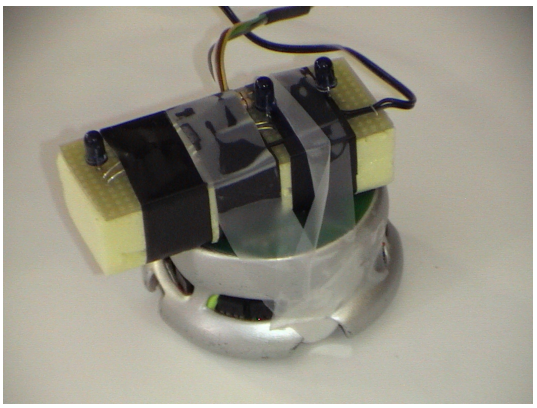
Controllo della traiettoria di un WMR mediante retroazione da web-cam

Caregaro Negrin Marco, Nicolettis Andrea, Pucci Paolo

23/06/2006



Università Degli Studi di Padova

Laurea in Ingegneria dell'Automazione

Corso di Progettazione di Sistemi di Controllo

Abstract:
Questo progetto riguarda l'implementazione di un algoritmo per il controllo della traiettoria di un WMR *Khepera* mediante retroazione da web-cam. In particolare, si possono distinguere i seguenti aspetti fondamentali:

- **rilevamento della posizione attuale del robot**: su quest'ultimo sono stati installati dei LED all'infrarosso, disposti opportunamente, mentre alla web-cam è stato applicato un filtro all'infrarosso. L'immagine acquisita, dopo essere stata trasformata mediante una matrice di proiezione prospettica che permette di eliminare l'errore di parallasse, viene processata con opportuni algoritmi di visione per ricavare la posizione reale del robot
- **generazione dei segnali di controllo**: l'utente, tramite una comoda interfaccia grafica, può specificare la traiettoria che il robot deve inseguire. In base a questo riferimento, e alla retroazione degli algoritmi di visione, vengono generati i comandi di feed-froward e di feed-back per il *Khepera*, mediante opportuni algoritmi di controllo

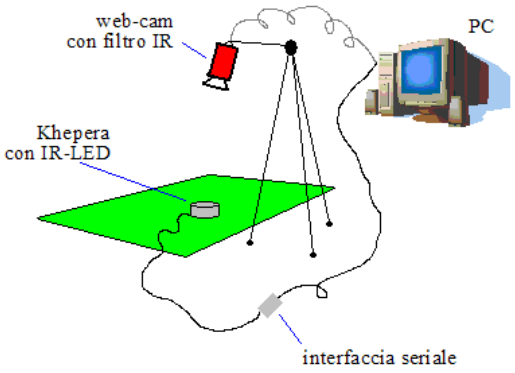


Fig. 1: Illustrazione schematica del sistema

Indice

1 Introduzione

Il problema del controllo di robot mobili, siano essi veicoli su ruote o manipolatori, trova applicazione in vaste aree ed è oggetto di numerosi studi ed approfondimenti in letteratura. In particolare, l'interesse del presente progetto si concentra su una specifica tipologia di veicoli su ruote e precisamente quelli costituiti da due ruote disposte sullo stesso asse e comandate indipendentemente da due motori, come appunto il *Khepera*.

Lo scopo ultimo del progetto è quello di realizzare un controllo sulla posizione del robot, controllo che poi potrà essere adattato ai diversi compiti richiesti.

La caratteristica principale di questo controllo è però quella di utilizzare un segnale in retroazione proveniente da un ulteriore algoritmo che utilizza l'immagine fornita da una telecamera, la quale individua la posizione del veicolo nel campo di azione. Questa organizzazione del sistema di controllo pone ovviamente tutta una serie di problemi, che vanno dalla scelta del metodo di individuazione del robot da parte della telecamera, alla definizione del percorso da seguire, all'interazione tra le due parti dell'algoritmo (quella relativa al sistema di visione ed il controllo vero e proprio del robot) e così via.
Si capisce dunque come ci siano molti aspetti da considerare che introducono notevoli limitazioni sulle scelte progettuali possibili, non ultimo il fatto che si ha a che fare con un sistema anolonomo, caratteristica che pone dei vincoli sui movimenti possibili, costringendo ad esempio il veicolo ad una serie di manovre complicate per spostarsi in una posizione vicina. Questa caratteristica è tipica dei veicoli a ruote e, per avere un'idea concreta delle limitazioni presenti, basta pensare ad esempio alle manovre eseguite da un'automobile al momento del parcheggio.

Tutte le scelte effettuate devono ovviamente tenere conto dell'hardware a disposizione, in modo tale da non richiedere prestazioni superiori a quelle raggiungibili dal sistema fisico. L'hardware utilizzato nell'esperienza verrà descritto nel seguito, in modo da avere un'idea dell'apparato in questione.

Una volta impostato il controllo del sistema, si è ovviamente passati alla definizione dell'applicazione finale dello stesso, da cui dipendono ovviamente ulteriori scelte relative al controllo e soprattutto alla determinazione delle traiettorie che il veicolo deve seguire. Al fine di evidenziare l'effettivo funzionamento dell'algoritmo di controllo progettato si è scelto di realizzare una " città virtuale" nella quale il robot deve muoversi in maniera adeguata, inserendo anche delle manovre più complicate come ad esempio un parcheggio.

State of the art

La letteratura sull'argomento, come si può facilmente intuire, è vastissima: nella bibliografia si è voluto proporne una selezione, che comprende solo i riferimenti esplicitamente utilizzati e quelli di maggior interesse, sia per quanto riguarda differenti approcci da quelli proposti qui, sia per quanto riguarda eventuali evoluzioni di quanto già fatto.
In particolare, [1] e [4] affrontano il problema del controllo di WMR con metodi "tradizionali", mentre [5] e [6] utilizzano controlli predittivi e *fuzzy-logic*.
Più in generale, [2] e [3] affrontano problemi di tracking e controllo di robot mediante retroazione da web-cam.
Per quanto riguarda sensori e algoritmi di visione, si rimanda a [7] e [8].
Per la teoria riguardante i sistemi olonomi e anolonomi si è fatto riferimento soprattutto a [9].
Per approfondimenti sul *trajectory planning* infine, si vedano [10], [11], [12] e [13].

2 Analisi del sistema

2.1 Sistemi olonomi e anolonomi

Il sistema oggetto del controllo è già stato classificato come anolonomo [1], si ritiene pertanto utile dare una breve descrizione di questa tipologia di sistemi, unitamente alla controparte rappresentata dai sistemi olonomi.
Si consideri un generico stato $q \in \mathbb{R}$ relativo ad un sistema di interesse e si supponga che quest'ultimo sia soggetto a dei vincoli del tipo:

$$a_i(q) : \mathbb{R}^n \to \mathbb{R}^n \qquad i = 1, \ldots, k\,(k < n)$$

$$a_i^T(q) \cdot \dot{q} = 0$$

Considerando un singolo vincolo del tipo descritto, esso si traduce nell'impossibilità da parte dello stato del sistema di muoversi lungo la retta rappresentante il vincolo stesso, ma è possibile il movimento su un piano perpendicolare alla direzione della retta.

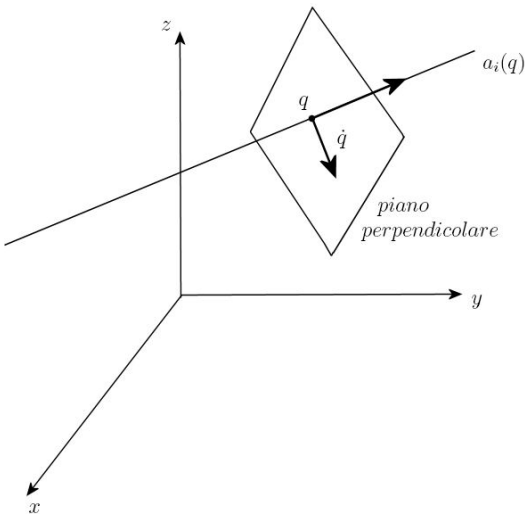


Fig. 2: Vincolo sulla velocità

Evidentemente il numero di vincoli determina tutta una serie di direzioni lungo le quali non è possibile muoversi. Ad esempio, nel caso di due vincoli, i movimenti sono possibili ovunque tranne che sul piano definito dai due vettori relativi ai vincoli stessi.
A questo punto ci si può chiedere se i vincoli $\{a_i(q) \quad i = 1, \ldots, k\}$ obblighino a stare su una determinata superficie. Ciò non è detto, perché in realtà i vincoli su posizione e velocità sono di tipo istantaneo, perciò in un determinato istante un certo movimento può risultare proibito ma questo non pregiudica il raggiungimento di una qualsiasi configurazione finale. Si pensi ad esempio al caso dell'automobile: uno spostamento laterale non è sicuramente consentito ma, con un certo numero di manovre, è possibile portare il mezzo in una qualsiasi posizione.
Si avrebbero dei vincoli complessivi sulla posizione se l'ipersuperficie indotta da $a_i(q)$ fosse rappresentabile nella forma $h(q) = 0$ e allora, in questo caso, non tutte le configurazioni sarebbero raggiungibili dal sistema. Si nota quindi la differenza tra i due casi esaminati, con vincoli solamente sulle velocità e con vincoli globali anche sulla posizione.

[1] Altri tipici esempi di sistemi anolonomi sono l'uniciclo, l'automobile, la bicicletta.

Si possono dare pertanto le definizioni di sistema olonomo e anolonomo.

Definizione 2.1 (Sistema olonomo) *Un sistema si dice* olonomo *se, dati k vincoli*

$$\{a_1(q), \ldots, a_k(q)\}$$

esistono k funzioni di questo tipo:

$$h_1(q) = 0, \ldots, h_k(q) = 0, \qquad h_i : \mathbb{R}^n \to \mathbb{R}$$

La definizione appena data equivale a dire che lo stato q appartiene ad una ipersuperficie di dimensione $n - k$ immersa in uno spazio di dimensione n.

Generalizzando quanto appena detto al caso in cui ci sia un numero di vincoli inferiore a k, si può affermare quanto segue:

Definizione 2.2 *Dati i vincoli* $\{a_1(q), \ldots, a_k(q)\}$ *definiti nel seguente modo:*

$$a_i^T(q) \cdot \dot{q} = 0$$

e considerate le funzioni $h_1(q), \ldots, h_m(q)$*, il sistema si dice:*

- olonomo *se* $m = k$*;*
- parzialmente anolonomo *se* $0 < m < k$*;*
- anolonomo *se* $m = 0$*.*

Riassumendo la differenza tra le due tipologie di sistemi, si può affermare che lo stato $q \in \mathbb{R}^n$ nel caso di un sistema olonomo è vincolato a stare su una superficie di dimensione $k < n$ che può essere definita anche nel seguente modo implicito:

$$S \triangleq \{q \in \mathbb{R}^n : h_i(q) = 0,\ i = 1, ..., k\}$$

Quella appena data è una descrizione globale del sistema, mentre dal punto di vista locale si possono indicare degli altri vincoli caratterizzanti, questa volta sulle velocità[2]:

$$a_i^T(q) \cdot \dot{q} = 0$$

Considerando questi ultimi vincoli scritti e conoscendo lo stato attuale del sistema si è in grado di definire i movimenti possibili, che ovviamente dipendono dal numero di vincoli presenti. I vincoli locali pertanto implicano in questo caso l'esistenza di vincoli globali.

Un sistema anolonomo invece ha dei vincoli locali sulle velocità dello stesso tipo ma questi non generano alcun vincolo globale sullo stato $q \in \mathbb{R}^n$. Ciò esprime appunto il fatto che una qualsiasi posizione è raggiungibile, anche se il numero di passi che devono essere compiuti per giungere allo stato finale può essere elevato.

Un'ulteriore considerazione riguarda l'integrabilità o meno dei vincoli. In particolare, un sistema si dice completamente integrabile, e dunque olonomo, quando i vincoli pfaffiani sulla velocità:

$$a_i^T(q) \cdot \dot{q} = 0$$

[2]In particolare i vincoli riportati di seguito sono denominati *vincoli pfaffiani*

possono essere ottenuti differenziando rispetto al tempo le k equazioni del tipo $h_i(q) = 0$:

$$\frac{\partial h_i(q)}{\partial q} = a_i^T(q) \qquad \Longrightarrow \qquad \frac{d}{dt}(h_i(q)) = \frac{\partial h_i(q)}{\partial q} \cdot \dot{q} = a_i^T(q) \cdot \dot{q} = 0$$

Nel caso in cui ciò non sia possibile il sistema viene ovviamente detto anolonomo.

Oltre alla rappresentazione appena data, facente uso di vincoli locali sulla posizione, è possibile rappresentare un generico sistema, sia esso olonomo o anolonomo, in un modo alternativo. Si tratta semplicemente di esprimere il sistema mediante le direzioni lungo le quali esso può muoversi, invece che fare uso dei vincoli che indicano le direzioni lungo cui il movimento è proibito. Si ottengono in questo modo delle equazioni del moto che, al loro interno, contengono anche i vincoli.

Considerando i sistemi anolonomi con vincoli pfaffiani possiamo descrivere quanto appena detto mediante due equazioni che si riferiscono al caso di sistemi senza drift e di sistemi con drift:

$$\dot{q} = g_1(q)u_1 + \ldots + g_m(q)u_m \qquad \text{(senza drift)}$$

$$\dot{q} = f(q) + g_1(q)u_1 + \ldots + g_m(q)u_m \qquad \text{(con drift)}$$

dove:

- $u_i \in \mathbb{R},\ i = 1, \ldots, m$ sono gli ingressi al sistema;
- $g_i : \mathbb{R}^n \to \mathbb{R}^n,\ i = 1, \ldots, m$ sono delle funzioni del vettore di stato $q \in \mathbb{R}^n$;
- $f(q)$ è una generica funzione del vettore di stato $q \in \mathbb{R}^n$.

Evidentemente, nel caso di sistemi senza drift, una combinazione degli ingressi per cui $\dot{q} = 0$ implica l'assenza di movimento, cosa che invece non è assicurata nel caso di sistemi con drift vista la presenza della funzione $f(q)$.
Inoltre è possibile dimostrare che un qualsiasi sistema non lineare è esprimibile come un sistema con drift. Dato il generico sistema non lineare $\dot{x} = f(x, u)$ è infatti possibile introdurre la nuova variabile di stato:

$$q = \begin{bmatrix} x \\ v \end{bmatrix}$$

con $\dot{v} = u$. A questo punto si può scrivere:

$$\dot{q} = \begin{bmatrix} f(x, v) \\ 0 \end{bmatrix} + \begin{bmatrix} 0 \\ I \end{bmatrix} v$$

ed il sistema così ottenuto è evidentemente un sistema con drift.

I sistemi senza drift possono invece essere suddivisi in due ulteriori categorie:

- sistemi del primo ordine;
- sistemi in catena.

I primi sono rappresentabili nel seguente modo:

$$\dot{q} = u\dot{Y} = q \cdot u^T - u \cdot q^T$$

con $Y \in SO(m)$.
I secondi invece sono descritti dalla seguente equazione (nel caso $m = 2$):

$$\dot{q} = g_1 \cdot u_1 + g_2 \cdot u_2$$

dove:

$$g_1 = \begin{bmatrix} 1 \\ 0 \\ q_2 \\ \vdots \\ q_{n-1} \end{bmatrix} \qquad \text{e} \qquad g_2 = \begin{bmatrix} 0 \\ 1 \\ 0 \\ \vdots \\ 0 \end{bmatrix}$$

Per un generico sistema diventa perciò importante poter analizzare l'olonomia o meno dello stesso. Spesso per fare ciò si riportano, mediante opportuni cambi di variabile, le equazioni del sistema nella forma caratteristica dei sistemi in catena, pur essendo possibile fare tale analisi anche avendo equazioni del sistema di altro tipo. Ciò perché, nel caso di sistemi in catena, si conoscono una serie di algoritmi di controllo che permettono di portare più agevolmente lo stato dalla configurazione iniziale, q_0, alla configurazione finale voluta, q_f.
Si procede definendo un nuovo stato e dei nuovi ingressi:

$$\begin{cases} z = f_1(q) & \text{nuovo stato} \\ v = f_2(q, u) & \text{nuovi ingressi} \end{cases}$$

dove entrambe le trasformazioni sono ovviamente invertibili. A questo punto si passa a considerare il sistema nel nuovo stato, definendo configurazioni iniziale e finale, z_0 e z_f, a partire da q_0 e q_f mediante la trasformazione $f_1(q)$. Si applica successivamente l'algoritmo scelto al fine di ottenere la successione di ingressi $v_1(t), \ldots, v_m(t)$ che eseguono quanto voluto. Infine, mediante l'inversa della trasformazione $f_2(q, u)$ ci si riporta agli ingressi originari, determinando pertanto la successione $u_1(t), \ldots, u_m(t)$.

Ovviamente, per assicurare che una determinata configurazione scelta possa effettivamente essere raggiunta dal sistema, è necessario eseguire un'analisi sulla controllabilità del processo in esame. Si riportano perciò le relative definizioni.

Definizione 2.3 (Sistema controllabile) *Dato un generico sistema non lineare*

$$\dot{x} = f(x, u) \qquad \begin{array}{l} x \in \mathbb{R}^n \\ u \in U \subseteq \mathbb{R}^m \end{array}$$

esso viene detto controllabile in $M \subset \mathbb{R}^n$ *se, dati* $x_0, x_f \in M$, *esistono* $T > 0$ *e* $u(\cdot) : [0, T] \to U$ *tali che* $x(0) = x_0$ *e* $x(T) = x_f$.

Il sistema viene detto *controllabile* se $M = \mathbb{R}^n$. Quanto appena scritto equivale ad affermare che è possibile trovare una successione di ingressi $u(0), \ldots, u(T)$ che portano lo stato dalla configurazione iniziale a quella finale voluta.

Definizione 2.4 *Dati* $T > 0$ *e un insieme aperto* $V \subset \mathbb{R}^n$ *si definisce il seguente insieme:*

$$\mathcal{R}^V(x_0, T) = \{x \in \mathbb{R}^n : \exists u : [0, T] \to U : x(0) = x_0,\ x(T) = x_f,\ x(t) \in V\ \forall t \in [0, T]\}$$

L'insieme appena definito contiene tutti gli stati che sono raggiungibili a partire da x_0 nel tempo T e che rimangono sempre in V.

Definizione 2.5 *Si definisce inoltre:*

$$\mathcal{R}_T^V(x_0) = \bigcup_{0<\tau\leq T} \mathcal{R}^V(x_0, \tau)$$

Definizione 2.6 (Sistema LA) *Dato un generico sistema non lineare*

$$\dot{x} = f(x,u) \qquad \begin{array}{l} x \in \mathbb{R}^n \\ u \in U \subseteq \mathbb{R}^m \end{array}$$

esso si dice localmente accessibile (LA) *da x_0 se $\forall V$ intorno di x_0 e $\forall T > 0$ risulta che:*

$$\mathcal{R}_T^V \supset \Omega$$

dove Ω è un generico insieme aperto non vuoto.

Definizione 2.7 (Sistema STLC) *Dato un generico sistema non lineare:*

$$\dot{x} = f(x,u) \qquad \begin{array}{l} x \in \mathbb{R}^n \\ u \in U \subseteq \mathbb{R}^m \end{array}$$

esso si dice small time locally controllable (STLC)[3] *se $\forall V$ intorno di x_0 e $\forall T > 0$ risulta che:*

$$\mathcal{R}_T^V \supset \Omega$$

dove Ω è un intorno di x_0.

Si possono a questo punto fornire le seguenti implicazioni:

Proposizione 2.1

$$Sistema\ STLC \Longrightarrow Sistema\ controllabile \Longrightarrow Sistema\ LA$$

Nel caso particolare di sistemi senza drift vale anche il viceversa:

$$Sistema\ STLC \Longleftrightarrow Sistema\ controllabile \Longleftrightarrow Sistema\ LA$$

Date tutte queste definizioni è necessario ora definire dei criteri che permettano di determinare se un sistema rientra in una delle tipologie sopra elencate. Si rende necessario a tale scopo definire il concetto di distribuzione.

Definizione 2.8 (Distribuzione) *Data una successione di vettori $\{g_1, \ldots, g_m\}$ si definisce distribuzione l'insieme generato da questi vettori:*

$$\Delta = span\{g_1, \ldots, g_m\}$$

Una distribuzione si dice involutiva *se è chiusa rispetto alle parentesi di Lie*[4]*:*

$$\Delta\ involutiva \Longleftrightarrow \forall f, g \in \Delta, \quad [f,g] \in \Delta$$

[3] Per brevità spesso si definisce il sistema semplicemente come *localmente controllabile (LC)*

[4] Dati due vettori f e g, si definisce l'operatore detto *parentesi di Lie* nel seguente modo:

$$[f,g] = \frac{\partial g}{\partial x} f - \frac{\partial f}{\partial x} g$$

Definizione 2.9 (Chiusura involutiva) *Si definisce* chiusura involutiva *di una distribuzione, e si indica con* $\bar{\Delta}$*, la più piccola distribuzione contenente* Δ *tale che se* $f, g \in \bar{\Delta}$ *allora* $[f, g] \in \bar{\Delta}$*.*

Vediamo ora alcune proprietà relative alle distribuzioni.

Proposizione 2.2 *Data una distribuzione* $\Delta_0(q) = span\{g_1(q), \ldots, g_m(q)\}$ *valgono le seguenti relazioni:*

- $\Delta_{i+1}(q) = \Delta_i(q) \bigcup span\{[g_j, v]\}$*, con:* $v \in \Delta_i(q)$ *e* $j = 1, \ldots, m$*;*
- $\Delta_{i+1} \supseteq \Delta_i$*;*
- $\Delta_{i+1} = \Delta_i \Longrightarrow \Delta_{i+2} = \Delta_i = \bar{\Delta}(q)$*.*

$\Delta_0(q)$ *si dice involutiva se* $\bar{\Delta} = \Delta_0$*, cioè se, facendo delle combinazioni, rimango comunque dentro allo stesso insieme.*

Un'ultima definizione relativa alle distribuzioni prima di passare ai teoremi che permettono di definire se un sistema è controllabile o meno è quella di distribuzione regolare.

Definizione 2.10 (Distribuzione regolare) *Una distribuzione* $\Delta(q)$ *si dice* regolare *se il relativo rango è costante per ogni* q*.*

A questo punto è possibile enunciare un paio di teoremi che, partendo dall'analisi delle distribuzioni appena introdotte, permettono di determinare le caratteristiche di interesse del sistema.

Teorema 2.1 (di Frobenius) *Se una distribuzione è regolare e involutiva allora essa è integrabile e il relativo sistema di controllo è olonomo e si può dimostrare che vale anche il viceversa.*

Teorema 2.2 (di Chow) *Dato il sistema:*

$$\dot{q} = g_1(q)u_1 + \ldots + g_m(q)u_m$$

esso è localmente controllabile (STLC) in q *se* $\bar{\Delta}(q) = \mathbb{R}^n$*, cioè se il rango della distribuzione è* n*.*

Avendo a disposizione tutti i mezzi esposti finora, è possibile verificare se una determinata configurazione può effettivamente essere raggiunta dal sistema partendo dalla condizione iniziale. Successivamente si può, come detto in precedenza, eseguire un cambio di variabili in modo da poter portare in sistema nella forma "in catena" e sfruttare uno degli algoritmi di controllo possibili che garantiscono il raggiungimento del risultato voluto.

2.2 Wheeled Mobile Robots (WMR)

Un interessante esempio di sistemi anolonomi che trovano vasta applicazione in ambito industriale è dato dai robot mobili su ruote. Ovviamente si può trovare un gran numero di diverse configurazioni per quanto riguarda questi veicoli, variando il numero di ruote, la loro disposizione, il controllo su di esse, ecc. Evidentemente essi rappresentano un caso tipico di sistemi anolonomi, essendo soggetti a vincoli che impediscono determinati movimenti, a seconda delle disposizione delle ruote. Principalmente gli obiettivi del controllo su questi oggetti sono di due tipi:

- inseguimento di una determinata traiettoria;

- stabilizzazione della posizione, cioè spostamento da un punto ad un altro.

Si vedrà che il secondo dei due obiettivi elencati presenta dei problemi maggiori rispetto al primo, che pur genera delle problematiche relative all'inseguimento.

Il modello più semplice per quanto riguarda un WMR è rappresentato dall'uniciclo, una singola ruota che si muove sul piano senza slittare. Più avanti si vedrà che il nostro robot può essere ricondotto ad un uniciclo ad esso equivalente. Le coordinate generalizzate che vanno a definire lo stato sono le coordinate sul piano e l'orientamento della ruota:

$$q = (x, y, \vartheta)$$

Il vincolo rappresentato dal fatto che la ruota non può slittare lateralmente si può esprimere nel seguente modo:

$$A(q)\dot{q} = \dot{x}\sin\vartheta - \dot{y}\cos\vartheta = 0$$

Esprimendo quindi le velocità ammissibili come una combinazione lineare dei vettori che generano lo spazio nullo della matrice $A(q)$[5], si può ottenere il modello cinematico del primo ordine:

$$\begin{bmatrix} \dot{x} \\ \dot{y} \\ \dot{\vartheta} \end{bmatrix} = g_1(q)v + g_2(q)\omega = \begin{bmatrix} \cos\vartheta \\ \sin\vartheta \\ 0 \end{bmatrix} v + \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} \omega$$

dove v e ω sono presi come ingressi di controllo e sono rispettivamente la velocità lineare della ruota e la relativa velocità angolare.

Date le definizioni della precedente sezione è possibile eseguire un'analisi del sistema. Indicando al solito la parentesi di Lie tra g_1 e g_2 come $[g_1, g_2]$, si può verificare che la condizione che garantisce la locale accessibilità data nel teorema 2.2 è soddisfatta:

$$rank\begin{pmatrix} g_1 & g_2 & [g_1, g_2] \end{pmatrix} = n = 3$$

Visto che stiamo considerando un sistema senza deriva, la proposizione 2.1 garantisce anche la controllabilità.

Infine, si vuole evidenziare l'effettiva possibilità di rendere il modello nella forma in catena già esposta. Definendo le seguenti trasformazioni per quanto riguarda le coordinate:

$$\begin{aligned} \chi_1 &= \vartheta \\ \chi_2 &= x\cos\vartheta + y\sin\vartheta \\ \chi_3 &= x\sin\vartheta - y\cos\vartheta \end{aligned}$$

e scegliendo una retroazione del tipo:

$$v = u_2 + \chi_3 u_1 \qquad \omega = u_1$$

si ottiene il seguente sistema in catena:

$$\begin{aligned} \dot{\chi}_1 &= u_1 \\ \dot{\chi}_2 &= u_2 \\ \dot{\chi}_3 &= \chi_2 u_1 \end{aligned}$$

che presenta tutte le caratteristiche esposte nella sezione precedente.

[5]La matrice $A(q)$ è definita, partendo dai vincoli sul sistema a_i, nel seguente modo:

$$A(q) = [a_1, \ldots, a_k] \in \mathbb{R}^{n\times k}$$

2.3 Sistema di visione

La caratteristica dei robot mobili su ruote (WMR) di essere dei sistemi anolonomi pone delle difficoltà per quanto riguarda la determinazione della posizione degli stessi nel piano cartesiano. Precisamente, si rende necessaria una derivazione della posizione del sistema per ottenere la velocità lineare e successivamente il modello cinematico del sistema deve essere integrato per ottenere la posizione del veicolo nel piano cartesiano. Evidentemente queste operazioni di integrazione e derivazione introducono degli errori che vanno ad accumularsi e rendono il valore di posizione ottenuto sempre meno accurato.

Un approccio interessante per evitare questo tipo di problemi è quello di utilizzare un sistema di visione che permetta di ottenere direttamente la posizione del robot richiesta dal controllore.

Ovviamente l'utilizzo della telecamera per estrarre informazioni riguardo al robot e all'ambiente circostante richiede un'adeguata calibrazione che spesso, a causa dell'incertezza sui parametri, può essere imprecisa e portare a degli errori che affliggono poi anche le informazioni relative alla posizione e all'orientamento del veicolo.

Per risolvere questo problema si fa uso di sistemi che prevedono un controllore in grado di adattare il comando alle incertezze relative alla calibrazione della telecamera, oltre che alle incertezze associate ai parametri meccanici del sistema.

Ci sono due modi diversi di procedere: si può generare la traiettoria di riferimento nello spazio della telecamera, oppure lavorare direttamente nello spazio delle operazioni. Ovviamente l'anello di controllo deve essere chiuso nello stesso spazio utilizzato per la generazione della traiettoria di riferimento.
Lavorare nello spazio della telecamera è indubbiamente più complesso, ma è indispensabile se la telecamera non è calibrata sufficientemente bene.
È possibile comunque definire delle matrici di trasformazione che permettano di passare da un piano ad un altro.

Si vede pertanto come l'introduzione di un sistema di visione da un lato abbia eliminato il problema legato agli errori d'integrazione, ma dall'altro abbia introdotto delle altre problematiche da risolvere.

Vediamo ora come è possibile rappresentare il modello del WMR nello spazio della telecamera e nello spazio delle operazioni.
Per quanto riguarda quest'ultimo, si vedrà che è possibile esprimere il modello cinematico nel seguente modo:

$$\dot{q} = S(q)v \tag{2.3.1}$$

dove $q(t), \dot{q}(t) \in \mathbb{R}^3$ sono definiti come:

$$q = [\ x_c \quad y_c \quad \vartheta\ ]^T \qquad \dot{q} = [\ \dot{x}_c \quad \dot{y}_c \quad \dot{\vartheta}\ ]^T \tag{2.3.2}$$

nelle quali $x_c(t)$, $y_c(t)$ e $\vartheta(t) \in \mathbb{R}^1$ denotano rispettivamente la posizione e l'orientamento del centro di massa del veicolo[6], $\dot{x}_c(t)$ e $\dot{y}_c(t)$ indicano le due componenti della velocità lineare del centro di massa, mentre $\dot{\vartheta}(t)$ rappresenta la velocità angolare del centro di massa. Infine la matrice $S(q) \in \mathbb{R}^{3\times 2}$ è definita nel seguente modo:

$$S(q) = \begin{bmatrix} \cos\vartheta & 0 \\ \sin\vartheta & 0 \\ 0 & 1 \end{bmatrix} \tag{2.3.3}$$

[6]Si assume che il centro di massa coincida con il centro dell'asse di rotazione del robot.

Il vettore velocità $v(t) \in \mathbb{R}^2$, ultima variabile da definire, ha la seguente forma:

$$v = [\ v_1 \quad v_2\ ]^T = [\ v_l \quad \dot{\vartheta}\ ]^T \tag{2.3.4}$$

con $v_l \in \mathbb{R}^1$ indicante la velocità lineare del centro di massa del veicolo.

Per quanto riguarda lo spazio della telecamera, si assume che il robot debba soddisfare lo stesso modello cinematico appena esposto:

$$\dot{\bar{q}} = S(\bar{q})\bar{v} \tag{2.3.5}$$

dove $\bar{q}(t) = [\ \bar{x}_c(t) \quad \bar{y}_c(t) \quad \bar{\vartheta}(t)\ ]^T \in \mathbb{R}^3$ rappresenta la posizione e l'orientamento del WMR nello spazio della telecamera e $\bar{v}(t) = [\ \bar{v}_1(t) \quad \bar{v}_2(t)\ ]^T \in \mathbb{R}^2$ indica le velocità lineare e angolare del veicolo nello spazio della telecamera.

Ovviamente sono richieste alcune accortezze di tipo tecnico sul sistema comprendente lo spazio di lavoro del robot e la telecamera fissata sopra di esso, in modo tale che:

- la telecamera sia in grado di catturare immagini attraverso l'intero piano di lavoro del robot;
- il sistema di visione sia in grado di determinare la posizione del centro di massa del veicolo riconoscendo delle caratteristiche fisiche, nel nostro caso dei LED opportunamente montati;
- il sistema di visione sia in grado di determinare l'orientamento del veicolo e quindi la direzione verso cui esso si sta muovendo, riconoscendo delle caratteristiche addizionali, nel nostro caso mediante ulteriori LED.

Si noti che non è necessario che il piano immagine sia parallelo al piano su cui si muove il veicolo, poiché mediante un'opportuna matrice di trasformazione (calcolata automaticamente nel nostro caso), è possibile eliminare la "distorsione" introdotta dalla prospettiva. Una volta ricondotti a questa situazione (piano immagine parallelo al piano di lavoro), il passaggio dallo spazio della telecamera allo spazio operativo è definito da ulteriori opportune matrici di trasformazione.
Per esprimere la posizione del veicolo nello spazio delle operazioni partendo dalla posizione nello spazio della telecamera si fa uso del cosiddetto modello pin-hole (a foro di spillo), ottenendo quanto segue (si osservi a riguardo anche la figura 3):

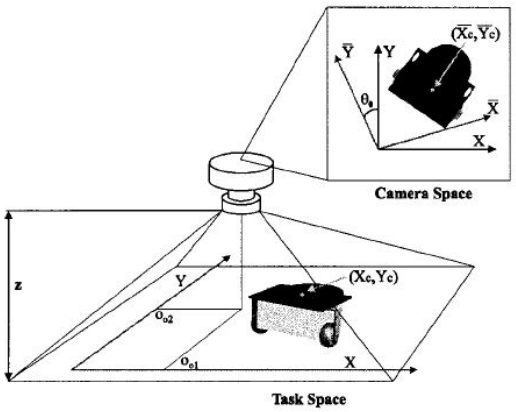


Fig. 3: Configurazione del sistema robot-telecamera

$$\begin{bmatrix} \bar{x}_c(t) \\ \bar{y}_c(t) \end{bmatrix} = HR\vartheta_0 \left(\begin{bmatrix} x_c(t) \\ y_c(t) \end{bmatrix} - \begin{bmatrix} O_{o1} \\ O_{o2} \end{bmatrix} \right) + \begin{bmatrix} O_{i1} \\ O_{i2} \end{bmatrix} \tag{2.3.6}$$

dove $H \in \mathbb{R}^{2\times 2}$ è una matrice diagonale, definita positiva e costante definita come segue:

$$H = \begin{bmatrix} \alpha_1 & 0 \\ 0 & \alpha_2 \end{bmatrix} \tag{2.3.7}$$

$\alpha_1, \alpha_2 \in \mathbb{R}^1$ sono delle costanti positive definite nel seguente modo:

$$\alpha_1 = \beta_1 \frac{\lambda}{z} \qquad \alpha_2 = \beta_2 \frac{\lambda}{z} \tag{2.3.8}$$

$z \in \mathbb{R}^1$ rappresenta l'altezza costante del centro ottico della telecamera rispetto al piano delle operazioni su cui si muove il veicolo. $\lambda \in \mathbb{R}^1$ è una costante che rappresenta la focale della telecamera mentre le costanti positive β_1 e β_2 indicano i fattori di scala della telecamera lungo le due direzioni.
La matrice $R(\vartheta_0)$ è una matrice di rotazione costante così definita:

$$R(\vartheta_0) = \begin{bmatrix} \cos\vartheta_0 & -\sin\vartheta_0 \\ \sin\vartheta_0 & \cos\vartheta_0 \end{bmatrix} \tag{2.3.9}$$

ϑ_0 rappresenta la rotazione in senso orario del sistema di coordinate nello spazio della telecamera rispetto a quello nello spazio delle operazioni.
$\begin{bmatrix} O_{o1} & O_{o2} \end{bmatrix}^T \in \mathbb{R}^2$ denota la proiezione del centro ottico della telecamera sullo spazio delle operazioni, mentre $\begin{bmatrix} O_{i1} & O_{i2} \end{bmatrix}^T \in \mathbb{R}^2$ indica il centro dell'immagine, definito come l'intersezione tra l'asse ottico ed il piano immagine.
Per ottenere il legame tra $\bar{v}_1(t)$ e $v_1(t)$, si prende la derivata di 2.3.6 e si sostituisce 2.3.1 al posto della derivata di 2.3.2, ottenendo le seguenti relazioni:

$$\begin{bmatrix} \dot{\bar{x}}_c \\ \dot{\bar{y}}_c \end{bmatrix} = \begin{bmatrix} v_1\alpha_1\cos(\vartheta+\vartheta_0) \\ v_1\alpha_2\sin(\vartheta+\vartheta_0) \end{bmatrix} \tag{2.3.10}$$

Moltiplicando $\dot{\bar{x}}_c$ per $(\frac{1}{\alpha_1})\cos(\vartheta+\vartheta_0)$ e $\dot{\bar{y}}_c$ per $(\frac{1}{\alpha_2})\sin(\vartheta+\vartheta_0)$ e sostituendo 2.3.5 al posto di $\dot{\bar{x}}_c$ e $\dot{\bar{y}}_c$, si ottiene la seguente espressione:

$$\begin{bmatrix} \frac{1}{\alpha_1}\cos\vartheta_0\cos(\vartheta+\vartheta_0) \\ \frac{1}{\alpha_2}\sin\vartheta_0\sin(\vartheta+\vartheta_0) \end{bmatrix} = \begin{bmatrix} v_1\cos^2(\vartheta+\vartheta_0) \\ v_1\sin^2(\vartheta+\vartheta_0) \end{bmatrix} \tag{2.3.11}$$

Che può essere anche scritta nella forma:

$$v_1 = T_1\bar{v}_1 \tag{2.3.12}$$

dove $T_1(\vartheta(t), \bar{\vartheta}(t)) \in \mathbb{R}^1$ è una funzione scalare positiva definita come:

$$T_1 = \frac{1}{\alpha_1}\cos\bar{\vartheta}\cos(\vartheta+\vartheta_0) + \frac{1}{\alpha_2}\sin\bar{\vartheta}\sin(\vartheta+\vartheta_0) > \gamma_1 \tag{2.3.13}$$

dove tutte le variabili sono state definite in precedenza, tranne γ_1 che è una costante positiva.

Per legare ora le velocità angolari nei due casi si parte sempre dalla relazione 2.3.5, eliminando $\bar{v}_1(t)$ nelle prime due righe dell'equazione vettoriale e ottenendo che:

$$\dot{\bar{y}}_c = \dot{\bar{x}}_c \tan\bar{\vartheta} \tag{2.3.14}$$

e quindi, sostituendo le espressioni riportate in 2.3.10 nella precedente espressione si ottiene:

$$\tan(\vartheta + \vartheta_0) = \frac{\alpha_1}{\alpha_2} \tan \bar{\vartheta} \tag{2.3.15}$$

Da questa espressione, derivando ed eseguendo qualche passaggio algebrico, si ottiene:

$$\dot{\vartheta} = T_2 \dot{\bar{\vartheta}} \tag{2.3.16}$$

dove $T_2(\vartheta(t)) \in \mathbb{R}^1$ è una funzione scalare positiva così definita:

$$T_2 = \frac{\alpha_1}{\alpha_2} \cos^2(\vartheta + \vartheta_0) + \frac{\alpha_2}{\alpha_1} \sin^2(\vartheta + \vartheta_0) > \gamma_2 \tag{2.3.17}$$

dove tutte le variabili sono state definite in precedenza, tranne γ_2 che è una costante positiva.

Facendo riferimento ora alle espressioni 2.3.12 e 2.3.16 è possibile definire la trasformazione invertibile generica tra le velocità nello spazio delle operazioni e quelle nello spazio della telecamera:

$$v = T_0 \bar{v} \tag{2.3.18}$$

dove $T_0(\vartheta(t), \bar{\vartheta}(t)) \in \mathbb{R}^2$ è definita come:

$$T_0 = \begin{bmatrix} T_1 & 0 \\ 0 & T_2 \end{bmatrix} \tag{2.3.19}$$

e tutte le variabili in gioco sono state definite nelle formule fin qui riportate.

Si vedrà più oltre che nel nostro caso si è preferito lavorare direttamente nello spazio operativo, considerato che il livello di accuratezza nell'inseguire il riferimento non era tale da poter risentire pesantemente di errori di calibrazione della telecamera. Ciò ha permesso di semplificare notevolmente il procedimento.

2.4 Sistema di controllo

Posture stabilization e trajectory following

Gli obbiettivi principali che si vogliono ottenere dal controllo di un WMR possono essere sostanzialmente due:

- *posture stabilization*: far muovere il veicolo dallo stato $q_d(t_0) = [x_d(t_0), y_d(t_0), \vartheta_d(t_0]$ allo stato $q_d(t_f) = [x_d(t_f), y_d(t_f), \vartheta_d(t_f]$
- *trajectory following*: far inseguire al veicolo una data traiettoria

Contrariamente a quanto ci si aspetterebbe, dalla teoria si ricava che la seconda operazione risulta più facile da risolvere della prima nel caso di WMR non olonomi: una spiegazione intuitiva di questo fatto può essere data in termini di un confronto tra il numero delle variabili controllate (le uscite del sistema) e il numero degli ingressi di controllo. Si consideri l'uniciclo: per questo tipo di veicolo sono disponibili due ingressi, ma sono necessarie tre variabili per determinare la sua configurazione (x, y, ϑ). Perciò risolvere il problema di *posture stabilization* richiede l'annullamento di tre errori di configurazione indipendenti; nel problema di *trajectory following*, invece, l'uscita da annullare $e_p = \begin{bmatrix} e_x \\ e_y \end{bmatrix}$ ha le stesse

dimensioni dell'ingresso: $u, e_p \in \mathbb{R}^2$.
A riguardo, dalla teoria si ricava che l'uniciclo non è stabilizzabile ad un punto mediante feed-back lineare (teorema di Brockett): il problema di *posture stabilization* richiede quindi un regolatore discontinuo e/o tempo-variante. Il problema di *trajectory following*, vedremo invece, può essere risolto anche con un feed-back lineare, se, ovviamente, la traiettoria di riferimento è compatibile con i vincoli di anolonomia del sistema.

Traiettorie e tempi di campionamento

In questa esperienza ci occuperemo solamente di *trajectory following*: come già accennato, il compito dell'algoritmo di controllo è in questo caso quello di far seguire al veicolo la traiettoria desiderata. È opportuno ricordare che, per definizione, una traiettoria non definisce solo un percorso, cioè una successione di punti per i quali passare, ma implicitamente anche una legge oraria, cioè gli istanti ai quali si deve passare per tali punti.
Nella nostra applicazione si è data la possibilità all'utente di specificare i punti per i quali passare, garantendo che il robot vada da un punto ad un altro in un tempo specificato e costante dt. È chiaro che ciò permette, implicitamente, di assegnare anche una legge temporale alla traiettoria, dal momento che specificando punti più vicini tra loro rallenta l'esecuzione del moto, mentre scegliere punti a maggiore distanza aumenta la velocità richiesta per passare da uno all'altro.
Come in quasi tutti i sistemi di controllo a tempo discreto, per il corretto funzionamento del regolatore è opportuno che il tempo di campionamento dt sia costante, e soprattutto sufficiente a garantire l'esecuzione di tutte le operazioni previste. Ciò ha richiesto, nel nostro caso, un'attenta analisi di diversi aspetti, tra cui: un'analisi del tempo massimo (*worst case*) richiesto per completare un ciclo dell'algoritmo, dopo aver chiaramente tolto tutte le operazioni che si possono eseguire off-line: questo tempo massimo è proprio il tempo di campionamento dt scelto, e poi l'introduzione di un loop di attesa, da eseguire nei casi in cui il tempo richiesto per i calcoli fosse minore del dt prefissato.

Feed-forward e feed-back

La generazione dei comandi da dare al robot prevede, in generale, un contributo a catena aperta (feed-forward), ed un contributo in retroazione (feed-back): i vantaggi di questa duplice azione sono ben noti. Gli algoritmi presentati, per entrambi i contributi, dipendono ovviamente dal modello del robot utilizzato, ma sono facilmente estendibili a WMR di tipo diverso.
Si vedrà che, per la nostra applicazione, gli algoritmi noti in letteratura non sono in grado di offrire prestazioni soddisfacenti: ciò è dovuto all'elevato periodo di campionamento dt, che ha reso necessaria l'introduzione di algoritmi *ad hoc*.

Ulteriori problematiche affrontate

Oltre a quelle già esposte, nello sviluppo del controllore vanno considerate altre problematiche, tipiche di sistemi in catena chiusa: in particolare la saturazione degli attuatori (esiste una velocità massima alla quale le ruote del robot possono muoversi), la quantizzazione dei riferimenti (le velocità assegnabili alle ruote devono essere multiple di una velocità minima), il tempo di assestamento della velocità intorno al suo riferimento (il

comando di una nuova velocità richiede del tempo per essere eseguito).

3 Apparato hardware

Nella presente sezione verrà fornita una breve descrizione di tutto l'hardware utilizzato nel corso dell'esperienza: si potrà così facilmente notare l'effettiva semplicità dello stesso.

3.1 WMR Khepera

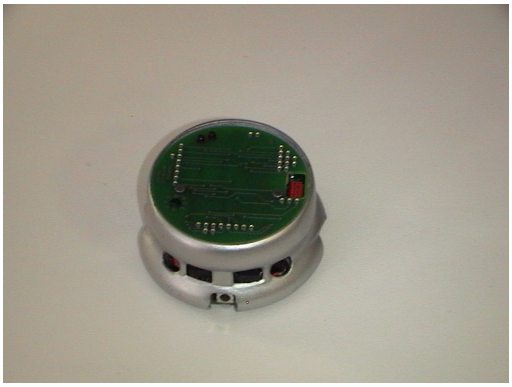


Fig. 4: WMR Khepera

L'oggetto del controllo è il veicolo riportato in figura 4 , le cui specifiche sono riportate nella tabella 1.

KHEPERA II SPECIFICATIONS	
Elements	**Technical information**
Processor	Motorola 68331, 25MHz
RAM	512 Kbytes
Flash	512 Kbytes Programmable via serial port
Motion	2 DC brushed servo motors 25:1 reduction gearbox Incremental encoders: 600 pulses per revolution of the wheel (roughly 12 pulses per mm of robot motion) Speed control, position control
Speed	Max: 60 cm/s, Min: 2 cm/s
Sensors	8 Infra-red proximity and ambient light sensors with up to 100 mm range Power consumption
I/O	3 Analog Inputs (0-4.3V, 8bit)
Power	Power Adapter Rechargeable NiMH Batteries
Autonomy	1 hour, moving continuosly. Additional turrets will reduce battery life.
Communication	Standard Serial Port RS232: 9.6, 19.2, 38.4, 57.6, 115.2 kbps
Extension Bus	Expansion modules can be added to the robot using the K-Extension bus.
Size	Diameter: 70mm Height: 30mm
Weight	Approx 80g
Payload	Approx 250g

Tab. 1: Specifiche Khepera

3.2 Sistema di LED

In figura 5 è riportata la basetta su cui sono stati montati i LED da usare per determinare, mediante opportuni algoritmi, la posizione e l'orientamento del veicolo. Il circuito è molto semplice, in quanto è costituito da una resistenza posta in serie a ciascun LED, dimensionata opportunamente in base al voltaggio del trasformatore di alimentazione.

La basetta è stata successivamente fissata alla parte superiore del veicolo in modo tale da rendere quest'ultimo localizzabile mediante la web-cam (figura 6). La retta congiungente i tre led è perpendicolare all'asse di rotazione delle ruote, ed il led centrale è posizionato in corrispondenza del centro di rotazione del veicolo

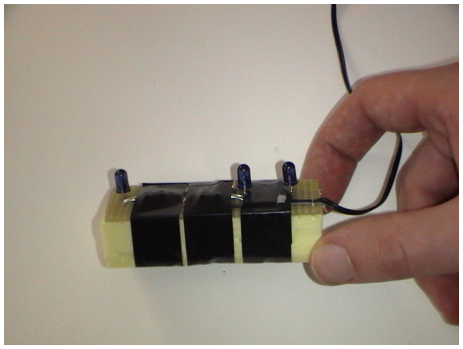


Fig. 5: LED per la localizzazione del WMR

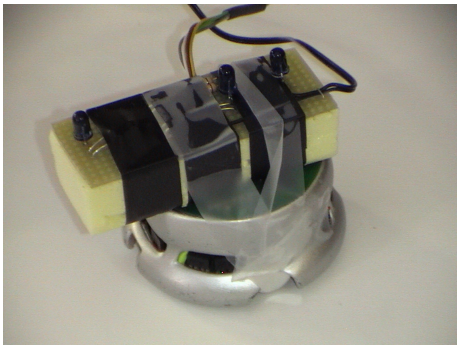


Fig. 6: Basetta montata sul WMR

Nella tabella 2 vengono riportate le specifiche dei led, trascritte dal data sheet:

LED TECHNICAL DATA	
Case	5mm waterclear
Viewing angle	20°
Power	1.6V
Current	20mA typ., 30mA max.
Wave lenght	940nm (infrared)
Intensity	8mW

Tab. 2: Specifiche Led

3.3 Web-cam e filtro

Per eseguire l'acquisizione dell'immagine è stata utilizzata una semplice web-cam, riportata in figura 7 , le cui specifiche sono riportate nella tabella 3.

Fig. 7: Webcam

SPECIFICHE WEBCAM	
Risoluzione	640×480
Frame rate	15 frame/s
Connessione	USB 2.0

Tab. 3: Specifiche web-cam

Allo scopo di rilevare solamente la porzione di luce di interesse si è utilizzato un filtro all'infrarosso, opportunamente fissato alla web-cam (figure 8 e 9).

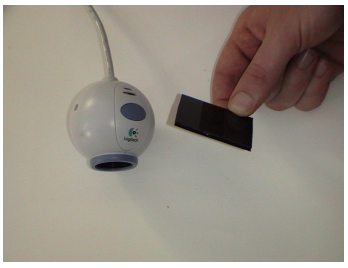


Fig. 8: Web-cam e filtro infrarosso

Fig. 9: Web-cam e filtro fissato su di essa

3.4 Sistema di comunicazione

Per la comunicazione tra l'unità di calcolo (PC) e il veicolo è stata utilizzata l'interfaccia seriale (figura 10) fornita con il veicolo stesso.

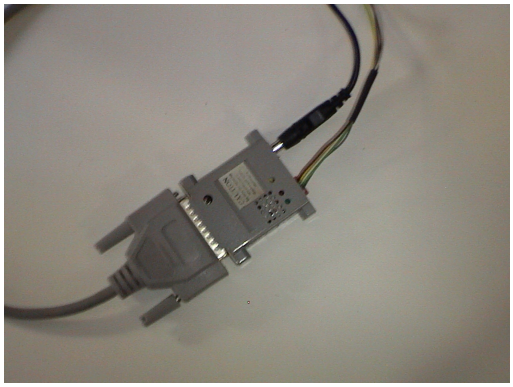


Fig. 10: Cavo seriale per la comunicazione

3.5 Ambiente per la sperimentazione

Per quanto riguarda l'area di lavoro su cui il WMR si può muovere, si è scelto di utilizzare una riproduzione realizzata su cartone di una semplice strada di città (figura 11), comprendente una rotatoria ed un parcheggio.

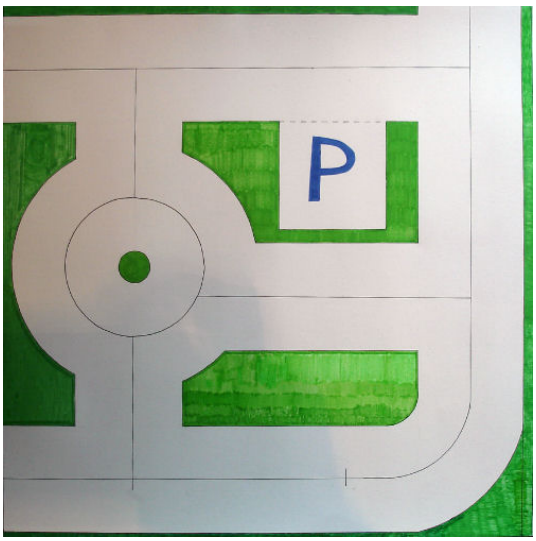


Fig. 11: Rappresentazione della città virtuale

L'immagine di questo piano di lavoro è stata poi utilizzata anche per realizzare l'interfaccia grafica di cui si parlerà nel seguito.

4 Realizzazione del controllo

4.1 Sistema di visione

4.1.1 Acquisizione immagine

Per l'acquisizione dell'immagine è stato utilizzato un file pre-compilato, corredato dalla relativa libreria: *vfm.m* e *vfm.dll*[7]. Il file sfrutta le risorse video del sistema per acquisire l'immagine e permetterne il suo utilizzo in MATLAB. Per la configurazione è sufficiente digitare nella riga di comando `vfm('configsource')`, per definire la sorgente di acquisizione, e `vfm('configformat')` per definire il formato dell'immagine da importare. Le finestre che appariranno alla digitazione dei suddetti comandi sono riportate nelle figure 12, 13, 14, 15.

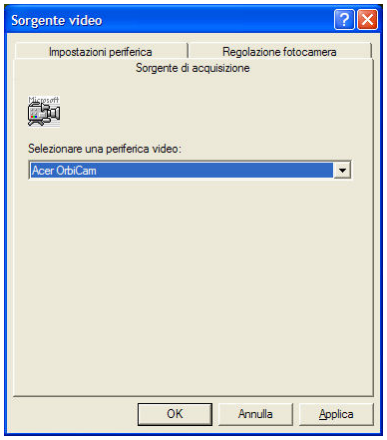


Fig. 12: *configsource* - Sorgente di acquisizione

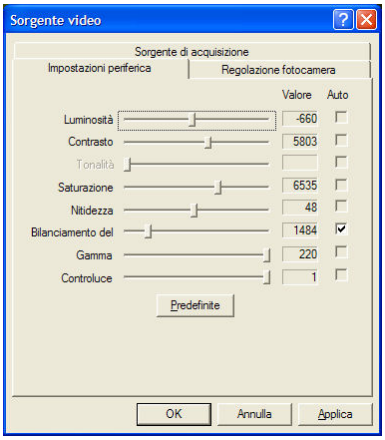


Fig. 13: *configsource* - Impostazioni periferica

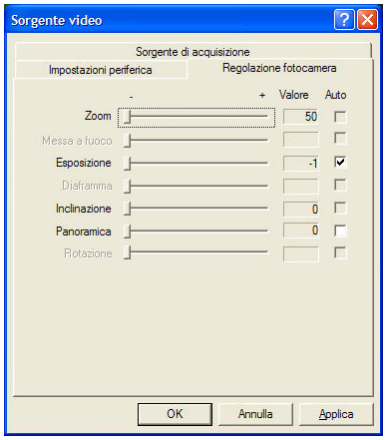


Fig. 14: *configsource* - Regolazione fotocamera

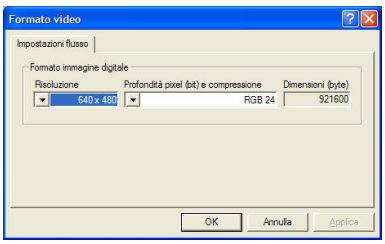


Fig. 15: *configformat* - Impostazioni di flusso

[7] Autore: School of Computing Sciences, Norwich, UK, *www2.cmp.uea.cac.uk*

Si possono notare la semplicità e l'estrema intuitività di tutti i comandi e le opzioni presenti.

Dopo aver impostato tutti i parametri iniziali si passa all'acquisizione vera e propria dell'immagine, eseguita per mezzo del comando `I=vfm('grab',F)`.
`I` rappresenta la matrice d'immagine in formato RGB[8] ($M \times N \times 3 \times F$) ed `F` è il numero di frame che verrà acquisito in MATLAB.

4.1.2 Problema della prospettiva d'immagine

L'acquisizione per mezzo di web-cam introduce ovviamente un problema di prospettiva dell'immagine (la web-cam in generale non sarà infatti collocata esattamente sopra il rettangolo di lavoro): per riportarsi all'immagine "corretta" è necessario il calcolo della matrice di trasformazione prospettica che porta dal piano della videocamera al piano originale.
Il calcolo di tale matrice è stato eseguito mediante un'acquisizione preliminare del campo di lavoro, al fine di identificarne le coordinate P_{image} dei quattro vertici. Sapendo la posizione reale P_{real} dei precedenti punti, grazie alla funzione MATLAB

```
maketform(P_real,P_image)
```

si può trovare la matrice geometrica `T` di trasformazione che mappa i punti P_{image} nei punti P_{real}.
La trasformazione così calcolata va applicata, mediante la funzione `imtransform`, ad ogni singolo frame acquisito, al fine di riportarlo nelle proporzioni vere: l'immagine così ottenuta rappresenta l'immagine dall'alto del piano di lavoro.

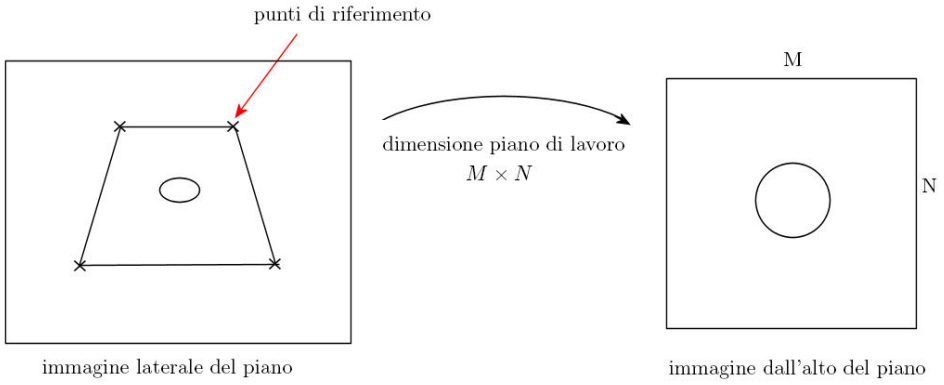


Fig. 16: Trasformazione dell'immagine acquisita

Nonostante questo accorgimento, si è in ogni caso cercato di acquisire l'immagine posizionando la web-cam sopra il piano di lavoro, in modo che l'errore di parallasse non fosse molto elevato.
Un ulteriore vantaggio della tecnica utilizzata risiede nel fatto che eventuali distorsioni introdotte dalla lente della telecamera vengono decisamente ridotte.

4.1.3 Utilizzo degli infrarossi

La tecnica che prevede la visione all'infrarosso è la più sicura dal punto di vista dei disturbi legati alle condizioni di luminosità e ai riflessi non desiderati. Lo svantaggio di questo

[8] Red Green Blue

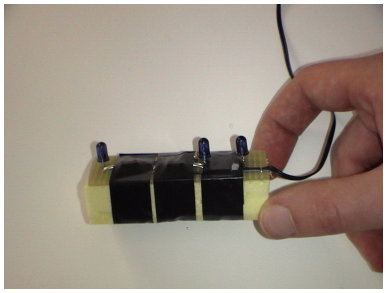


Fig. 17: Schedina con LED all'infrarosso

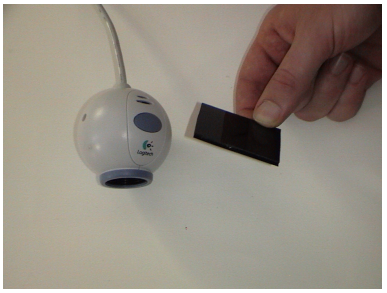


Fig. 18: Web-cam e filtro all'infrarosso

approccio è che richiede l'utilizzo di dispositivi non comunemente usati e quindi più difficili da reperire.

Nella presente esperienza si è applicato alla web-cam un filtro all'infrarosso in grado di isolare solamente i segnali luminosi di interesse, cioè quelli provenienti dai LED installati sul *Khepera*.
Le figure (17) e (18) illustrano la schedina sui quali sono installati i LED all'infrarosso, e la web-cam con il filtro all'infrarosso utilizzati per l'acquisizione.

Gli obiettivi dell'acquisizione sono rappresentati da tre LED all'infrarosso aventi un angolo di apertura di circa 20°. Quest'ultimo valore è stato supposto sufficiente considerando che la web-cam sia posizionata sopra il piano di lavoro con una sufficiente accuratezza.

La struttura circuitale del supporto dei LED è rappresentata da una semplice resistenza dimensionata appositamente, considerando il voltaggio del trasformatore utilizzato per l'alimentazione dei LED.

4.1.4 Tecnica di identificazione della posizione dell'obiettivo

Al fine di identificare la posizione del veicolo è sufficiente un unico LED.
Il programma di acquisizione utilizzato in questa prima fase di progetto si basa quindi sulla seguente logica:

1. Acquisizione dell'immagine;
2. Trasformazione in bianco e nero[9];
3. Trasformazione di prospettiva mediante la matrice calcolata in precedenza;
4. Limitazione dell'area di ricerca nel settore previsto dal filtro di Kalman;
5. Ricerca dei punti con valore 1;
6. Essendoci un unico LED, la sua posizione può essere ottenuta con la media geometrica dei punti trovati.

Per limitare il tempo di ricerca del punto voluto si è utilizzato il filtro di Kalman senza ingressi, definito dalle seguenti matrici (ovviamente queste valgono solo in questa prima

[9]La matrice di immagine contiene a questo punto solo due valori: 0 e 1. La conversione in bianco e nero si basa su un *thresholding* dell'intensità luminosa dell'immagine originale

fase di progetto, durante la quale i LED vengono spostati a mano, e serve solo per testare l'algoritmo di visione):

$$\bar{x} = \begin{bmatrix} x \\ v_x \\ y \\ v_y \end{bmatrix}$$

Stato composto da posizione e velocità del punto cercato

$$A = \begin{bmatrix} 1 & 0.9 & 0 & 0 \\ 0 & 1 & 0 & 0 \\ 0 & 0 & 1 & 0.9 \\ 0 & 0 & 0 & 1 \end{bmatrix} \quad Q = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0.5 & 0 & 0 \\ 0 & 0 & 1 & 0 \\ 0 & 0 & 0 & 0.5 \end{bmatrix}$$

$$C = \begin{bmatrix} 1 & 0 & 0 & 0 \\ 0 & 0 & 1 & 0 \end{bmatrix} \qquad R = \begin{bmatrix} 80 & 0 \\ 0 & 80 \end{bmatrix}$$

La matrice R si riferisce all'errore della web-cam, mentre la matrice Q è relativa all'errore del modello. Vale la pena di ricordare che il precedente modello è stato utilizzato per testare l'efficienza dell'algoritmo spostando manualmente il LED nel piano immagine ed acquisendone l'immagine.
Una non corretta calibrazione del filtro di Kalman, porta ad avere un inseguimento che, anche se buono, non risulta robusto. Una possibile alternativa può essere rappresentata dal filtro particellare: quest'ultimo garantisce una robustezza superiore, che va pagata però con una complessità di calcolo più elevata.
Si tenga presente che nella nostra applicazione ridurre il tempo di elaborazione è fondamentale, ed entrambi i suddetti filtri possono rappresentare l'anello debole del sistema real-time se non studiati ed implementati ad-hoc.

4.1.5 Orientamento del veicolo

Al fine di poter ottenere l'orientamento del veicolo oggetto del controllo si è scelto di posizionare tre LED in linea al di sopra del veicolo stesso, posti però a distanze diverse tra loro, come rappresentato in figura.

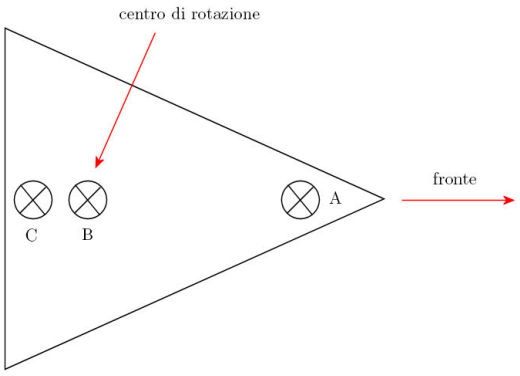


Fig. 19: Disposizione dei tre LED

La scelta effettuata prevede il posizionamento del LED centrale in corrispondenza del centro di rotazione del robot, in modo tale da poter essere preso come riferimento per la determinazione della posizione. L'orientamento invece è ricavato prendendo la retta passante per i tre punti e calcolandone il coefficiente angolare.

4.1.6 Identificazione dei tre punti

All'interno della procedura che porta all'identificazione dei punti rappresentanti la posizione dei tre LED si possono distinguere i seguenti passi:

1. Acquisizione immagine (RGB).

2. Trasformazione di prospettiva.

3. Ricerca all'interno dell'immagine dei punti con intensità luminosa[10] < 30: questi sono i punti candidati per la ricerca dei LED. Sicuramente i punti candidati sono in numero maggiore di tre (tipicamente un centinaio), ciò è dovuto a due motivi: prima di tutto un LED crea un alone luminoso, e non un singolo punto. Inoltre, nonostante il filtro all'infrarosso, ci sono dei punti "falsi" dovuti a rumore sovrapposto all'immagine.

4. Implementazione del classificatore " k-means".

   Consideriamo quest'ultimo punto. Il classificatore k-means ha la particolarità di conoscere in partenza il numero di punti o classi da trovare (nel nostro caso i punti da trovare sono 3). Lo schema di principio dell'algoritmo è il seguente:

   a) inizializzazione con tre punti casuali, B_1, B_2, B_3, presi tra i candidati, e con distanza reciproca compatibile con la disposizione reale dei LED
   b) dato M, insieme dei punti da classificare, si calcola per ogni elemento di M la distanza dai tre punti iniziali dati che, per il momento, rappresentano i baricentri (non aggiornati) delle tre classi cercate. L'elemento in esame, $e_i \in M$, viene posto nella classe relativa al baricentro più vicino a lui.
   c) una volta terminata la scansione degli elementi di M si ricalcolano i tre baricentri $B_1^\star$, $B_2^\star$ e $B_3^\star$. Con questi tre baricentri si itera nuovamente al punto precedente ciclicamente: dopo qualche ciclo ognuno dei punti $B_1^\star$, $B_2^\star$ e $B_3^\star$ converge al baricentro reale dei tre aloni principali individuati nella figura

   Normalmente sono sufficienti $4-5$ cicli per ottenere un'ottima stima dei baricentri cercati. Grazie all'inizializzazione introdotta, e riportata al punto a, invece bastano due cicli dell'algoritmo k-means.

5. Dati i tre baricentri B_1, B_2 e B_3, basandosi sulla distanza tra di essi si identificano i punti A, B e C indicati in figura 19. Questa identificazione si basa su una serie di condizioni `if`.

6. Determinazione dell'orientamento: per ricavare l'orientamento del veicolo si utilizzano le coordinate dei punti C ed A, in modo tale da sfruttare la distanza maggiore tra di essi al fine minimizzare l'errore commesso.

   Dati quindi $C = (x_C, y_C)$ e $A = (x_A, y_A)$ si calcola l'orientamento mediante la seguente formula:

   $$\vartheta = atan2(y_A - y_C, x_A - x_C)$$

   dove l'angolo calcolato può variare all'interno dell'intervallo $[-\pi, \pi]$.

[10] Si ricordi che il nero è rappresentato dal valore 255 mentre il bianco dal valore 0. Utilizzare un *thresholding* direttamente dell'immagine RGB permette di eliminare la conversione in bianco e nero, computazionalmente molto pesante

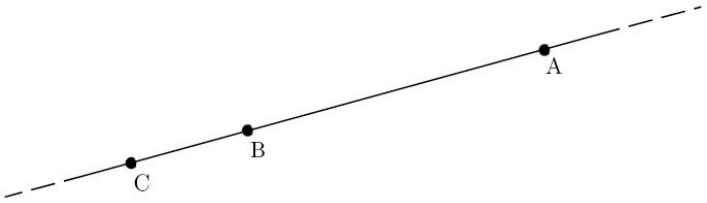


Fig. 20: Retta passante per i tre punti individuati

7. La posizione del veicolo viene infine definita facendo riferimento al punto $B = (x_B, y_B)$:

$$x = x_B$$
$$y = y_B$$

Si tenga presente che, nel caso non vengano rilevati i punti per un qualsiasi motivo, la funzione restituisce in uscita il valore `Inf`. In quest'ultimo caso si può agire in modi diversi: se si sta facendo uso del filtro di Kalman, ad esempio, si può considerare l'utilizzo di quello che prevede la perdita di pacchetti. In altri casi si può scegliere di bloccare il robot, e inviare un messaggio di errore, oppure di basare il controllo sulla stima precedente ecc.

Tornando alla stima della posizione attuale del robot mediante filtro di Kalman o filtro particellare, per aiutare la ricerca da parte dell'algoritmo di visione, la scelta effettuata in questa fase è quella di non limitare la ricerca in una zona ristretta, visto che la dimensione ridotta dell'immagine ($500 \times 500\, px$) fa sì che ciò non influisca in modo rilevante sul tempo di elaborazione (circa per il 2%). Questo approccio porta ad un considerevoile vantaggio visto che, non dovendo predire il punto attorno al quale effettuare la ricerca dell'obiettivo, non è necessario l'impiego del filtro di Kalman, con conseguente guadagno in termini di tempo di calcolo. Come già detto, la scelta fatta risulta valida vista la dimensione ridotta del campo di lavoro, $50 \times 50\, cm$, e la risoluzione della telecamera non troppo elevata, 640×480. Tuttavia, nonostante non ne sia richiesto l'impiego in questo caso, si effettuerà ugualmente lo studio del filtro di Kalman tempo variante, da utilizzare nel caso si disponga di un hardware più avanzato.

4.1.7 Filtro di Kalman tempo variante

Partiamo considerando le equazioni del modello, che, come si vedrà nel prossimo capitolo, hanno la seguente forma:

$$\dot{x} = v \cos\vartheta$$
$$\dot{y} = v \sin\vartheta$$
$$\dot{\vartheta} = \frac{v_s - v_d}{2l}$$

dove:

- v_s: velocità della ruota sinistra;
- v_d: velocità della ruota destra;
- l: distanza tra le due ruote;
- $v = \frac{v_d + v_s}{2} = \sqrt{\dot{x}^2 + \dot{y}^2}$

Scegliendo di considerare l'angolo $\vartheta(t)$ costante in tutto il periodo ed uguale al valore $\vartheta(k+1)$ alla fine del periodo stesso, nel calcolo della posizione è possibile passare ad un sistema ridotto e quindi di più facile trattazione.

Definendo le seguenti grandezze:

$$S = Y = \begin{bmatrix} x \\ y \\ \vartheta \end{bmatrix}$$

è possibile considerare la seguente evoluzione di stato a tempo discreto:

$$\begin{cases} S(k+1) = A_k S(k) + B_k u(k) + Q w_1(k) \\ Y(k) = C_k S(k) + R w_2(k) \end{cases} \tag{4.1.1}$$

con $w_1(k) \perp w_2(k)$ e $u(k) = [v_s \; v_d]^T$ come ingresso.

Esplicitando le tre componenti si troverà che l'equazione può anche essere riscritta nel seguente modo:

$$\begin{cases} x(k+1) = x(k) + \dfrac{v_d(k) + v_s(k)}{2} \cos\left(\vartheta(k) + \dfrac{v_s(k) - v_d(k)}{2l} dt\right) dt \\ y(k+1) = y(k) + \dfrac{v_d(k) + v_s(k)}{2} \sin\left(\vartheta(k) + \dfrac{v_s(k) - v_d(k)}{2l} dt\right) dt \\ \vartheta(k+1) = \vartheta(k) + \dfrac{v_s(k) - v_d(k)}{2l} dt \end{cases} \tag{4.1.2}$$

Avendo riportato anche queste espressioni possiamo ora calcolare le espressioni per le matrici che compaiono in (4.1.1).

$$A_k = \begin{bmatrix} \frac{\partial x(k+1)}{\partial x(k)} & \frac{\partial x(k+1)}{\partial y(k)} & \frac{\partial x(k+1)}{\partial \vartheta(k)} \\ \frac{\partial y(k+1)}{\partial x(k)} & \frac{\partial y(k+1)}{\partial y(k)} & \frac{\partial y(k+1)}{\partial \vartheta(k)} \\ \frac{\partial \vartheta(k+1)}{\partial x(k)} & \frac{\partial \vartheta(k+1)}{\partial y(k)} & \frac{\partial \vartheta(k+1)}{\partial \vartheta(k)} \end{bmatrix}_k = \begin{bmatrix} 1 & 0 & a \\ 0 & 1 & b \\ 0 & 0 & 1 \end{bmatrix}_k \tag{4.1.3}$$

con:

$$a = \left[-\frac{v_d(k) + v_s(k)}{2} \sin\left(\vartheta(k) + \frac{v_s(k) - v_d(k)}{2l} dt\right) dt \right]_k$$

$$b = \left[\frac{v_d(k) + v_s(k)}{2} \cos\left(\vartheta(k) + \frac{v_s(k) - v_d(k)}{2l} dt\right) dt \right]_k$$

$$B_k = \begin{bmatrix} \frac{\partial x(k+1)}{\partial v_s(k)} & \frac{\partial x(k+1)}{\partial v_d(k)} \\ \frac{\partial y(k+1)}{\partial v_s(k)} & \frac{\partial y(k+1)}{\partial v_d(k)} \\ \frac{\partial \vartheta(k+1)}{\partial v_s(k)} & \frac{\partial \vartheta(k+1)}{\partial v_d(k)} \end{bmatrix} = \begin{bmatrix} c - d & c + d \\ e + f & e - f \\ \frac{dt}{2l} & -\frac{dt}{2l} \end{bmatrix}_k \tag{4.1.4}$$

con:

$$c = \frac{1}{2} \cos\left(\vartheta(k) \frac{v_s(k) - v_d(k)}{2l} dt\right) dt$$

$$d = \frac{v_d(k) + v_s(k)}{4l} \sin\left(\vartheta(k) + \frac{v_s(k) - v_d(k)}{2l} dt\right) dt$$

$$e = \frac{1}{2} \sin\left(\vartheta(k) \frac{v_s(k) - v_d(k)}{2l} dt\right) dt$$

$$f = \frac{v_d(k) + v_s(k)}{4l} \cos\left(\vartheta(k) + \frac{v_s(k) - v_d(k)}{2l} dt\right) dt$$

$$C_k = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix}_k = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.1.5}$$

Una volta definite tutte le precedenti espressioni è finalmente possibile ottenere le equazioni relative al filtro di Kalman, per quanto riguarda la predizione:

$$\begin{aligned} \hat{S}(k+1|k) &= A_k S(k) + B_k u(k) \\ P(k+1|k) &= A_k P(k|k) A_k^T + Q \end{aligned} \tag{4.1.6}$$

e l'aggiornamento:

$$\begin{aligned} K(k+1) &= P(k+1|k) C_k^T (C_k P(k+1|k) C_k^T + R)^{-1} \\ \hat{S}(k+1|k+1) &= \hat{S}(k+1|k) + K(k+1)(y(k+1) - C_k \hat{S}(k+1|k)) \\ P(k+1|k+1) &= P(k+1|k) - K(k+1) C_k P(k+1|k) \end{aligned} \tag{4.1.7}$$

Vediamo ora le equazioni relative ad un sistema più accurato del precedente, ottenuto nel caso in cui non si faccia l'approssimazione di considerare l'angolo costante. Vale ancora l'evoluzione riportata in (4.1.1) ma stavolta le equazioni che si ottengono considerando le singole componenti sono le seguenti:

$$\begin{cases} x(k+1) = x(k) + \dfrac{v_d(k) + v_s(k)}{2} \dfrac{2l}{v_s(k) - v_d(k)} \left[\sin\left(\vartheta(k) + \dfrac{v_s(k) - v_d(k)}{2l} dt \right) - \sin \vartheta(k) \right] dt \\ y(k+1) = y(k) + \dfrac{v_d(k) + v_s(k)}{2} \dfrac{2l}{v_s(k) - v_d(k)} \left[\cos \vartheta(k) - \cos\left(\vartheta(k) + \dfrac{v_s(k) - v_d(k)}{2l} dt \right) \right] dt \\ \vartheta(k+1) = \vartheta(k) + \dfrac{v_s(k) - v_d(k)}{2l} dt \end{cases}$$

E di conseguenza anche le espressioni per le matrici saranno diverse.

$$A_k = \begin{bmatrix} \frac{\partial x(k+1)}{\partial x(k)} & \frac{\partial x(k+1)}{\partial y(k)} & \frac{\partial x(k+1)}{\partial \vartheta(k)} \\ \frac{\partial y(k+1)}{\partial x(k)} & \frac{\partial y(k+1)}{\partial y(k)} & \frac{\partial y(k+1)}{\partial \vartheta(k)} \\ \frac{\partial \vartheta(k+1)}{\partial x(k)} & \frac{\partial \vartheta(k+1)}{\partial y(k)} & \frac{\partial \vartheta(k+1)}{\partial \vartheta(k)} \end{bmatrix}_k = \begin{bmatrix} 1 & 0 & a \\ 0 & 1 & b \\ 0 & 0 & 1 \end{bmatrix}_k \tag{4.1.8}$$

con:

$$\begin{aligned} a &= l \frac{v_d(k) + v_s(k)}{v_s(k) - v_d(k)} \left[\cos\left(\vartheta(k) + \frac{v_s(k) - v_d(k)}{2l} dt \right) - \cos \vartheta(k) \right] dt \\ b &= l \frac{v_d(k) + v_s(k)}{v_s(k) - v_d(k)} \left[\sin\left(\vartheta(k) + \frac{v_s(k) - v_d(k)}{2l} dt \right) - \sin \vartheta(k) \right] dt \end{aligned}$$

$$B_k = \begin{bmatrix} \frac{\partial x(k+1)}{\partial v_s(k)} & \frac{\partial x(k+1)}{\partial v_d(k)} \\ \frac{\partial y(k+1)}{\partial v_s(k)} & \frac{\partial y(k+1)}{\partial v_d(k)} \\ \frac{\partial \vartheta(k+1)}{\partial v_s(k)} & \frac{\partial \vartheta(k+1)}{\partial v_d(k)} \end{bmatrix} = \begin{bmatrix} c & d \\ e & f \\ \frac{dt}{2l} & -\frac{dt}{2l} \end{bmatrix}_k \tag{4.1.9}$$

con:

$$c = -\frac{2lv_d(k)}{(v_s(k)-v_d(k))^2}\left[\sin\left(\vartheta(k)+\frac{v_s(k)-v_d(k)}{2l}dt\right)-\sin\vartheta(k)\right]dt+ \\ +\frac{v_d(k)+v_s(k)}{v_s(k)-v_d(k)}\cos\left(\vartheta(k)+\frac{v_s(k)-v_d(k)}{2l}dt\right)\frac{dt^2}{2}$$

$$d = \frac{2lv_s(k)}{(v_s(k)-v_d(k))^2}\left[\sin\left(\vartheta(k)+\frac{v_s(k)-v_d(k)}{2l}dt\right)-\sin\vartheta(k)\right]dt- \\ -\frac{v_d(k)+v_s(k)}{v_s(k)-v_d(k)}\cos\left(\vartheta(k)+\frac{v_s(k)-v_d(k)}{2l}dt\right)\frac{dt^2}{2}$$

$$e = -\frac{2lv_d(k)}{(v_s(k)-v_d(k))^2}\left[\cos\vartheta(k)-\cos\left(\vartheta(k)+\frac{v_s(k)-v_d(k)}{2l}dt\right)\right]dt+ \\ +\frac{v_d(k)+v_s(k)}{v_s(k)-v_d(k)}\sin\left(\vartheta(k)+\frac{v_s(k)-v_d(k)}{2l}dt\right)\frac{dt^2}{2}$$

$$f = \frac{2lv_s(k)}{(v_s(k)-v_d(k))^2}\left[\cos\vartheta(k)-\cos\left(\vartheta(k)+\frac{v_s(k)-v_d(k)}{2l}dt\right)\right]dt- \\ -\frac{v_d(k)+v_s(k)}{v_s(k)-v_d(k)}\sin\left(\vartheta(k)+\frac{v_s(k)-v_d(k)}{2l}dt\right)\frac{dt^2}{2}$$

$$C_k = \begin{bmatrix} 1 & 0 & 0 \\ 0 & 1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \tag{4.1.10}$$

Una volta definite tutte le matrici le equazioni per il filtro di Kalman sono le stesse.

4.2 Sistema di controllo

4.2.1 Modello del sistema e generazione del feed-forward

Si consideri la figura (21), che illustra le principali grandezze del veicolo alle quali si farà riferimento in questa sezione: in particolare v_S e v_D rappresentano le velocità delle ruote sinistra e destra rispettivamente, v è la velocità tangenziale corrispondente, L è la lunghezza del "semiasse" (cioè la distanza tra le ruote ed il centro del veicolo), infine ϑ è l'orientamento dell'uniciclo, misurato in senso orario, a partire dall'asse delle ascisse.

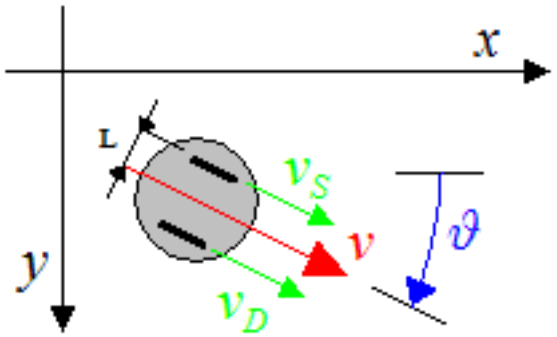


Fig. 21: Illustrazione del Khepera

Il vincolo di non slittamento impone che sia:

$$\tan(\vartheta) = \frac{\dot{y}}{\dot{x}} \tag{4.2.1}$$

nella quale x e y rappresentano le coordinate attuali del centro del veicolo, per le quali si ha anche che:

$$\begin{cases} \dot{x} &= v\cos(\vartheta) \\ \dot{y} &= v\sin(\vartheta) \end{cases}$$

Si ha inoltre:

$$v = \sqrt{\dot{x}^2 + \dot{y}^2} = \frac{v_D + v_S}{2} \tag{4.2.2}$$

mentre per la velocità angolare vale:

$$\omega = \dot{\vartheta} = \frac{v_S - v_D}{2L} \tag{4.2.3}$$

Dalle (4.2.2) e (4.2.3) si possono ricavare le velocità v_S e v_D da assegnare al robot, una volta che siano note v e ω, essendo infatti:

$$\begin{cases} v_S &= v + L\omega \\ v_D &= v - L\omega \end{cases}$$

Le precedenti relazioni permettono dunque di trattare il veicolo a due ruote precedente come un robot con un'unica ruota, disposta al centro del robot stesso, e della quale si possono controllare la velocità angolare e tangenziale.

Il problema che si vuole quindi risolvere in questa sezione è quello di trovare le grandezze nominali $v = v_d$ e $\omega = \omega_d$ che permettano al veicolo precedente di inseguire una traiettoria desiderata del tipo:

$$(x_d(t), y_d(t)) \quad t \in [0, T]$$

Dalla (4.2.1) si ha:

$$\vartheta = ATAN2(\dot{y}, \dot{x}) \tag{4.2.4}$$

dove $ATAN2$ indica l'arcotangente su quattro quadranti, non definita solo se entrambi gli argomenti sono nulli.
Derivando la precedente espressione rispetto al tempo, imponendo $x = x_d(t)$ e $y = y_d(t)$ si ricava:

$$\dot{\vartheta}_d(t) = \omega_d(t) = \frac{\ddot{y}_d(t)\dot{x}_d(t) - \ddot{x}_d(t)\dot{y}_d(t)}{\dot{x}_d^2(t) + \dot{y}_d^2(t)} \tag{4.2.5}$$

Dalla (4.2.2), imponendo allo stesso modo $x = x_d(t)$ e $y = y_d(t)$, si ricava poi:

$$v_d(t) = \sqrt{\dot{x}_d^2(t) + \dot{y}_d^2(t)} \tag{4.2.6}$$

Le ultime due espressioni trovate permettono, in teoria, di ricavare $v_d(t)$ e $\omega_d(t)$ una volta che sia stata assegnata la traiettoria desiderata:

$$(x_d(t), y_d(t)) \quad t \in [0, T]$$

Il problema principale del metodo appena illustrato è che richiede la conoscenza delle derivate prima e seconda della traiettoria di riferimento, il che rende, nel nostro caso,

di fatto molto impreciso l'inseguimento della traiettoria: bisogna infatti ricordare che l'algoritmo di controllo andrà implementato su una piattaforma Hardware che garantisce intervalli di campionamento dt dell'ordine di qualche decimo di secondo. Si consideri a riguardo l'approssimazione discreta della derivata prima:

$$\dot{f}(t) \simeq \frac{f(t+dt) - f(t)}{dt}$$

e della derivata seconda:

$$\ddot{f}(t) \simeq \frac{\dot{f}(t+dt) - \dot{f}(t)}{dt} \simeq \frac{f(t+2dt) - 2f(t+dt) + f(t)}{dt^2}$$

è evidente come l'approssimazione introdotta peggiori all'aumentare di dt.
I risultati sperimentali hanno confermato che il metodo ora illustrato non offre buone prestazioni con un tempo di campionamento dt dell'ordine dei decimi di secondo, com'è il nostro caso, a meno di non far procedere molto lentamente il robot. Quest'ultima soluzione, però, oltre ad essere poco "efficiente", ha anche un altro svantaggio: le velocità v_S e v_D del robot non possono essere impostate su un valore qualsiasi, ma solo su valori multipli di $v_{min} = 8[mm/s]$; è evidente quindi che minori sono le velocità desiderate, maggiori saranno gli effetti di questa "quantizzazione". Viceversa, avere riferimenti di velocità maggiori, riduce in percentuale l'effetto dell'errore, che sappiamo essere compreso in $\pm v_{min}$.
Le considerazioni precedenti inducono a cercare un algoritmo diverso per il calcolo dei riferimenti di velocità: si consideri a riguardo la figura (22).

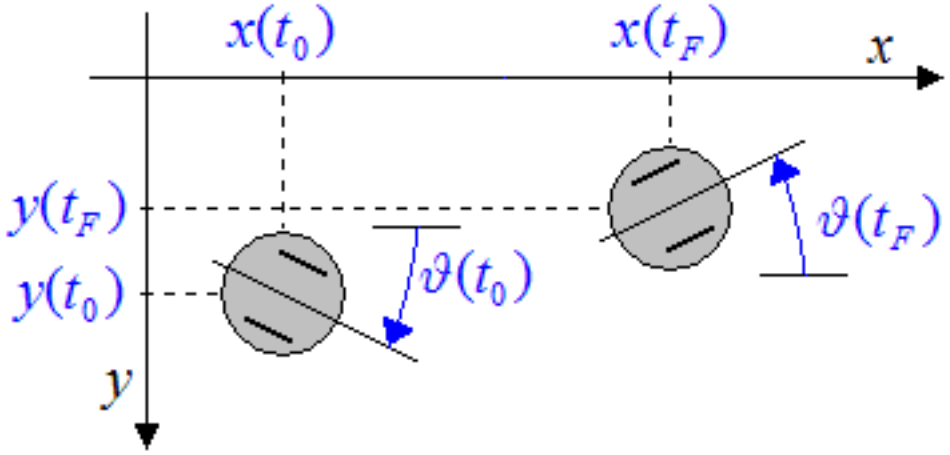


Fig. 22: Derivazione del feed-forward

Il nostro problema è quello di portare il robot dallo stato $q(t_0) = \begin{bmatrix} x(t_0) \\ y(t_0) \\ \vartheta(t_0) \end{bmatrix}$ allo stato $q(t_F) = \begin{bmatrix} x(t_F) \\ y(t_F) \\ \vartheta(t_F) \end{bmatrix}$ in un tempo $dt = t_F - t_0$.

Integrando le equazioni:

$$\begin{cases} \dot{x}(t) &= v(t)\cos(\vartheta(t)) \\ \dot{y}(t) &= v(t)\sin(\vartheta(t)) \end{cases}$$

nell'intervallo $[t_0, t_F]$ si ottengono le seguenti:

$$x(t_F) = x(t_0) + \int_{t_0}^{t_F} v(t)\cos(\vartheta(t))dt \tag{4.2.7}$$

$$y(t_F) = y(t_0) + \int_{t_0}^{t_F} v(t)\sin(\vartheta(t))dt \tag{4.2.8}$$

Ricordando che le velocità di riferimento $v_S(t)$ e $v_D(t)$ in ingresso al robot possono essere aggiornate solamente all'inizio dell'intervallo $[t_0, t_F]$, si ha:

$$\begin{cases} |v_S(t)| = |v_S(t_0)| & \forall t \in [t_0, t_F] \\ |v_D(t)| = |v_D(t_0)| & \forall t \in [t_0, t_F] \end{cases}$$

Per intervalli di campionamento dt sufficientemente brevi, si può supporre che valga, più in generale, la seguente espressione:

$$\begin{cases} v_S(t) = v_S(t_0) & \forall t \in [t_0, t_F] \\ v_D(t) = v_D(t_0) & \forall t \in [t_0, t_F] \end{cases}$$

perciò dalle (4.2.2) e (4.2.3) si ricava anche:

$$\begin{cases} v(t) = v(t_0) & \forall t \in [t_0, t_F] \\ \dot{\vartheta}(t) = \dot{\vartheta}(t_0) & \forall t \in [t_0, t_F] \end{cases}$$

che sostituite nelle (4.2.7) e (4.2.8) danno:

$$\frac{x(t_F) - x(t_0)}{v(t_0)} = \int_{t_0}^{t_F} \cos(\vartheta(t))dt = \int_{t_0}^{t_F} \frac{\cos(\vartheta(t))\dot{\vartheta}(t)}{\dot{\vartheta}(t)}dt = \frac{1}{\dot{\vartheta}(t_0)}\sin(\vartheta(t))\Big|_{t_0}^{t_F}$$

$$\frac{y(t_F) - y(t_0)}{v(t_0)} = \int_{t_0}^{t_F} \sin(\vartheta(t))dt = \int_{t_0}^{t_F} \frac{\sin(\vartheta(t))\dot{\vartheta}(t)}{\dot{\vartheta}(t)}dt = -\frac{1}{\dot{\vartheta}(t_0)}\cos(\vartheta(t))\Big|_{t_0}^{t_F}$$

dalle quali si ottiene infine:

$$\dot{\vartheta}(t_0) = \underbrace{v(t_0)\frac{\sin(\vartheta(t_F)) - \sin(\vartheta(t_0))}{x(t_F) - x(t_0)}}_{(\star)} = \underbrace{v(t_0)\frac{\cos(\vartheta(t_0)) - \cos(\vartheta(t_F))}{y(t_F) - y(t_0)}}_{(\star\star)} \tag{4.2.9}$$

La precedente relazione permette, nota $v(t_0)$, di ricavare $\dot{\vartheta}(t_0)$ tale da portare il robot dallo stato $q(t_0) = \begin{bmatrix} x(t_0) \\ y(t_0) \\ \vartheta(t_0) \end{bmatrix}$ allo stato $q(t_F) = \begin{bmatrix} x(t_F) \\ y(t_F) \\ \vartheta(t_F) \end{bmatrix}$.
Per quanto riguarda $v(t_0)$, essa si può trovare integrando la (4.2.2), in modo da ottenere:

$$v(t_0) = \sqrt{\left(\frac{x(t_F) - x(t_0)}{t_F - t_0}\right)^2 + \left(\frac{y(t_F) - y(t_0)}{t_F - t_0}\right)^2} \tag{4.2.10}$$

Si noti che $v(t_0)$ risulta definita per ogni configurazione scelta, mentre per quanto riguarda $\dot{\vartheta}(t_0)$ conviene utilizzare

$(\star)$ nel caso in cui $y(t_F) = y(t_0)$ o se $\cos(\vartheta(t_0)) = \cos(\vartheta(t_F))$
$(\star\star)$ nel caso in cui $x(t_F) = x(t_0)$ o se $\sin(\vartheta(t_0)) = \sin(\vartheta(t_F))$

In questo modo $\dot{\vartheta}(t_0)$ risulta non definita solo quando $y(t_F) = y(t_0)$ e $x(t_F) = x(t_0)$, cioè quando $v(t_0) = 0$, che corrisponde ad una situazione di pura rotazione del robot. In questo caso si può allora porre:

$$\dot{\vartheta}(t_0) = \frac{\vartheta(t_F) - \vartheta(t_0)}{t_F - t_0} \tag{4.2.11}$$

Si noti che la precedente espressione si potrebbe sempre utilizzare al posto della (4.2.9) per il calcolo di $\dot{\vartheta}(t_0)$, tuttavia quest'ultima, pur essendo computazionalmente più pesante, sembra garantire, a livello sperimentale, migliori prestazioni. Ciò appare indubbiamente strano, dal momento che la (4.2.11) sembra essere la migliore approssimazione di $\dot{\vartheta}(t_0)$. In realtà bisogna considerare prima di tutto l'elevato valore del periodo di campionamento, che, come già spiegato, rende imprecisa l'approssimazione discreta della derivata. Inoltre bisogna ricordare che entrambi i metodi utilizzano, per il calcolo di $v(t_0)$, un'approssimazione che, non è da escludere, possa farsi sentire maggiormente nel caso si applichi la (4.2.11) invece che la (4.2.9).

Scelta di $\vartheta(k)$
Dal momento che, nella nostra applicazione, la traiettoria di riferimento viene fornita dall'utente in maniera discontinua, "per punti" successivamente interpolati da una *spline*, la scelta dei valori di $\vartheta(k)$ non è univoca, a differenza di quanto avviene nel continuo, dove, per il vincolo di non slittamento, si deve avere $\vartheta = ATAN2(\dot{y}, \dot{x})$. L'idea che sorge spontanea è quindi quella di dare, come riferimento:

$$\vartheta_d(k) = ATAN2\left(y_d(k+1) - y_d(k), x_d(k+1) - x_d(k)\right)$$

La relazione precedente implica che, all'istante k, si vuole che il veicolo sia già orientato nella direzione del punto $(x_d(k+1), y_d(k+1))$.
Tuttavia, come si intuisce facilmente dalla figura (23), per avere una traiettoria più *"smooth"* conviene porre invece:

$$\vartheta_d(k) = ATAN2\left(y_d(k+1) - y_d(k-1), x_d(k+1) - x_d(k-1)\right)$$

Una volta ottenute, con il metodo precedente, v e ω, da queste si ricavano v_S e v_D: in condizioni ideali, e con un dt sufficientemente piccolo, questi due riferimenti dovrebbero permettere al robot di seguire abbastanza fedelmente la traiettoria. Tuttavia bisogna tenere conto di diversi fattori che disturbano il sistema:

- le ruote del veicolo possono slittare
- le velocità impostabili nel robot devono essere multiple di $v_{min} = 8[mm/s]$
- le velocità sono soggette ad un valore massimo
- il controllo di velocità non è ideale, e la velocità reale impiega un tempo non nullo per assestarsi al riferimento

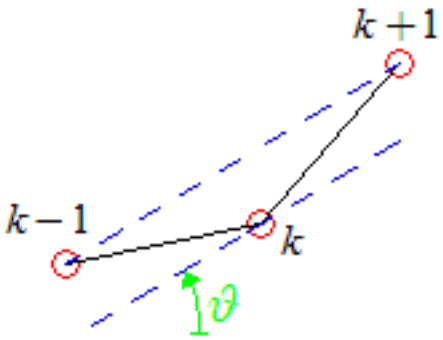


Fig. 23: Scelta di $\vartheta(k)$

- il modello potrebbe essere leggermente sbagliato (in particolare la lunghezza del semiasse e il raggio delle ruote)

È per questo che si rende necessaria l'introduzione di un controllo in retroazione che riesca a compensare gli errori nell'inseguimento della traiettoria desiderata.

4.2.2 Algoritmi di feed-back

Verranno descritti ora quattro diversi algoritmi che sono stati utilizzati per progettare il controllo in retroazione del sistema. Dei primi tre (per i quali si rimanda a [1]) si darà solo un cenno, dal momento che i risultati con essi ottenuti non sono stati soddisfacenti. Il motivo principale di ciò è che tutti i primi tre algoritmi si basano in massima parte sulla conoscenza delle derivate della traiettoria di riferimento, che, come già osservato, nel nostro caso provoca molte imprecisioni, visto l'elevato valore di dt.

Feed-back con linearizzazione del modello in prossimità della traiettoria

Questo algoritmo sfrutta una linearizzazione dell'uniciclo attorno alla traiettoria.
Si definisce l'errore di inseguimento della traiettoria come:

$$e = \begin{bmatrix} e_1 \\ e_2 \\ e_3 \end{bmatrix} = \begin{bmatrix} \cos(\vartheta) & \sin(\vartheta) & 0 \\ -\sin(\vartheta) & \cos(\vartheta) & 0 \\ 0 & 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} x_d - x \\ y_d - y \\ \vartheta_d - \vartheta \end{bmatrix} \tag{4.2.12}$$

e si introducono due ingressi virtuali, secondo le:

$$\begin{aligned} v &= v_d \cos(e_3) - u_1 \\ \omega &= \omega_d - u_2 \end{aligned} \tag{4.2.13}$$

Dalle ultime due relazioni si ricava:

$$\dot{e} = \begin{bmatrix} 0 & \omega_d & 0 \\ -\omega_d & 0 & 0 \\ 0 & 0 & 0 \end{bmatrix} e + \begin{bmatrix} 0 \\ \sin e_3 \\ 0 \end{bmatrix} v_d + \begin{bmatrix} 1 & 0 \\ 0 & 0 \\ 0 & 1 \end{bmatrix} \cdot \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} \tag{4.2.14}$$

Si introduce a questo punto il seguente feed-back lineare:

$$\begin{aligned} u_1 &= -k_1 e_1 \\ u_2 &= -k_2 sign(v_d(t)) e_2 - k_3 e_3 \end{aligned}$$

Scegliendo i guadagni k_1,k_2,k_3 in modo che:

$$k_1 = k_3 = 2\xi a \qquad k_2 = \frac{a^2 - \omega_d^2(t)}{|v_d(t)|} \qquad \xi \in (0,1), a > 0$$

si ottiene, per il sistema a catena chiusa, la seguente equazione caratteristica:

$$(\lambda + 2\xi a)(\lambda^2 + 2\xi a\lambda + a^2)$$

che ha un autovalore reale in $-2\xi a$ e una coppia di autovalori complessi coniugati con frequenza naturale a e coefficiente di smorzamento ξ.
Il problema è che k_2 tende a infinito quando $v_d \to 0$: a questo problema si può ovviare imponendo $a = a(t) = \sqrt{\omega_d^2(t) + bv_d^2(t)}$, con $b > 0$, cosicché:

$$k_1 = k_3 = 2\xi\sqrt{\omega_d^2(t) + bv_d^2(t)} \qquad k_2 = b|v_d(t)|$$

Il parametro b rappresenta un grado di libertà ulteriore a disposizione del progettista.
In termini delle grandezze originali, si ottiene la seguente legge di controllo:

$$\begin{aligned} v &= v_d\cos(\vartheta_d - \vartheta) + k_1[\cos(\vartheta)(x_d - x) + \sin(\vartheta)(y_d - y)] \\ \omega &= \omega_d + k_2 sign(v_d)[\cos(\vartheta)(x_d - x) - \sin(\vartheta)(y_d - y)] + k_3(\vartheta_d - \vartheta) \end{aligned}$$

Feed-back non lineare mediante equazione di Lyapunov

Si consideri ancora una volta l'equazione (4.2.14), e si ponga:

$$\begin{aligned} u_1 &= -k_1(v_d(t), \omega_d(t))e_1 \\ u_2 &= -\overline{k}_2 v_d(t)\tfrac{\sin(e_3)}{e_3}e_2 - k_3(v_d(t), \omega_d(t))e_3 \end{aligned} \tag{4.2.15}$$

con $\overline{k}_2 > 0$ costante, $k_1(\cdot,\cdot)$ e $k_3(\cdot,\cdot)$ funzioni positive e continue.
Si può dimostrare allora che, se:

- v_d e ω_d sono limitate
- $\dot{v}_d$ e $\dot{\omega}_d$ sono limitate
- $v_d(t) \neq 0$ quando $t \to \infty$
- $\omega_d(t) \neq 0$ quando $t \to \infty$

allora la legge di controllo (4.2.15) rende l'origine $e = 0$ asintoticamente stabile.
La dimostrazione è basata sull'utilizzo della funzione di Lyapunov

$$V = \frac{\overline{k}_2}{2}(e_1^2 + e_2^2) + \frac{e_3^2}{2}.$$

Dalle equazioni (4.2.12), (4.2.13) e (4.2.15) si ottiene, per il sistema originale, la seguente legge di controllo:

$$\begin{aligned} v &= v_d\cos(\vartheta_d - \vartheta) + k_1(v_d, \omega_d)[\cos(\vartheta)(x_d - x) + \sin(\vartheta)(y_d - y)] \\ \omega &= \omega_d + \overline{k}_2 v_d\frac{\sin(\vartheta_d - \vartheta)}{\vartheta_d - \vartheta}[\cos(\vartheta)(x_d - x) - \sin(\vartheta)(y_d - y)] + k_3(v_d, \omega_d)(\vartheta_d - \vartheta) \end{aligned}$$

Come nel caso precedente, si possono scegliere i guadagni in modo che:

$$k_1(v_d(t), \omega_d(t)) = k_3(v_d(t), \omega_d(t)) = 2\xi\sqrt{\omega_d^2(t) + bv_d^2(t)} \qquad \overline{k}_2 = b$$

con $b > 0$ e $\xi \in (0,1)$.

Dynamic feedback linearization

Si definisca un nuovo vettore:

$$\eta = (x, y)$$

Dalle equazioni dell'uniciclo si ricava:

$$\dot{\eta} = \begin{bmatrix} \dot{x} \\ \dot{y} \end{bmatrix} = \begin{bmatrix} \cos\vartheta & 0 \\ \sin\vartheta & 0 \end{bmatrix} \cdot \begin{bmatrix} v \\ \omega \end{bmatrix}$$

Indicando con a l'accelerazione lineare dell'uniciclo, ed introducendo un'ulteriore variabile $\xi = v$ si può scrivere:

$$v = \xi \quad , \quad \dot{\xi} = a \quad , \quad \Rightarrow \quad \dot{\eta} = \xi \begin{bmatrix} \cos\vartheta \\ \sin\vartheta \end{bmatrix}$$

Differenziando ulteriormente si trova:

$$\ddot{\eta} = \dot{\xi} \begin{bmatrix} \cos\vartheta \\ \sin\vartheta \end{bmatrix} + \xi\dot{\vartheta} \begin{bmatrix} -\sin\vartheta \\ \cos\vartheta \end{bmatrix} = \underbrace{\begin{bmatrix} \cos\vartheta & -\xi\sin\vartheta \\ \sin\vartheta & \xi\cos\vartheta \end{bmatrix}}_{B} \cdot \begin{bmatrix} a \\ \omega \end{bmatrix}$$

nella quale la matrice B risulta non singolare se $\xi = v \neq 0$. Sotto questa ipotesi si può allora porre:

$$\begin{bmatrix} a \\ \omega \end{bmatrix} = B^{-1} \begin{bmatrix} u_1 \\ u_2 \end{bmatrix}$$

da cui:

$$\ddot{\eta} = \begin{bmatrix} \ddot{\eta}_1 \\ \ddot{\eta}_2 \end{bmatrix} = \begin{bmatrix} u_1 \\ u_2 \end{bmatrix} = u \tag{4.2.16}$$

Essendo:

$$B^{-1} = \begin{bmatrix} \cos(\vartheta) & \sin(\vartheta) \\ -\frac{\sin(\vartheta)}{\xi} & \frac{\cos(\vartheta)}{\xi} \end{bmatrix}$$

si ottiene il seguente compensatore:

$$\begin{array}{rcl} \dot{\xi} & = & u_1\cos\vartheta + u_2\sin\vartheta \\ v & = & \xi \\ \omega & = & \frac{u_2\cos\vartheta - u_1\sin\vartheta}{\xi} \end{array} \tag{4.2.17}$$

Si noti che:

$$\begin{bmatrix} \ddot{x} = u_1 \\ \ddot{y} = u_2 \end{bmatrix}$$

Per quest'ultimo sistema, che risulta lineare e disaccoppiato, si può introdurre la seguente retroazione PD, che garantisce l'inseguimento della traiettoria $(x_d(t), y_d(t))$:

$$\begin{array}{rcl} u_1 & = & \ddot{x}_d(t) + k_p(x_d(t) - x) + k_d(\dot{x}_d(t) - \dot{x}) \\ u_2 & = & \ddot{y}_d(t) + k_p(y_d(t) - y) + k_d(\dot{y}_d(t) - \dot{y}) \end{array} \tag{4.2.18}$$

con i guadagni $k_p > 0$, $k_d > 0$.

Feed-back a quadranti

In questa sezione viene presentato un nuovo algoritmo di feed-back, da noi ideato, appositamente progettato per garantire buone prestazioni anche nel caso in cui il tempo di campionamento dt sia piuttosto elevato. Abbiamo già notato, infatti, come tutti i precedenti compensatori non fossero in grado di garantire buoni risultati, a meno di non far procedere molto lentamente il veicolo: in questo caso, invece, la retroazione risulta molto "robusta", anche nel caso di intervalli di campionamento particolarmente elevati. Questo algoritmo, inoltre, non richiede la conoscenza delle derivate della traiettoria di riferimento (ciò che rendeva gli altri compensatori inadatti alla nostra applicazione), non richiede $v_d(t) \neq 0 \quad \forall t$ e $\omega_d(t) \neq 0 \quad \forall t$, e risulta molto meno sensibile ai parametri del modello. Il tutto, però, a discapito di una minor precisione nella regolazione.

L'idea di base è molto semplice: si consideri a riguardo la figura (24), nella quale ϑ

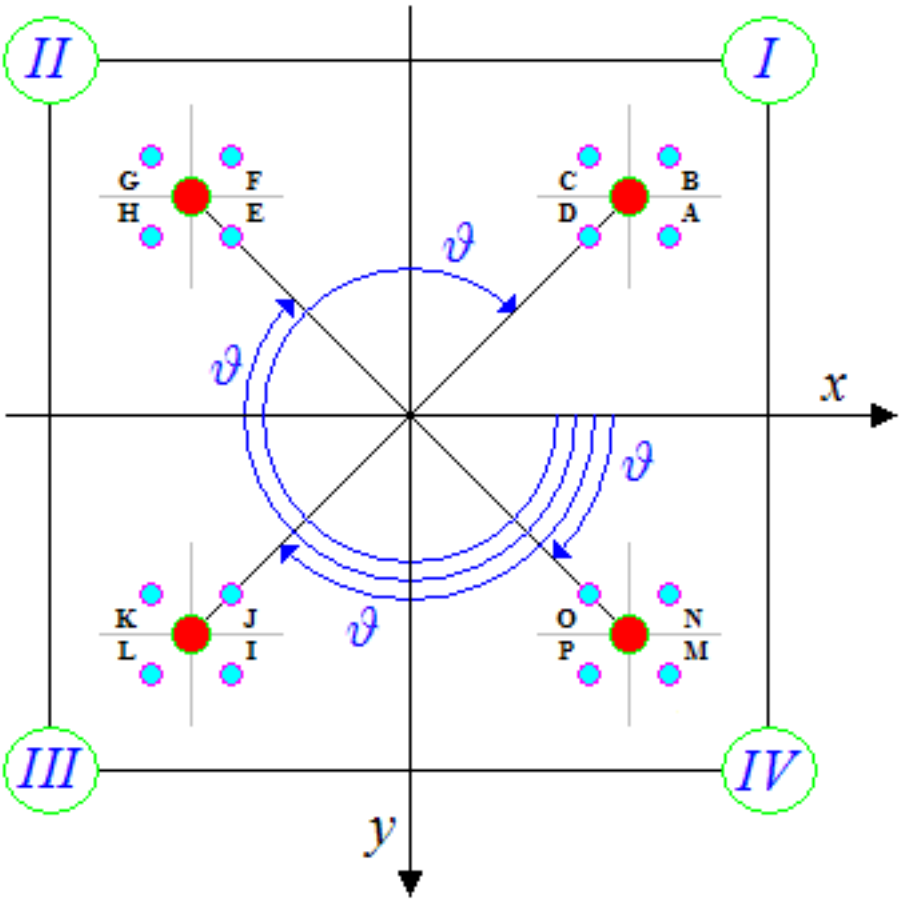


Fig. 24: Feed-back a quadranti

rappresenta, al solito, l'orientamento attuale del robot: in generale $\vartheta(t)$ può appartenere ad uno qualsiasi dei quattro quadranti **I**, **II**, **III** e **IV**. Per ognuno di questi quadranti, il cerchio rosso rappresenta la posizione attuale del robot, mentre i quattro cerchi azzurri individuano le possibili aree dentro le quali si vorrebbe che il robot fosse: supponiamo ad esempio che ϑ appartenga al primo quadrante, e che quindi la posizione attuale $q = (x, y)$

del robot sia quella indicata dal cerchio rosso in alto a destra. La posizione desiderata $q_d = (x_d, y_d)$, in generale, apparterrà ad una delle quattro aree indicate con $\mathcal{A}$, $\mathcal{B}$, $\mathcal{C}$ o $\mathcal{D}$. L'algoritmo si basa sul seguente ragionamento:

- $q_d \in \mathcal{A} \Rightarrow$ conviene aumentare ω, in modo da orientare il robot nella direzione corretta, ruotandolo in senso orario
- $q_d \in \mathcal{B} \Rightarrow$ conviene aumentare v, in modo da poter far recuperare al robot (che è nella direzione corretta) il ritardo rispetto al riferimento
- $q_d \in \mathcal{C} \Rightarrow$ conviene diminuire ω, in modo da orientare il robot nella direzione corretta, ruotandolo in senso antiorario
- $q_d \in \mathcal{D} \Rightarrow$ conviene diminuire v, in modo da eliminare l'anticipo del robot (che è nella direzione corretta) rispetto al riferimento

In maniera simile si può procedere per gli altri quattro quadranti.
Un'immediata obiezione potrebbe sorgere considerando casi simili a quello di figura (25): il punto desiderato q_d (rappresentato, al solito, dal cerchio azzurro), pur appartenendo a $\mathcal{M}$, risulta molto spostato alla destra del robot: intuitivamente quindi servirebbe un aumento di ϑ per orientare correttamente il veicolo. Il precedente algoritmo, invece, impone semplicemente un aumento di v, visto che, essendo $q_d \in \mathcal{M}$, il robot è apparentemente nella direzione corretta, ed è soltanto in ritardo.

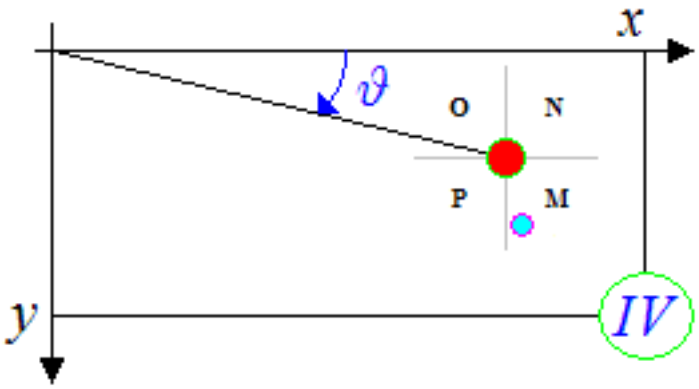


Fig. 25: Feed-back a quadranti

A prima vista, quindi, in casi come questi l'algoritmo sembra non essere efficace. Si osservi però che, aumentando v e non variando ω, dopo qualche istante il punto q_d cadrà in $\mathcal{P}$, ed allora l'algoritmo comanderà l'aumento di ω, in modo da riportare il robot al giusto orientamento.
Generalizzando a tutti i casi possibili, e ponendo:

$$e_x = x_d - x$$
$$e_y = y_d - y$$

si ottiene la seguente tabella:

caso	comando	e_x	e_y	$\sin\vartheta$	$\cos\vartheta$
$\mathcal{A}$	aumento ω	>0	>0	<0	>0
$\mathcal{B}$	aumento v	>0	<0	<0	>0
$\mathcal{C}$	riduco ω	<0	<0	<0	>0
$\mathcal{D}$	riduco v	<0	>0	<0	>0
$\mathcal{E}$	riduco v	>0	>0	<0	<0
$\mathcal{F}$	aumento ω	>0	<0	<0	<0
$\mathcal{G}$	aumento v	<0	<0	<0	<0
$\mathcal{H}$	riduco ω	<0	>0	<0	<0
$\mathcal{I}$	riduco ω	>0	>0	>0	<0
$\mathcal{J}$	riduco v	>0	<0	>0	<0
$\mathcal{K}$	aumento ω	<0	<0	>0	<0
$\mathcal{L}$	aumento v	<0	>0	>0	<0
$\mathcal{M}$	aumento v	>0	>0	>0	>0
$\mathcal{N}$	riduco ω	>0	<0	>0	>0
$\mathcal{O}$	riduco v	<0	<0	>0	>0
$\mathcal{P}$	aumento ω	<0	>0	>0	>0

dalla quale, per ispezione, si ricava:

$$\begin{cases} \text{aumento } \omega & \Leftrightarrow & e_x \sin\vartheta < 0 & \text{e} & e_y \cos\vartheta > 0 \\ \text{riduco } \omega & \Leftrightarrow & e_x \sin\vartheta > 0 & \text{e} & e_y \cos\vartheta < 0 \\ \text{aumento } v & \Leftrightarrow & e_y \sin\vartheta > 0 & \text{e} & e_x \cos\vartheta > 0 \\ \text{riduco } v & \Leftrightarrow & e_y \sin\vartheta < 0 & \text{e} & e_x \cos\vartheta < 0 \end{cases}$$

e, semplificando ulteriormente, si ricavano questi quattro casi possibili:

$$\begin{cases} e_x e_y \sin\vartheta\cos\vartheta > 0 & , & e_y \sin\vartheta > 0 & \Rightarrow & \text{aumento } v \\ e_x e_y \sin\vartheta\cos\vartheta > 0 & , & e_y \sin\vartheta < 0 & \Rightarrow & \text{riduco } v \\ e_x e_y \sin\vartheta\cos\vartheta < 0 & , & e_x \sin\vartheta > 0 & \Rightarrow & \text{riduco } \omega \\ e_x e_y \sin\vartheta\cos\vartheta < 0 & , & e_x \sin\vartheta < 0 & \Rightarrow & \text{aumento } \omega \end{cases}$$

Rimane da quantificare l'entità del contributo del regolatore rispetto al segnale di feed-forward. Le soluzioni plausibili sono molte, tuttavia bisogna considerare i seguenti aspetti:

- il contributo del regolatore dovrebbe aumentare all'aumentare dell'errore, sia per quanto riguarda un errore su v, che su ω
- è indispensabile introdurre dei fattori di scala k_v e k_ω sui contributi di v e ω, per poter tarare opportunamente il regolatore

La prima idea è stata quella di implementare quindi un regolatore del tipo:

$$\begin{cases} e_x e_y \sin\vartheta\cos\vartheta > 0 & \Rightarrow & \begin{cases} v_{FB} = sign(e_y \sin\vartheta) k_v \sqrt{e_x^2 + e_y^2} \\ \omega_{FB} = 0 \end{cases} \\ e_x e_y \sin\vartheta\cos\vartheta < 0 & \Rightarrow & \begin{cases} \omega_{FB} = -sign(e_x \sin\vartheta) k_\omega \sqrt{e_x^2 + e_y^2} \\ v_{FB} = 0 \end{cases} \end{cases}$$

I risultati sperimentali, tuttavia, hanno mostrato che, con questo tipo di regolatore, o k_ω ha un valore troppo basso (cioè di fatto il contributo su ω è insufficiente), oppure ha un valore troppo alto (cioè si instaurano delle rotazioni troppo brusche all'aumentare

dell'errore $\sqrt{e_x^2 + e_y^2}$). L'idea quindi è stata quella di cercare di diminuire l'effetto di tale errore su ω_{FB}, in modo che valori, anche elevati, di $\sqrt{e_x^2 + e_y^2}$, non provocassero variazioni troppo accentuate di ω_{FB}. La soluzione finale adottata è stata allora la seguente:

$$\omega_{FB} = -sign(e_x \sin\vartheta) k_\omega \sqrt[4]{e_x^2 + e_y^2}$$

4.2.3 Algoritmo di controllo con solo feed-back

Oltre a quelli precedenti, è stato implementato e testato anche un altro algoritmo per il controllo in retroazione, il quale si differenzia dai primi per il fatto che non prevede, in parallelo, un contributo di feed-forward. L'idea di base è molto semplice: si consideri l'algoritmo utilizzato per generare il feed-forward, che calcola le velocità $v_d(k)$ e $\omega_d(k)$ per portare il robot dallo stato $q_d(k) = \begin{bmatrix} x_d(k) \\ y_d(k) \\ \vartheta_d(k) \end{bmatrix}$ allo stato $q_d(k+1) = \begin{bmatrix} x_d(k+1) \\ y_d(k+1) \\ \vartheta_d(k+1) \end{bmatrix}$. Se, ad ogni step, si sostituisce al posto dello stato desiderato $q_d(k)$ lo stato reale $q(k) = \begin{bmatrix} x(k) \\ y(k) \\ \vartheta(k) \end{bmatrix}$, si ottengono dei diversi valori di $v_d(k)$ e $\omega_d(k)$ che, teoricamente, portano il robot dallo stato attuale $q(k)$ allo stato $q_d(k+1)$ desiderato all'istante successivo.
Le prestazioni ottenute con questo controllore sono state però modeste: i due svantaggi principali di questo tipo di soluzione sono

- l'assenza di feed-forward, e quindi di tutti i vantaggi che esso implica
- l'impossibilità di tarare l'algoritmo

5 Implementazione e risultati

5.1 Descrizione dell'algoritmo

In questa sezione viene descritto il funzionamento dell'algoritmo implementato, e vengono presentati i risultati ottenuti dalle prove effettuate.
All'interno del programma si possono distinguere le seguenti fasi:

- **configurazione web-cam**: l'utente può settare i parametri della web-cam tramite un'interfaccia grafica. Successivamente il robot viene posizionato sui quattro angoli del piano di lavoro, che vengono acquisiti in modo tale da poter definire la matrice di proiezione prospettica, utilizzata per eliminare l'errore di prospettiva introdotto dal sistema di visione.

- **inizializzazione robot**: viene creata la connessione tra PC e Khepera via cavo seriale.

- **definizione della traiettoria**: l'utente, tramite una comoda interfaccia grafica, può specificare la traiettoria che il veicolo deve seguire, attraverso una serie di punti selezionati con il mouse, successivamente interpolati da una *spline*. Il punto iniziale è già fornito dal programma in corrispondenza della posizione del robot sul piano.

- **generazione del segnale di feed-forward**: basandosi sulla traiettoria specificata dall'utente, il programma calcola, off-line, il segnale di comando per ottenere il suo inseguimento, facendo riferimento al modello del robot.

- **ciclo principale**: come si vede dalla figura (26), a questo punto il programma esegue ciclicamente, fino al completamento della traiettoria, le seguenti operazioni.
  Per prima cosa viene acquisita l'immagine dell'area di lavoro, processata per ottenere la posizione del robot. Quest'ultima, assieme alla traiettoria di riferimento, viene elaborata dall'algoritmo di controllo, che fornisce il segnale di retroazione da inviare al robot in aggiunta al segnale di feed-forward, calcolato precedentemente (off-line). Prima di inviare al robot il nuovo ingresso, si attende la fine del periodo di campionamento precedente, in modo da garantire un tempo di ciclo costante. In questa fase viene anche fornita all'algoritmo di visione una stima della posizione futura del robot, utilizzata per centrare la zona di ricerca dei punti da acquisire e limitarla.

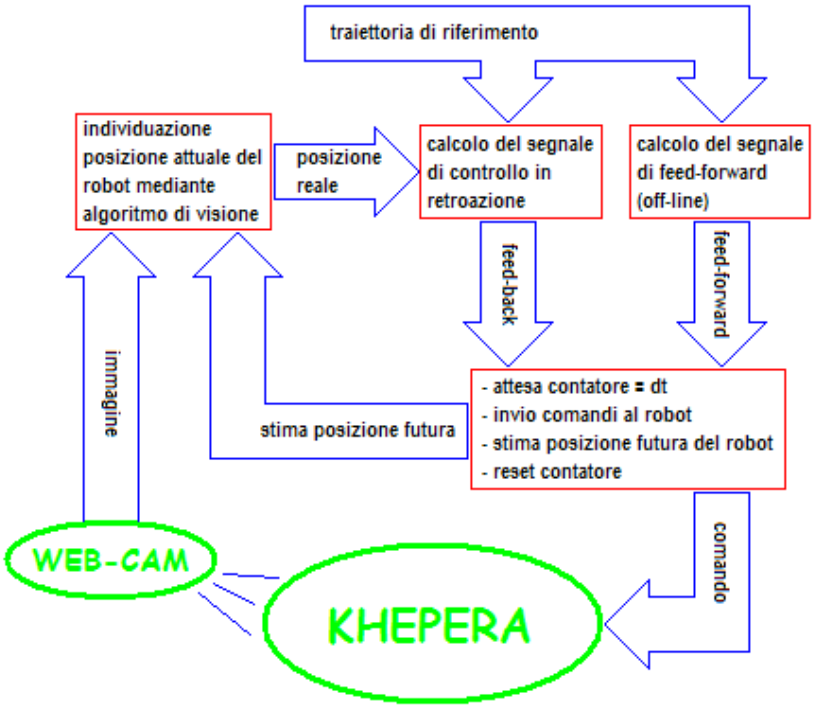


Fig. 26: Schema del programma principale

Durante il movimento del robot, è possibile seguire la traiettoria seguita dallo stesso direttamente dal monitor del PC: viene infatti eseguita e registrata un'animazione grafica che disegna il veicolo istante per istante, insieme alla traiettoria voluta e quella effettiva (si veda la figura 27).

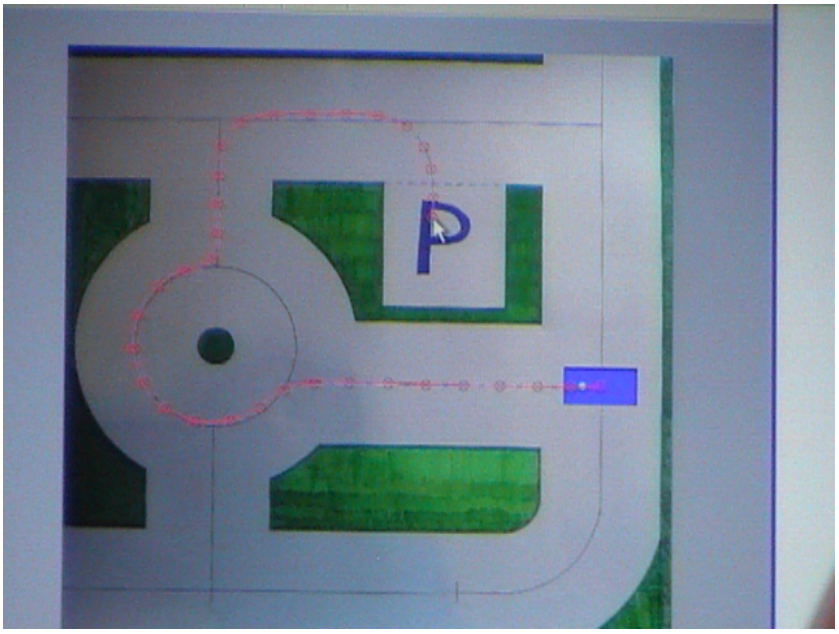


Fig. 27: Illustrazione dell'interfaccia grafica

5.2 Risultati sperimentali

La figura (28) riporta un esempio dei risultati tipicamente ottenuti in fase sperimentale.

É evidente che il solo contributo del feed-forward non è sufficiente a far seguire correttamente al veicolo la traiettoria desiderata. Ciò dipende in massima parte dal fatto che la superficie utilizzata come piano di lavoro causava uno slittamento eccessivo delle ruote del robot, che non si presentava su altri tipi di superfici, ad esempio sul legno.
L'introduzione del feed-back, come si può vedere, risolve questo problema, portando il veicolo ad un inseguimento decisamente migliore.
Rappresentare mediante figure le prestazioni del controllore implementato risulta piuttosto complesso: sono stati realizzati quindi una serie di video che permettono di apprezzare immediatamente i risultati ottenuti.

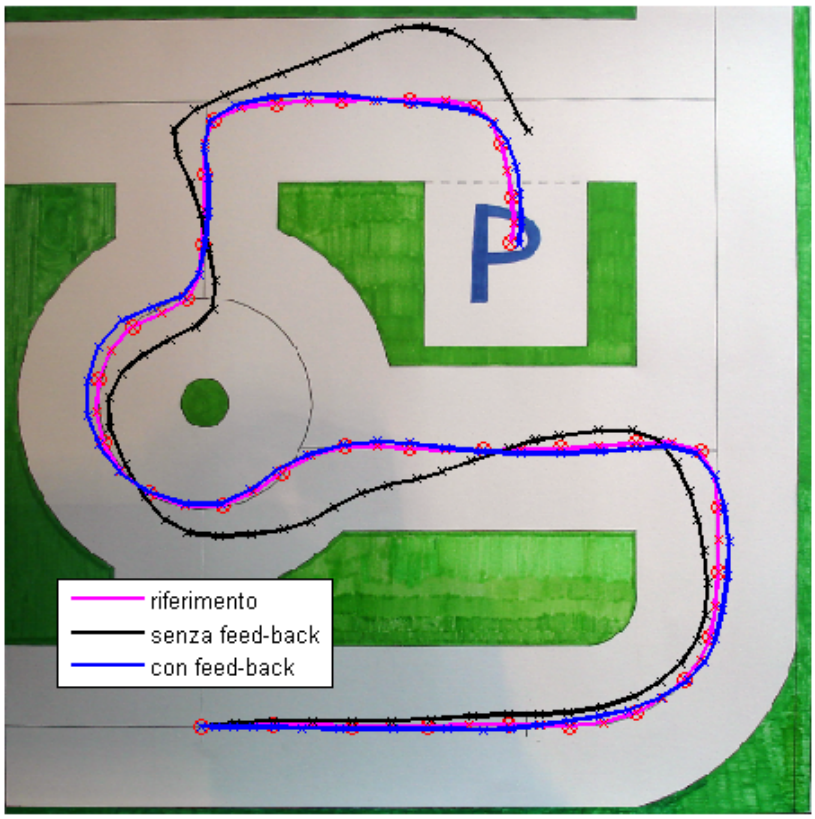


Fig. 28: Risultati sperimentali

6 Conclusioni

Con la presente esperienza si è realizzato un sistema di controllo che, pur facendo uso di un apparato hardware molto semplice, permette il conseguimento di risultati piuttosto soddisfacenti.

Sicuramente i miglioramenti possibili sono notevoli, soprattutto per quanto riguarda l'ottimizzazione del codice relativo all'acquisizione video, al fine di contenere il tempo di calcolo complessivo. Si è visto infatti che la parte dell'algoritmo relativa al tracking del robot impegna le risorse di calcolo per almeno l'80% del tempo di ciclo totale. Una possibile soluzione a questo problema consiste sicuramente nell'implementazione dell'algoritmo in un linguaggio di programmazione di più basso livello, ad esempio il *C*, al posto del linguaggio `MATLAB`®, scelto per la comodità di utilizzo e la rapidità di sviluppo che esso offre.
La riduzione del tempo di ciclo permetterebbe inoltre di implementare gli algoritmi presentati nelle sezioni precedenti, non realizzati proprio per mantenere il tempo di reazione del sistema entro limiti accettabili.

I possibili sviluppi futuri possono essere molteplici, tra i quali:

- estensione dello spazio operativo
- introduzione di tutte le problematiche relative allo spostamento in una città reale (semafori, precedenze, pedoni, ecc.)
- tracking e controllo di un numero maggiore di veicoli, eventualmente interagenti tra di loro
- analisi e risoluzione delle problematiche relative a *collision detection* e *collision avoidance*
- generazione automatica di traiettorie (*path planning*)
- integrazioni con sistemi `GPS`

Riferimenti bibliografici

[1] Giuseppe Oriolo, Alessandro De Luca, Marilena Vendittelli: *WMR Control via Dynamic Feedback Linearization: design, implementation, and experimental validation*, IEEE Transactions on control systems technology, vol.10, n. 6 (11/2002)

[2] Warren E.Dixon, Darren M. Dawson, Erkan Zergeroglu, Aman Behal: *Adaptive Tracking Control of a WMR via an Uncalibrated Camera System*, IEEE Transactions on systems, man and cybernetics, vol.31, n. 3 (06/2001)

[3] Master Shepherds (Burelli Luca, Caruso Giuseppe, Diquigiovanni Simone, Rossetti Andrea): *Simulazione di Robopastorizia*, progetto per il corso di 'Controllo dei processi', prof. Frezza, università degli studi di Padova (a.a. 2002/2003)

[4] Alessandro De Luca, Giuseppe Oriolo, Marilena Vendittelli: *Control of Wheeled Mobile Robots: An Experimental Overview*, progetto svolto per il Dipartimento di Informatica e Sistemistica, Università degli Studi di Roma "La Sapienza"

[5] Martin Seyr, Stefan Jakubek: *Mobile Robot Preditive Trajectory Tracking*, progetto svolto presso la 'Vienna Uniersity of Technology, Institute of Mechanics and Mechatronics

[6] Xianhua Jiang, Yuichi Motai, Xingquan Zhu: *Predictive fuzzy logic controller for trajectory tracking of a mobile robot*, IEEE Mid-Summer workshop on Soft Computing in Industrial Applications, Helsinki university of technology, Finland (05/2005)

[7] GianLuca Mariottini: *Sensori per la Visione*, SIRS Lab - Università degli studi di Siena, (01-02/2005)

[8] A.K. Jain, M.N. Murty, P.J. Flynn: *Data Clustering: a review*, ACM computing surveys, vol.31, n.3 (09/1999)

[9] Giuseppe Oriolo: *Control of Nonholonomic Systems*, Dottorato di ricerca in Ingegneria dei Sistemi, DIS, Università di Roma 'La Sapienza'

[10] Zoltàn Zoller, Peter Zentay: *Constant Kinetic Energy Robot Trajectory Planning*, Periodica Polytechnica Ser. Mech. Eng. Vol 43, N.2 (1999)

[11] Steven M. LaValle: *Planning Algorithms*, Cambridge University Press (2006)

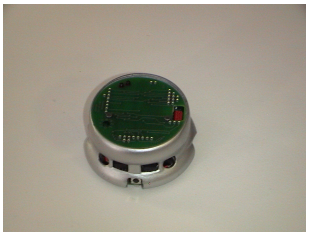
[12] Richard M. Murray, S. Shankar Sastry: *Nonholomic motion planning: steering using sinusoids*

[13] A.A.V.V.: *Robot Motion Planning and Control*, ed. Jean-Paul Laumond, Laboratoire d'Analye ed d'Architecture des Systèmes, Centre National de la Recherche Scientifique

Marco Caregaro Negrin

Andrea Nicolettis

Paolo Erasmo Pucci



Mr. Khepera

Mrs. Web-cam