(q) = U_{att}(q) + U_{rep}(q)$$

Il potenziale repulsivo a sua volta può essere visto come la sovrapposizione di diversi potenziali repulsivi individuali generati da ciascun ostacolo e pertanto può essere scritto come:

$$U(q) = U_{att}(q) + \sum_i U_{rep_i}(q)$$

avendo indicato con U_{rep_i} il potenziale generato dall'ostacolo i-esimo.

Un'idea pratica di quello che accade in tali condizioni è data dalla Figura 4.4.1.

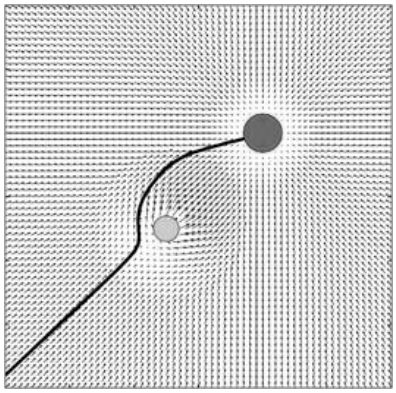

Fig. 4.4.1 *– Esempio di potenziale artificiale repulsivo ed attrattivo.*

Supponiamo ora che la funzione potenziale costruita sia differenziabile; allora ad ogni valore della posizione q il gradiente di tale funzione, indicato come $\nabla U(q)$, è, secondo il suo senso fisico, un vettore che punta localmente nella direzione che incrementa maggiormente la funzione $U(q)$.

Affinché il campo potenziale così definito possa agire da funzione di navigazione per la guida del robot all'interno dello spazio considerato, la sua componente attrattiva, responsabile del completamento del *task*, viene generalmente scelta in modo che sia zero all'obiettivo ed aumenti al crescere della distanza da esso. Al contrario, invece, la componente repulsiva viene costruita in modo da assumere un valore elevato o

idealmente infinito quando il robot è vicino agli ostacoli, e da decrescere fino ad annullarsi man mano che questo si allontana da essi .

Tenendo presente tale implementazione otteniamo allora che il robot dovrà muoversi sempre in modo da diminuire il più possibile il valore assunto dalla funzione potenziale nella posizione occupata; in altre parole la forza in grado di guidarlo verso l'obiettivo e mantenerlo lontano dagli ostacoli coinciderà semplicemente con il gradiente del potenziale artificiale cambiato di segno, cioè avremo:

$$F(q) = F_{att}(q) + F_{rep}(q) = -\nabla U_{att}(q) - \nabla U_{rep}(q)$$

La forza $F(q)$ risultante è allora allo stesso modo un vettore che punta per ogni valore della posizione q lungo la direzione nella quale viene localmente minimizzata la funzione potenziale U; d'altra parte tale forza può anche essere interpretata come il vettore velocità con cui il robot, supposto puntiforme, viene guidato al completamento del *task*.

Per il potenziale attrattivo la scelta usuale, che rispetta quanto detto in precedenza, è quella di un paraboloide centrato sull'obiettivo, ovvero di una funzione che cresce con il quadrato della distanza da esso:

$$U_{att}(q) = \tfrac{1}{2}\, k_{att} \left\| q - q_{goal} \right\|_2^2$$

avendo considerato per il calcolo della distanza la norma euclidea.

Se rappresentiamo graficamente la funzione ottenuta abbiamo la Figura 4.4.2.

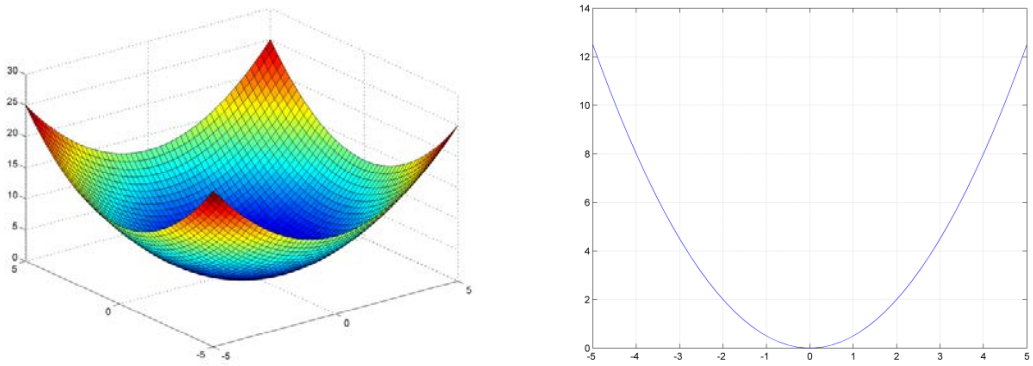

***Fig. 4.4.2** – Esempio di funzione potenziale*

Scegliendo opportunamente il fattore di scala costante k_{att} otteniamo l'effetto di aumentare o diminuire l'intensità con cui il target attira il robot durante la navigazione. Infatti se si calcola il suo gradiente lungo la posizione q

$$\nabla U(q) = k_{att}\,(q - q_{goal})$$

si ottiene un campo vettoriale proporzionale, secondo la costante k_{att}, alla differenza tra la posizione attuale e quella dell'obiettivo e diretto fuori da esso. Ovviamente, quindi, più lontano risulta il robot dal punto desiderato più grande è l'ampiezza del campo attrattivo a cui esso è sottoposto e dunque intuitivamente più grande sarà la velocità con cui esso viaggia.

Imponendo un vettore velocità proporzionale al vettore del campo di forze, il robot viene guidato in modo autonomo verso l'obiettivo e rallenta man mano che si approssima ad esso; al contrario, data la forma quadratica del potenziale in funzione della distanza, avremo che essa cresce illimitatamente fintanto che q si allontana da esso dando vita ad una velocità elevata quando il robot è distante dal target desiderato.

Il potenziale repulsivo, invece, ha il compito di mantenere il robot lontano dagli ostacoli, siano essi noti a priori oppure rilevati durante il movimento dai sensori di bordo, garantendo un percorso privo di collisioni. Abbiamo visto che siamo in grado di ottenere questo effetto facendo si che ogni ostacolo produca localmente un aumento, idealmente infinito, della funzione potenziale dal momento che la navigazione ha come obiettivo la sua minimizzazione. Graficamente quello che si ottiene è ad esempio visibile in Figura 4.4.3.

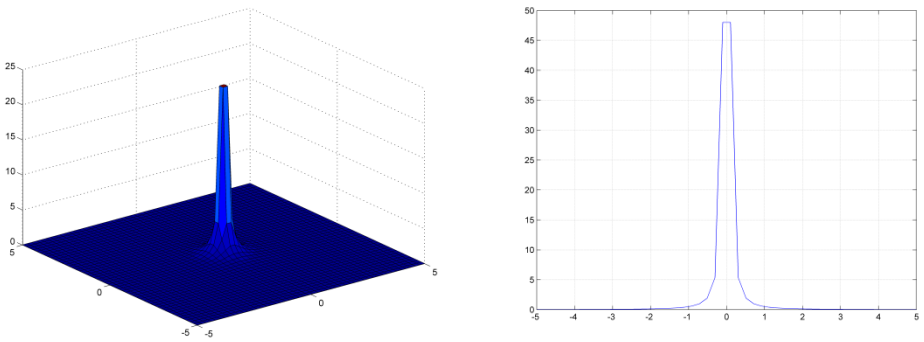

***Fig. 4.4.3** – Esempio di funzione potenziale repulsiva.*

Tale potenziale repulsivo deve quindi essere più forte quando il robot è vicino all'ostacolo ed avere una influenza via via decrescente, fino all'annullamento, quando questo è lontano. Data la natura lineare del problema il potenziale repulsivo può essere generato come sovrapposizione dei singoli effetti repulsivi dovuti a ciascun ostacolo presente nell'area di lavoro, cioè avremo:

$$U_{rep}(q) = \sum_{i} U_{rep_i}(q)$$

E' ragionevole considerare che l'influenza di un singolo ostacolo sul potenziale generato sia limitata all'ambiente nelle sue immediate vicinanze e che scompaia completamente ad una certa distanza da esso; un ostacolo lontano dal robot non deve cioè respingerlo ed al contrario l'ampiezza del potenziale repulsivo deve aumentare notevolmente

quando il robot si avvicina rischiando la collisione. Per tenere conto di tale effetto, e dell'influenza spazialmente limitata, una possibilità semplice di descrivere il potenziale generato dall'ostacolo i-esimo è data dalla funzione seguente:

$$\begin{cases} U_{rep_i}(q) = \frac{1}{2}\ k_{obst_i}\left(\frac{1}{d_{obst_i}(q)}\right)^2 - \left(\frac{1}{d_0}\right)^2 & if\ d_{obst_i}(q) \le d_0 \\ U_{rep_i}(q) = 0 & if\ d_{obst_i}(q) > d_0 \end{cases}$$

in cui d_{obst_i} rappresenta la minima distanza, in norma euclidea, dell'ostacolo i-esimo dalla posizione attuale q, mentre k_{obst} è un fattore di scala costante ed infine d_0 determina la soglia entro la quale l'ostacolo influenza il potenziale, e dunque il moto del robot. Una possibile funzione alternativa può essere invece costruita facendo ricorso alla forma gaussiana, ponendo ad esempio:

$$U_{rep}(q) = k_{obst_i}\, e^{-\delta\, d^2_{obst_i}}$$

e progettando opportunamente i valori di k_{obst} e di δ per tenere in conto del peso e della dimensione dell'ostacolo considerato. Supponendo ad esempio di avere un ostacolo posizionato nell'origine le due soluzioni presentate in precedenza, normalizzando il coefficiente di peso, portano ad esempio ai diagrammi di Figura 4.4.4.

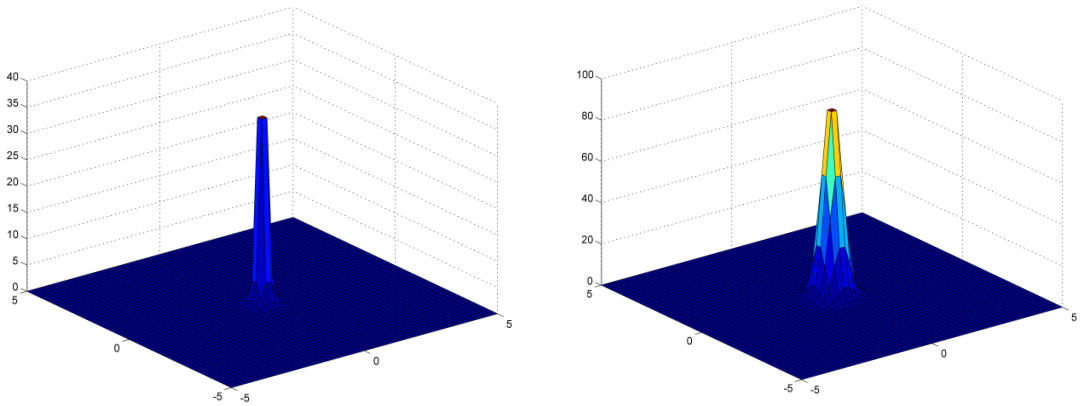

Fig. 4.4.4 *– Esempi di potenziali repulsivi. A sinistra il primo tipo visto, a destra quello*

La prima delle funzioni presentate ha lo svantaggio di avere una derivata non continua sul bordo dell'area di influenza, dettata dal parametro d_0. Tale aspetto, come si vedrà in dettaglio nel seguito del lavoro, porta ad avere alcuni problemi all'atto dell'implementazione. Si può realizzare una soluzione alternativa costruendo una funzione *ad hoc* basata sulle linee guida viste finora; in particolare si richiede che questa consenta di agire virtualmente da muro nella zona dell'ostacolo e nello stesso tempo

permetta di specificare il valore che essa, ed eventualmente le sue derivate successive, assumono proprio sul bordo dell'area in questione.

Più nel dettaglio, allora, una funzione adatta è quella polinomiale, scritta mediante interpolazione in funzione della distanza dall'ostacolo *i-esimo*; il grado di tale polinomio deve consentire di specificare tutte le condizioni al contorno desiderate.

$$\begin{cases} U_{rep_i}(q) = a_0 + a_1\, d_{obst_i}(q) + a_2\, d^2_{obst_i}(q) + \cdots a_n d^n_{obst_i}(q) & if\ d_{obst_i}(q) < d_0 \\ U_{rep_i} = 0 & if\ d_{obst_i}(q) > d_0 \end{cases}$$

Tali condizioni, che nel caso visto saranno $n+1$, consentono di specificare una volta per tutte il valore dei coefficienti che pesano ciascuno dei termini. Supponendo, ad esempio, di voler imporre il valore assunto dalla funzione in tre diversi punti e poi annullare questa e le sue due prime derivate sul bordo dell'area, è necessario fare ricorso ad un polinomio di quinto grado, ovvero determinare le sei costanti $a_0 \dots a_5$. Ipotizziamo che il bordo si trovi ad una distanza 3ε dal centro dell'area, possiamo ad esempio scegliere

$$U_{rep}(0) = f_0 \qquad U_{rep}(\varepsilon) = f_1 \qquad U_{rep}(2\varepsilon) = f_2$$

$$U_{rep}(3\varepsilon) = 0 \qquad U_{rep}{}'(3\varepsilon) = 0 \qquad U_{rep}{}''(3\varepsilon) = 0$$

la soluzione è data direttamente dal sistema di equazioni:

$$\begin{bmatrix} a_0 \\ a_1 \\ a_2 \\ a_3 \\ a_4 \\ a_5 \end{bmatrix} = \begin{bmatrix} 1 & 0 & 0 & 0 & 0 & 0 \\ 1 & \varepsilon & \varepsilon^2 & \varepsilon^3 & \varepsilon^4 & \varepsilon^5 \\ 1 & 2\varepsilon & (2\varepsilon)^2 & (2\varepsilon)^3 & (2\varepsilon)^4 & (2\varepsilon)^5 \\ 1 & 3\varepsilon & (3\varepsilon)^2 & (3\varepsilon)^3 & (3\varepsilon)^4 & (3\varepsilon)^5 \\ 0 & 1 & 2(3\varepsilon) & 3(3\varepsilon)^2 & 4(3\varepsilon)^3 & 5(3\varepsilon)^4 \\ 0 & 0 & 2 & 6(3\varepsilon) & 12(3\varepsilon)^2 & 20(3\varepsilon)^3 \end{bmatrix}^{-1} \begin{bmatrix} f_0 \\ f_1 \\ f_2 \\ 0 \\ 0 \\ 0 \end{bmatrix}$$

che permette di trovare i parametri necessari.

Chiaramente f_0, f_1 ed f_2 dovranno assumere valori sufficientemente elevati da garantire le proprietà di *avoidance* di cui si è discusso. Un esempio del risultato che si ottiene è dato dalla figura seguente:

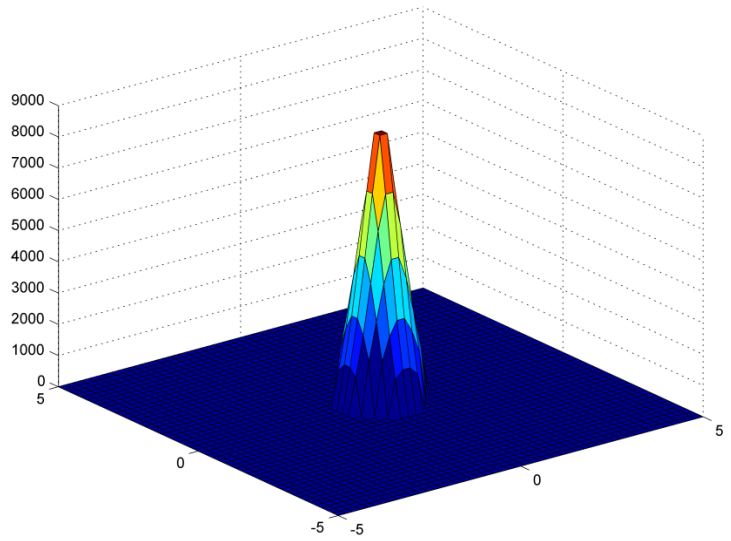

Fig. 4.4.5 *– Esempio di potenziale repulsivo polinomiale.*

Detto ciò, per ottenere un'opportuna funzione di navigazione non dobbiamo far altro che calcolare il gradiente di tale potenziale cambiato di segno ed imporre che esso coincida con la forza che guida il robot lontano dagli ostacoli $F_{rep_i}(q) = -\nabla U_{rep_i}(q)$.

L'approccio al *path planning* basato sulla definizione di un campo potenziale artificiale rappresenta dunque una tecnica semplice ed intuitiva che sfrutta un principio fisico di tipo energetico. Per un ambiente statico e completamente noto a priori, il potenziale può essere valutato offline producendo direttamente il profilo di velocità da applicare al robot, nell'ipotesi che possa essere considerato puntiforme; tale profilo è costruito per fare in modo che esso si muova lungo il campo energetico dal punto di partenza fino all'obiettivo. Tuttavia la tecnica può essere facilmente estesa ed applicata in versione online, così da poter tenere conto, oltre che delle informazioni sull'ambiente note a priori, anche della presenza di eventuali ostacoli incontrati lungo il percorso e rilevati dai sensori a bordo del robot mobile. Questa caratteristica particolarmente vantaggiosa consente di ottenere in sostanza quello che abbiamo chiamato *obstacle avoidance*; infatti è sufficiente determinare una volta per tutte il potenziale attrattivo dovuto all'obiettivo e quello repulsivo dovuto agli ostacoli noti, mentre si aggiorna ad ogni passo il potenziale repulsivo generato dagli ostacoli imprevisti e rilevati dai sensori di bordo durante il movimento.

In realtà, sebbene particolarmente semplici da implementare ed efficaci in buona parte delle situazioni di interesse pratico, i metodi basati sul potenziale artificiale, così come descritto nella sua forma generale, presentano diversi svantaggi; innanzitutto sono sensibili alla presenza di minimi locali nella funzione costruita, che possono ad esempio

comparire a causa della simmetria dell'ambiente, oppure della presenza di ostacoli di forma concava, inoltre nell'attraversamento di corridoi, l'utilizzo della sola ultima informazione proveniente dai sensori può portare ad un moto oscillatorio del robot. In particolare se si pensa ad un ostacolo concavo si può immaginare il caso in cui in una particolare posizione q^* in cui la forza attrattiva verso l'obiettivo risulta simmetrica alla forza repulsiva dovuta alle facce dell'ostacolo, dando vita ad un minimo locale che attira il robot lasciandolo in stallo. Allo stesso modo un minimo si può avere anche in assenza di ostacoli concavi se si ha la particolare situazione di un ambiente perfettamente simmetrico, sull'asse del quale il potenziale attrattivo e repulsivo potrebbe risultare equilibrato e non consentire il proseguimento, annullando il suo gradiente.

Per far fronte a tali problemi sono state sviluppate svariate soluzioni negli anni, con l'obiettivo di migliorare la strategia originale agendo sulla definizione del campo repulsivo. Una prima interessante modifica è quella di considerare la forza repulsiva generata dall'ostacolo non solamente in funzione della distanza del robot da esso, ma anche in funzione dell'orientazione relativa che esiste tra i due all'istante considerato; in questo modo il potenziale non resta più fissato punto per punto, ma viene legato al movimento del robot. Anche intuitivamente tale variazione risulta ragionevole dal momento che l'urgenza di evitare un ostacolo parallelo al moto del robot deve essere ovviamente inferiore di quella che si genera quando il robot si sta muovendo direttamente verso di esso. Una seconda possibile estensione per migliorare la forma del potenziale repulsivo è quella di evitare di considerare ostacoli che non andranno ad influire sulla velocità di movimento del robot; ad esempio ostacoli dietro al robot sono ininfluenti quando esso si sta muovendo in avanti.

La tecnica presentata risulta particolarmente interessante per quanto riguarda il problema di navigazione studiato in questa tesi. Il controllo ottimo che si è pensato di utilizzare, infatti, è già di per sé basato sull'idea di minimizzare una cifra di merito che pesi stato e vettore degli ingressi e dunque l'inserimento di un termine che costituisce il potenziale artificiale risulta assolutamente naturale. In particolare quel che si può fare è allora vedere gli ostacoli come dei vincoli di tipo *soft*, ovvero descriverli attraverso componenti penalizzanti all'interno della cifra di merito; in questo modo il processo di ottimizzazione farà in modo di risolvere il problema di controllo, mantenendosi lontano dai vincoli. A tal proposito, dunque, il potenziale repulsivo discusso finora appare naturalmente come una funzione di penalizzazione perfetta per raggiungere il risultato desiderato; se si fa in modo di rendere quasi infinito il suo valore nei pressi degli ostacoli reali, l'algoritmo di soluzione *MPC* garantirà implicitamente l'assenza di collisioni. Per quanto riguarda il concetto di potenziale attrattivo, invece, è sufficiente notare che il problema di controllo ottimo consente per definizione di portare il sistema verso un equilibrio, che può essere fatto coincidere, senza perdita di generalità, con l'obiettivo desiderato; pertanto senza alcun accorgimento particolare otteniamo con questo approccio il completamento del *task* in sicurezza.

4.5 - Vector Field Histogram.

Il *"vector field histogram"* è un metodo di *obstacle avoidance* real time proposto nel 1991 da Johann Borenstein e Yoram Koren in [10] che permette simultaneamente di rilevare la presenza di ostacoli ignoti, evitare possibili collisioni e di guidare il robot verso il completamento del *task*, esattamente come accade nella tecnica vista in precedenza. Tale soluzione si sviluppa a partire dall'idea di generare una griglia nello spazio cartesiano a due dimensioni e di considerarla come modello del mondo in cui vive il robot. Tale modello, inizialmente vuoto, viene continuamente popolato durante il movimento utilizzando le letture ottenute dai sensori di bordo, che registrano la presenza di ostacoli nelle vicinanze. Contemporaneamente all'aggiornamento di tale mondo la tecnica prevede una fase di sintesi delle informazioni raccolte per poter arrivare alla definizione degli ingressi di controllo necessari al robot. Più nel dettaglio, una finestra della mappa, valutata attorno alla posizione attuale del robot, viene in primo luogo convertita in un istogramma in coordinate polari contenente informazioni sull'occupazione spaziale distribuite a trecentosessanta gradi attorno ad esso. In particolare, la nuova mappa sarà composta da un certo numero di settori angolari di ampiezza finita, ciascuno dei quali contenente un valore indice della densità polare degli ostacoli nella direzione considerata.

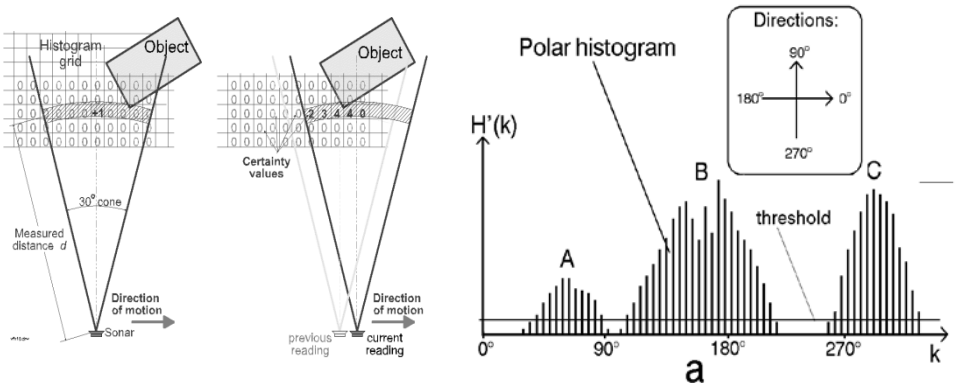

Fig. 4.5.1 *– VFH. Grid map a sinistra, polar histogram a destra.*

Nella fase di controllo vera e propria l'algoritmo non deve far altro che scegliere il settore più vantaggioso, tra tutti quelli definiti in precedenza, sulla base della più bassa densità di ostacoli rilevata ed infine sterzare il robot in tale direzione, scegliendo poi come velocità di movimento lineare un valore costante o eventualmente funzione della densità nei settori adiacenti.

Sostanzialmente, dunque, la tecnica *VFH* può essere implementata in tre distinti passi. Nel primo passo viene costruita una mappatura in un sistema di coordinate cartesiane dello spazio attorno al robot all'interno della quale vengono inserite tutte le

informazioni accumulate dai sensori di prossimità; in particolare lo spazio viene suddiviso in celle a ciascuna delle quali viene associato un valore di probabilità di occupazione aggiornato di volta in volta dai nuovi dati disponibili. Nel secondo passo la mappa in due dimensioni deve essere ridotta ad una struttura in coordinate polari in forma di istogramma; per preservare ed isolare le informazioni utili riguardanti gli ostacoli viene selezionata una finestra attiva di dimensioni prefissate attorno alla posizione attuale del robot, così da avere una visione corretta dell'ambiente circostante. Tale finestra viene così mappata in una struttura monodimensionale, nota come istogramma polare, composta da n settori angolari di ampiezza prefissata. In questo modo per ogni settore si ottiene in uscita dall'istogramma la probabilità di occupazione derivante dalle letture dei sensori. Infine il terzo passo della procedura *VFH* non fa altro che determinare la direzione di sterzo richiesta al robot, corrispondente ad uno dei settori maggiormente liberi dell'istogramma costruito, e simultaneamente adattare la velocità di movimento del robot in accordo alla densità degli ostacoli registrati nelle vicinanze.

Sfruttando tale costruzione un tipico istogramma polare è costituito da picchi, ovvero settori con una alta densità di occupazione, e valli, ovvero dei settori a bassa densità di ostacoli. Per valutare la direzione di sterzo opportuna verso l'obiettivo, vengono innanzitutto identificati come candidati tutti i settori corrispondenti a valli sufficientemente ampie per la larghezza del veicolo (con una certa soglia dipendente dalla sua cinematica); tra questi viene poi scelto quello che minimizza una determinata funzione di costo che tiene conto ad esempio dell'allineamento del robot rispetto al *target*, della differenza tra la direzione corrente e la direzione dell'obiettivo oppure della differenza tra la direzione scelta in precedenza per il robot e quella che si deve scegliere.

La tecnica *vector field histogram* pone rimedio ad alcune limitazioni dei metodi basati sulla definizione di un campo potenziale. Una prima considerazione, di natura puramente pratica, riguarda il fatto che nel *VFH* l'influenza di misure disturbate dai sensori viene minimizzata dal momento che la costruzione della mappa è in realtà una media di tutte le informazioni raccolte durante la navigazione; questo accorgimento riduce ad esempio l'oscillazione che si registra nel movimento lungo corridoi stretti utilizzando il metodo del potenziale artificiale. In tal caso, infatti, la presenza di informazioni cumulate fa si che la mappa vari poco da lettura a lettura, e che pertanto la direzione di viaggio resti ben definita ed in un certo senso meno dipendente dalla rilevazione attuale.

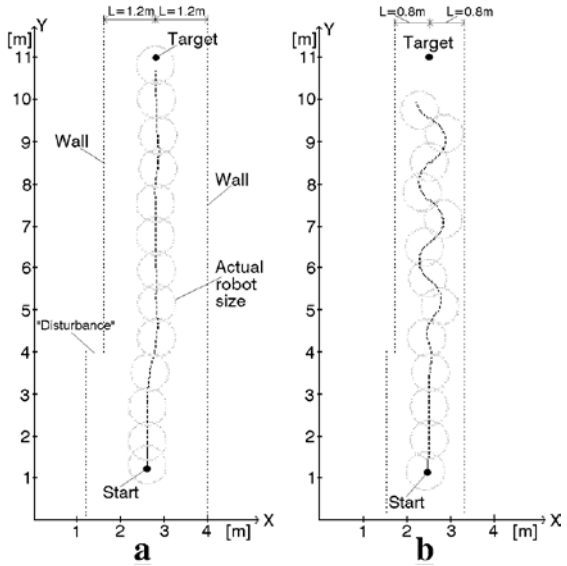

Fig. 4.5.2 *–VFH e potenziale artificiale. Attraversamento del corridoio.*

Inoltre l'aspetto più importante è dato dal fatto che, in tale soluzione, non sono state definite né forze repulsive né forze attrattive e pertanto il robot non può restare intrappolato in minimi locali, ma il completamento del task è garantito dalla scelta della velocità in accordo alla funzione di costo utilizzata.

Tale tecnica, per quanto interessante dal punto di vista teorico, e nonostante i vantaggi appena descritti, è stata tuttavia giudicata non utile ai fini del problema di navigazione considerato dal momento che si è scelto di non ricorrere ad un sistema di sensori di prossimità a bordo del robot. Inoltre, l'assenza di una vera e propria cifra di merito e la dipendenza diretta dalla direzione di movimento del robot rendono difficile l'integrazione con il metodo *MPC* che si è scelto di utilizzare.

4.6 - Modello del movimento umano.

Un'ulteriore interessante soluzione al problema di navigazione ed *obstacle avoidance* è quella che abbiamo incluso nella categoria di approcci *behavior based*, incentrata sull'analisi del comportamento umano di fronte a situazioni simili; in tale ambito il lavoro guida è quello di Schoner, Fajen e Warren [11]. Per quanto distante dall'implementazione che si intende effettuare, il lavoro è presentato poiché può essere facilmente interpretato come particolare applicazione del metodo del potenziale artificiale.

L'idea di base utilizzata è quella di costruire un modello del sistema al quale dare in ingresso la direzione di movimento attuale del robot ed informazioni riguardanti la posizione relativa dell'obiettivo e quella degli ostacoli, rappresentate in coordinate polari. A partire da questo, supponendo di muoversi a velocità costante, si può notare che, nella realtà, l'accelerazione angolare con cui un soggetto tende a ruotare aumenta con la distanza angolare dell'obiettivo, decresce con la sua distanza lineare ma nello stesso tempo decresce con la distanza angolare e lineare di eventuali ostacoli. Tenendo presente questi comportamenti, ed indicando con ϕ la direzione angolare, il modello adottato per descrivere formalmente tale legge di moto è il seguente:

$$\ddot{\phi} = -b\dot{\phi} - k_g(\phi - \psi_g)(e^{(-c_1 d_g)} + c_2) + \sum_i k_{oi}(\phi - \psi_{oi})e^{-c_3|\phi-\psi_{oi}|}e^{-c_4\, d_{oi}}$$

dove $b, k_g, c_1, c_2, c_3, c_4$ e k_{oi} sono opportuni coefficienti. Ancora $\dot{\phi}$ rappresenta la velocità angolare del soggetto, ψ_g e d_g sono rispettivamente la distanza angolare e lineare dell'obiettivo all'istante attuale mentre ψ_{oi} e d_{oi} costituiscono le equivalenti distanze riferite all'ostacolo *i-esimo*. Si nota che tale equazione è composta da tre termini; un termine che rappresenta una sorta di smorzamento, un termine legato all'obiettivo ed un termine dipendente invece dalla presenza di ostacoli.

In particolare, il termine smorzante oppone all'accelerazione angolare del soggetto la velocità con cui esso sta ruotando secondo un coefficiente b ed agisce riducendo eventuali oscillazioni. Il termine dovuto all'obiettivo spinge la direzione attuale del robot verso di esso, la sua forza è incrementata proporzionalmente con l'angolo e nello stesso tempo decresce esponenzialmente con la distanza lineare; come desiderato tale componente non va mai a zero a causa della distanza dall'ostacolo grazie alla costante c_2, assicurando che anche obiettivi lontani siano in grado di guidare il robot; il parametro k_g agisce da rigidezza modulando l'ampiezza di tale componente. Infine la componente dovuta agli ostacoli spinge il robot a puntare lontano da essi; la sua influenza cresce proporzionalmente all'angolo dell'ostacolo quando questo è piccolo, ma decresce esponenzialmente con esso quando questo diviene grande; ciò implica che quando il robot si gira verso l'ostacolo la sua repulsione cresce, ma solo fino ad una certa soglia limite; in aggiunta tale componente decresce esponenzialmente a zero man mano che la distanza dall'ostacolo cresce, così da fare in modo che il moto non sia

influenzato da ostacoli troppo distanti, anche se effettivamente sta puntando nella direzione di collisione futura.

In realtà abbiamo detto che tale modello del comportamento umano può essere interpretato come una particolare versione di potenziale artificiale che varia durante il movimento del robot in funzione della sua dinamica. In tal senso il campo complessivo può essere scritto in funzione non più della distanza, ma della direzione attuale come:

$$\varphi(\phi) = \varphi_g(\phi) + \sum_i \varphi_{oi}(\phi)$$

dopodiché il controllo effettivo del robot avviene secondo la legge, derivata dalla precedente

$$\ddot{\phi} = \frac{d\varphi}{d\phi} - b\dot{\phi}$$

che è del tutto equivalente a quella presentata all'inizio del paragrafo, con l'aggiunta del termine di smorzamento a ridurre le oscillazioni.

Tale campo potenziale differisce da quello tradizionale usato per la navigazione proprio perché l'orizzonte su cui viene costruito continua a variare assieme al movimento del robot stesso, e pertanto, anche con ostacoli fissati, non risulta definito a priori punto per punto, ma dipende in qualche modo dal suo comportamento.

La componente dovuta all'obiettivo della legge di controllo può essere espressa come funzione potenziale prendendo il suo integrale rispetto a ϕ :

$$\varphi_g = \frac{1}{2} k_g(\phi - \psi_g)^2 (e^{-c_1 d_g} + c_2)$$

quello che si ottiene è semplicemente un paraboloide centrato all'angolazione dell'obiettivo ψ_g. Di nuovo si nota che tale termine non è più un valore costante, ma varia assieme a ψ_g e d_g con il movimento del robot; la distanza dall'obiettivo agisce ancora da scalatura per ridurre l'effetto attrattivo quando il robot è vicino ad esso. Quando l'agente è allineato con l'obiettivo non vi è alcuna accelerazione angolare dovuta alla componente potenziale dell'obiettivo; questo corrisponde in realtà al fatto che la direzione di viaggio ha raggiunto un minimo della sua funzione potenziale; quando invece il robot non è diretto verso l'obiettivo l'accelerazione fa in modo che esso venga sterzato fino a far discendere nuovamente tale curva al suo minimo. Accanto a questo la presenza di un termine smorzante fa in modo che vengano limitate le oscillazioni attorno al punto di equilibrio, e dunque nel movimento in linea retta per il completamento del *task*. Graficamente otteniamo ad esempio un comportamento simile a quello mostrato in Figura 4.6.1:

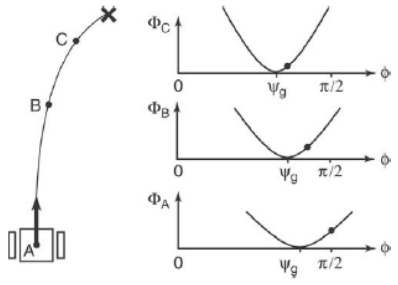

Fig. 4.6.1 *– Modello del movimento umano. Potenziale dell'obiettivo.*

La componente dovuta agli ostacoli può anch'essa essere espressa come una funzione potenziale costruita sulla direzione di viaggio del robot; in particolare prendendo ancora una volta il suo integrale rispetto a ϕ

$$\phi_{oi} = k_o \frac{c_3|\phi - \psi_{oi}| + 1}{c_3^2} e^{-c_3|\psi - \phi_{oi}|} e^{-c_4 d_{oi}}$$

Tale potenziale non è altro che un picco centrato in corrispondenza dell'angolazione dell'ostacolo ψ_{oi} ; la funzione tuttavia è realizzata in modo che la sua altezza decresca esponenzialmente con la distanza da esso per fare in modo che ostacoli distanti abbiano un effetto trascurabile sullo sterzo del robot. Vediamo un esempio in Figura 4.6.2.

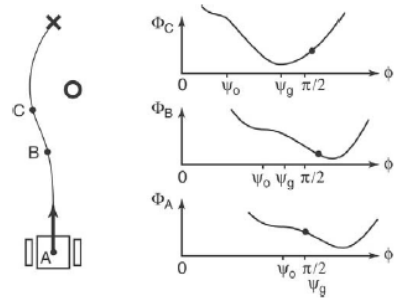

Fig. 4.6.2 *–Modello del movimento umano. Potenziale influenzato da un ostacolo.*

Di nuovo si noti che questo potenziale varia, a causa dei termini che lo compongono, con la dinamica del robot. Quando questo punta verso un ostacolo il valore della funzione potenziale fa si che esso acceleri per cambiare velocemente la sua direzione di viaggio mentre, una volta superato, ovvero al crescere del valore assoluto dell'angolo relativo che compare nell'espressione precedente, la sua influenza decade immediatamente. Va tenuto presente che, a differenza del metodo potenziale tradizionale, abbiamo ora un potenziale generato dall'ostacolo caratterizzato da un massimo finito, e non più illimitato; tale accorgimento è necessario per consentire al robot di passare da un lato all'altro dell'ostacolo se necessario.

Nella funzione potenziale complessiva, ottenuta unendo le componenti viste in precedenza, potrebbero ancora verificarsi dei minimi locali. Tuttavia questo non rappresenta più un problema come nel caso tradizionale. Infatti in quest'ultimo caso un minimo locale è una posizione in cui il robot, a causa di un gradiente nullo, è costretto ad annullare la propria velocità lineare e restare fermo, interrompendo il percorso verso l'obiettivo. Nel modello presentato, invece, il campo potenziale controlla solamente la direzione di sterzo mentre la velocità di traslazione viene supposta costante fino al completamento del *task*; in tal caso allora minimi locali rappresentano semplicemente particolari valori della direzione nei quali il robot viene attirato durante la manovra di aggiramento di un ostacolo. In generale quindi il robot nella sua navigazione seguirà la direzione di sterzo imposta dai minimi locali, ma questi convergeranno ad un unico minimo globale una volta che tutti gli ostacoli saranno superati e la via verso l'obiettivo resterà libera.

Un'estensione di tale tecnica può essere realizzata tenendo in conto anche l'ampiezza effettiva degli ostacoli e le dimensioni fisiche del robot; in tal caso inoltre, per evitare possibili collisioni nell'attraversamento di passaggi stretti, si può aggiungere un controllo della velocità lineare che dipenda in qualche modo dal valore del potenziale dovuto agli ostacoli, ovvero si può scegliere ad esempio:

$$v = \max\{v_{max}\, e^{-k_v \varphi_o} - \varepsilon\, , 0\}$$

prendendo un valore opportuno per la costante ε si può fare in modo che la velocità si annulli quando il potenziale generato dall'ostacolo, φ_o, è sufficientemente grande. In questo modo quando non vi sono ostacoli, il potenziale da essi generato è nullo e la velocità di movimento è pari a quella massima mentre quando vi sono ostacoli nelle vicinanze, oppure vicini all'allineamento con il robot in movimento, il potenziale che ne deriva è elevato e la velocità decresce.

In questo modo si può vedere allora che, fissata la velocità di rotazione $\dot{\phi}$, e detta v la velocità di traslazione, il robot viaggia con un raggio di curvatura $v/\dot{\phi}$; in tal senso allora se v decresce perché l'ostacolo è vicino, anche il raggio di curvatura decresce, producendo nella pratica una manovra di aggiramento dell'ostacolo maggiormente aggressiva, esattamente come ci si aspetterebbe in un comportamento naturale.

Anche tale tecnica viene per il momento scartata per quanto riguarda il problema di navigazione considerato dal momento che, in una prima fase, si è scelto di affrontare l'intero problema trascurando, per quanto possibile, la direzione di moto del robot e preoccupandosi piuttosto della sua posizione in coordinate cartesiane.

4.7 – Conclusioni.

In sintesi, dopo aver riportato le tecniche più diffuse in letteratura per risolvere il problema di *obstacle avoidance*, si è visto che il metodo che maggiormente si presta ad una implementazione pratica nel contesto predittivo è quello dei potenziali artificiali. In tal caso infatti la navigazione priva di collisioni del robot è garantita, almeno nel caso ideale, dal ricorso al gradiente di una funzione potenziale costruita sull'obiettivo e sugli ostacoli. Questo può dunque essere visto come problema di minimizzazione di una cifra di merito e fatto rientrare agevolmente nella procedura di ottimizzazione che risolve il problema *MPC.* Più nel dettaglio, si può fare in modo che l'avvicinamento agli ostacoli generi un aumento significativo della cifra di costo, garantendo così che la sua minimizzazione produca un movimento privo di collisioni; in realtà questo corrisponde ad inserire nel problema di ottimizzazione dei vincoli sotto forma di elementi penalizzanti e non come vere e proprie limitazioni fisiche sul sistema.

5 - CONTROLLO MPC DI UN EPUCK

5.1-Introduzione.

Negli ultimi decenni il controllo ottimo di tipo *model predictive control* è diventato una delle metodologie più utilizzate per la soluzione di problemi di controllo multivariabile, sia nell'ambito teorico, sia nelle applicazioni pratiche.

Una delle metodologie di controllo che è capace di gestire dei vincoli rigidi sul sistema in modo non conservativo è il controllo *MPC*. Questo consiste in una strategia di controllo basata sulla soluzione online, a ciascun istante di campionamento, di un problema di ottimizzazione matematica costruito sul modello dinamico del sistema da controllare. In tale problema di ottimizzazione, l'evoluzione predetta del sistema lungo un orizzonte prefissato è ottimizzata rispetto ad una data funzione di costo; il risultato del problema di ottimizzazione è una sequenza di controllo ottima per gli istanti successivi, della quale solo il primo elemento viene applicato realmente al sistema controllato. All'istante di campionamento successivo l'orizzonte predittivo è fatto scorrere in avanti, ed il problema di controllo ottimo su orizzonte finito è risolto nuovamente sulla base di una nuova misura dello stato reale. Tale tecnica, denominata *Receding Horizon Control*, è descritta in modo approfondito in [12] ed attualmente viene considerata implicitamente quando si parla di controllo *MPC*.

Il concetto di controllo *Receding Horizon* per un sistema *SISO* può essere compreso facilmente se si guarda allo schema di Figura 5.1.1.

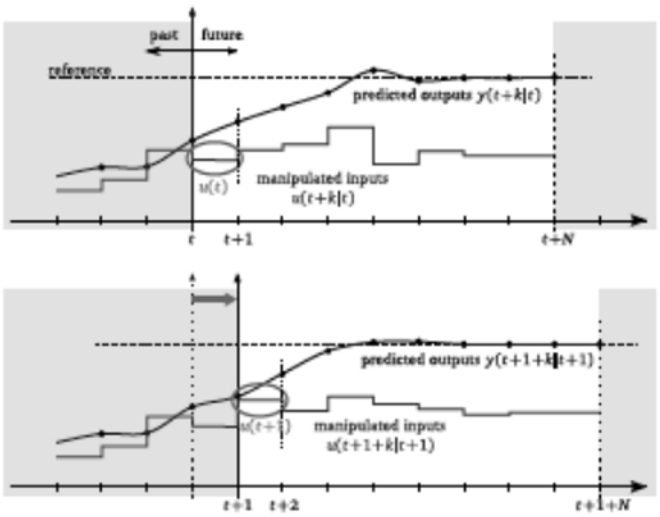

Fig. 5.1.1 *– MPC con Receding Horizon.*

All'istante t il problema di controllo ottimo viene risolto sullo stato iniziale $x(t)$ lungo un orizzonte predittivo di lunghezza N fissata. Dalla sequenza di ingressi ottimi prodotta $U_t = \{u(t) \ \dots \ u(t+N-1)\}$ il primo elemento viene applicato al sistema reale per la durata del periodo di campionamento.

All'istante successivo $(t+1)$ l'orizzonte predittivo è spostato in avanti di un passo, una nuova misura dello stato $x(t+1)$ è disponibile, ed il problema di controllo ottimo è risolto nuovamente con le informazioni aggiornate.

La nuova sequenza di controllo sarà generalmente diversa dalla sequenza di controllo precedente. Per questo motivo, facendo scorrere l'orizzonte predittivo ed utilizzando le nuove misure dello stato del sistema, si ottiene un certo livello di robustezza contro gli errori di modellazione ed i disturbi. In altre parole *RHC* introduce un meccanismo di *feedback* nel sistema, il cui controllo diventa quindi di tipo *closed-loop*.

Nel tempo MPC è diventato una metodologia di controllo molto diffusa nell'industria di processo, dove le dinamiche ed i tempi di campionamento sono relativamente lenti.

Il motivo della popolarità di *MPC,* sia nell'industria che nel mondo accademico, è semplice. Il controllo predittivo è in grado di garantire ottimalità, rispetto a certe misure di prestazione, rispettando contemporaneamente vincoli rigidi sul valore assunto dagli stati del sistema e sulle variabili di controllo. Una delle principali limitazioni di *MPC* è sempre stata, tuttavia, la sua sostanziale complessità computazionale, soprattutto se confrontata a quella di controllori di tipo classico.

Nell'ultimo decennio *MPC* è diventato una soluzione di interesse per un più ampio raggio di applicazioni. Vi sono in particolare tre fattori importanti che hanno contribuito a questo sviluppo. Il primo di questi è il notevole progresso tecnologico, che consente lo sviluppo di processori sempre più veloci, dal costo ridotto, miniaturizzati ed efficienti dal punto di vista del consumo energetico. Il secondo importante fattore è lo sviluppo di algoritmi di ottimizzazione più potenti ed affidabili, che ampliano notevolmente le possibilità di controllo di *MPC*. Infine, come ultimo fattore, è bene ricordare anche i progressi nella teoria stessa del controllo predittivo; in particolare sono stati sviluppati risultati in merito alla stabilità ed alla applicabilità della tecnica che possono essere approfonditi in [13].

5.2 – La tecnica MPC.

A questo punto consideriamo più da vicino la tecnica *MPC.* Come suggerisce il nome, il controllo predittivo *model based* è realizzato a partire da un modello dinamico del sistema da controllare. Nell'ambito del controllo di sistemi lineari a tempo discreto si fa riferimento di solito alla forma classica:

$$\begin{cases} x(t+1) = Ax(t) + Bu(t) \\ y(t) = Cx(t) \end{cases}$$

in cui $x(t) \in R^n, u(t) \in R^m$ ed $y(t) \in R^p$ sono rispettivamente lo stato del sistema, l'azione di controllo, e l'uscita all'istante di tempo t. Le matrici A, B, C sono la matrice di sistema, la matrice di ingresso, e la matrice di uscita. Tale sistema è ipotizzato essere soggetto a vincoli sui valori ammissibili di stati ed ingressi; in particolare $x(t) \in \mathbb{X}$ ed $u(t) \in \mathbb{U}$, in cui i due *set* $\mathbb{X}$ ed $\mathbb{U}$ si considerano, senza perdita di generalità, convessi e contenenti l'origine.

Supponiamo ora di considerare il solo problema di regolazione. In tal caso l'obiettivo è quello di guidare in modo ottimo lo stato del sistema verso l'origine, soddisfacendo, lungo tutto l'intervallo, i vincoli dati. Affinché questo sia possibile, è necessario prima di tutto verificare che la coppia (A, B) sia controllabile.

Nella tecnica viene implementata la strategia *Receding Horizon* descritta in precedenza. Dunque ad ogni istante di tempo deve essere risolto un problema di controllo ottimo su orizzonte finito. La funzione di costo V^N che caratterizza tale problema di ottimizzazione è definita genericamente come

$$V^N(x(t), U_t) = \sum_{k=t}^{t+N-1} l\big(x(k), u(k)\big) + V^f\big(x(t+N)\big)$$

dove U_t indica la sequenza di ingressi lungo l'orizzonte predittivo $\{u(t) \dots u(t+N-1)\}$.

Nell'espressione di V^N il termine N è l'orizzonte di predizione e gli elementi del vettore di stato soddisfano l'equazione dinamica $x(t+1) = Ax(t) + Bu(t)$. La funzione di costo $l\big(x(k), u(k)\big)$ è una funzione definita positiva dello stato e della variabile di controllo; solitamente la scelta ricade sulla funzione quadratica

$$l\big(x(k), u(k)\big) = \|x(k)\|_Q^2 + \|u(k)\|_R^2 = x(k)^T Q\, x(k) + u(k)^T R\, u(k)$$

Le due componenti rappresentano il quadrato della norma euclidea di ciascun termine, pesata secondo due matrici definite positive, Q ed R, applicate rispettivamente su stato e controllo. In modo simile anche la funzione di costo terminale V^f è una funzione definita positiva. Per questa ragione, e come vedremo meglio in seguito, questa è scelta solitamente nella forma

$$V^f\big(x(t+N)\big) = \|x(t+N)\|_P^2 = x(t+N)^T P\, x(t+N)$$

con una matrice di peso dello stato finale P anch'essa definita positiva. Tale funzione è legata al concetto di legge ausiliaria, ovvero una legge stabilizzante di tipo $u(k) = Kx(k)$ che, vedremo in seguito, è necessario definire per specificare correttamente il problema *MPC.*

In aggiunta ai vincoli esterni, ovvero all'appartenenza di stato e variabile di controllo ai relativi insiemi di ammissibilità $\mathbb{X}$ ed $\mathbb{U}$, è necessario aggiungere il cosiddetto vincolo terminale nella forma

$$x(t+N) \in \mathbb{X}^f \subseteq \mathbb{X}$$

ovvero richiedere che lo stato finale $x(t+N)$, ricavato dall'ottimizzatore, rientri all'interno di un determinato set $\mathbb{X}^f$ anch'esso centrato nell'origine.

In realtà il ricorso a cifre di costo quadratiche, come quelle viste, nella formulazione del problema di ottimizzazione, non è strettamente necessario. L'uso di norme politopiche, ad esempio, è vantaggioso da un punto di vista computazionale dal momento che porta ad un problema di programmazione lineare *(LP)*, per cui esistono solutori efficienti e quindi computazionalmente poco dispendiosi. Tuttavia questo tipo di funzione di costo porta generalmente ad un comportamento peggiore dell'anello chiuso, se comparato al caso precedente, e pertanto viene evitato; ulteriori approfondimenti si possono trovare in [14].

La sequenza di valori ottimi per la variabile di controllo è quindi ottenuta come argomento della minimizzazione della funzione di costo in presenza dei vincoli:

$$U_t^* = \{u(t|t), \dots u(t+N-1|t)\} = \arg\min_{U_t}\left(V^N(x(t), U_t)\,, \begin{cases} x(k) \in \mathbb{X} \\ u(k) \in \mathbb{U} \\ x(t+N) \in \mathbb{X}^f \end{cases}\right)$$

Tale soluzione deve essere calcolata ad ogni istante di tempo, dopodiché il primo elemento della sequenza, $u(t|t)$, viene applicato al sistema reale e gli altri scartati. L'esecuzione iterativa della misura dello stato attuale, dell'ottimizzazione sulla base delle informazioni aggiornate, e dell'applicazione della variabile di controllo predetta al sistema, può essere interpretata infine come una legge di controllo implicita e tempo invariante, ovvero riassunta nella forma

$$k_N(x(t)) = u(t|t)$$

Applicando questa la dinamica dell'anello chiuso diventa:

$$x(t+1) = Ax(t) + B\, k_N(x(t))$$

Quello che ci si può chiedere è allora se il sistema ottenuto sia stabile, e se, al passo successivo, il problema di controllo predittivo sia ancora ben posto.

Per poter discutere delle proprietà del controllo *MPC* è necessario introdurre la nozione di *set* positivamente invariante che troviamo descritta in [15]. In particolare un set Ω è detto positivamente invariante *(PI)* per il sistema autonomo $x(t+1) = f\big(x(t)\big)$ se, per ogni stato iniziale contenuto in Ω, la soluzione $x(t)$ resta confinata in Ω per tutti i successivi istanti di tempo. Per estensione, se consideriamo il controllore *state feedback* $k\big(x(t)\big)$, abbiamo che il set Ω è positivamente invariante per il sistema in anello chiuso se, per ciascuno stato $x(t) \in \Omega$ vale la relazione $Ax(t) + Bk\big(x(t)\big) \in \Omega$, che garantisce che al passo successivo lo stato sarà ancora confinato nel *set*.

5.2.1 – Condizioni per la stabilità.

La stabilità di *MPC* può essere dimostrata utilizzando il criterio di Lyapunov e sfruttando la cifra di costo ottima V^{N^*}, prodotta dall'ottimizzazione, come *Lyapunov function*. In particolare si ottengono delle condizioni sul set terminale $\mathbb{X}^f$ e sulla funzione di peso finale $V^f\big(x(t+N)\big)$, che garantiscono quanto desiderato, e che sono spiegate nel dettaglio in [13]. Ci possiamo limitare allora ad elencarne i punti fondamentali.

Innanzitutto tutti gli stati all'interno del set terminale devono soddisfare i vincoli di ammissibilità sul valore della variabile di stato, ovvero $\mathbb{X}^f$ deve essere un insieme chiuso, contenente l'origine e tale che $\mathbb{X}^f \subseteq \mathbb{X}$. Allo stesso modo anche i vincoli posti sulla variabile di controllo devono essere soddisfatti se il sistema è controllato attraverso la legge ausiliaria, ovvero l'ingresso calcolato dalla legge ausiliaria locale, $k^f(x)$, deve risiedere in $\mathbb{U}$ per ogni $x \in \mathbb{X}^f$. In secondo luogo il set terminale $\mathbb{X}^f$ deve essere positivamente invariante rispetto alla legge ausiliaria $k^f(x)$, ovvero in altre parole

$$x(k+1) = Ax + Bk^f\big(x(k)\big) \in \mathbb{X}^f \quad \forall x(k) \in \mathbb{X}^f$$

Infine, la funzione di costo finale V^f deve essere una *Lyapunov function* per il sistema controllato attraverso la legge ausiliaria, ovvero deve essere scelta in modo che valga la relazione

$$V^f\big(Ax(k) + Bk^f\big(x(k)\big)\big) \leq V^f\big(x(k)\big) - l\big(x(k), k^f\big(x(k)\big)\big) \qquad \forall\, x(k) \in \mathbb{X}^f$$

Le prime due assunzioni garantiscono l'ammissibilità di tutti gli stati all'interno del set terminale e, in corrispondenza di questi, di tutte le azioni di controllo generate dalla legge ausiliaria. La terza assunzione garantisce il rispetto persistente, ovvero valido ad ogni passo, dei vincoli sugli stati e sugli ingressi di controllo all'interno dell'orizzonte predittivo di N passi. L'ultima ipotesi, infine, è quella che garantisce la stabilità richiedendo che il costo terminale lungo la traiettoria del sistema in anello chiuso dalla legge ausiliaria sia non crescente. Per una dimostrazione più dettagliata di tale risultato si rimanda alla letteratura.

5.2.2 – Scelte progettuali.

Dunque la stabilità del sistema e le proprietà di *MPC* risiedono sostanzialmente nella scelta della funzione di costo finale V^f che deve rispettare la condizione vista. In particolare possono essere adottate due strategie in merito al comportamento tenuto nei confronti dello stato finale. La prima di queste consiste semplicemente nel supporre che la legge di controllo ausiliaria sia di tipo $u(k) = 0$. Chiaramente, affinché questo sia possibile, è necessario che il sistema di partenza sia stabile, ovvero che la matrice $(A + BK)$ abbia autovalori nel cerchio di raggio unitario con $K = 0$. La seconda soluzione è quella che fa riferimento all'utilizzo di una legge di controllo ausiliaria stabilizzante nella forma generica vista $u(k) = Kx(k)$, con un guadagno scelto in modo che $(A + BK)$ sia asintoticamente stabile.

Richiamando il principio di ottimalità, tipico dei problemi di ottimo ad orizzonte infinito, si può allora pensare di scegliere $V^f = V_\infty^*$, cioè prendere il suo valore ottimo quando $N \to \infty$. Indipendentemente dalla scelta dell'orizzonte di predizione N, si ritornerebbe quindi ad avere la stessa proprietà di ottimalità del controllo su orizzonte infinito. D'altra parte, a causa dei vincoli sul problema, il valore V_∞^* è sconosciuto. Una idea comune in MPC è allora quella di scegliere V^f come se fosse il costo dell'equivalente problema su orizzonte infinito non vincolato, $\tilde{V}_\infty$:

$$\tilde{V}_\infty(x) = \min_U \sum_{k=0}^{\infty} l\big(x(k), u(k)\big) = \min_U \sum_{k=0}^{\infty} \|x(k)\|_Q^2 + \|u(k)\|_R^2$$

La soluzione di questo problema non è altro che il classico *LQR* a tempo discreto e, pertanto, il valore ottimo della cifra di merito può essere calcolato come:

$$\tilde{V}_\infty^*(x) = \|x\|_P^2 \; = \; x^T P \, x$$

in cui la matrice P rappresenta l'unica soluzione definita positiva dell'equazione algebrica di Riccati *(ARE)* a tempo discreto:

$$P = Q + A^T(P - PB(R + B^T PB)^{-1} \, B^T P)A$$

Inoltre sappiamo anche che il controllore ottimo che rende minima tale cifra è lineare ed è costituito da un guadagno K, indicato solitamente come guadagno di Kalman, che può essere calcolato semplicemente come:

$$K = -(R + B^T PB)^{-1} B^T PA$$

Una scelta pratica più semplice può essere quella di determinare innanzitutto K^f con un qualsiasi metodo di assegnamento degli autovalori, e poi calcolare P dall'equazione di Lyapunov discreta:

$$(A + BK)^T P(A + BK) - P = \; (Q + K^T RK)$$

Nel caso in cui la matrice A sia stabile, invece, la scelta più semplice è quella vista di porre $K = 0$ e di conseguenza l'equazione precedente si semplifica ed il calcolo di P si ottiene come:

$$A^T P A - P = -Q$$

Per poter ottenere una regione di attrazione il più grande possibile per il controllore *MPC*, $\mathbb{X}^f$ viene generalmente scelto come massimo invariante positivo *(MPI)* del sistema in anello chiuso con la legge ausiliaria $x(t+1) = (A + BK)x(t) \in \mathbb{X}$.

Per poter implementare nella pratica un controllore *MPC* è necessario risolvere ad ogni istante di campionamento il problema di ottimizzazione finora descritto. Se le funzioni di costo lungo l'orizzonte e quella terminale sono scelte secondo le linee guida viste, allora la funzione di costo risultante $V^N\big(x(t), u(t{:}\, t+N-1|t)\big)$ del problema è di tipo quadratico. Le matrici di peso Q ed R su stato e variabile di controllo sono date ed il peso terminale P e la legge ausiliaria sono ottenute facilmente risolvendo *offline* l'equazione algebrica di Riccati del problema *LQR* non vincolato associato.

L'unica questione ancora aperta riguarda solamente il calcolo del set terminale $\mathbb{X}^f$. Molto semplicemente, se la funzione terminale scelta è di tipo $V^f = \frac{1}{2}\|x\|_P^2$, si potrebbe pensare di prendere come set una sua curva di livello, ovvero scegliere un valore c tale per cui:

$$\mathbb{X}^f = \{x : \|x\|_P^2 \leq c\}$$

Tuttavia, nel caso in cui i set $\mathbb{X}, \mathbb{U}$ e $\mathbb{X}^F$ vengano scelti politopici, ovvero possano essere rappresentati come intersezione di un insieme finito di sottospazi, allora il problema di ottimizzazione risultante è ancora di tipo quadratico e pertanto relativamente semplice da risolvere.

Dunque, in letteratura, si assume generalmente che tali vincoli siano di tipo politopico; tale ipotesi deve essere inoltre fatta solamente sulla forma di $\mathbb{X}$ e di $\mathbb{U}$ dal momento che, data la natura lineare del sistema, anche il massimo invariante positivo con cui si sceglie $\mathbb{X}^f$ sarà politopico.

In questo per ciascun set poliedrico che descrive i vincoli di ammissibilità, $\mathbb{X}, \mathbb{U}$ e $\mathbb{X}^f$, si è utilizzata la rappresentazione matriciale, ovvero la definizione:

$$\mathbb{X} = \{x : H_x\, x \leq K_x\} \qquad \mathbb{U} = \{u : H_u\, u \leq K_u\} \qquad \mathbb{X}^f = \{x_f : H_f\, x_f \leq K_f\}$$

Ne risulta quindi che il tutto è un problema di programmazione quadratica *(QP)* che può essere risolto con un basso costo computazionale utilizzando i moderni algoritmi di ottimizzazione; questo aspetto consente quindi di applicare la tecnica anche a sistemi con tempi di campionamento ridotti, e pertanto sarà l'approccio adottato nel seguito.

Si noti che esistono efficienti algoritmi in grado di calcolare set *MPI* anche nel caso si richieda che $\mathbb{X}^f$ sia politopico.

5.3 – Implementazione MPC su Epuck per la stabilizzazione.

Dopo aver descritto la tecnica di controllo *MPC* torniamo a considerare il problema di controllo del robot Epuck, ed in particolare concentriamoci nuovamente sulla ricerca, tramite approccio predittivo questa volta, di una soluzione al problema di parcheggio.

Fino ad ora abbiamo visto come sia possibile descrivere il robot mobile come un modello ad uniciclo e come si possa costruire una legge linearizzante in grado di semplificare, all'esterno, il problema di controllo. Abbiamo visto poi come, in questa situazione, sia possibile scegliere gli ingressi di a_x ed a_y del sistema linearizzato a partire dai riferimenti di posizione x_r ed y_r e poi applicare il tutto tramite un *feedback* dinamico al sistema reale. Tale approccio costringe tuttavia a progettare un ulteriore sistema di controllo per la pianificazione della traiettoria da imporre al robot. In realtà sarebbe desiderabile unificare i due problemi e risolvere contemporaneamente il problema di stabilizzazione e quello di pianificazione, così da avere un sistema in grado di agire completamente in tempo reale senza bisogno di progettare alcun percorso, ma semplicemente avvicinandosi all'obiettivo.

Una possibile soluzione è allora quella di applicare direttamente un controllo di tipo *MPC* che, come abbiamo visto, ha il vantaggio di poter gestire esplicitamente dei vincoli operativi sul problema. Inoltre, alla luce dell'obiettivo finale di coordinamento di una flotta di robot, la soluzione predittiva consente di avere a disposizione il vettore delle azioni future lungo l'orizzonte scelto, che può essere utilizzato in un contesto di tipo distribuito.

In dettaglio torniamo quindi a considerare il sistema ad uniciclo ottenuto dall'applicazione a tempo discreto del *feedback* linearizzante descritto in precedenza; ricordiamo a tal proposito che la forma del sistema, applicando la discretizzazione di Eulero in avanti, è la seguente:

$$\begin{cases} \dot{x} = v\,cos\phi \\ \dot{y} = v\,sin\phi \\ \dot{\phi} = \omega \end{cases} \xrightarrow[a\ tempo\ discreto]{feedback\ linearizzante} \begin{cases} z_1^+ = z_1 + \tau z_3 \\ z_3^+ = z_3 + \tau a_x \\ z_2^+ = z_2 + \tau z_4 \\ z_4^+ = z_4 + \tau a_y \end{cases} \qquad z = \begin{bmatrix} x \\ v_x \\ y \\ v_y \end{bmatrix}$$

Dopo aver scelto opportunamente il tempo di campionamento τ si ottiene il sistema lineare in forma matriciale:

$$\xi(t+1) = \begin{bmatrix} 1 & \tau & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \tau \\ 0 & 0 & 0 & 1 \end{bmatrix} \xi(t) + \begin{bmatrix} 0 & 0 \\ \tau & 0 \\ 0 & 0 \\ 0 & \tau \end{bmatrix} u(t)$$

che come già notato è in forma diagonale a blocchi, ovvero costituita da due sottosistemi disaccoppiati che rappresentano rispettivamente la dinamica lungo l'asse delle ascisse e delle ordinate del riferimento cartesiano. Non solo, è immediato verificare che, su ogni sottosistema, la dinamica residua è quella di un doppio integratore, relativamente semplice da controllare. Sebbene, come già discusso, la tecnica di discretizzazione adottata sia stata poi abbandonata nel seguito, le considerazioni e le prove inerenti il controllore di cui si sta discutendo sono state fatte mantenendo tale rappresentazione, e pertanto vengono riportate come tali.

Ricordiamo anche la legge lineare sullo stato che abbiamo utilizzato inizialmente per verificare la validità dell'approccio; questa, infatti, tornerà utile come legge ausiliaria nel problema *MPC*.

In particolare abbiamo visto:

$$u(t) = \begin{bmatrix} -a_1 & -a_3 & 0 & 0 \\ 0 & 0 & -a_2 & -a_4 \end{bmatrix} \xi(t) = k_{aux}\, \xi(t)$$

in cui i parametri che compaiono nella matrice k_{aux} sono opportuni guadagni stabilizzanti.

Sinteticamente il problema di controllo *MPC* è formulato come ottimizzazione di una cifra di costo V^N su di un orizzonte finito di durata N; tale ottimizzazione ha come argomento in uscita il vettore contenente il valore delle variabili di controllo per i successivi N passi in grado di portare il sistema sulla traiettoria ottimale rispettando eventuali vincoli su stato o ingressi imposti in fase di progetto. A partire da questo, secondo il criterio *Receding Horizon (RH)* già discusso, si utilizza solo il primo dei campioni prodotti ed al passo successivo si ripete l'intera procedura. Nel caso in questione, essendo il sistema lineare, una cifra di merito vantaggiosa dal punto di vista computazionale è quella di tipo quadratico su stato e variabile di controllo, esattamente come abbiamo visto nel caso generale

$$V^N = \sum_{k=t}^{t+N-1} \|\xi(k)\|_Q^2 + \|u(k)\|_R^2 + \|\xi(t+N)\|_P^2$$

i pesi Q, R (e di conseguenza P) potranno essere scelti come diagonali a blocchi, proprio grazie al disaccoppiamento ottenuto sulle dinamiche del sistema.

Per quanto riguarda la scelta della matrice P, che pesa lo stato finale dell'ottimizzazione all'istante $t+N$, possiamo utilizzare semplicemente la legge ausiliaria proposta:

$$u(k) = K_{aux}\,\xi(k) \qquad \rightarrow \qquad \xi(k+1) = (A + BK_{aux})\,\xi(k)$$

e nell'ipotesi che la matrice della dinamica ottenuta sia stabile, utilizzare l'equazione seguente per il calcolo di P:

$$(A + BK_{aux})^T P (A + BK_{aux}) - P + (Q + K_{aux}^T\, R\, K_{aux}) = 0$$

Fatto questo il problema *MPC* può essere riformulato sfruttando il modello dinamico del sistema per arrivare ad una scrittura vettoriale della cifra di merito; infatti, noto lo stato attuale $\xi(t)$, possiamo scrivere la predizione lungo l'orizzonte agli istanti $(t+k)$ successivi in funzione della variabile di ottimizzazione $u(t\colon t+N-1)$, ottenendo:

$$\xi(t) = I\,\xi(t)$$

$$\xi(t+1) = A\,\xi(t) + B\,u(t)$$

$$\xi(t+2) = A\xi(t+1) - Bu(t+1) = A^2\xi(t) + AB\,u(t) + B\,u(t+1)$$

$$\xi(t+k) = A^k\xi(t) + A^{k-1}B\,u(t) + \cdots AB\,u(t+k-2) + Bu(t+k-1)$$

L'intera predizione lungo l'orizzonte $t \ldots t+N$ può allora essere riassunta al modo seguente:

$$\underbrace{\begin{bmatrix} \xi(t) \\ \xi(t+1) \\ \vdots \\ \xi(t+N) \end{bmatrix}}_{\Xi_t} = \underbrace{\begin{bmatrix} I \\ A \\ \vdots \\ A^N \end{bmatrix}}_{\mathcal{A}} \xi(t) + \underbrace{\begin{bmatrix} 0 & 0 & \cdots & 0 \\ B & 0 & \cdots & 0 \\ \vdots & \vdots & \cdots & \vdots \\ A^{N-1}B & A^{N-2}B & \cdots & B \end{bmatrix}}_{\mathcal{B}} \underbrace{\begin{bmatrix} u(t) \\ u(t+1) \\ \vdots \\ u(t+N-1) \end{bmatrix}}_{U_t}$$

Applicata al caso in questione, questa permette di riscrivere la cifra di merito in modo compatto facendo scomparire la sommatoria. Il termine Ξ_t sarà composto da $N+1$ elementi mentre il termine U_t, rispetto al quale si vuole ottimizzare, sarà un vettore di N componenti:

$$V^N = \|\Xi_t\|^2_{\begin{bmatrix} Q & \ddots \\ \ddots & P \end{bmatrix}} + \|U_t\|^2_{\begin{bmatrix} R & \ddots \\ \ddots & R \end{bmatrix}} = \|\mathcal{A}\xi(t) + \mathcal{B}U_t\|^2_{\mathcal{Q}} + \|U_t\|^2_{\mathcal{R}}$$

I due pesi su stato ed ingresso di controllo sono ora costituiti dalle matrici allargate:

$$\mathcal{Q} = \begin{bmatrix} Q & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & P \end{bmatrix} \qquad \mathcal{R} = \begin{bmatrix} R & 0 & 0 \\ 0 & \ddots & 0 \\ 0 & 0 & R \end{bmatrix}$$

entrambe sono per costruzione diagonali a blocchi e la prima contiene anche il peso sullo stato finale P.

In altre parole il problema di ottimizzazione diventa quello di calcolare il minimo rispetto al vettore $U_t = \{u(t) \dots u(t+N-1)\}$ della funzione quadratica vista.

$$\min_{U_t} \xi^T(t)\mathcal{A}^T\mathcal{Q}\mathcal{A}\xi(t) + 2\xi^T(t)\mathcal{A}^T\mathcal{B}U_t + U_t^T(\mathcal{B}^T\mathcal{Q}\mathcal{B} + \mathcal{R})U_t$$

Tale approccio permette, come abbiamo visto, di aggiungere al problema eventuali vincoli sul valore della variabile di controllo o delle variabili di stato come, nel caso considerato, possono essere i limiti dell'area di lavoro o la velocità massima di movimento del robot; accanto a questi è necessario imporre anche un vincolo terminale, ovvero richiedere che lo stato $\xi(t+N)$ appartenga ad un opportuno set Ξ^f invariante rispetto all'applicazione della legge ausiliaria; ad esempio:

$$\xi(t+N) \in \Xi^f \qquad \rightarrow \qquad \Xi^f = \{x: x^T P x \leq c\}$$

In questo modo siamo in grado di scrivere una legge di controllo tempo invariante che stabilizza il sistema, ovvero che porta il robot nell'origine, qualsiasi sia la sua posizione iniziale:

$$\xi(t) = [x \quad v\,cos\phi \quad y \quad v\,sin\phi] \qquad (noto)$$

$$\min_{U_t} V^N\big(\xi(t), u(t) \dots u(t+N-1)\big) \quad \rightarrow \quad U_t = \{u(t) \dots u(t+N-1)\} \rightarrow u(t|t) = \begin{bmatrix} a_x \\ a_y \end{bmatrix}$$

I valori degli ingressi ω e v sono poi calcolati secondo quanto discusso al Capitolo 3 ed applicati direttamente al robot dopo la conversione nelle due velocitàω_R ed ω_L. Particolare attenzione deve essere prestata all'integrazione numerica, come si è già evidenziato.

5.4 – MPC per obstacle avoidance.

A questo punto, dopo aver implementato un controllore *MPC* in grado di risolvere, in combinazione con il *feedback* linearizzante, il problema del parcheggio, prendiamo in considerazione il problema di *obstacle avoidance* con l'obiettivo di inserire nel controllore anche la capacità di produrre un moto privo di collisioni. Per semplificare il problema limitiamoci innanzitutto a considerare ostacoli fissi ed in posizione nota, trascurando il comportamento combinato di più robot che sarà trattato in seguito.

Abbiamo visto che, in letteratura, esistono svariate soluzioni al problema ed abbiamo brevemente sintetizzato i pregi ed i difetti di quelle principali nel capitolo precedente. Nel caso in esame, avendo scelto di utilizzare *MPC* come algoritmo per la soluzione del problema di parcheggio di ogni singolo robot, la tecnica di *obstacle avoidance* che meglio si presta è quella dei potenziali artificiali. In tal caso infatti il problema di navigazione è già di per sé formulato nei termini di ottimizzazione di una determinata funzione di costo, dunque risulta sufficiente aggiungere alla cifra di merito una penalità legata alla presenza di ostacoli lungo il percorso.

In particolare ricordiamo che, data la linearità del sistema in esame, la funzione a cui si è pensato di fare ricorso è composta da un termine quadratico nello stato e nel vettore di ingressi; a questa sovrapponiamo un termine penalizzante, che invece è di tipo non lineare. Si ottiene allora la seguente espressione all'istante di tempo attuale per J_t:

$$\sum_{k=t}^{t+N-1} \left(\|\xi(k)\|_Q^2 + \|u(k)\|_R^2 + \sum_{p=1}^{M} \begin{cases} \dfrac{1}{\left\| E\left(\xi(k) - \xi_{obst_p}\right)\right\|^2} - \dfrac{1}{d_{\lim}^2} & \left\| E\left(\xi(k) - \xi_{obst_p}\right)\right\| \le d_{\lim} \\ 0 & \left\| E\left(\xi(k) - \xi_{obst_p}\right)\right\| > d_{\lim} \end{cases} \right)$$

in cui N indica l'orizzonte di predizione prescelto mentre M il numero di ostacoli noti lungo il percorso. La matrice E,invece, è semplicemente una matrice di selezione che consente di estrarre dal vettore di stato, costituito rispettivamente da posizione e velocità lungo gli assi cartesiani, le sole componenti relative alla posizione, da confrontare con la quella dell'ostacolo:

$$\xi(k) = \begin{vmatrix} x \\ \dot{x} \\ y \\ \dot{y} \end{vmatrix} \quad \xi(k)_{obst_p} = \begin{vmatrix} x_o \\ 0 \\ y_o \\ 0 \end{vmatrix} \quad \rightarrow \quad E = \begin{vmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{vmatrix}$$

Utilizzando tale cifra di merito l'idea è quella di valutare per ogni stato predetto lungo l'orizzonte la distanza da ciascuno degli ostacoli; se questa risulta inferiore al valore di soglia d_{lim}, ovvero in altre parole se l'ostacolo influenza il moto del robot nella

posizione considerata, la cifra di merito complessiva viene aumentata di un valore sempre crescente man mano che tale distanza si riduce, secondo la filosofia del potenziale artificiale. La cifra di penalizzazione così costruita assume un valore illimitato in corrispondenza della condizione di collisione; ne deriva quindi in uscita dall'algoritmo un valore ottimo degli ingressi che mantiene l'intera traiettoria al di fuori del set di ostacoli, come desiderato.

Abbiamo già discusso gli svantaggi di una componente penalizzante scritta secondo tale funzione ed abbiamo visto che, nel seguito, sarà più utile ai fini pratici una funzione polinomiale costruita *ad hoc;* nuovamente, tuttavia, tale formulazione è stata mantenuta nel presente capitolo dal momento che le prove e le analisi iniziali sono state costruite su questa. La forma più corretta sarà poi utilizzata nel seguito per risolvere il problema di coordinamento.

Tale approccio, idealmente corretto, è tuttavia soggetto principalmente a due problemi di natura pratica. Il primo di questi è legato all'implementazione a tempo discreto del sistema da controllare; senza le dovute attenzioni si può infatti arrivare alla situazione in cui l'ostacolo non è sufficientemente grande per essere considerato dall'algoritmo. In particolare, può accadere che tra due istanti di tempo consecutivi la posizione sia incrementata di un valore superiore a due volte quello di soglia, rendendo di fatto l'ostacolo del tutto trasparente alla legge di controllo e portando il robot alla collisione. Il problema in tal caso risiede nella massima velocità di movimento consentita e nel tempo di campionamento scelto; basse velocità e piccoli valori del periodo di campionamento consentono di rimediare al problema, tuttavia vedremo in seguito che quest'ultimo parametro non può essere scelto liberamente. Ciò che è necessario fare è allora considerare un valore di soglia d_{lim} comparabile con le scelte fatte in precedenza, ovvero imporre una minima dimensione agli ostacoli che il sistema è in grado di evitare; in questo modo eventuali ostacoli più piccoli dovranno essere allargati di un valore di sicurezza sufficiente a scongiurare la situazione descritta.

Il secondo dei problemi che nasce dall'approccio proposto è legato alla complessità computazionale del problema di ottimizzazione. Sebbene la funzione di costo utilizzata per valutare l'influenza degli ostacoli sia relativamente semplice, il numero di questi può infatti essere abbastanza elevato; ciò si nota soprattutto se si pensa che, per descrivere un ostacolo reale di dimensioni finite nella filosofia del potenziale artificiale, è necessario immaginare che il suo perimetro sia composto da tanti piccoli ostacoli. Questo aspetto, unito al fatto che la valutazione della funzione di costo deve essere fatta lungo tutto l'orizzonte predittivo, costringe a mantenere limitata la dimensione dell'orizzonte stesso o ad incorrere in tempi di calcolo che possono divenire notevolmente elevati. D'altra parte la riduzione dell'orizzonte influenza notevolmente le prestazioni ottenute, dal momento che l'efficienza del sistema è data esattamente dalla capacità di tener conto anche dell'effetto degli ostacoli lontani sulle posizioni future, che conferisce una notevole "intelligenza" al comportamento del robot. Inoltre un'estensione dei tempi di calcolo per ogni singolo passo della legge di controllo

costringe a fare ricorso a dei tempi di campionamento elevati, andando ad aggravare la situazione descritta in precedenza. Per porre rimedio a questa eccessiva complessità è necessario allora prestare attenzione all'algoritmo utilizzato per l'ottimizzazione, ed in particolare fare in modo di rendere la soluzione del problema il più possibile agevolata.

5.5 – Risoluzione numerica degli algoritmi.

Il problema di ottimizzazione ottenuto nel precedente paragrafo risulta complessivamente non convesso, a causa del contributo dato dagli ostacoli, e pertanto richiede di ricorrere ad un algoritmo di minimizzazione ben più complesso di quello che potrebbe essere utilizzato nel caso quadratico di navigazione in ambiente libero. A tal proposito è stato scelto innanzitutto di testare in ambiente *MATLAB* la funzione *fminunc* che permette di risolvere il problema di ottimizzazione di cifre di costo non lineari in assenza di vincoli espliciti. Quest'ultima caratteristica non crea particolari problemi dal momento che la scelta di ricorrere al potenziale artificiale, per la gestione delle collisioni, ha permesso di considerare gli ostacoli come vincoli di tipo *soft*, inserendoli in modo implicito all'interno della funzione di merito.

Utilizzando tale algoritmo con un orizzonte di predizione sufficientemente lungo da garantire buone prestazioni, si ottiene il risultato cercato, infatti in uno dei casi di test utilizzati abbiamo la traiettoria mostrata in Figura 5.5.1.

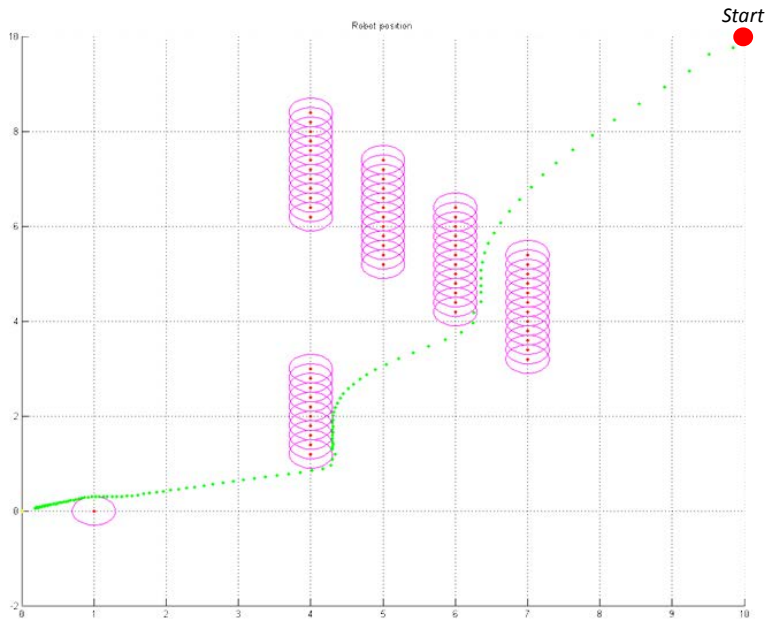

Fig. 5.5.1 *–Obstacle avoidance MPC. Prova con fminunc*

Tuttavia, come ci si aspettava, i tempi di calcolo necessari a completare un singolo ciclo di ottimizzazione risultano complessivamente molto elevati, e diventano praticamente insostenibili nei pressi degli ostacoli, lasciando il tutto valido solo per scopi di simulazione.

Per far fronte a tale problema si può tener presente che un algoritmo di ottimizzazione come quello considerato non fa altro che cercare ad ogni passo la direzione di minima discesa lungo la funzione di costo; tale direzione, corrispondente di fatto al gradiente della funzione rispetto alle variabili di ingresso, viene di norma valutata numericamente a partire dalla cifra di merito. In particolare allora si può pensare di calcolare in forma chiusa l'espressione di tale gradiente e fornirla in ingresso all'algoritmo in modo da semplificare buona parte del lavoro; il calcolo numerico viene infatti sostituito da una semplice valutazione della funzione, riducendo il carico computazionale e di conseguenza anche i tempi di elaborazione.

Nel caso in esame l'espressione finale della cifra di costo, ricavata precedentemente, si presta bene a tale tipo di calcolo. Si ricorda che tale cifra risulta

$$V_t^N = \|\mathcal{A}\,\xi(t) + \mathcal{B}\,U_t\|_Q^2 + \|U_t\|_{\mathcal{R}}^2 + V_{obst}^N$$

con il termine V_{obst}^N definito come:

$$\sum_{k=0}^{N-1}\sum_{p=1}^{M}\begin{cases}\dfrac{1}{\left\|E\left(\xi(t+k)-\xi_{obst_p}\right)\right\|^2}-\dfrac{1}{d_{\lim}^2} & \left\|E\left(\xi(t+k)-x_{obst_p}\right)\right\| \le d_{\lim}\\ 0 & \left\|E\left(\xi(t+k)-x_{obst_p}\right)\right\| > d_{\lim}\end{cases}$$

Per quanto riguarda la componente penalizzante dovuta agli ostacoli, possiamo esprimere nuovamente il termine ξ_{t+k} utilizzando la predizione; in particolare si può notare che al passo k, selezionando il *k-esimo* blocco riga delle matrici $\mathcal{A}$ e $\mathcal{B}$, si può scrivere

$$\xi(t+k) = \mathcal{A}_k\,\xi(t) + \mathcal{B}_k\,U_t$$

dove

$$\mathcal{A} = \begin{bmatrix}\mathcal{A}_0\\ \vdots\\ \mathcal{A}_N\end{bmatrix} \qquad\qquad \mathcal{B} = \begin{bmatrix}\mathcal{B}_0\\ \vdots\\ \mathcal{B}_N\end{bmatrix}$$

sono le matrici allargate del problema in forma vettoriale.

Possiamo suddividere il gradiente nelle due componenti legate rispettivamente al contributo di stato ed ingresso ed al contributo degli ostacoli. La prima di queste si ottiene facilmente ricordando la forma quadratica della funzione scelta:

$$V^N = \|\mathcal{A}\,\xi(t) + \mathcal{B}\,U_t\|_Q^2 + \|U_t\|_{\mathcal{R}}^2 = (\mathcal{A}\,\xi(t) + \mathcal{B}\,U_t)^T\,Q_c\,(\mathcal{A}\,\xi(t) + \mathcal{B}\,U_t) + U^T\,\mathcal{R}\,U$$

derivando rispetto all'intero vettore di ingressi U si ricava:

$$\frac{\partial V}{\partial U} = 2\,(\mathcal{A}\,\xi(t) + \mathcal{B}\,U_t)^T\,Q_c\,\mathcal{B} + 2\,U^T\mathcal{R}$$

Per quanto riguarda la componente dovuta agli ostacoli supponiamo innanzitutto per semplicità di avere un solo contributo da valutare, la funzione di costo risulta allora semplicemente:

$$V_2 = \sum_{k=0}^{N-1} \frac{1}{\|E\,(\xi(t+k) - \xi_{obst})\|^2} - \frac{1}{d_{\lim}^2}$$
$$= \sum_{k=0}^{N-1} \frac{1}{\|E\,(\mathcal{A}_k\xi(t) + \mathcal{B}_k U_t - \xi_{obst})\|^2} - \frac{1}{d_{\lim}^2}$$

la sua derivata rispetto al vettore degli ingressi U_t è quindi:

$$\frac{\partial V_2}{\partial U_t} = \sum_{k=0}^{N-1} \frac{-2}{\|E\,(\xi(t+k) - \xi_{obst})\|^4}\,(\mathcal{A}_k\xi(t) + \mathcal{B}_k U_t - \xi_{obst})^T\,E^T\,E\,\mathcal{B}_k$$

Per considerare il contributo complessivo di tutti gli ostacoli è allora sufficiente ad ogni passo predittivo, all'interno dell'orizzonte, valutare la condizione sulla distanza del punto $\xi(t+k)$ dall'ostacolo ξ_{obst} e ripetere l'espressione precedente per ognuno di quelli che risultano influenti ai fini del moto.

Utilizzando anche il gradiente appena calcolato nella funzione *fminunc* tuttavia, otteniamo ancora risultati non sufficienti a consentire un vero e proprio controllo online del robot, anche se si registra una evidente riduzione dei tempi di calcolo. In realtà, con lo stesso ragionamento utilizzato in precedenza, si può pensare di ottenere un miglioramento ulteriore fornendo all'algoritmo anche la forma esplicita della matrice Hessiana, ottenuta derivando rispetto al vettore di ingressi U_t il gradiente appena calcolato.

Il calcolo pratico dell'hessiano può essere fatto derivando nuovamente rispetto ad U_t il gradiente trasposto. In particolare per quanto riguarda la componente dovuta allo stato ed all'ingresso otteniamo semplicemente:

$$\frac{\partial^2 V}{\partial U_t^2} = 2\,\mathcal{B}^T\,\mathcal{Q}\,\mathcal{B} + 2\,\mathcal{R}$$

Per quanto riguarda la componente penalizzante dovuta agli ostacoli supponiamo invece nuovamente di avere un solo ostacolo influente ai fini del calcolo, quello che otteniamo è allora la matrice

$$\frac{\partial^2 V_2}{\partial U_t^2} = \sum_{k=0}^{N-1} \frac{8}{\|E\,(\xi(t+k) - \xi_{obst})\|^6}\, \mathcal{B}_k^T\, E^T\, E\, (\xi(t+k) - \xi_{obst})\, (\xi(t+k) - \xi_{obst})^T\, E^T\, E\, \mathcal{B}_k - \frac{2}{\|E\,(\xi(t+k) - \xi_{obst})\|^4}\, \mathcal{B}_k^T\, E^T\, E\, \mathcal{B}_k$$

In realtà, poi, otteniamo l'estensione al caso di ostacoli multipli ripetendo semplicemente, ad ogni nuovo passo di predizione, la stessa espressione per ciascuno degli ostacoli che rientra nell'area di influenza del moto, presente e futuro, del robot.

Tuttavia la funzione *fminunc* utilizzata finora non consente di aggiungere agevolmente anche questa informazione al problema di ottimizzazione; per questo motivo si è scelto di passare ad algoritmi di ottimizzazione più avanzati ed in particolare di ricorrere al toolbox *TOMLAB*.

Per applicare anche questo nuovo accorgimento al problema di ottimizzazione si ricorre alla funzione del toolbox citato chiamata *ucSolve*, che consente di risolvere problemi non lineari non vincolati specificando in forma esplicita anche il gradiente e l'hessiano. Questa volta i risultati che si ottengono, con le dovute attenzioni legate alla dimensione minima degli ostacoli, sono soddisfacenti sia dal punto di vista della soluzione trovata, che dal punto di vista dei tempi di calcolo necessari. Con un orizzonte di predizione sufficientemente lungo da garantire buone prestazioni e con tempi di calcolo inferiori a $0.2\ s$ per passo nei punti più critici otteniamo ad esempio la traiettoria mostrata in Figura 5.5.2.

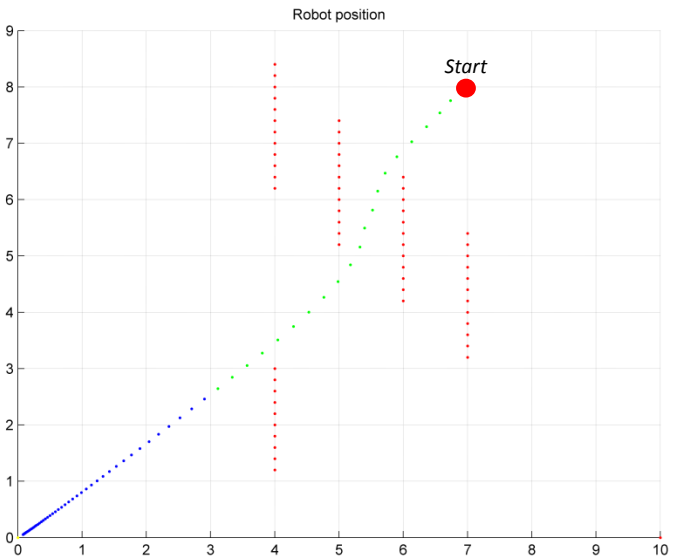

***Fig. 5.5.2** –Obstacle avoidance con ucSolve() e legge ausiliaria.*

Dai risultati ottenuti il metodo descritto finora appare valido per la soluzione del problema di navigazione online dei robot che si vuole risolvere e pertanto verrà adottato nel seguito.

Un'ultima idea per rendere la soluzione del problema più rapida, dal punto di vista globale questa volta, può essere quella di fare ricorso ad una legge ausiliaria nei tratti in cui non sono rilevati ostacoli nelle vicinanze; in questo modo infatti si evita di utilizzare la procedura di ottimizzazione, sostituita da dalla legge ausiliaria definita a priori, e si possono ottenere miglioramenti nei termini di tempo di calcolo complessivo. Risulta sensata l'idea di far ricorso a tale legge quando il robot ha ormai lasciato alle spalle tutti gli ostacoli, e l'unico problema rimasto è quello di semplice navigazione verso l'obiettivo.

Per valutare la validità della tecnica descritta finora si è scelto di utilizzare in simulazione il modello non lineare del robot reale, includendo anche le saturazioni sulle variabili di controllo; in particolare allora è necessario considerare che ad ogni passo di campionamento deve essere aggiunta, oltre all'ottimizzazione *MPC,* anche la procedura di linearizzazione. Questa in realtà non rappresenta un problema dal punto di vista computazionale, si faccia riferimento al Capitolo 3.

5.6 – Conclusioni.

In questo capitolo si è dimostrato, attraverso la simulazione di un buon numero di casi, che il problema di *obstacle avoidance* durante la navigazione del robot verso un obiettivo può essere risolto utilizzando congiuntamente un controllo *MPC* e la definizione di un potenziale artificiale. In particolare da tale combinazione nasce una soluzione particolarmente efficiente, in grado di pesare, attraverso la predizione lungo l'orizzonte, anche gli ostacoli che influenzeranno il moto futuro del robot, e di risolvere pertanto anche problemi particolarmente complessi come l'attraversamento di un corridoio.

Inoltre è stata valutata la possibilità di implementare effettivamente in tempo reale tale strategia; per far questo si è fatto ricorso ad algoritmi di ottimizzazione avanzati che consentissero di specificare, accanto alla funzione di costo, anche il suo gradiente ed il suo hessiano in forma esplicita, così da ridurre notevolmente il carico computazionale ed i tempi calcolo. Il tutto è stato applicato al modello del robot a disposizione appoggiandosi ad un anello di controllo interno per la linearizzazione in *feedback* dinamica e validando i risultati.

Si sottolinea ancora una volta che le funzioni di penalizzazione utilizzate in questa sezione non sono poi quelle utilizzate per risolvere il problema di coordinamento di cui si discuterà nel resto della trattazione.

6- MPC PER IL COORDINAMENTO

6.1 - Introduzione

Dopo aver visto come la tecnica *MPC,* unita ad una linearizzazione in *feedback,* consenta di gestire il problema di controllo di movimento di un singolo robot ad uniciclo, consideriamo una sua estensione al caso multi agente. In particolare ricordiamo anche che l'utilizzo della procedura descritta in precedenza permette di gestire online la generazione di una traiettoria verso l'obiettivo e che è possibile inserire naturalmente nel problema di ottimizzazione una cifra di penalizzazione in grado di tenere conto del problema di *obstacle avoidance*.

Per gestire il problema di coordinamento tra robot si considera ora il set di robot come un unico sistema e si risolve il problema di controllo che ne emerge con un approccio distribuito, così come anticipato nel Capitolo 1. A tal proposito è la particolare natura del problema a rendere vantaggiosa una visione di questo tipo dal momento che ogni robot costituisce, di fatto, un sottosistema a sé stante interconnesso agli altri solo tramite dei vincoli. Più nel dettaglio, infatti, ogni robot è caratterizzato da una propria dinamica del tutto indipendente da quella degli altri e l'unico fattore da tenere in conto, e che costringe a considerare per intero il problema di controllo del sistema, è la possibilità di collisione durante il movimento, che in altre parole può essere descritta come vincolo alle possibilità di moto di ciascun agente.

Abbiamo già detto che la scelta di *MPC,* per assolvere al compito di gestione del sistema, ha come principale effetto quello di rendere disponibile, ad ogni passo di campionamento, una predizione delle azioni future calcolate dall'algoritmo di ottimizzazione che consente, in maniera elegante, di prevedere ed evitare possibili collisioni.

In particolare, invece di implementare un controllore *MPC* centralizzato con compiti di supervisione, si è scelto di indagare l'efficacia di un approccio completamente distribuito in cui ciascun robot è fornito di un regolatore *MPC* locale, costruito a partire da quanto descritto al Capitolo 5. Questo ha la capacità di risolvere il singolo problema del parcheggio in un ambiente caratterizzato da ostacoli ed è in grado di dialogare con gli altri regolatori per scambiare informazioni in merito alla propria traiettoria predetta, così da arrivare, secondo la filosofia distribuita, ad una forma di consenso che renda la soluzione ottenuta il più possibile simile a quella globale centralizzata. Il motivo di questa scelta è già stato discusso, e si riassume sostanzialmente nella maggior efficienza

risolutiva di tanti piccoli problemi, rispetto ad un unico grande problema, ed inoltre nel fatto che esso si semplifica quando i singoli sottosistemi diventano solo minimamente interagenti o del tutto indipendenti dinamicamente tra loro.

Nel nostro caso questo significa che solo robot considerati vicini, o con traiettorie che possono sovrapporsi, dovranno preoccuparsi di concordare una soluzione comune per evitare la collisione e raggiungere l'ottimo globale del sistema se possibile. D'altra parte, inoltre, questo comportamento è quello che si rileva nei problemi reali di gestione del traffico in cui ciascun agente deve prendere una decisione locale, utilizzando le informazioni stimate dell'ambiente circostante, e costruire una soluzione che porti al consenso globale appoggiandosi eventualmente ad un set di regole convenzionali; basti pensare a quello che accade in un incrocio stradale.

Quest'ultimo aspetto fornisce in particolare uno spunto di ricerca su di un argomento notevolmente interessante; è infatti curioso, una volta implementato un opportuno algoritmo *MPC* distribuito che garantisca il funzionamento corretto nelle situazioni descritte, studiare come differenti meccanismi di priorità nella gestione delle collisioni portino a diverse soluzioni del problema di controllo complessivo. Tale aspetto sarà quindi uno degli obiettivi finali del lavoro che segue.

6.2 – MPC Robusto. Strategia tube based.

Prima di proseguire è necessario prendere in considerazione il problema di robustezza di un controllo *MPC*, ovvero valutare il comportamento che si ha quando, a causa degli effetti dell'incertezza, l'evoluzione predetta del sistema nominale è diversa dal comportamento reale attuale. In generale infatti, dal momento che è impossibile ottenere modelli perfetti di sistemi reali, e dal momento che vi sarà sempre un qualche disturbo esogeno agente sul sistema, la presenza di incertezza diventa un aspetto inevitabile per qualsiasi problema di controllo. A tal proposito, metodi di sintesi di controllori che tengono conto di informazioni note a priori sulle incertezze, come ad esempio i limiti o la caratterizzazione spettrale di un disturbo, sono indicati come metodi di controllo robusto. Il controllo lineare robusto, ed in particolare metodi H_2 o H_∞ trovano, al giorno d'oggi, un grande utilizzo nella pratica. D'altra parte, però, tali metodi non tengono direttamente conto della presenza di vincoli sui valori assunti dalle variabili di stato o dagli ingressi di controllo. Solitamente, quindi, i controllori sono sintetizzati per un problema ideale e non vincolato e solo a posteriori ulteriori misure sono adattate per assicurare che i vincoli siano soddisfatti con soluzioni *ad hoc*. Chiaramente questo, non solo rende l'analisi difficile, ma porta anche allo sviluppo di controllori maggiormente conservativi. Ancora, nella maggior parte delle applicazioni, l'obiettivo di controllo non è solo quello di stabilizzare il sistema ma quello di controllarlo in modo che la sua uscita insegua un dato valore di riferimento oppure una traiettoria di

riferimento. Inoltre, il numero e la qualità delle misure disponibili in applicazioni reali è generalmente limitato, così che spesso misure parzialmente errate dell'uscita del sistema sono l'unica fonte di informazione che può essere utilizzata per il controllo *(output feedback control)*.

Nel caso migliore, la cosa che può accadere quando un controllore nominale è utilizzato su di un sistema incerto è un degrado delle prestazioni. Tuttavia, se l'incertezza è grande, oppure se il sistema in anello chiuso ha margini di robustezza ridotti, il sistema controllato incerto può diventare instabile; è quindi il caso di occuparsi di controllo robusto. Questo è un problema ancor più complesso se si tratta di sistemi vincolati, dal momento che l'obiettivo del controllo non è solo quello di assicurare una robusta stabilità ma anche di soddisfare iterativamente i vincoli.

Nel caso di interesse prendiamo sistemi caratterizzati da un disturbo additivo non noto ma limitato del tipo:

$$x(k+1) = Ax(k) + Bu(k) + w(k)$$

con $w(k)$ appartenente ad un intorno compatto dell'origine $\mathbb{W}$ che siamo in grado di caratterizzare. Il problema è quello di trovare una legge di controllo in grado di garantire la convergenza, l'ottimalità ed il rispetto dei vincoli indipendentemente dal valore assunto dal disturbo.

Anche se il controllo *MPC* discusso finora fornisce forti risultati teorici riguardanti la sua stabilità nominale e la sua applicabilità, non tiene in conto del comportamento che si può avere quando l'evoluzione del sistema predetto differisce dal comportamento reale del sistema; si ritiene allora necessario prendere in considerazione l'aspetto di robustezza nella fase stessa di progettazione del controllore *MPC*.

La prima soluzione a tale problema è descritta in [16]; l'idea è quella di minimizzare la funzione obiettivo nel suo *worst case*, ovvero una funzione di costo che corrisponde al caso peggiore di incertezza. Tale metodo è solitamente indicato come *MPC minmax* ed ha come inconveniente la necessità di far ricorso ad algoritmi di calcolo computazionalmente molto dispendiosi e pertanto verrà trascurato. Una seconda soluzione, che è invece quella utilizzata nel seguito, prende il nome di *tube based MPC* e consente di giungere a risultati simili, senza complicare di troppo il problema da ottimizzare, una descrizione approfondita si trova in [17].

In particolare l'approccio *tube based* richiede di prendere in considerazione, accanto al sistema perturbato, un sistema nominale avente la stessa dinamica in assenza di disturbo; la sua equazione risulta cioè

$$\hat{x}(k+1) = A\,\hat{x}(k) + B\,\hat{u}(k)$$

Tale sistema sarà quello a cui applicare la tecnica *MPC,* esattamente come si farebbe per il progetto di un controllore che trascuri l'incertezza, per ricavare il valore ottimo

dell'ingresso $\hat{u}(t: t+N-1|t)$. Questo necessita chiaramente di aver ridefinito opportunamente tutti i vincoli, per applicarli alle nuove variabili di stato e di controllo.

L'ingresso per il sistema reale viene poi costruito in modo robusto utilizzando una legge ausiliaria che pesi la differenza tra lo stato del sistema nominale e quello, misurato, del sistema perturbato come

$$u(k) = \hat{u}(k) + K\big(x(k) - \hat{x}(k)\big)$$

Se si considera tale differenza come variabile d'errore, ovvero si definisce $z(k) = x(k) - \hat{x}(k)$, si identifica la sua dinamica come

$$z(k+1) = (A + BK)\, z(k) + w(k)$$

Si noti che essa è indipendente dal valore predetto $\hat{u}(k)$. A partire da questa equazione e per estensione di quanto visto in precedenza, se la matrice dell'anello chiuso $(A + BK)$ è asintoticamente stabile, essendo $w(k)$ un disturbo limitato, esiste un set, detto robustamente positivamente invariante *(RPI),* $\mathbb{Z}$ tale che:

$$z(t) \in \mathbb{Z} \qquad w(k) \in \mathbb{W} \quad \forall k \geq t \quad \rightarrow \quad z(t+i) \in \mathbb{Z} \quad \forall i \geq 0$$

o in altre parole tale che il sistema errore risulti sempre vincolato all'interno dell'insieme a patto che vi si trovi nell'istante iniziale, cioè:

$$(A + BK)\, \mathbb{Z} \oplus \mathbb{W} \subseteq \mathbb{Z}$$

All'atto pratico vincolare l'errore $z(k)$ significa, dunque, garantire che lo stato del sistema reale si manterrà in un intorno prefissato dello stato del sistema nominale, indipendentemente dal valore assunto dal disturbo, purché limitato. Tale risultato motiva quindi il nome *tube based* dell'approccio poiché otteniamo di fatto la garanzia che la traiettoria reale si mantenga all'interno di un tubo centrato su quella nominale.

Con il simbolo $\oplus$ si è indicata l'operazione di somma di Minkowski tra due set, definita più precisamente come:

$$C = A \oplus B = \{c = a + b : a \in A, b \in B\}$$

allo stesso modo ci sarà utile la differenza $\ominus$:

$$C = A \ominus B = \{c : c \oplus B \subseteq A\}$$

Riprendendo la definizione della variabile $z(k) = x(k) - \hat{x}(k) \in \mathbb{Z}$ si vede che è necessario impostare nuovamente il problema affinché siano rispettati i vincoli sulle variabili di stato e di ingresso ammissibili, ovvero garantire che:

$$x(k) \in \mathbb{X} \subseteq \mathbb{R}^n \qquad u(k) \in \mathbb{U} \subseteq \mathbb{R}^m$$

In particolare questo si ottiene costruendo, a partire dai primi, dei vincoli maggiormente restrittivi ed imponendo di conseguenza:

$$\hat{x}(k) \in \widehat{\mathbb{X}} = \mathbb{X} \ominus \mathbb{Z} \qquad \hat{u}(k) \in \widehat{\mathbb{U}} = \mathbb{U} \ominus K\mathbb{Z}$$

ad esempio, infatti, abbiamo per la variabile di stato la seguente relazione

$$x(k) = \hat{x}(k) + \big(x(k) - \hat{x}(k)\big) = \hat{x}(k) + z(k) \in \widehat{\mathbb{X}} \oplus \mathbb{Z} \subseteq \mathbb{X}$$

Allo stesso modo è necessario rivedere il concetto di set terminale applicato alla nuova legge ausiliaria $\hat{u}(k) = K\hat{x}(k)$. In particolare, allora, definiamo il set invariante $\widehat{\mathbb{X}}^f \subseteq \widehat{\mathbb{X}}$ tale che, per ogni $\hat{x}(k) \in \mathbb{X}^f$ e per ogni stato nominale che evolve secondo la legge dell'anello chiuso $\hat{x}(k+1) = (A + BK)\hat{x}(k)$, valgano le relazioni di appartenenza:

$$\begin{cases} \hat{x}(k+i) \in \widehat{\mathbb{X}}^f \\ K\hat{x}(k+i) \in \widehat{\mathbb{U}} \end{cases} \qquad \forall i \geq 0$$

A questo punto, per il nuovo problema di controllo, si costruisce ancora una volta la funzione di costo quadratica come quella vista in precedenza, ma questa volta basata su stato ed ingresso del sistema nominale:

$$V\big(\hat{x}(t), \hat{u}(t\colon t+N-1)\big) = \sum_{k=t}^{t+N-1} \|\hat{x}(k)\|_Q^2 + \|\hat{u}(k)\|_R^2 + V^f\big(\hat{x}(t+N)\big)$$

Il problema *MPC* da risolvere, tenendo presente questa cifra appena scritta ed i vincoli ristretti, è di conseguenza il seguente:

$$\min_{\hat{u}(t:t+N-1)} V\big(\hat{x}(t), \hat{u}(t\colon t+N-1)\big)$$

$subject\ to$

$\hat{x}(k+1) = A\hat{x}(k) + B\hat{u}(k)$

$\hat{x}(k) \in \widehat{\mathbb{X}} \quad \forall k = t\colon t+N-1$

$\hat{u}(k) \in \widehat{\mathbb{U}} \quad \forall k = t\colon t+N-1$

$\hat{x}(t+N) \in \mathbb{X}^f \subseteq \mathbb{X}$

Per quanto imposto esso garantisce che la soluzione ottenuta sia ammissibile anche per il sistema reale. Questa, a tal proposito, sarà la sequenza di ingressi ottimi per il modello nominale $\hat{u}(t\colon t+N-1|t)$ calcolata all'istante attuale; secondo il criterio *Receding Horizon*, di cui si è ampiamente discusso, viene quindi scelto solo il primo elemento $\hat{u}(t|t)$ e poi utilizzato direttamente per controllare il sistema perturbato secondo la legge robusta:

$$u(t) = \hat{u}(t|t) + K\big(x(t) - \hat{x}(t)\big)$$

In tal caso si noti che la traiettoria di stato del sistema nominale $\hat{x}(t)$ è in generale indipendente da quella del sistema reale $x(t)$; tuttavia la presenza della proprietà di invarianza garantisce che $z(t) = x(t) - \hat{x}(t)$ resti limitato, e pertanto necessariamente:

$$x(t) = \hat{x}(t) \oplus \mathbb{Z}$$

Questo permette di dire che, dal momento che *MPC* garantisce la convergenza di $\hat{x}(t)$ e che vale quanto appena visto, allora per lo stato del sistema reale varrà $x(t) \to \mathbb{Z}$ per $t \to \infty$.

Il fatto che nella trattazione appena descritta lo stato del sistema nominale non sia influenzato in alcun modo dall'evoluzione del sistema reale fa si che, in realtà, si stia implementando una sorta di controllo basato su di un anello aperto; questo non garantisce un sufficiente grado di robustezza a fronte di perturbazioni di modello. Un'idea per poter inserire un *feedback* dal sistema perturbato è allora quella di rivedere il problema di ottimizzazione aggiungendo come variabile anche lo stato iniziale del sistema nominale $\hat{x}(t)$. Accanto al nuovo grado di libertà si aggiunge un ulteriore vincolo esplicito che richieda alla differenza $x(t) - \hat{x}(t)$ di rimanere confinata all'interno di un set invariante.

In particolare allora il problema *MPC* robusto viene riformulato nel modo seguente:

$$V^*\big(x(t)\big) = \min_{\hat{x}(t), \hat{u}(t:t+N-1)} V\big(\hat{x}(t: t+N), \hat{u}(t: t+N-1)\big)$$

$subject\ to$:

$\hat{x}(k+1) = A\hat{x}(k) + B\hat{u}(k)$

$\hat{x}(k) \in \widehat{\mathbb{X}} \quad \forall k = t \dots t+N-1$

$\hat{u}(k) \in \widehat{\mathbb{U}} \quad \forall k = t \dots t+N-1$

$x(t+N) \in \mathbb{X}^f \subseteq \mathbb{X}$

$x(t) - \hat{x}(t) \in \mathbb{Z}$

Il risultato è ancora la sequenza di ingressi ottima $\hat{u}(t: t+N-1)$ alla quale si aggiunge anche lo stato attuale del sistema nominale $\hat{x}(t)$ che questa volta è per ipotesi legato allo stato del sistema perturbato a causa dell'ultimo vincolo imposto. La variabile di controllo per il sistema reale viene costruita di nuovo secondo l'approccio robusto come:

$$u(t) = \hat{u}(t|t) + K\big(x(t) - \hat{x}(t|t)\big)$$

Lo schema risultante del controllo costruito è ora a tutti gli effetti un anello chiuso, poiché il sistema nominale utilizzato per risolvere *MPC* è strettamente dipendente dal valore assunto dalle variabili di stato del sistema reale come mostrato in Figura 6.2.1.

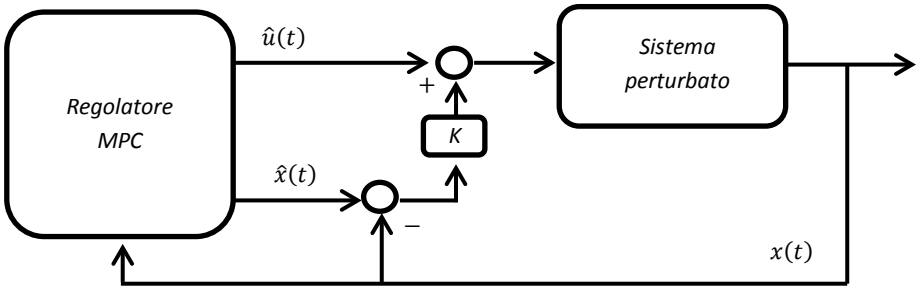

Fig. 6.2.1 *– Schema dell'MPC robusto.*

Quest'ultima soluzione di controllo sarà esattamente quella usata nel seguito per generare le traiettorie percorse da ciascuno dei robot; l'idea è infatti quella di utilizzare la garanzia di percorrenza all'interno del tubo per progettare un sistema di navigazione libero da collisioni; più nel dettaglio sarà interessante quindi tener traccia del percorso ideale e dell'incertezza con cui ciascuno dei passi viene inseguito.

6.3 – Problema di coordinamento.

Supponiamo, in generale, che il sistema di cui si vuole effettuare il controllo coordinato sia composto da M agenti uguali e che ciascuno di questi possa essere descritto mediante una equazione dinamica lineare a tempo discreto del tipo seguente

$$x^{[i]}(t+1) = A\, x^{[i]}(t) + B\, u^{[i]}(t)$$

e che a questa sia associata l'equazione che determina le variabili di uscita:

$$z^{[i]}(t) = C\, x^{[i]}(t)$$

Nel nostro caso specifico si nota che, come ricordato più volte, l'applicazione del controllo in *feedback* linearizzante permette di descrivere le dinamiche del singolo robot come un sistema lineare e tempo invariante descritto da un modello di tipo doppio integratore.

In dettaglio si scrive il vettore di stato come

$$x = \left(\underbrace{x_p \, , v_x}_{\text{Dinamica lungo x}} \, , \underbrace{y_p \, , v_y}_{\text{Dinamica lungo y}} \right)$$

dove x_p ed y_p rappresentano le coordinate lungo un sistema di riferimento cartesiano della posizione assunta dal robot, mentre v_x e v_y rappresentano le rispettive derivate, ovvero le velocità con cui esso si muove. Allora le matrici A e B, che compaiono nell'equazione dinamica, non sono altro che matrici diagonali a blocchi costituite da elementi di dimensione $2x2$ uguali tra loro:

$$A = \begin{bmatrix} A_s & 0 \\ 0 & A_s \end{bmatrix} \; B = \begin{bmatrix} B_s & 0 \\ 0 & B_s \end{bmatrix}$$

Tale osservazione ci consente di concentrare, quando necessario, l'attenzione su di un solo sottosistema ridotto, costituito dalla coppia $(A_s \, , B_s)$ che governa il moto lungo una direzione, considerando poi la restante parte del tutto equivalente alla prima. Utilizzando l'accorgimento descritto si è in grado, allora, di semplificare notevolmente il problema computazionale al momento dell'implementazione effettiva, poiché tutti i calcoli matriciali avranno dimensioni dimezzate.

Dal momento che la posizione del robot risulta direttamente accessibile, grazie all'utilizzo della telecamera e dell'algoritmo di riconoscimento, le variabili di uscita corrisponderanno semplicemente alle due coordinate cartesiane che la descrivono.

Accantonando per il momento questo aspetto pratico, che ci sarà utile nei paragrafi successivi, possiamo concentrarci sul problema di coordinamento vero e proprio. In particolare prendiamo ora l'*i-esimo* agente di cui è noto il modello analitico nella forma descritta in precedenza. Per questo supponiamo di aver definito un vettore di riferimenti di posizione lungo l'orizzonte di predizione N:

$$\tilde{y}^{[i]}_{[t:t+N-1]} = \{\tilde{y}_k \, , k = t \dots t + N - 1\}$$

A partire da tale vettore possiamo ricavare i riferimenti equivalenti per lo stato e per l'ingresso, utili nella definizione del problema di controllo, ricorrendo ad una sorta di osservatore alimentato dalla differenza tra la posizione di riferimento e l'equivalente predetta:

$$\tilde{x}^{[i]}(k+1) = A \, \tilde{x}^{[i]}(k) + B \, \tilde{u}^{[i]}(k) + G^x \left(\tilde{y}^{[i]}(k+1) - C \, \tilde{x}^{[i]}(k) \right)$$

$$\tilde{u}^{[i]}(k+1) = \tilde{u}^{[i]}(k) + G^u \left(\tilde{y}^{[i]}(k+1) - C \, \tilde{x}^{[i]}(k) \right)$$

Affinché questo sia possibile è necessario che la matrice che governa il nuovo sistema sia asintoticamente stabile; in altre parole, dette

$$\tilde{A} = \begin{bmatrix} A & B \\ 0 & I \end{bmatrix} \qquad \tilde{C} = [C \quad 0] \qquad \tilde{G} = \begin{bmatrix} G^x \\ G^u \end{bmatrix}$$

è necessario che il guadagno $\tilde{G}$ sia scelto in modo che l'anello chiuso abbia come dinamica una matrice $(\tilde{A} - \tilde{G}\,\tilde{C})$ Schur, ovvero essendo nel caso a tempo discreto, avente autovalori tutti all'interno di un cerchio di raggio unitario. In questo modo abbiamo la garanzia che la convergenza dei due riferimenti sia asintoticamente stabile.

Coerentemente con l'idea di sfruttare il controllo robusto per garantire opportune proprietà di limitatezza dell'errore di inseguimento della traiettoria, si può iniziare definendo un insieme quadrato di lato 2ε all'interno del quale deve risiedere la differenza tra due successivi riferimenti; indicando questo con $B_\epsilon(0)$

$$\tilde{y}_{t+1}^{[i]} \in \tilde{y}_t^{[i]} \oplus B_\epsilon(0)$$

in questo modo, implicitamente, si richiede di mantenere limitata anche la velocità ideale di movimento del robot che da ultimo dipende proprio dall'evoluzione del riferimento.

A partire da questa condizione, inoltre, possiamo utilizzare l'equazione di regime per definire la condizione stazionaria corrispondente con l'uscita $\tilde{y}_t^{[i]}$. Dalla precedente si ottiene la seguente espressione che permette di definire il *set* Δ_i^{SS}

$$\begin{bmatrix} x_{t+1}^{[i]\ SS} - x_t^{[i]\ SS} \\ u_{t+1}^{[i]\ SS} - u_t^{[i]\ SS} \end{bmatrix} \in \begin{bmatrix} I - A & -B \\ -C & 0 \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ I \end{bmatrix} B_\epsilon(0) = \Delta_i^{SS}$$

Per quanto riguarda i riferimenti $\tilde{x}$ ed $\tilde{u}$ possiamo poi scrivere

$$\begin{bmatrix} \tilde{x}_{t+1}^{[i]} - x_{t+1}^{[i]\ SS} \\ \tilde{u}_{t+1}^{[i]} - u_{t+1}^{[i]\ SS} \end{bmatrix}$$
$$= \left(\tilde{A} - \tilde{G}\,[C \quad 0]\right) \begin{bmatrix} \tilde{x}_t^{[i]} - x_t^{[i]\ SS} \\ \tilde{u}_t^{[i]} - u_t^{[i]\ SS} \end{bmatrix}$$
$$+ \left(\tilde{A} - \tilde{G}\,[C \quad 0]\right) \begin{bmatrix} x_t^{[i]\ SS} - x_{t+1}^{[i]\ SS} \\ u_t^{[i]\ SS} - u_{t+1}^{[i]\ SS} \end{bmatrix}$$

in questa formulazione l'ultimo termine appartiene, per definizione, al set Δ_i^{SS} appena introdotto, mentre la matrice $\left(\tilde{A} - \tilde{G}\,[C \quad 0]\right) = \tilde{F}$ è per ipotesi stabile. Se allora si riscrive la relazione come

$$\begin{bmatrix} \tilde{x}_{t+1}^{[i]} - x_{t+1}^{[i]\ SS} \\ \tilde{u}_{t+1}^{[i]} - u_{t+1}^{[i]\ SS} \end{bmatrix} = \tilde{F} \begin{bmatrix} \tilde{x}_t^{[i]} - x_t^{[i]\ SS} \\ \tilde{u}_t^{[i]} - u_t^{[i]\ SS} \end{bmatrix} + w_t^{[i]} \qquad w_t^{[i]} \in \Delta_i^{SS}$$

è possibile utilizzarla per definire un nuovo *set* Δ^{xu} che risulterà robustamente positivamente invariante *(RPI)* per tale equazione.

Questo permette a sua volta di avere la garanzia che:

$$\tilde{y}_t^{[i]} - C\,\tilde{x}_t^{[i]} \in -[C \quad 0]\,\Delta^{xu} = \Delta^y \qquad \rightarrow \qquad \tilde{y}_{t+1}^{[i]} - C\,\tilde{x}_t^{[i]} \in \Delta^y \oplus B_\epsilon(0) \quad \forall t$$

e ciò vale per ogni istante di tempo a patto di aver imposto la prima condizione sul sistema.

A questo punto abbiamo visto che l'implementazione robusta di un controllo *MPC* richiede che sia definito un sistema nominale lungo il quale svolgere l'ottimizzazione; in particolare, tenendo presente le considerazioni fatte finora, scegliamo per questo la stessa struttura utilizzata per l'osservatore:

$$\hat{x}_{t+1}^{[i]} = A\,\hat{x}_t^{[i]} + B\,\hat{u}_t^{[i]} + G^x\left(\tilde{y}_{t+1}^{[i]} - C\,\tilde{x}_t^{[i]}\right)$$

Dopo aver fatto ricorso al sistema nominale per risolvere il problema di ottimizzazione predittiva, l'effettivo valore della variabile di controllo da applicare al sistema si ottiene combinando la soluzione ottenuta con un termine lineare, basato su di una legge ausiliaria K, in grado di pesare il disallineamento tra il sistema reale e quello nominale:

$$u_t^{[i]} = \hat{u}_t^{[i]} + K\left(x_t^{[i]} - \hat{x}_t^{[i]}\right) = \hat{u}_t^{[i]} + K\,\varepsilon_{\mathrm{t}}^{[i]}$$

dove è definito l'errore di inseguimento nominale come $\varepsilon_t^{[i]} = x_t^{[i]} - \hat{x}_t^{[i]}$.

È possibile vedere che la sua dinamica è dettata dall'equazione

$$\varepsilon_{t+1}^{[i]} = (A + BK)\,\varepsilon_t^{[i]} + v_t^{[i]} \qquad v_t^{[i]} = -G^x\left(\tilde{y}_{t+1}^{[i]} - C\,\tilde{x}_t^{[i]}\right) \in -G^x\left(\Delta^y \oplus B_\epsilon(0)\right)$$

e che pertanto $F_k = (A + BK)$ deve risultare anch'essa Schur. Inoltre, ricordando quanto definito in precedenza per il vincolo su $\left(\tilde{y}_{t+1}^{[i]} - C\,\tilde{x}_t^{[i]}\right)$, è possibile limitare l'insieme di appartenenza della componente di disturbo $v_t^{[i]}$ e definire, per quanto detto, un suo *invariant set* $\mathcal{E}_i$ di modo che sia valida la garanzia:

$$x_t^{[i]} \in \hat{x}_t^{[i]} \oplus \mathcal{E}_i$$

A questo punto tale risultato permette di arrivare a quanto desiderato per il problema di coordinamento, ovvero, consente di definire con sicurezza un *set*, nell'intorno della traiettoria di riferimento, all'interno del quale si ha garanzia di trovare la traiettoria reale e pertanto tutta l'informazione di cui abbiamo bisogno per evitare collisioni tra agenti multipli. In dettaglio è sufficiente aggiungere a ciascun *MPC* il vincolo

$$C\left(\hat{x}_k^{[i]} - \tilde{x}_k^{[i]}\right) \in \Delta_i^z \qquad k = t \ldots t + N - 1$$

con l'insieme Δ_i^z completamente arbitrario, ovvero richiedere che la posizione del sistema nominale sia nell'intorno della posizione del riferimento, per poter ottenere di conseguenza la garanzia che

$$C\, x_k^{[i]} \in C\, \tilde{x}_k^{[i]} \oplus \mathcal{E}_i \oplus \Delta_i^z = C\, \tilde{x}_k^{[i]} \oplus Z_i$$

cioè che la posizione reale non sia mai al di fuori del set Z_i centrato sulla posizione di riferimento, nota a priori. Questo, in un'ottica di coordinamento nel movimento di vari robot, sarà l'intervallo di incertezza che ciascun agente può comunicare direttamente al vicino, assieme alla sua traiettoria di riferimento, per evitare la collisione.

Con una struttura di questo tipo siamo dunque in grado di conferire alla soluzione del problema *MPC* le garanzie necessarie a far sì che l'evoluzione del sistema reale sia sempre predicibile all'interno di un *range* di incertezza prefissato. Tuttavia, il ricorso ad una formulazione robusta della strategia di controllo, implica una complicazione del problema di minimizzazione che, come abbiamo già visto, non coinvolgerà più solamente il vettore delle variabili di ingresso lungo l'orizzonte di predizione, ma anche lo stato iniziale del sistema nominale, per evitare che la sua evoluzione sia disallineata da quella del sistema reale. Nel nostro caso, inoltre, dovremo aggiungere il valore del riferimento di posizione all'istante $t+N$, così da fare in modo che esso evolva naturalmente verso il *set point* finale della navigazione. In particolare abbiamo allora come variabile di ottimizzazione:

$$\xi_t^{[i]} = \left\{ \hat{x}_t^{[i]} \;\middle|\; \hat{u}_t^{[i]} \dots \hat{u}_{t+N-1}^{[i]} \;\middle|\; \bar{y}_{t+N}^{[i]} \right\}$$

nel caso in questione la sua dimensione sarà di $4 + 2N + 2$.

Il problema di ottimizzazione risolto da ciascun agente risulterà quindi nella forma:

$$\min_{\hat{x}_t^{[i]} \,|\, \hat{u}_t^{[i]} \dots \hat{u}_{t+N-1}^{[i]} \,|\, \bar{y}_{t+N}^{[i]}} V_i^N \left(\hat{x}_t^{[i]}, \hat{U}_{[t:t+N-1]}^{[i]}, \bar{y}_{t+N}^{[i]} \right)$$

$$V_i^N = \sum_{k=t}^{t+N-1} \left\| \hat{x}_k^{[i]} - \tilde{x}_k^{[i]} \right\|_Q^2 + \left\| \hat{u}_k^{[i]} - \tilde{u}_k^{[i]} \right\|_R^2 + \left\| \hat{x}_{t+N}^{[i]} - \bar{x}_{t+N}^{[i]} \left(\bar{y}_{t+N}^{[i]} \right) \right\|_P^2 + \gamma \left\| \bar{y}_{t+N}^{[i]} - \tilde{y}_{t+N-1}^{[i]} \right\|^2 + \left\| \bar{y}_{t+N}^{[i]} - \bar{y}_{setpoint}^{[i]} \right\|_T^2 + V_i^{coll}$$

La cifra di merito V_i^N, che consente di implementare il tutto, dovrà invece comprendere una penalità, rispettivamente, sull'errore di inseguimento della traiettoria e della variabile di controllo di riferimento lungo l'orizzonte predittivo, relativo allo stato finale del sistema nominale, un termine che garantisca la corretta evoluzione del riferimento di posizione ed infine la componente penalizzante dovuta all'*obstacle avoidance* già

discussa nei capitoli precedenti. Con V_i^{coll} è infatti indicata la parte della cifra di merito utilizzata per garantire che vengano evitate collisioni tra sottosistemi.

Tale cifra deve essere minimizzata in presenza dei vincoli di cui si è discusso finora, al fine di garantire le proprietà di robustezza cercate. In sintesi è necessario imporre innanzitutto che lo stato iniziale del sistema nominale non si discosti eccessivamente dallo stato reale misurato, ovvero che sia

$$x_t^{[i]} - \hat{x}_t^{[i]} \in \mathcal{E}_i$$

inoltre che il nuovo termine $\bar{y}_{t+N}$ si mantenga nell'intorno $B_\epsilon(0)$ dell'ultimo campione, cioè

$$\bar{y}_{t+N}^{[i]} - \tilde{y}_{t+N-1}^{[i]} \in B_\epsilon(0)$$

Accanto a questi, per ogni passo lungo l'orizzonte predittivo, devono essere esplicitati i set di ammissibilità per le variabili $\hat{x}$ ed $\hat{u}$ ricavati dagli equivalenti insiemi relativi al sistema reale, ovvero

$$\hat{x}_k^{[i]} \in \hat{\mathbb{X}}_i \qquad \hat{u}_k^{[i]} \in \hat{\mathbb{U}}_i \qquad \forall\, k = t \dots t+N-1$$

e, ancora, imposto il vincolo fittizio, e di dimensione arbitraria, che consente di limitare lo scostamento della variabile di stato nominale dal riferimento

$$C\left(\hat{x}_k^{[i]} - \tilde{x}_k^{[i]}\right) \in \Delta^z \qquad \forall\, k = t \dots t+N-1$$

Infine la tecnica *MPC* robusta richiede anche di imporre i cosiddetti vincoli terminali, sulla base del valore scelto per la legge di controllo ausiliaria stabilizzante; in altre parole, come abbiamo visto, è necessario vincolare lo stato finale dell'orizzonte di predizione ad appartenere ad un opportuno insieme invariante rispetto all'applicazione delle legge ausiliaria stessa ed allo stesso modo anche la variabile di controllo relativa, cioè si richiede che valgano:

$$\hat{x}_{t+N}^{[i]} - \bar{x}_{t+N}^{[i]}\left(\bar{y}_{t+N}^{[i]}\right) \in \Sigma_i$$

$$\bar{x}_{t+N}^{[i]}\left(\bar{y}_{t+N}^{[i]}\right) \in \hat{\mathbb{X}} \oplus \Sigma_i$$

$$\bar{u}_{t+N}^{[i]}\left(\bar{y}_{t+N}^{[i]}\right) \in \hat{\mathbb{U}} \oplus K\Sigma_i$$

dove si è scritto sinteticamente

$$\bar{x}_{t+N}^{[i]}\left(\bar{y}_{t+N}^{[i]}\right) = A\,\tilde{x}_{t+N-1}^{[i]} + B\,\tilde{u}_{t+N-1}^{[i]} + G^x\left(\bar{y}_{t+N}^{[i]} - C\tilde{x}_{t+N-1}^{[i]}\right)$$

Esso rappresenta l'evoluzione al tempo $t+N$ dello stato dell'osservatore, realizzato a partire dal nuovo valore di riferimento $\bar{y}_{t+N}$ in uscita all'ottimizzatore. Il *set* Σ_i è calcolato come *RPI* per l'equazione ausiliaria

$$\delta x_{t+1} = (A + BK)\delta x_t + w_t = F\delta x_t + w_t \qquad w_t \in \mathbb{W}_{arb}$$

Tali relazioni andranno ovviamente ad influenzare il modo con cui viene scelta, nell'algoritmo, la variabile di ottimizzazione $\bar{y}_{t+N}$ da cui dipendono i valori finali dei riferimenti $\bar{x}_{t+N}$ ed $\bar{u}_{t+N}$.

Riassumendo, dunque, siamo in grado di controllare, con l'approccio visto, il movimento del singolo robot verso l'obiettivo formulando il tutto come problema di inseguimento di una traiettoria di riferimento; in particolare, però, la struttura dei vincoli fa sì che si abbia la garanzia che la traiettoria percorsa realmente sia vincolata a stare in un intorno ben preciso di quella di riferimento, e con una incertezza nota a priori. In questo modo è possibile risolvere il problema di *collision avoidance* semplicemente ricorrendo ad una implementazione distribuita in cui ciascun agente trasmette ai suoi vicini la sequenza ideale dei suoi passi successivi lungo l'orizzonte predittivo, assieme all'incertezza con cui questi saranno percorsi. Tale informazione andrà allora a costituire il set di ostacoli, con relativa dimensione, che ciascuno degli agenti restanti dovrà tenere in conto nella soluzione della propria ottimizzazione. Ovviamente questo necessita di definire correttamente un protocollo di comunicazione e la tipologia dell'algoritmo utilizzato per raggiungere una sorta di consenso globale ed avere la garanzia che le proprietà siano rispettate.

Tralasciando per il momento questo aspetto più complesso, si può quindi ritornare alla cifra di merito aggiungendo la componente penalizzante dovuta agli ostacoli, come visto in precedenza nel caso di ostacoli fissi. Questa volta, però, il set di ostacoli non sarà definito a priori ma composto online. In dettaglio una prima possibilità è quella di scegliere per l'agente *i-esimo* l'insieme:

$$\mathcal{O} = \left\{ obst: \quad obst \in C\ \tilde{x}_k^{[j]} \qquad k = t \dots t + N - 1 \quad j \neq i \right\}$$

ovvero supporre che tutta la traiettoria stimata per ciascun altro robot lungo l'orizzonte predittivo non possa essere intersecata; la dimensione di ciascuno di questi ostacoli fittizi può invece essere dettata dal valore dell'incertezza contenuto in Z_j e dalla dimensione fisica di un robot, con i relativi margini di sicurezza, così che idealmente l'intero tubo predetto possa essere evitato. In realtà, vedremo nel seguito, che tale scelta degli ostacoli può essere notevolmente semplificata notando che è sufficiente richiedere, durante l'ottimizzazione, che al medesimo istante di tempo non vi siano due riferimenti di robot diversi sovrapposti nella stessa posizione.

Dato che il termine penalizzante usato nel Capitolo 5 non garantisce continuità della cifra di costo e delle derivate successive sul bordo dell'area di influenza, generando malfunzionamenti nell'algoritmo di ottimizzazione, nelle realizzazioni successive esso sarà sostituito con quello polinomiale di cui si è già discusso.

6.4 – Implementazione.

Dopo aver definito formalmente il problema di controllo ed aver imposto tutte le condizioni necessarie al funzionamento dell'algoritmo, è necessario scendere nel dettaglio dell'implementazione effettiva, rivalutando il problema *MPC* alla luce delle informazioni note sul sistema reale.

I passi successivi saranno quindi quelli di identificare opportunamente le variabili reali di ottimizzazione, di riscrivere sulla base di queste la cifra di merito, di ridefinire correttamente i *set* di cui si è discusso, ed infine di generare a partire da questi la forma esatta dei vincoli. Tenendo presenti le esigenze pratiche verranno calcolate inoltre le matrici Jacobiana ed Hessiana della cifra di merito complessiva, da fornire in ingresso all'algoritmo di minimizzazione per rendere più veloce la convergenza, esattamente come discusso nel Capitolo 5.

In particolare prendiamo ancora una volta il singolo agente *i-esimo* e cerchiamo di esprimere la cifra di merito nella forma adatta agli algoritmi di soluzione di cui si è già discusso, dopodiché riformuliamo i vincoli nei termini della variabile di ottimizzazione utilizzata. Per semplicità sarà omesso l'indice $[i]$ utilizzato finora e per alleggerire la notazione l'istante temporale a cui appartiene ciascuna variabile sarà riportato ove necessario come pedice.

6.4.1 – La cifra di costo

Innanzitutto riprendiamo l'equazione del sistema nominale che abbiamo scelto come:

$$\hat{x}_{t+1} = A\hat{x}_t + B\hat{u}_t + G^x(\tilde{y}_{t+1} - C\,\tilde{x}_t)$$

e progettiamo opportunamente il guadagno G^x come discusso in precedenza. Allo stesso modo per il riferimento $\tilde{x}$ l'equazione è ancora quella dell'osservatore:

$$\tilde{x}_{t+1} = A\tilde{x}_t + B\tilde{u}_t + G^x(\tilde{y}_{t+1} - C\tilde{x}_t)$$

pertanto, se scriviamo la differenza $\hat{x}_{k+1} - \tilde{x}_{k+1}$, che compare nella cifra di merito, si ottiene:

$$\hat{x}_{k+1} - \tilde{x}_{k+1} = A\,(\hat{x}_k - \tilde{x}_k) + B\,(\hat{u}_k - \tilde{u}_k) \qquad k = t \dots t+N-2$$

$$\hat{x}_{t+N} - \tilde{x}_{t+N}(\bar{y}_{t+N}) = A(\hat{x}_{t+N-1} - \tilde{x}_{t+N-1}) + B\,(\hat{u}_{t+N-1} - \tilde{u}_{t+N-1})$$

Si ricordi che il riferimento $\tilde{y}_k$ è noto a priori in $k = t \dots t+N-1$ e di conseguenza lo sono anche $\tilde{x}_k$ ed $\tilde{u}_k$ lungo l'orizzonte predittivo, mentre il nuovo obiettivo $\bar{y}_{t+N}$ è un argomento dell'ottimizzazione e da esso dipende strettamente anche $\bar{x}_{t+N}$.

Note tali equazioni dinamiche possiamo riscrivere la predizione lungo l'orizzonte come nel caso di *MPC* generico già visto:

$\hat{x}_t - \tilde{x}_t = I\,(\hat{x}_t - \tilde{x}_t)$

$\hat{x}_{t+1} - \tilde{x}_{t+1} = A(\hat{x}_t - \tilde{x}_t) + B(\hat{u}_t - \tilde{u}_t)$

$\hat{x}_{t+2} - \tilde{x}_{t+2} = A^2(\hat{x}_t - \tilde{x}_t) + AB(\hat{u}_t - \tilde{u}_t) + B(\hat{u}_{t+1} - \tilde{u}_{t+1})$ e di conseguenza in forma matriciale:

$$\underbrace{\begin{bmatrix} \hat{x}_t - \tilde{x}_t \\ \vdots \\ \hat{x}_{t+N} - \tilde{x}_{t+N} \end{bmatrix}}_{\hat{X}_t - \tilde{X}_t} = \underbrace{\begin{bmatrix} I \\ \vdots \\ A^N \end{bmatrix}}_{\mathcal{A}} (\hat{x}_t - \tilde{x}_t) + \overbrace{\begin{bmatrix} 0 & \cdots & 0 \\ B & \cdots & \vdots \\ A^{N-1}B & \cdots & B \end{bmatrix}}^{\mathcal{B}} \underbrace{\begin{bmatrix} \hat{u}_t - \tilde{u}_t \\ \vdots \\ \hat{u}_{t+N-1} - \tilde{u}_{t+N-1} \end{bmatrix}}_{\hat{U}_t - \tilde{U}_t}$$

$$= [\mathcal{A} \quad \mathcal{B}] \begin{bmatrix} \hat{x}_t - \tilde{x}_t \\ \hat{U}_t - \tilde{U}_t \end{bmatrix}$$

Per la cifra di merito possiamo quindi eliminare la sommatoria, definire opportunamente le matrici di peso allargate $\mathcal{Q} = diag(Q \dots Q\ P)$ ed $\mathcal{R} = diag(R \dots R)$ e riscrivere il tutto (trascurando per il momento la penalizzazione dovuta agli ostacoli) come:

$$V^N = \left\|\hat{X}_t - \tilde{X}_t\right\|_{\mathcal{Q}}^2 + \left\|\hat{U}_t - \tilde{U}_t\right\|_{\mathcal{R}}^2 + \left\|\begin{bmatrix} I \\ I \end{bmatrix} \bar{y}_{t+N} - \begin{bmatrix} \tilde{y}_{t+N-1} \\ \bar{y}_{setpoint} \end{bmatrix}\right\|_{\begin{bmatrix} \gamma I & 0 \\ 0 & T \end{bmatrix}}^2$$

In questa le variabili di ottimizzazione abbiamo detto essere $\left\{\hat{x}_t\ ,\ \hat{U}_{[t:t+N-1]}\ ,\ \bar{y}_{t+N}\right\}$. Si definisce il vettore ξ_t come:

$$\xi_t = \begin{bmatrix} \hat{x}_t - \tilde{x}_t \\ \hat{U}_t - \tilde{U}_t \\ \bar{y}_{t+N} \end{bmatrix}$$

Così facendo, infatti, avremo che la funzione di costo V^N può essere riscritta sinteticamente come:

$$\| [\mathcal{A} \quad \mathcal{B} \quad 0]\, \xi_t \|_{\mathcal{Q}}^2 + \| [0 \quad I \quad 0]\, \xi_t \|_{\mathcal{R}}^2 + \left\|\begin{bmatrix} I \\ I \end{bmatrix} [0 \quad 0 \quad I]\, \xi_t - \underbrace{\begin{bmatrix} \tilde{y}_{t+N-1} \\ \bar{y}_{setpoint} \end{bmatrix}}_{\mathcal{Y}}\right\|_{\begin{bmatrix} \gamma I & 0 \\ 0 & T \end{bmatrix}}^2$$

e ulteriormente riassunta in forma matriciale raccogliendo tutti i termini comuni :

$$V^N = \left\| \underbrace{\begin{bmatrix} \mathcal{A} & \mathcal{B} & 0 \\ 0 & I & 0 \\ 0 & 0 & \begin{bmatrix} I \\ I \end{bmatrix} \end{bmatrix}}_{\mathbb{A}} \xi_t - \underbrace{\begin{bmatrix} 0 \\ 0 \\ y \end{bmatrix}}_{\mathbb{Y}} \right\|^2_{\underbrace{\begin{bmatrix} \mathcal{Q} & 0 & 0 \\ 0 & \mathcal{R} & 0 \\ 0 & 0 & \begin{bmatrix} \gamma I & 0 \\ 0 & T \end{bmatrix} \end{bmatrix}}_{\mathbb{Q}}} = \| \mathbb{A}\xi_t - \mathbb{Y} \|^2_{\mathbb{Q}}$$

A questo punto si può scrivere esplicitamente il contenuto del vettore ξ_t e suddividerlo nella sua componente nota e nella restante parte indipendente:

$$\xi_t = \begin{bmatrix} \hat{x}_t - \tilde{x}_t \\ \widehat{U}_t - \widetilde{U}_t \\ \bar{y}_{t+N} \end{bmatrix} = \begin{bmatrix} \hat{x}_t \\ \widehat{U}_t \\ \bar{y}_{t+N} \end{bmatrix} - \begin{bmatrix} \tilde{x}_t \\ \widetilde{U}_t \\ 0 \end{bmatrix} = \hat{\xi}_t - \tilde{\xi}_t$$

tenendo presente che $\tilde{\xi}_t$ è un vettore completamente noto ad ogni istante di tempo. Il problema di minimizzazione *MPC*, in assenza della componente di *collision avoidance,* diventa quindi:

$$\begin{aligned} \min_{\hat{\xi}_t} V^N(\hat{\xi}_t) &= \min_{\hat{\xi}_t} \left\| \mathbb{A}\,\hat{\xi}_t - \left(\mathbb{A}\tilde{\xi}_t + \mathbb{Y}\right) \right\|_{\mathbb{Q}} = \cdots \\ &= \min_{\hat{\xi}_t} \; \hat{\xi}_t^T \mathbb{A}^T \mathbb{Q}\, \mathbb{A}\, \hat{\xi}_t - 2\left(\mathbb{Y} + \mathbb{A}\tilde{\xi}_t\right)^T \mathbb{Q}\, \hat{\xi}_t + \left(\mathbb{A}\tilde{\xi}_t + \mathbb{Y}\right)^T \mathbb{Q} \left(\mathbb{A}\tilde{\xi}_t + \mathbb{Y}\right) \end{aligned}$$

tale forma quadratica deve essere risolta in presenza dei vincoli di cui si è discusso finora, che dovranno anch'essi essere ricondotti ad una funzione del vettore incognito $\hat{\xi}_t$.

6.4.2 – Formalizzazione dei vincoli.

Allo scopo di formalizzare correttamente i vincoli discussi consideriamo nuovamente la forma dell'*i-esimo* sistema sotto controllo che ricordiamo essere:

$$\begin{cases} x(k+1) = \overbrace{\begin{bmatrix} 1 & \tau & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & \tau \\ 0 & 0 & 0 & 0 \end{bmatrix}}^{A_s} x(k) + \overbrace{\begin{bmatrix} 0 & 0 \\ \tau & 0 \\ 0 & 0 \\ 0 & \tau \end{bmatrix}}^{B_s} u(k) \\ z(k) = \underbrace{\begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix}}_{C_s} x(k) \end{cases}$$

In realtà, per avere un risultato più accurato rispetto a quanto visto nel capitolo precedente, è possibile fare ricorso alla discretizzazione esatta che porta ad avere il medesimo blocco A_s mentre per il blocco B_s la forma

$$B_s = \begin{bmatrix} \tau^2/2 \\ \tau \end{bmatrix}$$

Prendiamo il sottosistema ridotto formato dalle matrici (A_s, B_s, C_s). Una volta effettuati i calcoli sulla base di questo, che risulteranno di conseguenza computazionalmente più economici, il risultato finale sarà ottenuto assemblando opportunamente i vari blocchi trovati.

Utilizzando questa formulazione è possibile riscrivere l'equazione dell'osservatore in forma compatta come

$$\begin{bmatrix} \tilde{x}_{k+1}^S \\ \tilde{u}_{k+1}^S \end{bmatrix} = \underbrace{\begin{bmatrix} A_s & B_s \\ 0 & 1 \end{bmatrix}}_{\tilde{A}_s} \begin{bmatrix} \tilde{x}_k^S \\ \tilde{u}_k^S \end{bmatrix} + \underbrace{\begin{bmatrix} G^{x,S} \\ G^{u,S} \end{bmatrix}}_{\tilde{G}_s} \left(\tilde{y}_{k+1}^S - \underbrace{[C_s \quad 0]}_{\tilde{C}_s} \begin{bmatrix} \tilde{x}_k^S \\ \tilde{u}_k^S \end{bmatrix} \right)$$

e quindi, a partire dalla definizione del *set* quadrato $B_\epsilon(0)$, che nel caso ridotto sarà semplicemente un intervallo di lato 2ε, costruire:

$$\Delta_i^{SS} = -\begin{bmatrix} I_2 - A_s & -B_s \\ -C_s & 0 \end{bmatrix}^{-1} \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} B_\epsilon(0)$$

e ancora scrivere:

$$\delta x_{t+1} = \left(\tilde{A}_s + \tilde{G}_s \tilde{C}_s\right) \delta x_t^S + \tilde{w}_t \qquad \tilde{w}_t \in \left(\tilde{A}_s + \tilde{G}_s + \tilde{C}_s\right) \Delta_i^{SS} = \widetilde{\mathbb{W}}$$

A questo punto allora, ricordando che per costruzione la matrice $\left(\tilde{A}_s + \tilde{G}_s \tilde{C}_s\right)$ deve essere Schur, è possibile definire il set *RPI* Δ^{xu} come

$$\Delta^{xu} = \bigoplus_{k=0}^{\infty} \left(\tilde{A}_s + \tilde{G}_s \tilde{C}_s\right)^k \widetilde{\mathbb{W}}$$

oppure, all'atto pratico, una sua δ *outer approximation,* in modo da avere poi come cercato

$$\tilde{y}_{t+1}^S - C_s \tilde{x}_t^S \in -[C_s \quad 0] \Delta^{xy} \oplus B_\epsilon(0) = \Delta^y$$

Fatto questo è possibile concentrarsi sull'applicazione dei vincoli al problema di ottimizzazione da riscrivere attraverso la variabile $\hat{\xi}_t = \{\hat{x}_t \mid \hat{U}_t \mid \bar{y}_{t+N}\}$.

In particolare si prenda innanzitutto la prima delle condizioni viste, $x_t^{[i]} - \hat{x}_t^{[i]} \in \mathcal{E}_i$, e, a partire dal *set* costruito sul sottosistema ridotto $\mathcal{E}_i^S$, si ricavi la sua formulazione matriciale:

$$\mathcal{E}_i^S = \{\, \varepsilon^S :\ H_\varepsilon^S\, \varepsilon^S \le L_\varepsilon^S \,\} \qquad \rightarrow \qquad \mathcal{E}_i = \left\{ \varepsilon : \begin{bmatrix} H_\varepsilon^S & 0 \\ 0 & H_\varepsilon^S \end{bmatrix} \varepsilon \le \begin{bmatrix} L_\varepsilon^S \\ L_\varepsilon^S \end{bmatrix} \right\}$$

da questa, calcolata una volta per tutte *offline*, avremo a disposizione la forma necessaria all'algoritmo di ottimizzazione per imporre il primo vincolo:

$$H_\varepsilon(x_t - \hat{x}_t) \le L_\varepsilon \quad \rightarrow \quad -H_\varepsilon \le -H_\varepsilon x_t + L_\varepsilon \quad \rightarrow \quad -H_\varepsilon [I_4 \quad 0 \quad 0] \hat{\xi}_t \le -H_\varepsilon\, x_t + L_\varepsilon$$

Il secondo vincolo di cui è necessario tener conto riguarda il riferimento di posizione $\bar{y}_{t+N}$ ed in particolare abbiamo visto che è necessario richiedere che sia $\bar{y}_{t+N} - \tilde{y}_{t+N-1} \in B_\epsilon(0)$. Sfruttando il fatto che sul sistema singolo il *set* $B_\epsilon(0)$ è un semplice intervallo si può richiedere

$$-\varepsilon \le \bar{y}_{t+N}^S - \tilde{y}_{t+N-1}^S \le \varepsilon \quad \rightarrow \quad \begin{cases} \bar{y}_{t+N}^S \le \tilde{y}_{t+N-1}^S + \varepsilon \\ -\bar{y}_{t+N}^S \le -\tilde{y}_{t+N-1}^S + \varepsilon \end{cases}$$

di conseguenza ottenere per il sistema originale la condizione

$$\begin{cases} I\, \bar{y}_{t+N} \le \tilde{y}_{t+N-1} + \begin{bmatrix} \varepsilon \\ \varepsilon \end{bmatrix} \\ -I\, \bar{y}_{t+N} \le -\tilde{y}_{t+N-1} + \begin{bmatrix} \varepsilon \\ \varepsilon \end{bmatrix} \end{cases} \quad \rightarrow \quad \begin{bmatrix} I_2 \\ -I_2 \end{bmatrix} \bar{y}_{t+N} \le \begin{bmatrix} \tilde{y}_{t+N-1} + \begin{bmatrix} \varepsilon \\ \varepsilon \end{bmatrix} \\ -\tilde{y}_{t+N-1} + \begin{bmatrix} \varepsilon \\ \varepsilon \end{bmatrix} \end{bmatrix}$$

Quindi, ricordando che $\bar{y}_{t+N}$ è l'ultimo elemento del vettore $\hat{\xi}_t$, si ricava la forma finale da inserire nel problema:

$$\left[0 \quad 0 \quad \begin{bmatrix} I_2 \\ -I_2 \end{bmatrix} \right] \hat{\xi}_t \le \begin{bmatrix} \tilde{y}_{t+N-1} + \begin{bmatrix} \varepsilon \\ \varepsilon \end{bmatrix} \\ -\tilde{y}_{t+N-1} + \begin{bmatrix} \varepsilon \\ \varepsilon \end{bmatrix} \end{bmatrix}$$

Le condizioni successive devono essere scritte singolarmente per ciascuna variabile all'interno dell'orizzonte predittivo $k = t \dots t + N - 1$; per questo motivo è necessario trovare esplicitamente il valore dello stato del sistema nominale $\hat{x}_{t+k}$ in funzione dello stato iniziale $\hat{x}_t$ rispetto al quale si ottimizza, secondo l'approccio predittivo. In particolare è possibile definire ciascuna riga delle matrici allargate $\mathcal{A}$ e $\mathcal{B}$ come $\mathcal{A}_k$ e $\mathcal{B}_k$ e scrivere, sfruttando il modello per predire i valori futuri, la seguente relazione:

$$\hat{x}_{t+k} - \tilde{x}_{t+k} = [\mathcal{A}_k \quad \mathcal{B}_k \quad 0]\, \hat{\xi}_t - [\mathcal{A}_k \quad \mathcal{B}_k \quad 0]\, \tilde{\xi}_t$$

Tenendo presente tale risultato è possibile andare ad imporre il vincolo $C(\hat{x}_k - \tilde{x}_k) \in \Delta^z$ ricordando che il *set* Δ^z può essere scelto a piacere e che nel caso ridotto coincide anch'esso con un semplice intervallo di dimensione arbitraria. Si ottiene quindi la condizione:

$$\begin{cases} C\,(\hat{x}_k - \tilde{x}_k) \le \begin{bmatrix}\delta\\ \delta\end{bmatrix} \\ -C\,(\hat{x}_k - \tilde{x}_k) \le \begin{bmatrix}\delta\\ \delta\end{bmatrix} \end{cases} \quad \rightarrow \quad \begin{bmatrix} C \\ -C \end{bmatrix} [\mathcal{A}_k \quad \mathcal{B}_k \quad 0]\,\hat{\xi}_t \le \begin{bmatrix} C \\ -C \end{bmatrix} [\mathcal{A}_k \quad \mathcal{B}_k \quad 0]\,\tilde{\xi}_t + \begin{bmatrix}\delta\\ \delta\\ \delta\\ \delta\end{bmatrix}$$

Allo stesso modo è possibile inserire il vincolo $\hat{x}_k \in \hat{\mathbb{X}}$. In realtà questa volta è necessario costruire il set $\hat{\mathbb{X}}$ a partire dall'insieme di ammissibilità delle variabili di stato vere e proprie. Tale set sarà in generale più piccolo dal momento che, noto $x_k \in \mathbb{X}$ e ricordando che vale il primo dei vincoli di cui abbiamo discusso, si hanno le relazioni

$$x_t - \hat{x}_t \in \mathcal{E} \qquad x_t \in \mathbb{X} \qquad \rightarrow \qquad \hat{x}_t \in \mathbb{X} \ominus \mathcal{E} = \hat{\mathbb{X}}$$

Fatto questo non resta quindi che riformulare il vincolo nella sua notazione matriciale (H_x, L_x) e poi usare la relazione su $\hat{x}_{t+k}$ vista per imporre la condizione

$$H_x[\mathcal{A}_k \quad \mathcal{B}_k \quad 0]\hat{\xi}_t \le L_x + H_x\big([\mathcal{A}_k \quad \mathcal{B}_k \quad 0]\tilde{\xi}_t - \tilde{x}_{t+k}\big)$$

direttamente nel problema di ottimizzazione.

Infine, gli ultimi vincoli che restano da riformulare sono quelli relativi al set terminale, ovvero riguardanti il valore assunto dalle variabili all'istante $t+N$; in particolare è necessario richiedere che

$$\begin{bmatrix}\hat{x}_{t+N}\\ \bar{y}_{t+N}\end{bmatrix} \in \Xi^F$$

oppure, in altre parole, che siano soddisfatte le relazioni seguenti:

$$\begin{cases} \hat{x}_{t+N} - \bar{x}_{t+N}(\bar{y}_{t+N}) \in \Sigma \\ \bar{x}_{t+N}(\bar{y}_{t+N}) \in \hat{\mathbb{X}} \ominus \Sigma \\ \bar{u}_{t+N}(\bar{y}_{t+N}) \in \hat{\mathbb{U}} \ominus K\Sigma \end{cases}$$

Abbiamo già visto che Σ rappresenta un *set* positivamente robustamente invariante per l'equazione:

$$\delta x_{t+1} = (A + BK)\delta x_t + w_t = F\delta x_t + w_t \qquad w_t \in \mathbb{W}_{arb}$$

Dal momento che per ipotesi la matrice $F = (A + BK)$ è Schur il *set* terminale Σ si calcola come δ *outer approximation* [15] di:

$$\Sigma_o = \bigoplus_{k=0}^{\infty} F^k\, \mathbb{W}_{arb}$$

Nella formulazione (H_Σ, L_Σ) si ottiene:

$$\Sigma = \{\sigma : H_\Sigma \sigma \le L_\Sigma\}$$

A questo punto, per implementare il primo dei vincoli, ricordiamo che $\hat{x}_{t+N} - \bar{x}_{t+N}$ può essere ottenuto facendo evolvere i termini all'istante t con l'equazione del sistema allargato:

$$\hat{x}_{t+N} - \bar{x}_{t+N} = [\mathcal{A}_N \quad \mathcal{B}_N \quad 0]\left(\hat{\xi}_t - \tilde{\xi}_t\right)$$

e poi riconvertire il tutto come:

$$H_\Sigma(\hat{x}_{t+N} - \bar{x}_{t+N}) \leq L_\Sigma \quad \rightarrow \quad H_\Sigma\,[\mathcal{A}_N \quad \mathcal{B}_N \quad 0]\,\hat{\xi}_t \leq H_\Sigma\,[\mathcal{A}_N \quad \mathcal{B}_N \quad 0]\,\tilde{\xi}_t + L_\Sigma$$

ottenendo l'espressione nella forma corretta per essere aggiunta al problema *MPC*.

Il secondo vincolo sul riferimento può essere considerato utilizzando l'espressione dell'osservatore per far evolvere il riferimento $\tilde{x}_{t+N-1}$ al nuovo termine $\bar{x}_{t+N}$, in particolare si ottiene la funzione di $\bar{y}_{t+N}$ già vista:

$$\bar{x}_{t+N} = (A - G^x C)\,\tilde{x}_{t+N-1} + B\,\tilde{u}_{t+N-1} + G^x\,\bar{y}_{t+N}$$

Il *set* da imporre si calcola come differenza di Minkowski tra $\hat{\mathbb{X}}$ e Σ; può inoltre essere riscritto in forma matriciale come:

$$\hat{\hat{\mathbb{X}}} = \hat{\mathbb{X}} \ominus \Sigma \qquad \rightarrow \qquad \hat{\hat{\mathbb{X}}} = \left\{x : H_{\hat{x}}\,x \leq L_{\hat{x}}\right\}$$

ed in questo modo utilizzato nella relazione complessiva

$$H_{\hat{x}}\,G^x[0 \quad 0 \quad I_2]\hat{\xi}_t \leq L_{\hat{x}} - H_{\hat{x}}\left((A - G^x C)\,\tilde{x}_{t+N-1} + B\,\tilde{u}_{t+N-1}\right)$$

da applicare anch'essa al problema in modo diretto.

L'ultimo vincolo che abbiamo definito è quello sulla variabile di controllo, ovvero, nella pratica, sui valori ammissibili per l'accelerazione del robot; tuttavia, dal momento che si è imposto un vincolo sull'evoluzione della variabile di riferimento y_t, tale grandezza è implicitamente limitata, poiché limitata è la velocità ideale con cui il robot si muove e di conseguenza le sue variazioni. Per questo motivo tale vincolo viene all'atto pratico ignorato.

Per quanto riguarda la specifica dell'insieme di ammissibilità $\mathbb{X}$ delle variabili di stato ricordiamo che esso è composto dalla posizione e dalla velocità lungo i due assi del sistema di riferimento cartesiano; per questo motivo, nella pratica, tale set sarà costituito dai limiti imposti all'area di lavoro ed alla velocità massima del robot reale, ovvero costruito come prodotto cartesiano:

$$\mathbb{X} = [-x_m \quad x_m] \times [-v_{xm} \quad v_{xm}] \times [-y_m \quad y_m] \times [-v_{ym} \quad v_{ym}]$$

Nel paragrafo precedente abbiamo costruito un problema di ottimizzazione *MPC* per ciascun singolo robot con l'obiettivo di mettere a disposizione, in ciascun istante di tempo, la propria traiettoria di riferimento unita all'incertezza con qui questa sarà sicuramente percorsa. Accanto a questo il problema è predisposto per far si che il robot,

nel suo movimento, si mantenga ad una certa distanza da eventuali ostacoli, che dovranno poi essere identificati con gli altri robot nell'area considerata. Il problema *MPC* da risolvere è quindi riassumendo il seguente:

$$\min_{\{\hat{x}_t\,,\widehat{u}_{[t:t+N-1]},\bar{y}_{t+N}\}} \sum_{k=t}^{t+N-1} \left\|\hat{x}_k^{[i]} - \tilde{x}_k^{[i]}\right\|_Q^2 + \left\|\hat{u}_k^{[i]} - \tilde{u}_k^{[i]}\right\|_R^2 + \left\|\hat{x}_{t+N}^{[i]} - \bar{x}_{t+N}^{[i]}\left(\bar{y}_{t+N}^{[i]}\right)\right\|_P^2$$
$$+ \gamma \left\|\bar{y}_{t+N}^{[i]} - \tilde{y}_{t+N-1}^{[i]}\right\|^2 + \left\|\bar{y}_{t+N}^{[i]} - \bar{y}_{setpoint}^{[i]}\right\|_T^2 + V_i^{coll}$$

$subject\ to$:

$x_t^{[i]} - \hat{x}_t^{[i]} \in \mathcal{E}_i$

$\bar{y}_{t+N}^{[i]} - \tilde{y}_{t+N-1}^{[i]} \in B_\epsilon(0)$

$\hat{x}_k^{[i]} \in \widehat{\mathbb{X}}_i \qquad \hat{u}_k^{[i]} \in \widehat{\mathbb{U}}_i \qquad \forall\, k = t \dots t+N-1$

$C\left(\hat{x}_k^{[i]} - \tilde{x}_k^{[i]}\right) \in \Delta^z \qquad \forall\, k = t \dots t+N-1$

$\hat{x}_{t+N}^{[i]} - \bar{x}_{t+N}^{[i]}\left(\bar{y}_{t+N}^{[i]}\right) \in \Sigma_i$

$\bar{x}_{t+N}^{[i]}\left(\bar{y}_{t+N}^{[i]}\right) \in \widehat{\mathbb{X}} \oplus \Sigma_i$

$\bar{u}_{t+N}^{[i]}\left(\bar{y}_{t+N}^{[i]}\right) \in \widehat{\mathbb{U}} \oplus K\Sigma_i$

Per quanto riguarda il termine di *collision avoidance* si è scelto, per ottenere una funzione di costo che fosse continua e che avesse derivate continue in corrispondenza del bordo d_{lim} , di utilizzare una funzione di costo costruita *ad hoc*. In particolare ricordiamo che, affinché si possa ottenere l'effetto di *avoidance* dall'ostacolo considerato, è necessario fondamentalmente avere una funzione il cui valore salga velocemente con il diminuire della distanza da esso fino a valori idealmente illimitati; in questo modo infatti, l'algoritmo di ottimizzazione, che deve ridurre il valore complessivo della cifra di merito, farà in modo di mantenersi il più possibile distante dalla zona critica e quindi, all'atto pratico, aggirare l'ostacolo lungo il bordo dell'area considerata.

Secondo quanto già detto nel Capitolo 4 si è deciso di fare ricorso ad una funzione polinomiale, costruita interpolando tre punti, ed aggiungendo le condizioni desiderate sulle derivate nel punto di congiunzione. Più nel dettaglio si è scelto un polinomio di grado $n = 5$ che permette di specificare esplicitamente sei condizioni ovvero, nel nostro caso, di imporre il passaggio per i tre punti ed il valore della funzione e delle sue due prime derivate in corrispondenza dell'estremo.

Dopo aver posto come argomento del polinomio il quadrato della distanza tra il riferimento $\bar{y}_{t+N}$ e la posizione dell'ostacolo:

$$\rho = \left\| D\hat{\xi}_t - Cx_{obst} \right\|^2$$

si ha quindi come termine penalizzante:

$$f(\rho) = a_0 + a_1\rho + a_2\rho^2 + a_3\rho^3 + a_4\rho^4 + a_5\rho^5$$

Supponendo di porre l'ostacolo nell'origine e l'estremo in corrispondenza del valore 3ε le condizioni che si possono imporre per ottenere le garanzie desiderate sono allora, ad esempio:

$$f(0) = 1e^4 \qquad f(\varepsilon) = 5e^3 \qquad f(2\varepsilon) = 1e^3 \quad f(3\varepsilon) = 0 \quad f'(3\varepsilon) = 0 \qquad f''(3\varepsilon) = 0$$

6.4.3 – Dettagli sull'implementazione numerica.

Nel capitolo precedente è stato descritto il problema di ottimizzazione che costituisce l'algoritmo di *MPC* cooperativo sviluppato.

All'atto pratico abbiamo visto che il tutto può essere implementato ricordando che sono note le variabili $\tilde{y}_{[t:t+N-1]}$, $\tilde{x}_{[t:t+N-1]}$ ed $\tilde{u}_{[t:t+N-1]}$ e definendo sulla base di queste

$$\hat{\xi}_t = \begin{bmatrix} \hat{x}_t \\ \hat{u}_{[t:t+N-1]} \\ \bar{y}_{t+N} \end{bmatrix} \quad \tilde{\xi}_t = \begin{bmatrix} \tilde{x}_t \\ \tilde{u}_{[t:t+N-1]} \\ 0 \end{bmatrix} \quad \mathbb{A} = \begin{bmatrix} \mathcal{A} & \mathcal{B} & 0 \\ 0 & I & 0 \\ 0 & 0 & \begin{bmatrix} I \\ I \end{bmatrix} \end{bmatrix} \quad \mathbb{Y} = \begin{bmatrix} 0 \\ 0 \\ \tilde{y}_{t+N-1} \\ \bar{y}_{setpoint} \end{bmatrix} \quad \mathbb{Q}$$
$$= \begin{bmatrix} \mathcal{Q} & 0 & 0 \\ 0 & \mathcal{R} & 0 \\ 0 & 0 & \begin{bmatrix} \gamma I & 0 \\ 0 & T \end{bmatrix} \end{bmatrix}$$

Il problema di ottimizzazione diventa:

$$\min_{\hat{\xi}_t} \ \hat{\xi}_t^T \mathbb{A}^T \mathbb{Q}\, \mathbb{A}\, \hat{\xi}_t - 2\left(\mathbb{Y} + \mathbb{A}\tilde{\xi}_t\right)^T \mathbb{Q}\, \hat{\xi}_t + \left(\mathbb{A}\tilde{\xi}_t + \mathbb{Y}\right)^T \mathbb{Q} \left(\mathbb{A}\tilde{\xi}_t + \mathbb{Y}\right) + V_i^{coll}$$

da risolvere in presenza dei vincoli seguenti

$$-H_\varepsilon [I_4 \quad 0 \quad 0]\hat{\xi}_t \leq -H_\varepsilon\, x_t + L_\varepsilon$$

$$\begin{bmatrix} 0 & 0 & \begin{bmatrix} I_2 \\ -I_2 \end{bmatrix} \end{bmatrix} \hat{\xi}_t \leq \begin{bmatrix} \tilde{y}_{t+N-1} + \begin{bmatrix} \varepsilon \\ \varepsilon \end{bmatrix} \\ -\tilde{y}_{t+N-1} + \begin{bmatrix} \varepsilon \\ \varepsilon \end{bmatrix} \end{bmatrix}$$

$$\begin{bmatrix} C \\ -C \end{bmatrix} [\mathcal{A}_k \quad \mathcal{B}_k \quad 0]\, \hat{\xi}_t \leq \begin{bmatrix} C \\ -C \end{bmatrix} \tilde{x}_k + \begin{bmatrix} \delta \\ \delta \\ \delta \\ \delta \end{bmatrix}$$

$$H_x[\mathcal{A}_k \quad \mathcal{B}_k \quad 0]\hat{\xi}_t \leq L_x + H_x\big([\mathcal{A}_k \quad \mathcal{B}_k \quad 0]\tilde{\xi}_t - \tilde{x}_{t+k}\big)$$

$$H_\Sigma\, [\mathcal{A}_N \quad \mathcal{B}_N \quad 0]\, \hat{\xi}_t \leq H_\Sigma\, [\mathcal{A}_N \quad \mathcal{B}_N \quad 0]\, \tilde{\xi}_t + L_\Sigma$$

$$H_{\hat{x}}\, G^x[0 \quad 0 \quad I_2]\hat{\xi}_t \leq L_{\hat{x}} - H_{\hat{x}}\left((A - G^x C)\, \tilde{x}_{t+N-1} + B\, \tilde{u}_{t+N-1}\right)$$

ricavati definendo, secondo le linee guida viste, ciascuno dei singoli *set* che compaiono nell'espressione.

Se non fosse per il termine penalizzante, che tiene in conto della *collision avoidance*, tale problema di ottimizzazione sarebbe puramente quadratico e sottoposto a vincoli lineari, per cui risulterebbe relativamente semplice da risolvere, e di conseguenza computazionalmente poco dispendioso. A causa di quest'ultima componente, invece, torniamo di nuovo al problema, discusso in precedenza, di dover utilizzare un algoritmo di risoluzione che consenta di specificare esplicitamente oltre alla funzione di costo anche l'espressione in forma chiusa del suo Jacobiano e del suo Hessiano, così da facilitare la convergenza.

Consideriamo innanzitutto il calcolo dello Jacobiano. Per quanto riguarda la componente quadratica della cifra di costo nominale si ha semplicemente l'espressione

$$\frac{\partial V_1^N}{\partial \hat{\xi}_t} = 2\, \hat{\xi}_t^T\, \mathbb{A}^T \mathbb{Q}\, \mathbb{A} - 2\big(\mathbb{Y} + \mathbb{A}\tilde{\xi}_t\big)^T \mathbb{Q}$$

Per quanto riguarda, invece, il termine di penalizzazione relativo agli ostacoli si è scelto di utilizzare per la rappresentazione del vincolo una funzione polinomiale, come visto in dettaglio al Capitolo 4, ed applicare il tutto al riferimento $\bar{y}_{t+N}$ che deve produrre l'algoritmo di ottimizzazione.

In tal caso, ricordando che si è definita la variabile $\rho = \left\| D\hat{\xi}_t - Cx_{obst} \right\|^2$, per ciascuno degli ostacoli e per ciascun campione $t + k$ all'interno dell'orizzonte predittivo $k = 0 \dots N - 1$ la componente legata allo stato del sistema nominale ha come derivata la seguente

$$\frac{\partial f}{\partial \hat{\xi}_t} = \frac{\partial f}{\partial \rho}\frac{\partial \rho}{\partial \hat{\xi}_t} = (a_1 + 2a_2\rho + 3a_3\rho^2 + 4a_4\rho^3 + 5a_5\rho\text{^}4)\, 2\big(D\hat{\xi}_t - Cx_{obst}\big)^T D$$

Questa, estesa a ciascun ostacolo x_{obst}, va a costituire lo Jacobiano desiderato.

Per quanto riguarda l'Hessiano è necessario invece calcolare ciascun elemento come la derivata, fatta rispetto alla variabile di ottimizzazione $\hat{\xi}_t$, della precedente espressione trasposta, ottenendo:

$$\frac{\partial^2 f}{\partial \hat{\xi}_t^2} = 4D^T\left(D\hat{\xi}_t - Cx_{obst}\right)(2a_2 + 6a_3\rho + 12a_4\rho^2 + 20a_5\rho^3\ )\left(D\hat{\xi}_t - Cx_{obst}\right)^T D$$
$$+ (a_1 + 2a_2\rho + 3a_3\rho^2 + 4a_4\rho^3 + 5a_5\rho\text{^}4)\ 2D^T D$$

Ricorrendo all'espressione del vettore dei coefficienti, calcolato *offline,* a_v possiamo riscrivere il tutto nella forma sintetica utile per l'implementazione:

$$f'(\rho) = [0 \quad 1 \quad 2\rho \quad 3\rho^2 \quad 4\rho^3 \quad 5\rho^4]\ a_v\ 2\left(D\hat{\xi}_t - Cx_{obst}\right)^T D$$

$$f''(\rho) = 2\Big([0 \quad 1 \quad 2\rho \quad 3\rho^2 \quad 4\rho^3 \quad 5\rho^4]a_v\ D^T D$$
$$+ [0 \quad 0 \quad 2 \quad 6\rho \quad 12\rho^2 \quad 20\rho^3]a_v\left(D\hat{\xi}_t - Cx_{obst}\right)^T D\Big)$$

In questo modo siamo in grado di fornire direttamente all'algoritmo di ottimizzazione le informazioni necessarie al raggiungimento della convergenza, evitando il passaggio per la derivazione numerica delle funzioni trovate e, di conseguenza, aumentando le prestazioni del sistema.

Torniamo ora a considerare il problema di definizione del *set* $B_\epsilon(0)$; in particolare si è visto che non è possibile costruirlo sul sottosistema ridotto come intervallo $\pm\varepsilon$ a meno di non vincolare il sistema a percorrere delle traiettorie preferenziali troppo restrittive. Non potendo estendere il tutto ad un cerchio, e non volendo complicare il problema di calcolo dei *set* restanti, l'idea alternativa è allora quella di mantenere la formulazione a quadrato per i calcoli iniziali, e poi ricorrere a figure con un numero di lati maggiore per l'applicazione del vincolo su $\bar{y}_{t+N}$. Tale approccio risulta valido a patto di garantire che le nuove figure scelte definiscano in realtà un *set* più piccolo di quello di partenza, ovvero che il vincolo applicato realmente sia più restrittivo di quanto necessario. In questo modo, tuttavia, la maggior conservatività viene ripagata con l'aumento del numero di possibili direzioni preferenziali tra le quali il robot può scegliere, e pertanto con un miglioramento notevole della mobilità complessiva.

In dettaglio, per garantire quanto richiesto ed essere sicuri di non restringere eccessivamente la dimensione del set, si è scelto di far ricorso a figure geometriche inscritte in un cerchio di raggio ε ottenendo in questo modo ad esempio:

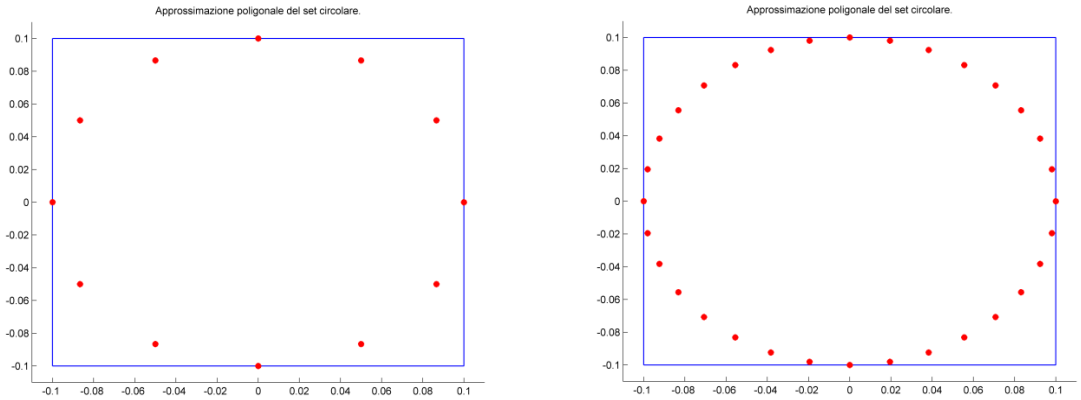

Fig. 6.4.2.1 *– Costruzione geometrica del set $B_\epsilon(0)$. Sulla sinistra un poligono con 12 lati, sulla destra un poligono con 32 lati.*

Per utilizzare questa struttura è necessario rivedere l'applicazione del vincolo $\bar{y}_{t+N}^{[i]} - \tilde{y}_{t+N-1}^{[i]} \in B_\epsilon(0)$ che finora era stato descritto semplicemente sfruttando il parametro ε come:

$$\begin{bmatrix} 0 & 0 & \begin{bmatrix} I_2 \\ -I_2 \end{bmatrix} \end{bmatrix} \hat{\xi}_t \leq \begin{bmatrix} \tilde{y}_{t+N-1} + \begin{bmatrix} \varepsilon \\ \varepsilon \end{bmatrix} \\ -\tilde{y}_{t+N-1} + \begin{bmatrix} \varepsilon \\ \varepsilon \end{bmatrix} \end{bmatrix}$$

A tal proposito è vantaggioso passare dall'insieme di vertici, che descrive la forma geometrica più complessa, alla formulazione matriciale $(H_{B\varepsilon}, L_{B\varepsilon})$ già vista per altri tipi di vincoli, e poi implementare il tutto come:

$$H_{B\varepsilon}(\bar{y}_{t+N} - \tilde{y}_{t+N-1}) \leq L_{B\varepsilon} \qquad \rightarrow \qquad [0 \quad 0 \quad H_{B\varepsilon}]\, \hat{\xi}_t \leq H_{B\varepsilon}\, \tilde{y}_{t+N-1} + L_{B\varepsilon}$$

Si ricordi poi, ancora una volta, che il membro di sinistra può essere valutato *offline* mentre il termine noto di tale disuguaglianza dipende da un parametro variante, e pertanto deve essere ricalcolato ogni volta *online*.

Terminata la costruzione, che ricordiamo può essere fatta *offline*, abbiamo a disposizione la nuova cifra di penalizzazione da inserire nel problema *MPC* per ottenere la garanzia di assenza di collisioni. Esattamente come fatto per i casi precedenti possiamo a questo punto calcolare esplicitamente lo Jacobiano e l'Hessiano di tale funzione così da semplificare il problema numerico di ottimizzazione e velocizzare complessivamente l'esecuzione dell'algoritmo.

6.4.4 – Definizione degli agenti mobili come ostacolo.

È necessario determinare a questo punto una legge di priorità per le manovre di gestione del movimento dei robot. L'idea è infatti, come abbiamo visto, quella di costruire online, ad ogni passo, un vettore contenente la posizione dei possibili ostacoli per ciascuno dei robot e poi fornirlo in ingresso all'algoritmo di ottimizzazione. Tali ostacoli saranno in particolare ricavati a partire dalle informazioni trasmesse dagli agenti vicini in merito alla loro traiettoria di riferimento $\tilde{y}_{[t:t+N-1]}$ oppure $\tilde{x}_{[t+N-1]}$. Sotto tale ipotesi, allora, è importante decidere, una volta stabilito il numero di robot nell'area di lavoro, quali di questi sono in grado di influenzare il moto dell'agente *i-esimo,* ovvero in parole povere quali traiettorie di riferimento dei vicini trasformare in ostacoli per il robot considerato.

Questa scelta è molto interessante poiché consente di risolvere correttamente il problema di "gestione del traffico" evidenziando le particolari dinamiche che si presentano nelle interazioni reali tra pedoni oppure automobili etc… Ad esempio, se si considera il problema dell'incrocio per tre agenti mobili, è possibile scegliere il classico sistema di priorità circolare che si ha nella guida su strada:

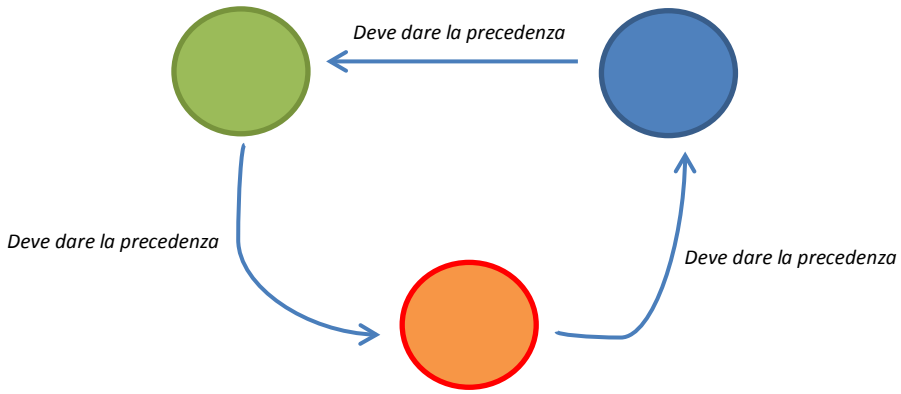

Fig. 6.4.4.1 *– Esempio di sistema di priorità circolare.*

oppure stabilire un ordine di priorità assoluto, come ad esempio:

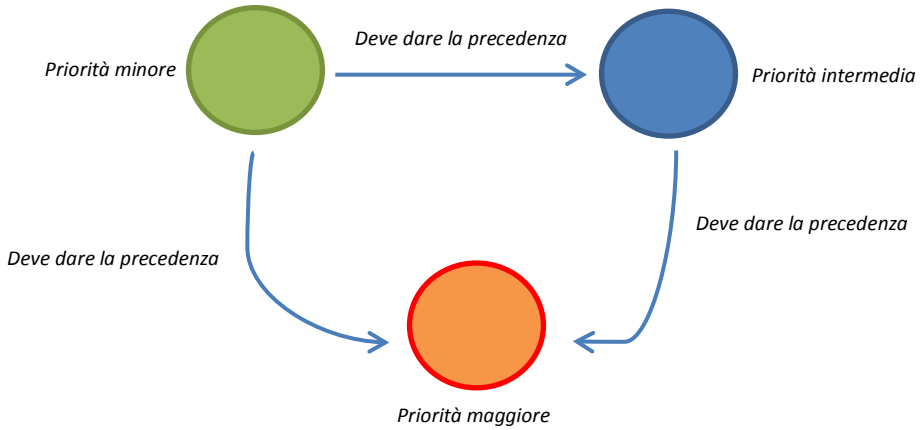

Fig. 6.4.4.2 *– Esempio di sistema di priorità assoluta.*

A seconda del caso scelto otteniamo allora un comportamento diverso del sistema complessivo che deve essere studiato. In particolare è necessario prestare attenzione a quali set di regole di precedenza garantiscano l'assenza di collisione tra gli agenti per il problema considerato, o d'altra parte l'assenza di posizioni di stallo.

7- IMPLEMENTAZIONE E RISULTATI

7.1 – Introduzione.

Nei capitoli precedenti abbiamo visto numerose soluzioni per il controllo e la stabilizzazione del singolo robot ed abbiamo verificato come, quella più semplice e congeniale ai nostri scopi, sia l'applicazione congiunta di un *feedback* dinamico linearizzante e di una tecnica *MPC* per sistemi lineari. Abbiamo visto poi come è possibile inserire in modo naturale, nel controllore sviluppato, la capacità di mantenere il robot lontano dagli ostacoli ed evitare di conseguenza le possibili situazioni di collisione. Infine si è costruita una tecnica in grado di risolvere il problema di coordinamento di robot multipli estendendo quanto visto nel caso singolo mediante l'utilizzo di un approccio *MPC* robusto ed una implementazione distribuita. Fatto questo si è discusso degli algoritmi di ottimizzazione disponibili e si è scelto di fare ricorso al solutore maggiormente efficiente presente nel pacchetto *TOMLAB,* che fornisce i medesimi risultati di un algoritmo generico in tempi computazionali più brevi, grazie alla possibilità di specificare agevolmente Jacobiano ed Hessiano della funzione di costo.

Tenendo presente questo panorama, ed alla luce delle considerazioni fatte finora, l'implementazione finale che si è scelto di utilizzare per i test può essere formalizzata nel problema seguente da applicare a ciascuno dei robot:

$$\min_{\hat{\xi}_t} \ \hat{\xi}_t^T \mathbb{A}^T \mathbb{Q}\, \mathbb{A}\, \hat{\xi}_t - 2\left(\mathbb{Y} + \mathbb{A}\tilde{\xi}_t\right)^T \mathbb{Q}\, \hat{\xi}_t + \left(\mathbb{A}\tilde{\xi}_t + \mathbb{Y}\right)^T \mathbb{Q}\left(\mathbb{A}\tilde{\xi}_t + \mathbb{Y}\right) + V_i^{coll}$$

$subject\ to$:

$$-H_\varepsilon [I_4 \quad 0 \quad 0]\hat{\xi}_t \leq -H_\varepsilon\, x_t + L_\varepsilon$$

$$[0 \quad 0 \quad H_{B\varepsilon}]\, \hat{\xi}_t \leq H_{B\varepsilon}\, \tilde{y}_{t+N-1} + L_{B\varepsilon}$$

$$\begin{bmatrix} C \\ -C \end{bmatrix} [\mathcal{A}_k \quad \mathcal{B}_k \quad 0]\, \hat{\xi}_t \leq \begin{bmatrix} C \\ -C \end{bmatrix} \tilde{x}_k + \begin{bmatrix} \delta \\ \delta \\ \delta \\ \delta \end{bmatrix}$$

$$H_x [\mathcal{A}_k \quad \mathcal{B}_k \quad 0]\hat{\xi}_t \leq L_x + H_x\left([\mathcal{A}_k \quad \mathcal{B}_k \quad 0]\tilde{\xi}_t - \tilde{x}_{t+k}\right)$$

$$H_\Sigma\, [\mathcal{A}_N \quad \mathcal{B}_N \quad 0]\, \hat{\xi}_t \leq H_\Sigma\, [\mathcal{A}_N \quad \mathcal{B}_N \quad 0]\, \tilde{\xi}_t + L_\Sigma$$

$$H_{\hat{x}}\, G^x [0 \quad 0 \quad I_2]\hat{\xi}_t \leq L_{\hat{x}} - H_{\hat{x}}\left(\left((A - G^x C)\, \tilde{x}_{t+N-1} + B\, \tilde{u}_{t+N-1}\right)\right)$$

Nell'espressione precedente si fa riferimento alle grandezze

$$\hat{\xi}_t = \begin{bmatrix} \hat{x}_t \\ \hat{u}_{[t:t+N-1]} \\ \bar{y}_{t+N} \end{bmatrix} \qquad \tilde{\xi}_t = \begin{bmatrix} \tilde{x}_t \\ \tilde{u}_{[t:t+N-1]} \\ 0 \end{bmatrix}$$

$$\mathbb{A} = \begin{bmatrix} \mathcal{A} & \mathcal{B} & 0 \\ 0 & I & 0 \\ 0 & 0 & \begin{bmatrix} I \\ I \end{bmatrix} \end{bmatrix} \quad \mathbb{Y} = \begin{bmatrix} 0 \\ 0 \\ \tilde{y}_{t+N-1} \\ \bar{y}_{setpoint} \end{bmatrix} \quad \mathbb{Q} = \begin{bmatrix} \mathcal{Q} & 0 & 0 \\ 0 & \mathcal{R} & 0 \\ 0 & 0 & \begin{bmatrix} \gamma I & 0 \\ 0 & T \end{bmatrix} \end{bmatrix}$$

Per quanto riguarda la cifra di merito penalizzante, applicata agli ostacoli per garantire la *collision* avoidance, si è scelta la forma polinomiale già discussa nel Capitolo 4, che ricordiamo essere:

$$\rho = \left\| D\hat{\xi}_t - Cx_{obst} \right\|^2$$

$$f(\rho) = a_0 + a_1\rho + a_2\rho^2 + a_3\rho^3 + a_4\rho^4 + a_5\rho^5$$

$$\sum_{p=0}^{M} \begin{cases} f\left(\left\| D\hat{\xi}_t - Cx_{obst,p} \right\|^2\right) & \left\| D\hat{\xi}_t - Cx_{obst,p} \right\|^2 \leq d_{lim} \\ 0 & \left\| D\hat{\xi}_t - Cx_{obst,p} \right\|^2 > d_{lim} \end{cases}$$

L'insieme degli M ostacoli che l'algoritmo prende in considerazione è costituito da tutti i riferimenti $\tilde{y}_{t+N}$ calcolati per gli altri robot interagenti, ovvero:

$$\mathcal{O}^{[i]} = \left\{ obst^{[i]} : obst^{[i]} = \tilde{y}_{t+N-1}^{[j]} \ \ \forall j \neq i \right\}$$

7.2 – Implementazione dell'algoritmo.

L'implementazione dell'intero algoritmo può essere schematizzata al modo seguente:

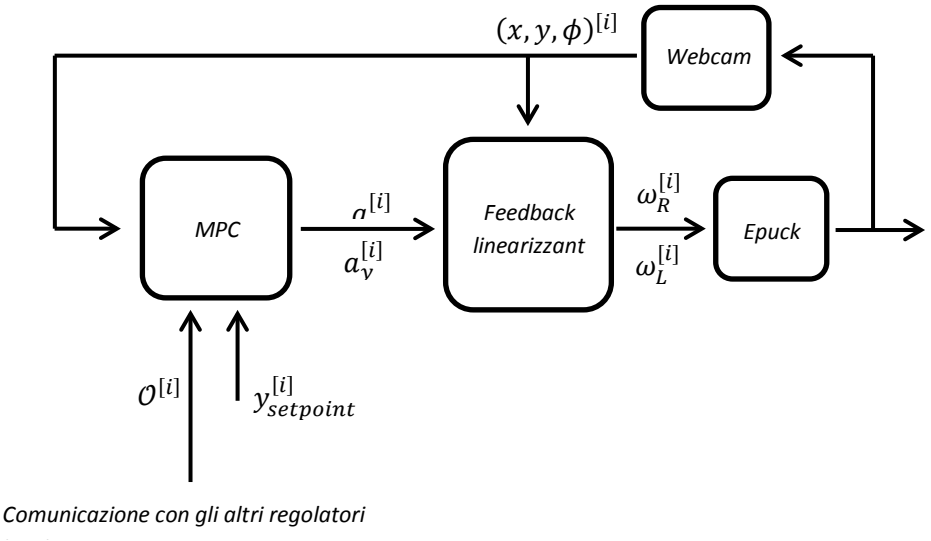

***Fig. 7.2.1**– Schema dell'algoritmo di coordinamento.*

Come descritto nel Capitolo 2 si è utilizzato uno strato *software,* implementato in *MATLAB,* per la gestione della *webcam* e della comunicazione *wireless* verso ciascun robot; in questo modo è possibile considerare sempre disponibile, ad ogni passo di campionamento, la posizione e l'orientamento dei singoli agenti sul piano di lavoro e nello stesso tempo è possibile comandare questi imponendo le velocità di rotazione delle due ruote.

L'anello di controllo interno che si occupa di realizzare il *feedback* linearizzante è implementato ad una frequenza superiore a quella dell'anello esterno *MPC;* questa scelta è possibile data la ridotta complessità computazionale dei calcoli da eseguire e nello stesso tempo garantisce una maggior efficacia nell'integrazione numerica in avanti della variabile di controllo accelerazione grazie alle ripetute rilevazioni dello stato del sistema. Il sistema di regolazione esterno, invece, necessita di tempi computazionali superiori per la risoluzione del problema di ottimizzazione e pertanto deve essere utilizzato con una frequenza ridotta; ad ogni ciclo ciascun robot comunica ai regolatori interessati la posizione del proprio riferimento futuro $\tilde{y}_{t+N}$ che potrà essere considerata come un ostacolo da evitare.

Tutti i calcoli riguardanti la definizione delle matrici del sistema, la costruzione degli equivalenti allargati per il problema *MPC*, l'impostazione dei vincoli e la costruzione dei *set* sono eseguiti *offline* prima di avviare l'esecuzione dell'algoritmo di controllo. I passi da eseguire *online* sono invece il rilevamento della posizione di ciascuno dei robot, la

costruzione della colonna di termini noti che compare nell'equazione dei vincoli, l'aggiornamento dell'insieme degli ostacoli sulla base delle regole di priorità scelte, l'ottimizzazione della cifra di merito vista, l'aggiornamento dei riferimenti secondo le equazioni dell'osservatore di cui si è già discusso nel Capitolo 6 ed infine la costruzione degli ingressi di controllo a_x ed a_y da inviare all'anello interno.

Secondo le indicazioni già fornite nella trattazione precedente, per facilitare il calcolo del valore ottimo della cifra di costo ed abbassare la complessità computazionale dell'algoritmo, riducendo di conseguenza i tempi di calcolo, è necessario costruire esplicitamente anche lo Jacobiano e l'Hessiano della funzione obiettivo. Anche questo può essere fatto *offline* direttamente nel momento della programmazione.

7.3 – Prove di posizionamento in simulazione.

Per testare il funzionamento dell'algoritmo presentato in questo lavoro si è scelto di fare ricorso innanzitutto ad un simulatore costruito sulla base del modello cinematico del singolo robot di cui si è discusso al Capitolo 2. Questo consente di ricostruire facilmente un'area di lavoro con uno svariato numero di robot interagenti non dovendo preoccuparsi delle tematiche pratiche riguardanti i tempi di campionamento reali e la complessità computazionale dell'algoritmo di calcolo, che deve essere ospitato interamente sul medesimo *PC,* emulando diversi regolatori con la capacità di comunicare tra loro. D'altra parte anche l'accesso diretto allo stato, senza dover passare dalla routine di gestione del sistema di visione, semplifica notevolmente le cose.

Completate le prove in simulazione si è deciso di verificare la validità dell'approccio anche sui sistemi reali ed innanzitutto si è scelto di limitare l'analisi a due soli robot interagenti. Sebbene gli algoritmi siano stati teoricamente ben definiti e ciascuna componente sia stata implementata e testata singolarmente, l'utilizzo combinato dei vari moduli presenta infatti alcune difficoltà pratiche e la taratura dei parametri richiede alcune accortezze.

Nel Capitolo 5 si è vista l'efficacia della tecnica *MPC* sviluppata per il controllo del movimento di un singolo robot; con il medesimo approccio la prima cosa che si è scelto di fare è stato di testare sul simulatore l'algoritmo di regolazione distribuito in presenza di un singolo robot. Lo scopo è quello di posizionarlo all'obiettivo finale in assenza di ostacoli, validando in questo modo la componente della cifra di merito che si occupa dell'evoluzione dei riferimenti intermedi. In particolare otteniamo il risultato di Figura 7.3.1 in cui vediamo in giallo la traiettoria ideale prodotta dall'ottimizzatore ed in verde la traiettoria effettivamente percorsa dal modello del robot ad uniciclo.

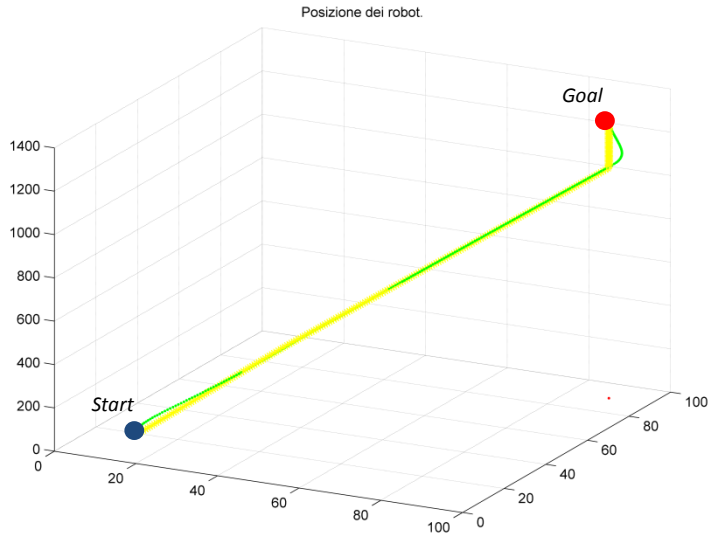

***Fig. 7.3.1**– Posizionamento di un singolo robot tramite algoritmo di MPC distribuito – simulazione.*

Si nota che, seppur con una leggera sovraelongazione, l'algoritmo è in grado di far evolvere il riferimento fino all'obiettivo e di vincolare il robot a seguirlo, completando il compito richiesto. L'entità della sovraelongazione è legata alla velocità ideale di spostamento e alla matrice dei guadagni scelta per l'osservatore in un senso che vedremo meglio nel seguito.

La stessa cosa può essere estesa alla navigazione parallela di due robot con traiettorie che non prevedono alcuna collisione. In tal caso otteniamo i risultati di Figura 7.3.2; il secondo robot è identificato dalla traiettoria in blu mentre in magenta si osserva la sequenza dei riferimenti generati dal suo regolatore.

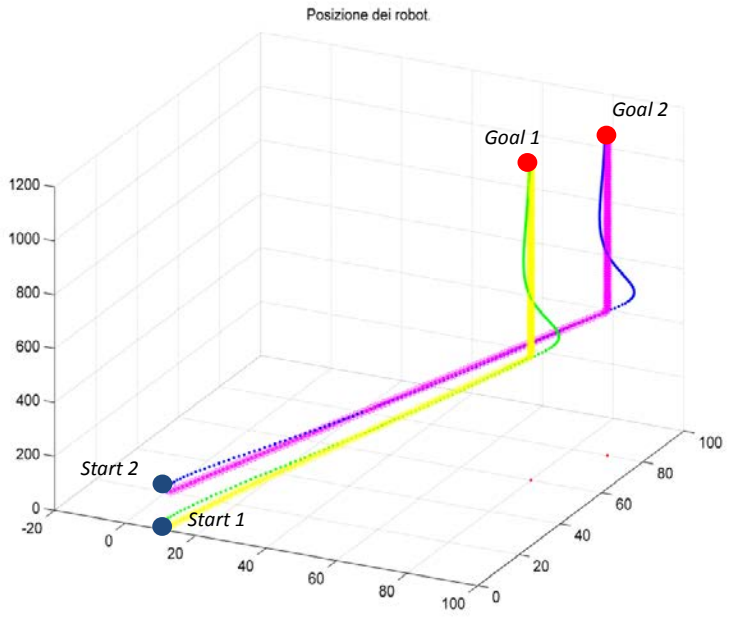

***Fig. 7.3.2**– Posizionamento di una coppia di robot tramite algoritmo di MPC distribuito in assenza di collisioni – simulazione.*

Di nuovo si può affermare che il simulatore e l'algoritmo di controllo che implementa funzionano correttamente.

7.4 – Prove di coordinamento in simulazione.

Il passo successivo è quello di valutare il funzionamento della componente di *avoidance* della cifra di merito scritta; per far questo si ricorre allo stesso setup visto in precedenza richiedendo tuttavia ai due robot di completare dei *task* di tipo concorrente, ovvero all'atto pratico di percorrere delle traiettorie che si sovrappongono. Si ottiene a tal proposito il risultato di Figura 7.4.1 in cui il significato dei colori è lo stesso visto nel diagramma precedente; per quanto riguarda la politica di gestione della collisione si è scelto innanzitutto di dare priorità ad uno dei due robot, costringendo l'altro alla manovra di evasione.

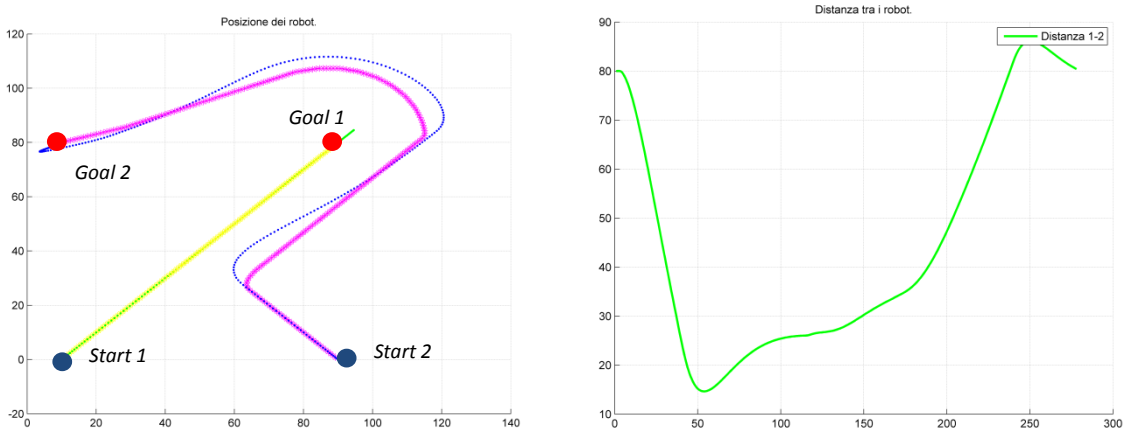

Fig. 7.4.1*– Coordinamento di due agenti con possibile collisione – simulazione.*

Il risultato ottenuto conferma il funzionamento della tecnica adottata, tuttavia la complicata manovra di aggiramento che il secondo agente deve compiere per lasciare priorità al primo nel completamento del *task* non rappresenta esattamente ciò che ci si può aspettare da un comportamento reale; l'idea migliore sarebbe infatti quella di far attendere il robot fino al passaggio del primo e solo poi riprendere con il movimento. Tale aspetto, di fondamentale importanza per la trattazione che stiamo facendo, sarà discusso con più dettaglio nel seguito.

Per le simulazioni successive si è scelto di fare riferimento ad un *set* di tre robot interagenti sul medesimo piano; a tal proposito, come accennato nel Capitolo 6, può essere interessante studiare il problema dell'incrocio stradale. Si sottolinea nuovamente che la costruzione di una regola per la gestione delle manovre anti collisione si riflette in realtà sulla composizione dei *set* di ostacoli $\mathcal{O}^{[i]}$ che ciascuno degli agenti considera nel proprio problema di ottimizzazione; a seconda della scelta dei robot in competizione si ottengono le diverse configurazioni possibili.

Supponiamo ad esempio di voler analizzare il comportamento che si ha quando ciascuno dei robot deve dare la precedenza a quello proveniente dalla propria destra; la configurazione che si ottiene è quella circolare già schematizzata. Il risultato della simulazione è visibile in Figura 7.4.2; interessante è anche osservare l'evoluzione della distanza tra ciascuno dei robot, a conferma dell'assenza di collisioni.

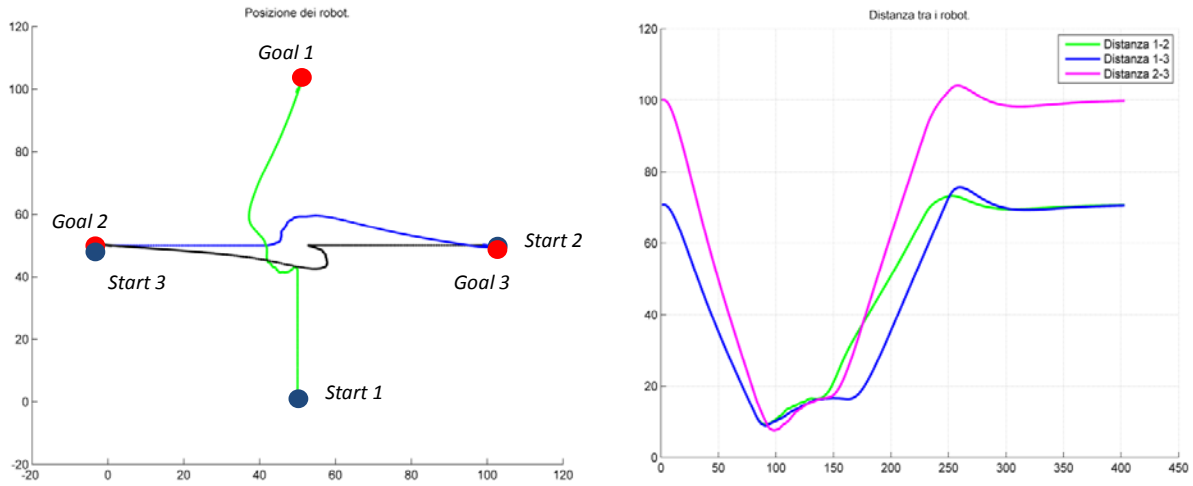

Fig. 7.4.2 *– Coordinamento di tre agenti in un incrocio.*
Priorità circolare – simulazione.

Allo stesso modo si potrebbe voler dare un ordine di precedenza tra i vari agenti; ad esempio si potrebbe chiedere al primo robot di muoversi in assenza di ostacoli, al secondo di avere come ostacolo solamente il primo ed infine al terzo di avere come ostacoli gli altri due. Il risultato che si ottiene è quello di Figura 7.4.3 in cui in verde vediamo il primo robot, in blu il secondo robot ed in viola il terzo. Di nuovo si può notare che per effetto della cifra di merito utilizzata la manovra di allontanamento di ciascuno dei robot assume un'ampia dimensione senza prevedere l'ipotesi di fermarsi ad attendere che l'ostacolo sia passato; tale aspetto porterà a concludere che la costruzione dei vincoli necessita di una diversa impostazione, oppure di una fine

taratura dei parametri che consenta di bilanciare al meglio ciascuna componente da ottimizzare.

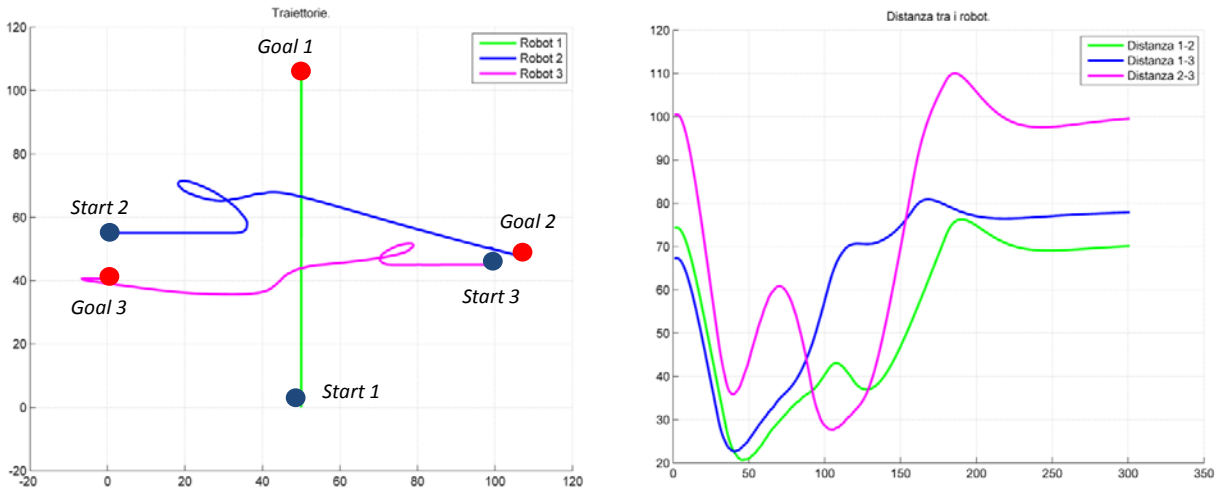

Fig. 7.4.3*– Coordinamento di tre agenti in un incrocio.*
Elenco di priorità – simulazione.

7.5 – Prove di posizionamento sul sistema reale.

Sulla base di quanto visto finora si è poi scelto di testare l'algoritmo sul sistema reale; in particolare, come accennato, si considera il problema di controllo di due soli agenti, distinti sul piano di lavoro dalla videocamera mediante *marker* di colore diverso. Il primo test effettuato è, come per la simulazione, quello di posizionamento contemporaneo in assenza di collisioni; il risultato è quello di Figura 7.5.1.

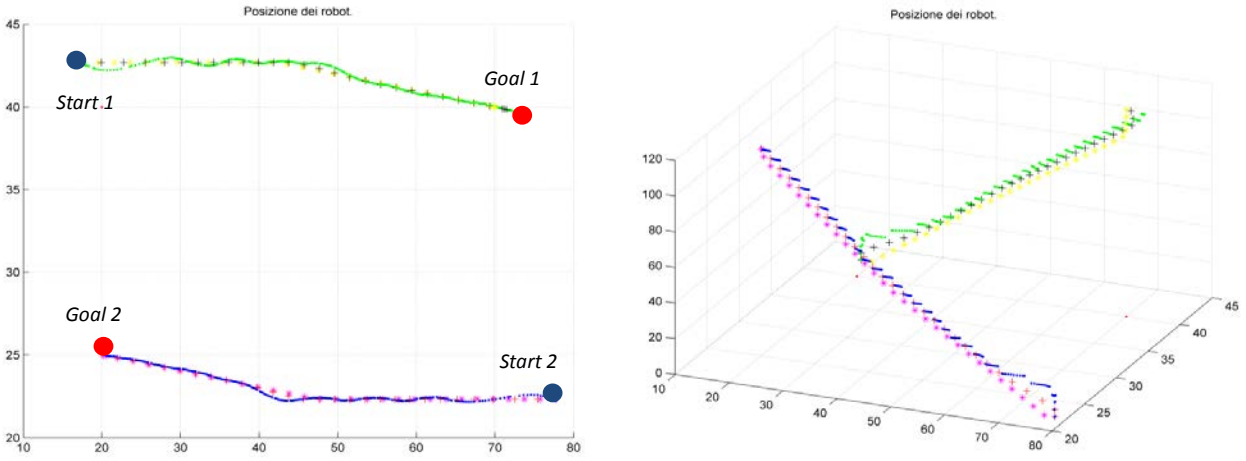

Fig. 7.5.1*– Coordinamento di due agenti in assenza di collisioni – sistema reale.*

L'andamento ottenuto conferma la validità dell'approccio seguito per quanto riguarda la componente della cifra di merito con la responsabilità di guidare il robot verso l'obiettivo; nello stesso tempo assicura il funzionamento del setup sperimentale nel suo complesso, dal rilevamento di posizioni ed orientamenti con riconoscimento dei due diversi agenti, alla comunicazione *wireless* contemporanea. Si può osservare inoltre come l'algoritmo assicuri che la traiettoria reale seguita da ciascuno dei robot, visibile in verde ed in blu in Figura 7.5.1, sia limitata all'intorno desiderato della traiettoria di riferimento rappresentata rispettivamente dalle croci di colore nero e rosso; questa, infine, seguirà opportunamente la sequenza di riferimenti ideale $\tilde{y}^{[i]}$ di cui si è ampiamente discusso nel Capitolo 6.

7.6 – Prove di coordinamento sul sistema reale.

Confermato il funzionamento della struttura di test realizzata è stato possibile procedere con una prova che prevede traiettorie in collisione, così da validare definitivamente l'algoritmo presentato in questo lavoro di tesi. Il primo risultato è quello di Figura 7.6.1.

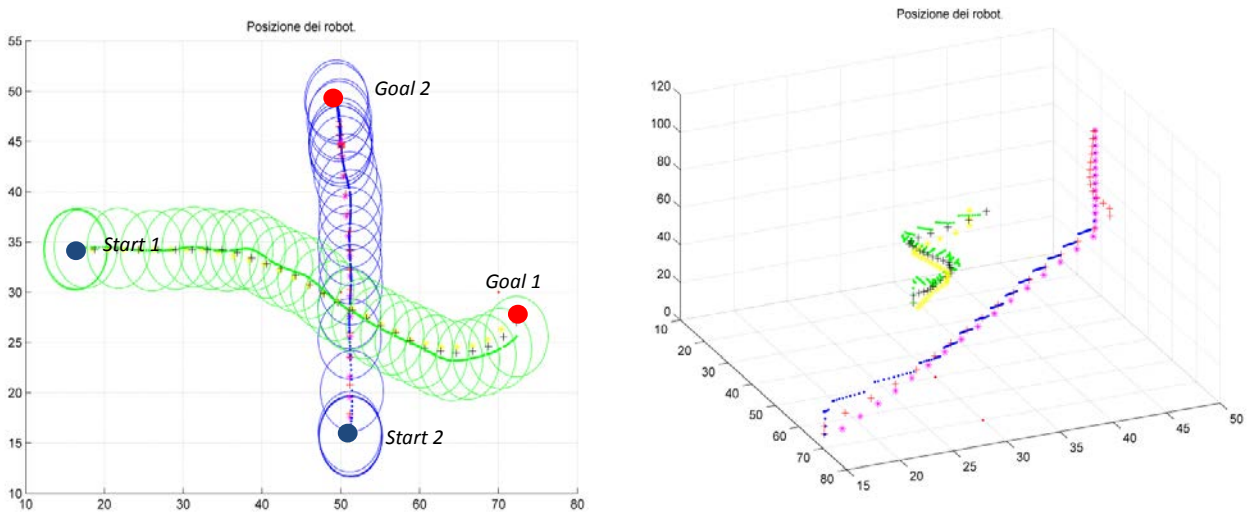

***Fig. 7.6.1**– Coordinamento di due agenti in presenza di collisioni. Precedenza al robot blu – sistema reale.*

Si nota che è stata lasciata la priorità nel movimento al robot in blu mentre il robot in verde è costretto ad una manovra di evasione per evitare la collisione.

La stessa prova può essere eseguita senza definire in modo specifico una priorità, ovvero lasciando che ciascuno degli agenti veda l'altro come ostacolo. Il risultato è quello di Figura 7.6.2.

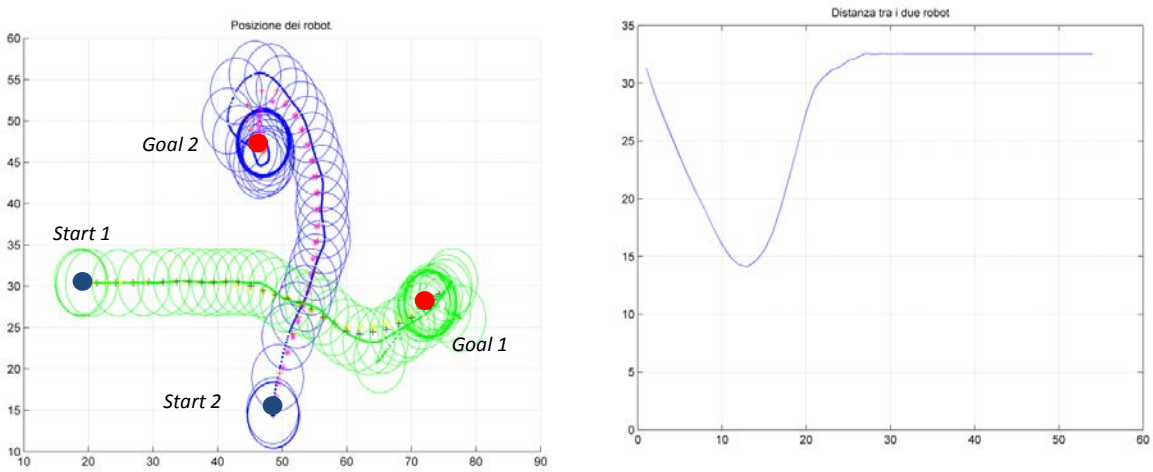

***Fig. 7.6.2**– Coordinamento di due agenti in presenza di collisioni. Precedenza non definita – sistema reale.*

Si vede bene che in questo caso anche il robot in blu è costretto ad alterare la sua traiettoria per evitare la futura collisione prevista dall'algoritmo; allo stesso modo il robot in verde continua la sua manovra che lo mantiene sufficientemente lontano dall'altro agente. Interessante è a tal proposito valutare la distanza che si ha tra i due robot visibile nell'immagine di destra; questa conferma la validità dell'approccio adottato per quanto riguarda l'*obstacle avoidance*.

L'ultima delle prove che si è scelto di effettuare è quella di movimento dei due robot lungo la medesima direzione; il risultato è quello visibile in Figura 7.6.3.

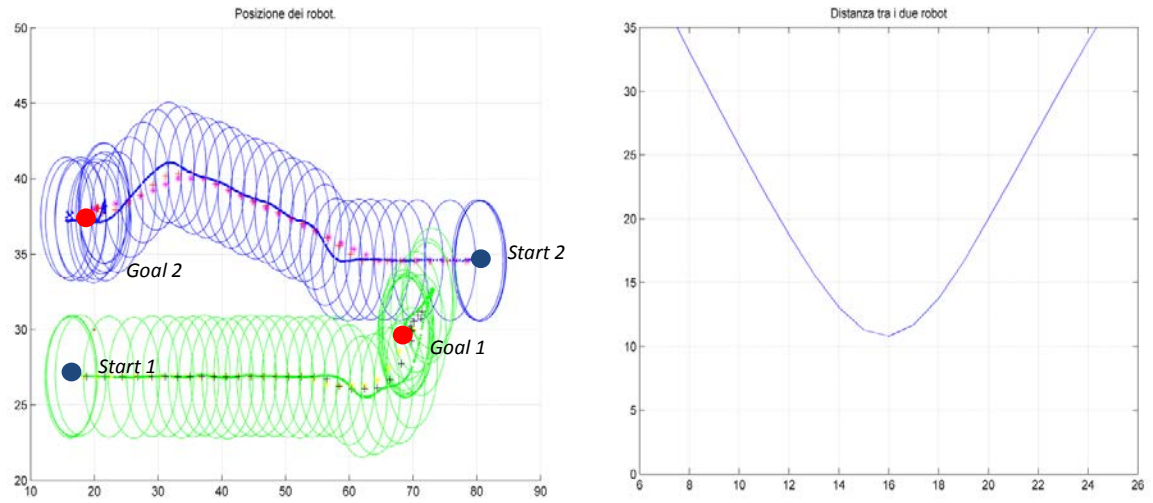

***Fig. 7.6.3**– Coordinamento di due agenti in presenza di collisioni. Precedenza non definita – sistema reale.*

Si può osservare ancora una volta la validità dell'algoritmo presentato al Capitolo 6. Entrambi i robot infatti generano una traiettoria opportunamente alterata per evitare la collisione; l'effetto ottenuto si apprezza facilmente dal grafico della distanza tra i centri dei due robot, che si mantiene al di sotto del valore critico.

In un problema simile al precedente se si aumenta l'intervallo di guardia tra i due robot si ottiene un comportamento come quello visibile in Figura 7.6.4.

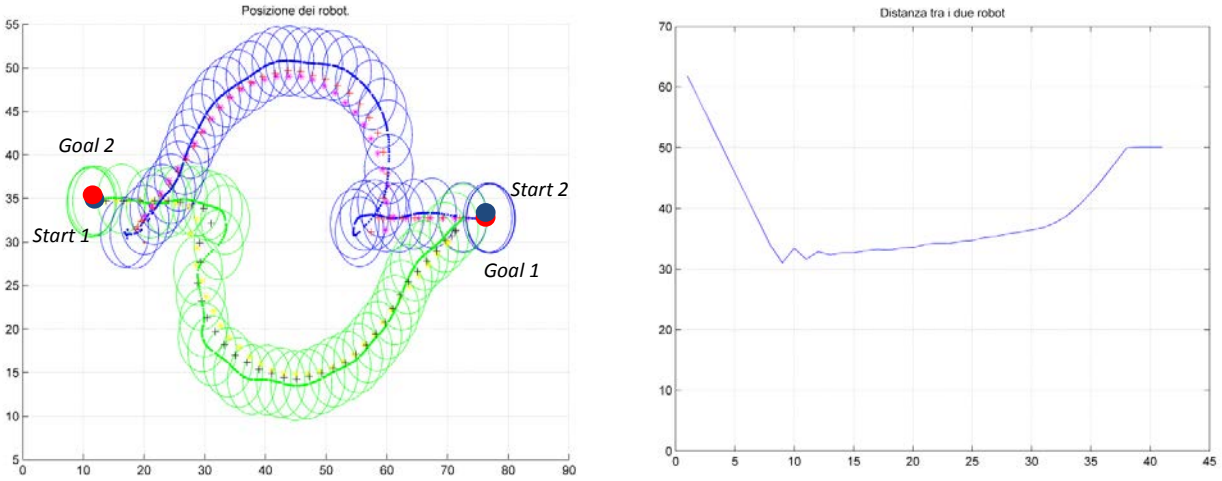

Fig. 7.6.4– *Coordinamento di due agenti in presenza di collisioni. Precedenza non definita. Intervallo di guardia aumentato – sistema reale.*

Da questo si può osservare che al variare dei parametri usati per l'algoritmo di controllo il comportamento complessivo ottenuto varia significativamente; la scelta pertanto richiede particolare attenzione.

In ogni caso, come si può vedere dall'immagine di destra, la prova ha esito positivo ed entrambi i robot completano il *task* richiesto senza collidere e senza aver pianificato a priori una traiettoria e senza un algoritmo di supervisione. Questo è possibile grazie all'approccio distribuito del problema *MPC* ed allo scambio di dati tra i due regolatori locali, così come descritto nei capitoli precedenti. La notevole ampiezza della manovra di evasione dimostra ancora una volta che è importante tarare opportunamente i parametri utilizzati nella legge di controllo.

8- CONCLUSIONI E SVILUPPI FUTURI

In questo lavoro di tesi è stata analizzata la possibilità di applicare una filosofia di controllo *MPC* con implementazione distribuita al problema di coordinamento di un set di robot mobili in movimento planare. Si è partiti dallo studio delle tecniche esistenti in letteratura per l'inseguimento di traiettorie di riferimento, progettate a priori, da parte di robot con struttura ad uniciclo; questo ha permesso di capire le problematiche connesse e sintetizzare un controllore linearizzante in grado di semplificare notevolmente il problema di posizionamento. In seguito si è affrontato il problema dell'*obstacle avoidance* e, attraverso una panoramica delle soluzioni presenti in letteratura, si è scelta una tecnica basata sui potenziali artificiali che ben si integra con la filosofia *MPC.* Successivamente è stato analizzato il funzionamento di un controllo *MPC* centralizzato allo scopo di governare un singolo robot in presenza di soli ostacoli fissi; questo ha permesso di evitare il ricorso ad una pianificazione a priori di una traiettoria libera verso l'obiettivo integrando il tutto nel medesimo problema di regolazione. Infine, sulla base di quanto visto in precedenza, è stata proposta una tecnica predittiva distribuita in grado di coordinare il moto di una piccola flotta di robot garantendo l'assenza di collisioni reciproche; per questo si è fatto ricorso ad un *MPC* robusto di tipo *tube based.*

Sulla base dei risultati presentati nel Capitolo 7 si può concludere che la tecnica proposta consente effettivamente di raggiungere il risultato desiderato, ovvero consente di gestire il posizionamento contemporaneo dei robot all'interno del medesimo ambiente ed in assenza di collisioni. Grazie alla comunicazione tra i regolatori locali a ciascun sottosistema si è evitato il ricorso ad un controllore di supervisione; l'idea di fondo è infatti quella di un algoritmo in grado di raggiungere un consenso in presenza di obiettivi in competizione tra loro.

In realtà si è visto anche che il comportamento tenuto dai dispositivi reali non è sempre quello che ci si aspetterebbe dal punto di vista intuitivo e che le dinamiche effettivamente generate dipendono in alcuni casi dalla scelta fatta per i numerosi parametri coinvolti nell'algoritmo. In particolare tale comportamento è stato associato all'aspetto che ha il potenziale repulsivo generato da ostacoli in movimento quando l'intervallo di sicurezza scelto per garantire l'assenza di collisioni è ampio. Sebbene questo consenta in ogni caso di risolvere il problema di posizionamento, infatti, un entità non compatibile con il potenziale attrattivo generato implicitamente dall'obiettivo con la costruzione *MPC* costringe un agente ad ampie manovre di aggiramento

dell'ostacolo. Nel caso particolare di incontro frontale con un robot a cui è assegnata una priorità superiore questo può portare addirittura alla "messa in fuga" dell'agente con priorità più bassa. Più in dettaglio questo è dovuto alla particolare forma che assume il vettore gradiente sul piano al momento della prevista collisione tra i rispettivi riferimenti, che ricordiamo avverrebbe dopo N passi; questo non permette infatti ad un robot di fermarsi in presenza dell'ostacolo, ma lo costringe a muoversi all'indietro mettendo da parte provvisoriamente la navigazione verso l'obiettivo.

Questo aspetto si apre allora alla possibilità di studiare una soluzione diversa per l'*avoidance* che consideri il vincolo di collisione in maniera *hard* e non, come fatto finora, come una componente di penalizzazione della cifra di merito; del resto l'approccio *MPC* possiede l'indubbio vantaggio di consentire la gestione di vincoli di questo tipo durante l'ottimizzazione.

Un secondo aspetto degno di nota riguarda la complessità computazionale dell'intero algoritmo di test sviluppato. Utilizzando, infatti, un solo calcolatore per emulare la presenza di diversi regolatori locali tra loro comunicanti, per implementare il *feedback* linearizzante interno e per gestire la videocamera che si occupa del rilevamento della posizione si è costretti a fare ricorso a dei tempi di campionamento piuttosto lunghi. Questi non influiscono eccessivamente sulle prestazioni dell'algoritmo a patto di mantenere limitata la velocità massima di movimento dei robot reali.

Per far fronte a tale situazione e velocizzare il tutto una soluzione è quella di semplificare la componente non lineare del problema di ottimo cercando di ricondursi ad un problema puramente lineare, per il quale esistono algoritmi molto efficienti di soluzione. A tal proposito si ricordi che i termini non lineari sono dovuti solamente alla presenza della componente penalizzante che garantisce l'*obstacle avoidance;* se questa fosse trasformata in un vincolo di tipo *hard* otterremmo allora un miglioramento anche in tale direzione, consentendo di giungere ad un problema puramente quadratico di più facile risoluzione.

Per migliorare l'efficienza del sistema realizzato sarebbe inoltre opportuno rivedere con attenzione ciascuna delle sezioni di codice, ricorrendo per quanto possibile ad implementazioni di più basso livello che consentano di evitare il ricorso all'ambiente *MATLAB*. In dettaglio si potrebbe pensare di trasferire l'anello di controllo più interno, ovvero il *feedback* linearizzante, sul microcontrollore a bordo di ciascun robot, facendo in modo di trasmettere come informazione anche la posizione rilevata dalla telecamera e se possibile integrarla con i dati provenienti dagli *encoders* presenti sulle ruote; questo permetterebbe di aumentare notevolmente la frequenza di campionamento migliorando l'accuratezza del sistema. Un secondo aspetto riguarda sicuramente l'implementazione dell'algoritmo di rilevamento tramite *webcam* che dovrebbe essere trasferito in linguaggio *C* facendo ricorso alle librerie grafiche *OpenCV.* Con la stessa idea sarebbe necessario anche valutare la disponibilità di efficienti algoritmi di ottimizzazione disponibili direttamente come librerie per il linguaggio *C*.

In conclusione, nonostante gli ampi margini di miglioramento a cui si è solamente accennato, è stato possibile verificare la validità dell'approccio distribuito presentato per quanto riguarda il problema di controllo coordinato del posizionamento di alcuni robot mobili in assenza di collisione.

9 - APPENDICE

9.1 – Firmware del microcontrollore.

Per quanto riguarda il *firmware* scritto sul microcontrollore dei robot utilizzati per i test, di cui si è discusso più in dettaglio nel Capitolo 2, si è fatto riferimento alle librerie fornite dal produttore e liberamente disponibili in rete per la gestione di basso livello dei singoli dispositivi di cui esso è dotato. La *routine* principale del programma viene invece riportata nel seguito.

```
#include < e_init_port.h>
#include < e_uart_char.h>
#include < matlab.h>
#include < e_motors.h>
#include < e_motors.h>
#include < e_led.h>
#include < e_epuck_ports.h>
#include <utility.h>
#include <stdlib.h>
#include <string.h>
#include <ctype.h>
#include <stdio.h>
#include <math.h>


#define uart_send_text(msg) do {
e_send_uart1_char(msg,strlen(msg));
while(e_uart1_sending()); } while(0)


#define PI 3.14159265358979
#define STEPSTO2PI 1300
#define T_SPEED 300

int main(void)
{
      int selector;
      long i;
      char car, buffer[20];
      int dato[2], speedl, speedr;
      dato[0]=1;
```

```
//Inizializzazione delle porte, della UART e dei
motori.
e_init_port();
e_init_uart1();
e_init_motors();
e_set_speed_left(0);
e_set_speed_right(0);


//Lettura della posizione del selettore; la scelta
della posizione zero permette di mettere in attesa il
micro così da consentire la programmazione.
selector=getselector();
if(selector==0){
     while(1);
     }


//Nel caso in cui il selettore non sia nella posizione
zero inizia il ciclo del programma, la     UART viene
pulita dai dati in attesa e il LED7 conferma il
risultato corretto del setup.
while(e_getchar_uart1(&car));
LED7 = 1;


//Inizio del loop infinito. Il micro attende che vi
sia un comando sulla seriale e nel caso questo sia
quello corretto legge i due interi seguenti e li
interpreta come velocità da assegnare alle ruote.


while(1){
while(!e_ischar_uart1());
e_getchar_uart1(&car);
if (car=='s'){
     while(e_receive_int_from_matlab(dato,2)==0);
     LED1 =1;
     for(i=0;i<30000;i++){
               asm("nop");
          }
     LED1= 0;
     LED4= 1;
     //Interpretazione dei due interi ricevuti come
velocità.
     speedl = (int) dato[0];
     speedr = (int) dato[1];
     //Saturazione delle velocità inviate al micro.
```

```
            if (speedl>1000) speedl = 1000;
            if (speedl<-1000) speedl =-1000;
            if (speedr>1000) speedr = 1000;
            if (speedr<-1000) speedr =1000;



                    //Attuazione della velocità.
            e_set_speed_left(speedl);
            e_set_speed_right(speedr);
            LED4 = 0;
            }//Chiusura dell'if in risposta al comando 's'
      }//Chiusura del loop infinito.
}//Termine del main.
```

9.2 – Codice *MATLAB* per la comunicazione.

Per quanto riguarda la componente lato *PC* che gestisce la comunicazione con il microcontrollore, ed in particolare che invia i comandi di velocità per le due ruote, è stato utilizzato il seguente codice *MATLAB.*

```
function invio_dati(speedl,speedr)


      global EpuckPort;
      %Saturazione degli ingressi di controllo. Serve ad
      evitare l'overflow della variabile intera.
      if (speedl>1000) speedl=1000;end
      if (speedl<-1000) speedl=-1000;end
      if (speedr>1000) speedr=1000;end
      if (speedr<-1000) speedr=1000;end


      %Costruzione del pacchetto nel formato int a 16 bit.
      data = [speedl speedr];

      int16(data);
      %Il carattere s comunica al robot che l'informazione
      successiva è costituita dalle due velocità.
      fwrite(EpuckPort,'s','char');
      fwrite(EpuckPort,2*2,'uint16');
      fwrite(EpuckPort,data,'int16');
```

end

In tal caso si vede il ricorso al formato intero a $16\,bit$ che consente di inviare correttamente le informazioni su ciascuna velocità in un pacchetto di due byte distinti.

9.3 – Codice *MATLAB* per la localizzazione via webcam.

Per quanto riguarda la parte *software* che gestisce la localizzazione via *webcam* dei due robot sul piano di lavoro si è fatto riferimento al codice seguente. Questo comprende anche una calibrazione empirica che consente il passaggio dal sistema di riferimento immagine al sistema di riferimento reale; procedure più sofisticate sono state valutate e non portano a vantaggi direttamente visibili data la semplicità del problema in esame.

```
function [x1,y1,theta1,x2,y2,theta2]=acquisizione_doppia()

global vid
%Acquisizione di un frame.
data1=getdata(vid,1);


data=imcomplement(data1);
diff_im = imsubtract(data(:,:,3), rgb2gray(data));
diff_im=medfilt2(diff_im,[3 3]);
diff_im=im2bw(diff_im,0.3);
diff_im=bwareaopen(diff_im,50);
bw=bwlabel(diff_im,8);
stats2=regionprops(bw,'Centroid');


diff_im2 = imsubtract(data1(:,:,3), rgb2gray(data1));
diff_im2=medfilt2(diff_im2,[3 3]);
diff_im2=im2bw(diff_im2,0.25);
diff_im2=bwareaopen(diff_im2,50);
diff_im2=imfill(diff_im2,'holes');
bw2=bwlabel(diff_im2,8);
stats=regionprops(bw2,'Centroid');


diff_im3 = imsubtract(data1(:,:,1), rgb2gray(data1));
diff_im3=medfilt2(diff_im3,[3 3]);
diff_im3=im2bw(diff_im3,0.24);
diff_im3=bwareaopen(diff_im3,50);
```

```
diff_im3=imfill(diff_im3,'holes');
bw3=bwlabel(diff_im3,8);
stats3=regionprops(bw3,'Centroid');


diff_im4 = imsubtract(data1(:,:,2), rgb2gray(data1));
diff_im4=medfilt2(diff_im4,[3 3]);
diff_im4=im2bw(diff_im4,0.07);
diff_im4=bwareaopen(diff_im4,50);
diff_im4=imfill(diff_im4,'holes');
bw4=bwlabel(diff_im4,8);
stats4=regionprops(bw4,'Centroid');


for object = 1:length(stats3)
    bc1 = stats3(object).Centroid;
end
x_r=bc1(1);
y_r=bc1(2);

for object = 1:length(stats4)
    bc2 = stats4(object).Centroid;
end
x_g=bc2(1);
y_g=bc2(2);


for object = 1:length(stats2)
    bc3 = stats2(object).Centroid;
end
x_b=bc3(1);
y_b=bc3(2);


for object = 1:length(stats)
    bc4 = stats(object).Centroid;
end
x_y=bc4(1);
y_y=bc4(2);


%Calcolo delle posizioni e delle orientazioni.
orientazione1=atan2(-(y_r-y_g),(x_r-x_g))*180/pi;
if (orientazione1<0) orientazione1=orientazione1+360; end
pm1 = [(x_r+x_g)/2 (y_r+y_g)/2];
x1=(640-pm1(1))*0.147;
y1=(pm1(2))*0.147;
```

```
theta1 = orientazione1*pi/180;


orientazione2=atan2(-(y_y-y_b),(x_y-x_b))*180/pi;
if (orientazione2<0) orientazione2=orientazione2+360; end
pm2 = [(x_y+x_b)/2 (y_y+y_b)/2];
x2=(640-pm2(1))*0.147;
y2=(pm2(2))*0.147;
theta2 = orientazione2*pi/180;

flushdata(vid);
end
```

La variabile *vid* che compare nel codice è il riferimento ad un oggetto globale di tipo videocamera definito ed impostato correttamente al di fuori della procedura. In particolare nel *setup* sperimentale utilizzato si utilizzano i comandi seguenti.

```
global vid;
format='YUY2_640x480';
vid = videoinput('winvideo', 1, format);
set(vid, 'FramesPerTrigger', Inf);
set(vid, 'ReturnedColorspace', 'rgb')
vid.FrameGrabInterval = 1;
c= onCleanup(@()stop(vid));
start(vid);
flushdata(vid);
```

L'oggetto videocamera viene creato utilizzando il *toolbox imaq*, viene scelto il numero di *frames* da catturare ad ogni *trigger*, lo spazio di colore da utilizzare, l'intervallo con cui le immagini acquisite vengono scartate ed infine la connessione viene aperta; la funzione *flushdata()* permette di ripulire il *buffer* che ha una capacità limitata.

BIBLIOGRAFIA

[1] F. Mondada, M. Bonani, X. Raemy, J. Pugh, C. Cianci, A. Klaptocz, S. Magnenat, J.C. Zufferey, D. Floreano and A. Martinoli. The e-puck, a Robot Designed for Education in Engineering. *Proceedings of the 9th Conference on Autonomous Robot Systems and Competitions, pp. 59-65, 2009.*

[2] Roger W. Brockett. Asymptotic Stability and Feedback Stabilization. *Differential Geometric Control Theory (R. W. Brockett, R. S. Millman and H. J. Sussmann, eds.), pp 181-191, Birkhauser, Boston, 1983.*

[3] Ti-Chung Lee, Kai-Tai Song, Ching-Hung Lee, Ching-Cheng Teng. Tracking control of unicycle-modeled mobile robots using a saturation feedback controller. *Control Systems Technology, IEEE Transactions on, vol.9, no.2, pp.305-318, Mar 2001.*

[4] Jiang, Zhong-Ping and Nijmeijer. Tracking control of mobile robots: a case study in backstepping. *Automatica, 33 (7), pp. 1393-1399.*

[5] J.M. Yang and J.H. Kim. Sliding mode control for trajectory tracking of nonholonomic wheeled mobile robots. *IEEE Transactions on Robotics and Automation, vol. 15, no.3, pp.578 -587 1999.*

[6] F. Diaz del Rio, G. Jimenez, J. Sevillano, S. Amaya and A. Balcells. A new method for tracking memorized paths: Application to unicycle robots. *Procedings of the 10th Mediterranean Conference on Control, 2002.*

[7] M. Aicardi, G. Casalino, A. Bicchi, A. Balestrino. Closed loop steering of unicycle like vehicles via Lyapunov techniques. *Robotics & Automation Magazine, IEEE, vol.2, no.1, pp.27-35, Mar 1995.*

[8] A. Yufka and O. Parlakatuna. Performance comparison of bug algorithms for mobile robots. *5th International Advanced Technologies Symposium (IATS'09), May 13-15, 2009.*

[9] O. Khatib. Real-time obstacle avoidance for manipulators and mobile robots. *International Journal of Robotics Research, vol.5, no.1, pp. 90–98, 1995.*

[10] J. Borenstein and Y. Koren. The vector field histogram - fast obstacle avoidance for mobile robots. *IEEE Transaction on Robotics and Automation, vol.7, no.3, pp. 278–288, 1991.*

[11] W.H. Huang, B.R. Fajen, J.R. Fink, W.H. Warren. Visual navigation and obstacle avoidance using a steering potential function. *Robotics and Autonomous Systems 54, 288-299, 2006.*

[12] E. B. Lee and L. Markus, *"Foundations of Optimal Control Theory.",* Wiley, New York, 1967.

[13] D.Q. Mayne, J.B. Rawlings, C.V. Rao, and P.O.M. Scokaert. Constrained model predictive control: Stability and optimality. *Automatica, 36(6), pp. 789–814, 2000.*

[14] C.V. Rao, J.B. Rawlings. Linear Programming and Model Predictive Control. *Journal of Process Control, 10, pp. 283–289, 2000.*

[15] S. V. Rakovic´, E. C. Kerrigan, K. I. Kouramas, and D. Q. Mayne. Invariant Approximations of the Minimal Robust Positively Invariant Set. *IEEE Transactions on automatic control, vol.50, no.3, Mar 2005.*

[16] P.J. Campo and M. Morari. Robust model predictive control. *Proceedings of American Control Conference, vol.2. pp. 1021-102, 1987.*

[17] D.Q. Mayne, M.M. Seron, V. Rakovic. Robust model predictive control of constrained linear systems with bounded disturbances. *Automatica, 41, pp. 219–224, 2005.*

[18] A. De Luca , G. Oriolo and M. Vendittelli. Stabilization of the unicycle via dynamic feedback linearization. *Proceedings of the 6th IFAC Symposium on Robotic Control, pp.397, 2000.*

[19] B.R. Fajen, W.H. Warren. Behavioral dynamics of steering, obstacle avoidance, and route selection. *Journal of Experimental Psychology: Human Perception and Performance, 29 (2), pp. 343–362, 2003.*