UNIVERSITÀ DI BRESCIA
FACOLTÀ DI INGEGNERIA
Dipartimento di Elettronica per l'Automazione

Laboratorio di Robotica Avanzata
Advanced Robotics Laboratory

Corso di Robotica Mobile
(Prof. Riccardo Cassinis)

Speedy Docking Marker Detection

Elaborato di esame di:

Claudio Gandelli, Gianpiero Preosti, Giuseppe Turelli

Consegnato il: **08 aprile 2009**

Sommario

Il software proposto in questo lavoro è stato realizzato per effettuare il riconoscimento di un pattern di LED, di seguito definito "marker", all'interno di immagini ambientali acquisite tramite telecamera. Il software determina posizione e dimensione del marker all'interno dell'immagine ottenuta dalla telecamera, con lo scopo di stimare la posizione relativa della telecamera rispetto al marker. Ad ogni iterazione comunica le coordinate elaborate ad un processo esterno.

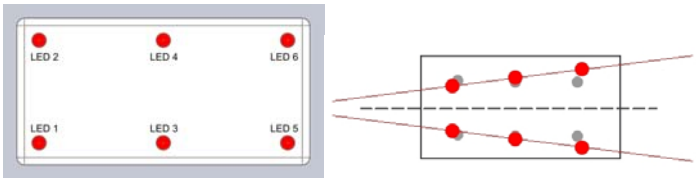
1. Introduzione

In questo elaborato abbiamo trattato le procedure di autolocalizzazione di un robot all'interno del suo ambiente di movimento. Gli elementi che compongono il sistema di autolocalizzazione sono:

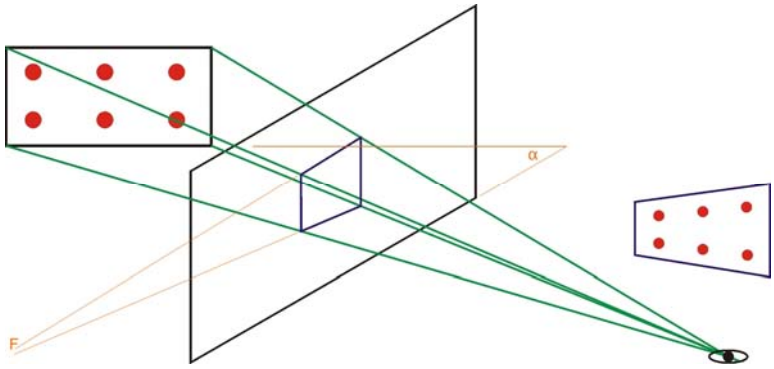
- il landmark: un marker attivo composto di sei LED rossi ad alta luminosità, disposti a comporre un pattern regolare sulla superficie di un box di materiale plastico, posizionato in corrispondenza della stazione di ricarica.
- La telecamera: è una webcam con risoluzione di 640x480 pixel, montata a bordo del robot.
- L'unità di calcolo: è sul robot stesso, nel nostro caso trattasi di un calcolatore portatile montato sul dorso di Speedy

1.1. Idea per l'autolocalizzazione

Il progetto è nato con l'idea di sfruttare la distorsione trapezoidale dell'immagine del marker di forma nota, ottenuta dalla webcam, per risalire alla posizione da cui è stato registrato lo scatto.



La deformazione trapezoidale è quel fenomeno di deformazione dell'immagine che si verifica quando il piano dell'immagine non è parallelo al piano dell'oggetto (la superficie frontale del marker) ma forma con esso un angolo α, mantenendo invece l'asse verticale dell'immagine parallela a quella dell'oggetto ripreso. L'immagine risulta così deformata prospetticamente e le linee orizzontali, originariamente parallele, nell'immagine acquisita convergono verso un punto denominato fuoco, gli oggetti più vicini risultano più grandi mentre quelli più lontani dal punto di osservazione rimpiccioliscono. Le linee verticali, al contrario, rimangono tali.



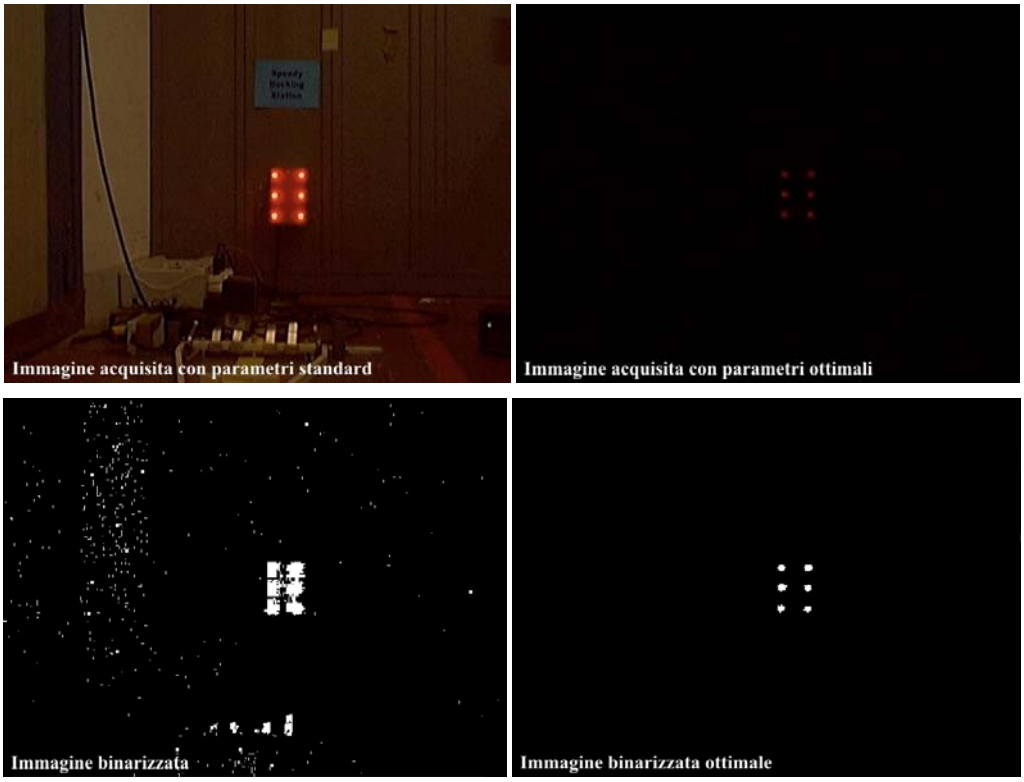
Nel nostro caso il robot si muove solo sul piano del pavimento e la telecamera si trova a un'altezza fissa da terra che coincide col centro del landmark. Da qualsiasi posizione del robot inquadrando il marker questo subirà la distorsione trapezoidale e potranno variare di volta in volta le lunghezze delle verticali che uniscono i LED corrispondenti sulle due file.

2. Il problema affrontato

L'implementazione del software secondo le idee esposte nel capitolo precedente si è scontrata con alcune limitazioni tecnologiche, dovute principalmente alla strumentazione a disposizione e alla necessità di contenimento dei costi per la messa in opera del sistema. Di seguito verranno trattati nello specifico gli aspetti problematici incontrati durante il lavoro.

2.1. Binarizzazione e riconoscimento dei Blob

Il primo problema affrontato riguarda la qualità delle immagini ottenute dalla webcam. Ecco alcuni esempi degli scatti che avrebbero dovuto permettere il riconoscimento del landmark.



2.2. Localizzazione

A causa della bassa risoluzione della telecamera la deformazione prospettica del landmark è assolutamente poco rilevante e non permette di determinare la posizione del robot.

Non è nemmeno possibile determinare se il robot si trovi alla destra o alla sinistra del marker in quanto lo scostamento verticale dei LED che compongono il marker non varia per un angolo frontale piuttosto ampio.

2.3. Comunicazione inter-processo

Questo lavoro si occupa esclusivamente dell'acquisizione di dati dall'ambiente circostante: essi vanno ulteriormente elaborati per poter poi prendere delle decisioni adeguate. Non è quindi pensabile prescindere dall'introduzione di un meccanismo per consentire la comunicazione di questo software con un processo esterno.

L'esigenza di mettere in comunicazione differenti processi è un problema assai comune sin dagli albori nel mondo dell'informatica, questo si traduce nella disponibilità di un discreto numero di meccanismi che permettono lo scambio di dati tra processi, tutti ampiamente collaudati.

Per poter individuare il metodo di comunicazione più adatto si devono vagliare attentamente tutte le esigenze di scambio dati con l'esterno. Prima di tutto è necessario specificare che il programma acquisisce in continuazione informazioni dall'ambiente e produce i propri risultati ad ogni acquisizione. Per questo motivo è necessario introdurre un flag che permetta al processo lettore di segnalare l'avvenuta lettura dei dati e al nostro software di indicarne, al contrario, la scrittura di nuovi.

Scendendo nel particolare, il software necessita dei seguenti input:

- segnale ad hoc di terminazione (non vengono utilizzati quelli forniti dal SO) che viene inviato dal programma "lettore", implementato e come una variabile booleana condivisa

E produce i seguenti output:

- le coordinate cartesiane (vedi Appendice A) dei 6 LED che compongono il Marker, rappresentate da 12 interi
- un segnalatore di "freschezza" dei dati, anche questo implementato come una variabile condivisa booleana

Un'altra caratteristica importante per la scelta della tecnologia da utilizzare è che la comunicazione deve avvenire solamente con processi residenti sulla medesima macchina (non è, quindi, strettamente necessario utilizzare un "socket TCP/IP"). Infine è stato scelto di sovrascrivere i dati qualora ne fossero disponibili di nuovi (i dati vecchi non sono più utili).

3. La soluzione adottata

In questo capitolo analizziamo le soluzioni alternative che abbiamo adottato per far fronte alla mancata efficacia del metodo basato su distorsione trapezoidale e ai problemi legati principalmente alle tecnologie in uso per la realizzazione di questo sistema.

3.1. Binarizzazione e riconoscimento dei Blob

La correzione manuale dei parametri di esposizione e guadagno della webcam ha permesso un filtraggio ottimale, nelle condizioni del laboratorio di robotica, del rumore indesiderato, e il processo di binarizzazione è calibrato in modo da scartare blob troppo piccoli, certamente estranei al landmark.

Abbiamo verificato che la distanza tra due LED allineati in verticale fosse inversamente proporzionale alla distanza dell'obiettivo della webcam dal marker, e constatato che è possibile usare questo indicatore per determinare tale distanza.

La discriminazione tra Destra e Sinistra però non può risolversi solo attraverso la misurazione dei pixel che separano i LED.

3.2. Localizzazione

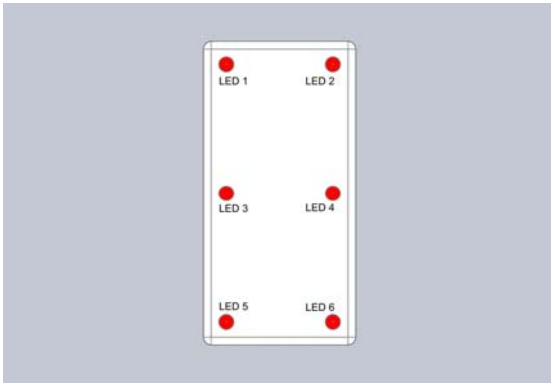
La mancata efficacia della misura di deformazione geometrica del marker ci ha indotto a ideare alcune strategie alternative per la localizzazione del robot, mantenendo gli stessi elementi per non incidere su tecnologie e costi del sistema. La soluzione finale ha curiosamente preso spunto proprio dal problema

che ha portato ad abbandonare l'idea originaria di utilizzare la tecnica della deformazione prospettica. Il problema, lo ricordiamo, è dato dal fatto che anche per grandi variazioni dell'angolo con cui si guarda il marker non si hanno variazioni sensibili delle posizioni dei baricentri dei blob relativi ai led del marker. Dato che proprio queste variazioni sono di entità trascurabile, si possono considerare i blob come allineati sia in verticale sia in orizzontale aggiungendo una piccola tolleranza.

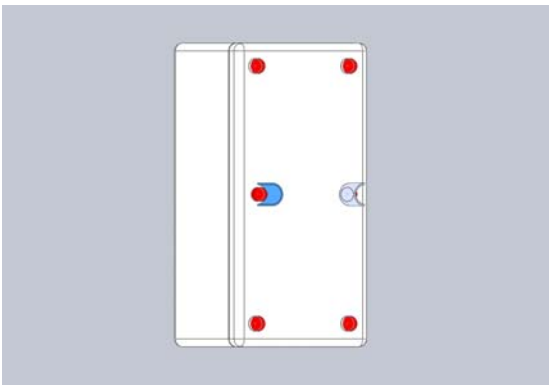
In primo luogo una possibile soluzione era stata identificata nel rendere le due sorgenti luminose centrali del marker fonti luminose direzionali, incassando i due LED nel box del marker e sovrapponendo due tubicini di plastica ai diodi luminosi. In questo modo le due luci centrali sarebbero state visibili alla videocamera solo se il robot si fosse trovato in un corridoio frontale al marker della larghezza di un angolo adeguata a permettere il suo approccio alla stazione di ricarica. Il problema fondamentale di questa soluzione deriva dall'impossibilità di discriminare se il robot, osservando solo i quattro led angolari, si trovi alla destra o alla sinistra del landmark e, di conseguenza, rende la decisione del movimento da compiere impossibile.

La necessità di discriminare il semipiano in cui si trovi il robot rispetto al marker ha portato in primo luogo all'idea di non incassare i LED centrali, ma lasciarli affiorare come gli altri dalla superficie, apponendo due barriere fisiche opache sporgenti per rendere LED diversi visibili da semipiani diversi, mentre lasciarli entrambi visibili quando il robot fosse stato allineato con la docking station. Nel modo inizialmente ideato però le barriere opache andavano ad interferire anche con la visibilità dei diodi angolari rendendo il processo di identificazione del marker assai poco preciso.

L'ultima soluzione adottata prevede la rotazione di 90° del marker in modo da ottenere due colonne di tre LED ciascuna.



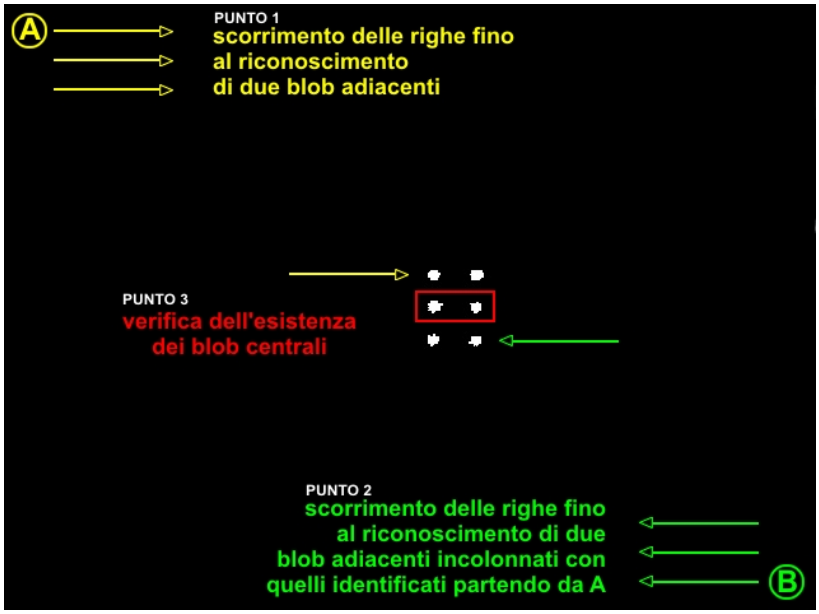
In questo modo la distanza verticale tra i led agli angoli è quasi raddoppiata rispetto a quella che si misurava col marker in orizzontale, inoltre l'oscuramento dei LED mediani tramite barriere opache ortogonali alla superficie del marker o tramite incassamento delle sorgenti luminose non ostruisce la visibilità delle sorgenti luminose rimanenti dalle diverse angolazioni di ripresa del robot, come dimostrano i modelli realizzati con un software di disegno tridimensionale impostando il punto d'osservazione per il rendering nelle posizioni raggiungibili dalla webcam di Speedy.



Vengono di seguito esplicitate le principali fasi dell'algoritmo di acquisizione compiute ad ogni iterazione:

1. ricerca, partendo dalla posizione A nell'immagine di riferimento, di due blob aventi baricentro sulla stessa riga;
2. trovati i blob al punto 1, vengono ricercati altri due blob, partendo dalla posizione B dell'immagine di riferimento, aventi fra loro baricentro nella stessa riga e baricentro nella stessa colonna degli omologhi di cui al punto 1;
3. completati i punti 1 e 2, vengono ricercati altri due blob, se presenti, che abbiano baricentro nella stessa colonna degli omologhi di cui al punto 1 e baricentro nella riga a metà tra i blob riconosciuti al punto 1 e quelli riconosciuti al punto 2;
4. i dati così raccolti vengono scritti nella memoria condivisa e viene settato il flag di freschezza a 1.

Avendo le posizioni dei blob è possibile sapere a quale distanza si trova il robot secondo una legge nota (vedi Appendice B) e se esso si trovi sulla sinistra oppure sulla destra rispetto al marker.



3.3. Comunicazione inter-processo

Tra le svariate tecnologie a disposizione, in base alle caratteristiche del software elencate al punto 2.3, sono state approfondite il "named pipe", il "socket TCP/IP", il "shared memory".

Lasciando i dettagli tecnici di ciascuno all'interesse del lettore (vedi bibliografia) verranno solamente esposte le motivazioni che hanno portato alla scelta di un meccanismo di "shared memory" protetto mediante un semaforo.

Il motivo per cui si è deciso di non utilizzare il "socket TCP/IP" è l'eccessivo overhead di comunicazione che questo avrebbe portato. Similmente la decisione di scartare il "named pipe" è stata dettata dalla scelta progettuale di sovrascrivere i dati vecchi (una volta prodotti nuovi dati si dovrebbe togliere dal "named pipe" quelli vecchi).

La soluzione più adatta è stata invidivuata nel "shared memory" protetto con un semaforo: questa soluzione permette, infatti, sia di sovrascrivere comodamente il contenuto della memoria condivisa sia di ridurre, rispetto al "socket TCP/IP", l'occupazione di risorse. Per l'intera comunicazione è stato sufficiente utilizzare un vettore di 14 interi dove le prime 12 posizioni contengono le coordinate presunte del Marker, la posizione 13 contiene il valore booleano di freschezza e la posizione 14 l'indicatore, sempre booleano, di terminazione. Entrambi i processi devono avere accesso sia in lettura sia in scrittura alla memoria condivisa, questo per permettere al lettore di segnalare l'avvenuto consumo dei dati ponendo a false la posizione 13 dell'array di comunicazione. Naturalmente, prima di poter effettuare alcuna operazione sulla memoria condivisa, è necessario ottenere il "lock" sul semaforo. Per permettere di testare adeguatamente il meccanismo è stato realizzato un semplice programma di lettura

dati che implementa il suddetto meccanismo di comunicazione e stampa i dati a console. E' compilabile sia in C che in C++ ed è quindi attingendo dal sorgente di quest'ultimo che dovrebbe essere sviluppato altro software che necessiti di comunicare con quello qui presentato.

3.3.1. Struttura della memoria condivisa

La memoria condivisa è un array di interi, contenente 14 valori.

ARRAY[0], ARRAY[1]: coordinate X e Y del LED 1.

ARRAY[2], ARRAY[3]: coordinate X e Y del LED 2.

ARRAY[4], ARRAY[5]: coordinate X e Y del LED 3.

ARRAY[6], ARRAY[7]: coordinate X e Y del LED 4.

ARRAY[8], ARRAY[9]: coordinate X e Y del LED 5.

ARRAY[10], ARRAY[11]: coordinate X e Y del LED 6.

ARRAY[12]: flag di freschezza dei dati.

ARRAY[13]: flag di terminazione del programma.

4. Modalità operative

4.1. Componenti necessari

Per la compilazione del software sono necessarie le librerie VisLib e il compilatore gcc, l'eseguibile prodotto invece sarà stand-alone e potrà essere eseguito da qualsiasi percorso sul filesystem locale.

4.2. Modalità di installazione

Il programma necessita delle librerie Vislib, che dovrebbero essere installate in "$(HOME)/vislib/". Una volta compilato tramite "make speedydock" l'esecuzione di "make install" copierà il programma in "usr/local/bin".

4.3. Modalità di taratura

In primo luogo è necessario effettuare l'allineamento della telecamera rispetto al centro del maker. La telecamera è fissata sul robot in modo che l'obiettivo si trovi ad un'altezza corrispondente a quella del centro del marker e lo stesso deve essere il più possibile allineato in modo che il sistema di riferimento della telecamera coincida con quello del marker.

Una volta allineata la webcam il sistema può esser tarato attraverso una serie di parametri che dipendono dall'ambiente in cui è operativo il robot, quali l'illuminazione ambientale, la distanza della stazione di carica dal marker, la sensibilità della videocamera.

TOL: è il valore, misurato in pixel, della massima differenza di coordinate per dichiarare due pixel allineati, sia in orizzontale che in verticale.

X_MIN: è il valore minimo dell'ordinata che può assumere un blob appartenente al marker. È stato definito empiricamente dall'immagine del marker scattata dalla posizione più vicina possibile che il robot può assumere.

X_MAX: è il valore massimo dell'ordinata che può assumere un blob appartenente al marker. È stato definito empiricamente dall'immagine del marker scattata dalla posizione più vicina possibile che il robot può assumere.

AREA_MIN: indica, in pixel, la minima area di un blob perchè questo sia preso in considerazione nel riconoscimento del pattern del marker.

BIN_TH: è la soglia per l'algoritmo di binarizzazione, indica il livello di discriminazione tra sorgente luminosa e sfondo.

SET_SHUTTER: è il valore di velocità dell'otturatore.

SET_GAIN: è il valore di guadagno della videocamera.

L'esperienza porta a consigliare che nelle operazioni di taratura è opportuno modificare solamente la velocità dell'otturatore e il valore soglia per la binarizzazione. Nel caso in cui il software fallisca nel riconoscere uno o più led è opportuno abbassare la soglia di binarizzazione e, nel caso non funzioni ancora, diminuire la velocità dell'otturatore prima di abbassare ulteriormente la soglia; nel caso contrario (viene visualizzata la scritta "too many blobs") si deve agire alzando la soglia di binarizzazione oppure aumentando la velocità dell'otturatore.

4.4. Problemi noti e suggerimenti

Il programma è stato realizzato per il sistema di orientamento del robot mobile Speedy, che dispone di una telecamera a bassa risoluzione e di un marker con sei LED rossi ad alta luminosità. I parametri di binarizzazione sono sensibili alle variazioni della luminosità ambientale e nel nostro caso sono stati ottimizzati per l'ambiente di esercizio attuale. Per ambienti con repentini cambiamenti di luminosità globale il sistema potrebbe subire malfunzionamenti dovuti alla difficoltà di binarizzazione dell'immagine e conseguente mancato riconoscimento del marker. A causa di possibili variazioni di luce interne al laboratorio, dovute all'irraggiamento solare, le prestazioni del software possono essere influenzate negativamente, si consiglia, nel caso si notassero malfunzionamenti, di ripetere la taratura del sistema.

5. Conclusioni e sviluppi futuri

Il sistema realizzato in questo lavoro, in seguito alle modifiche apportate all'idea iniziale per risolvere i problemi sorti in corso d'opera, si è concluso con l'automatizzazione del processo di acquisizione dei dati sulla posizione del marker mentre il robot è in movimento, questi dati vengono trasmessi a mezzo memoria condivisa ad un processo esterno che si occuperà delle azioni di guida del robot in risposta ai dati di posizione. Eventuali sviluppi di questo lavoro possono consistere nell'irrobustimento della rilevazione a variazioni ambientali di luminosità, per aumentare l'efficacia della rilevazione in ambienti più rumorosi di quello di esercizio attuale, o all'affinamento del marker sfruttando ancora il principio di apporre barriere fisiche alla diffusione della luce, in modo che si verifichi la variazione di alcuni parametri del marker a seconda dell'angolo da cui questo viene osservato (per esempio adottando sorgenti luminose non puntiformi interne al box e fessure per la loro osservazione).

6. Appendice A

6.1. Sistema di riferimento della telecamera

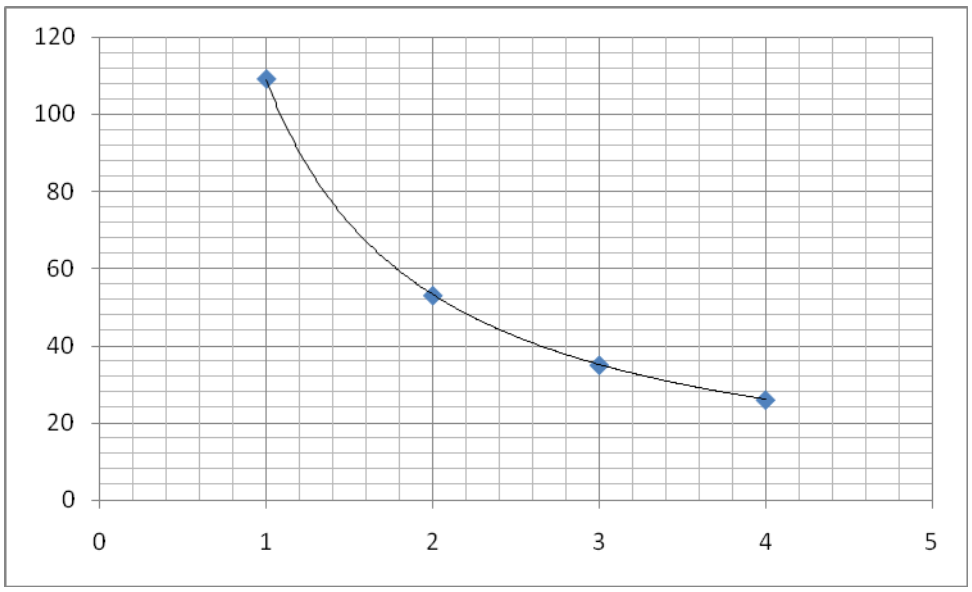
A seguito della rotazione di 90° del marker, che ha permesso di ottenere i vantaggi decisivi per l'autolocalizzazione del robot, il sistema di riferimento delle immagini è stato ruotato anch'esso di 90°, consentendoci di mantenere invariato l'algoritmo di matching per il riconoscimento del marker e di effettuare tutte le prove del caso con il marker orientato nei due modi testati. L'attuale sistema di coordinate è orientato come mostrato nella figura seguente e originato nell'angolo superiore sinistro dell'immagine.

7. Appendice B

7.1. Legame tra dimensione del marker e posizione del robot

La legge che lega la dimensione rilevata del marker e la distanza del robot dalla sorgente luminosa è una relazione di proporzionalità inversa per la quale la distanza *d*, in metri, del robot dal landmark si ottiene moltiplicando l'altezza del marker *h* misurata in pixel per un coefficiente *c* che abbiamo stimato attraverso una serie di misure sperimentali.

d = c/h (c=106)



In figura è rappresentata la relazione tra la distanza del robot dal marker (*x*) e le relative distanze tra i blob superiori e inferiori nelle immagini scattate dalla webcam (*y*).

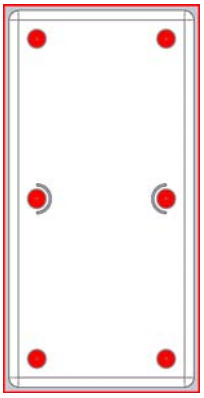
7.2. Test di avvicinamento

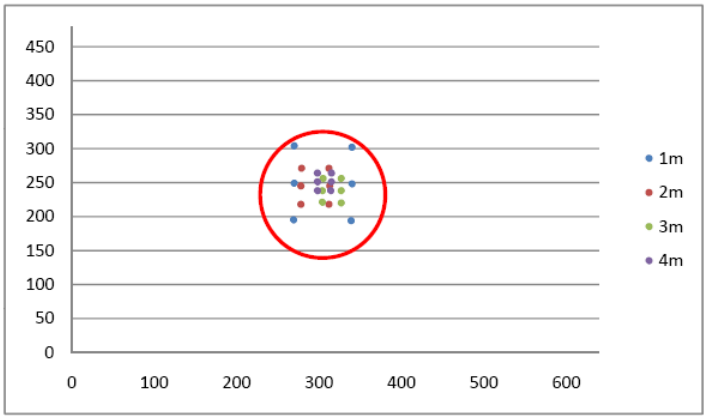
In laboratorio abbiamo effettuato alcune osservazioni dei dati raccolti dal nostro software di elaborazione delle immagini in alcune manovre di parcheggio assistito manualmente. Il robot, da una qualunque posizione di partenza all'interno di quell'angolo in cui sono visibili tutti e sei i LED luminosi, effettuando gli spostamenti necessari a mantenere marker nella zona centrale dell'immagine acquisita, termina la sua corsa con l'ingresso nella stazione di ricarica. Per risultati ottimali e perché l'angolo critico di avvicinamento sia il più ampio possibile consigliamo di ridurre lo spazio attualmente presente tra la sorgente luminosa e la stazione di ricarica.

7.2.1. Approccio frontale

Questa manovra di avvicinamento e posizionamento sulla docking station è stata osservata lasciando che il robot si muovesse in linea retta partendo da una posizione ottimale, ossia frontalmente al marker e già rivolto verso lo stesso. I blob sono stati rilevati nelle posizioni corrette al centro dell'immagine della webcam e la manovra si è conclusa correttamente

metri	dist L1-L5		X	Y	dist L2-L6		X	Y
1m	109	LED 1	195	269	108	LED 2	194	339
		LED 3	249	270		LED 4	248	340
		LED 5	304	270		LED 6	302	340
2m	53	LED 1	218	278	53	LED 2	218	312
		LED 3	245	278		LED 4	245	313
		LED 5	271	279		LED 6	271	312
3m	35	LED 1	221	304	36	LED 2	220	327
		LED 3	238	304		LED 4	238	327
		LED 5	256	305		LED 6	256	327
4m	26	LED 1	238	298	26	LED 2	238	314
		LED 3	251	298		LED 4	251	315
		LED 5	264	298		LED 6	264	315

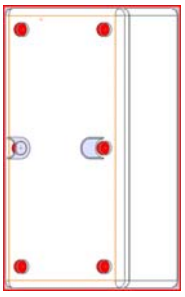


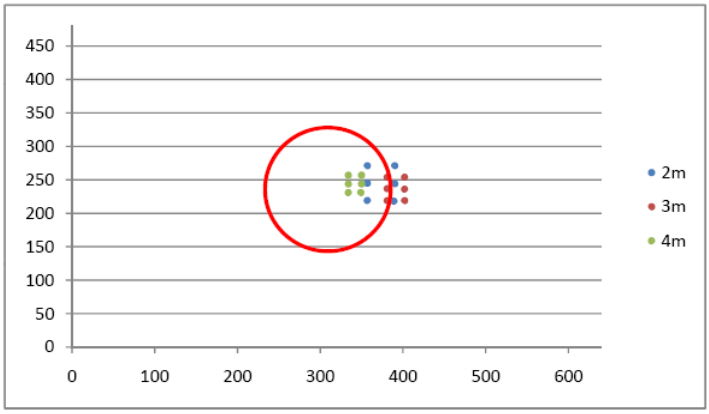


7.2.2. Approccio con angolo critico

In questa seconda manovra l'avvicinamento è avvenuto da un punto sul bordo del cono d'ingresso sulla base, la posizione più esterna in cui i sei LED sono ancora tutti visibili ed il parcheggio è consentito. Le manovre di approccio han dovuto correggere la direzione in relazione alla posizione in cui il marker era rilevato e la procedura si è conclusa con successo.

metri	dist L1-L5		X	Y	dist L2-L6		X	Y
2m	52	LED 1	219	357	53	LED 2	218	389
		LED 3	245	357		LED 4	244	390
		LED 5	271	357		LED 6	271	390
3m	35	LED 1	219	381	35	LED 2	219	402
		LED 3	237	381		LED 4	236	402
		LED 5	254	381		LED 6	254	402
4m	26	LED 1	231	334	26	LED 2	231	349
		LED 3	244	334		LED 4	244	350
		LED 5	257	334		LED 6	257	350




450
400
350
300
250
200
150
100
50
0
0
100
200
300
400
500
600
2m
3m
4m

Bibliografia

[1] Massimiliano Lunelli: “Deformazioni prospettiche”, da *www.matapp.unimib.it/~lunelli/geometria*

[2] Brian "Beej" Hall: “Beej's Guide to Unix IPC”, da *www.ecst.csuchico.edu/~beej/guide/ipc*.

[3] Evi Nemeth, Garth Snyde, Trent R. Hein: "Linux Administration Handbook" 2nd edition, *Prentice Hall.* October 30, 2006.

[4] Clark Atlanta University: “VisLib wiki”, da h*ttp://205.129.163.112/mediawiki*.

Indice