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)

Morguldoc 2: elaborazione di immagini per il riconoscimento di marker attivi

Elaborato di esame di: **Michele Rossi, Andrea Viola**

Consegnato il: **26 luglio 2004**

Sommario

Il progetto ha lo scopo di realizzare un'applicazione che sia in grado di riconoscere, mediante un sistema di visione costituito da una webcam montata su di un robot mobile, due marker artificiali attivi posizionati secondo una particolare configurazione ad una distanza massima di 4m e di calcolare le coordinate (x,y) dei loro due baricentri.

1. Introduzione

Morgul è l'acronimo di "MObile Robot for Guarding University Laboratory", sigla che identifica il progetto relativo al pattugliamento tramite robot mobile di alcune aree sensibili dislocate all'interno della facoltà di ingegneria di Brescia. Morgul è appunto il nome attribuito al robot Pioneer 3 che si farà carico di effettuare il pattugliamento.

Per muoversi in piena autonomia il robot ha montato su di sé un calcolatore portatile che ne gestisce e ne guida i comportamenti sfruttando i programmi appositamente realizzati in questo progetto.

L'operazione di pattugliamento è piuttosto onerosa in termini energetici e questo richiede che al robot sia messa a disposizione una stazione di erogazione nella quale possa eseguire la ricarica delle proprie batterie e di quelle del portatile.

Nasce quindi la necessità di realizzare un sistema che consenta a Morgul di raggiungere autonomamente la propria docking station e di connettersi ai morsetti per la ricarica delle batterie.

Per compiere questa operazione nella maniera corretta il robot deve essere in grado di stimare la propria posizione rispetto a quella della stazione: ciò implica la necessità di ideare una tecnica di visione che fornisca, durante la fase di avvicinamento, il disassamento del robot da quella traiettoria che invece gli garantirebbe di inserirsi nella stazione perfettamente allineato.

La strumentazione del sistema di visione è costituita da una webcam Philips ToUcam Pro II PCVC 840K montata sopra il robot per l'acquisizione delle immagini e connessa al calcolatore portatile per effettuare su queste ultime le opportune elaborazioni.

La posizione rispetto alla docking station sarà valutata in base alla disposizione con cui compariranno nelle immagini acquisite i due marker attivi posizionati sopra di essa. In base alla particolare configurazione adottata per i marker è infatti possibile ricavare una stima alquanto precisa del disassamento e della distanza del robot dalla stazione di ricarica.

Per ottenere le seguenti informazioni questo progetto si pone quindi lo scopo di realizzare un programma capace di:

- acquisire le immagini da webcam nel formato e nella risoluzione voluti;
- elaborare le immagini per individuare con appositi meccanismi di soglia la presenza dei marker attivi (o, in caso negativo, la loro assenza dallo spazio visivo analizzato);
- calcolare, in caso di corretta individuazione dei marker, le coordinate (x,y) dei loro baricentri. Queste verranno poi messe a disposizione quali parametri su cui regolare la traiettoria del robot per l'avvicinamento alla docking station;
- rendere disponibili ad altri programmi le informazioni relative alle coordinate (x,y) dei due baricentri per consentire successive elaborazioni.

2. Il problema affrontato

Il robot mobile Morgul ed il PC portatile montato su di esso utilizzano per il proprio funzionamento delle batterie che necessitano di essere periodicamente ricaricate. Per questo scopo è stato implementato un insieme di behaviour in grado di condurre il robot all'interno della propria stazione di ricarica (docking station). Per consentire un corretto posizionamento all'interno della docking station, si rende necessario un sistema capace di fornire ai behaviour le informazioni relative al disassamento rispetto alla rotta ottimale di approccio e una stima della distanza dalla docking station.

Scopo di questo elaborato è il progetto e la realizzazione di un sistema di visione che fornisca ai behaviour le informazioni necessarie per gestire correttamente l'avvicinamento alla stazione di ricarica.

Per risolvere il problema è necessario realizzare un sistema di visione che permetta di rilevare la posizione del robot rispetto alla docking station, elaborare le informazioni raccolte e memorizzarle in un formato che ne consenta poi l'utilizzo da parte dei behaviour.

Il sistema deve essere in grado di fornire i dati relativi al disassamento con un buon fattore di precisione in quanto il robot, per potersi collegare al dispositivo di ricarica, necessita di raggiungere la docking station mantenendo una rotta prossima a quella ottima. In questo modo è possibile consentire al robot un ingresso agevole nella stazione di ricarica.

3. La soluzione adottata

3.1. Webcam e marker

Poiché il robot Morgul risulta già dotato di una webcam Philips per l'acquisizione delle immagini, abbiamo deciso di utilizzare tale sensore per localizzare la docking station e di conseguenza per ricavare le informazioni per l'avvicinamento. L'utilizzo della webcam ben si presta ai nostri scopi in quanto permette di implementare un sistema di visione a basso costo e di precisione elevata.

Il riconoscimento della posizione rispetto alla docking station viene effettuato individuando uno o più marker opportunamente posizionati nell'ambiente. Il primo passo per la progettazione consiste nel determinare la tipologia di marker più conveniente alle nostre esigenze. La nostra scelta ricade su marker attivi, cioè in grado di emettere energia luminosa verso il sistema di visione. Marker di questo tipo permettono un più facile riconoscimento da parte della webcam sia in ambienti illuminati con luce naturale che con luce artificiale in quanto la loro luminosità, in opportune condizioni, risulta superiore a quella dell'ambiente. Per semplificare il riconoscimento dei marker si ipotizza che questi siano le fonti a più elevata luminosità presenti nello spazio di visione della webcam.

L'unico aspetto negativo nell'utilizzo di marker attivi è che per il loro funzionamento richiedono di essere alimentati: nel nostro caso la fonte di energia è costituita dalla docking station.

La scelta di impiegare due marker attivi è legata alla metodologia di localizzazione che verrà utilizzata per determinare la posizione del robot rispetto alla stazione di ricarica.

3.2. La metodologia di riconoscimento

Questa metodologia richiede la realizzazione di un particolare supporto che consenta di disporre i marker secondo una precisa configurazione tale da permettere una corretta stima del disassamento e della distanza del robot dalla docking station.

I marker sono montati lungo la stessa direttrice ma posizionati ad altezze diverse secondo i seguenti schemi, che riportano, rispettivamente, la vista laterale e frontale del supporto delle lampadine (Figura 1):

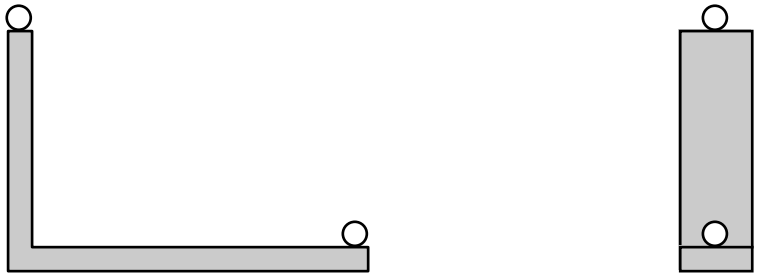

Figura 1

Nel caso in cui il robot sia perfettamente in asse con i due marker notiamo che la telecamera riconosce questi ultimi come due entità luminose posizionate verticalmente una sopra l'altra al centro dello spazio di visione (Figura 2):

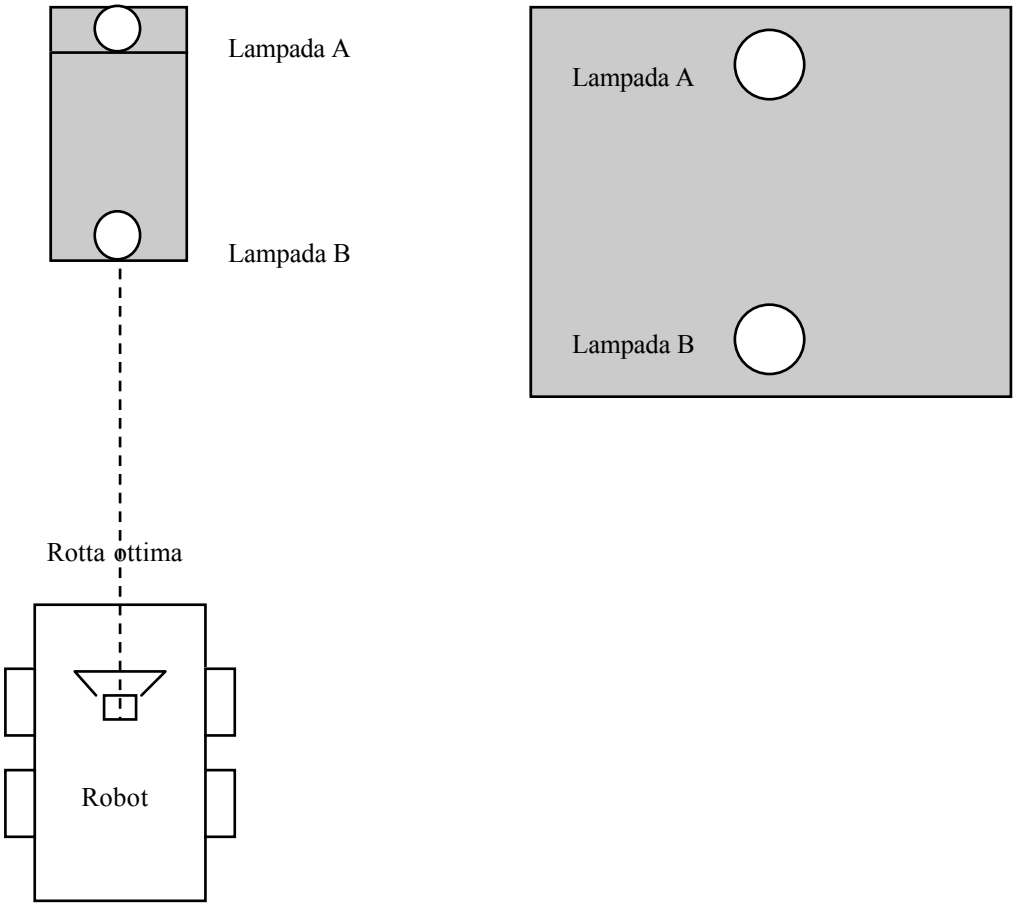

Figura 2

Se il robot si trova spostato a sinistra rispetto all'asse della rotta ottima i due marker saranno identificati da due entità luminose posizionate una sopra l'altra, in cui quella superiore si trova spostata a sinistra e quella inferiore spostata a destra rispetto al centro dello spazio di visione.

La Figura 3 mostra la situazione appena descritta:

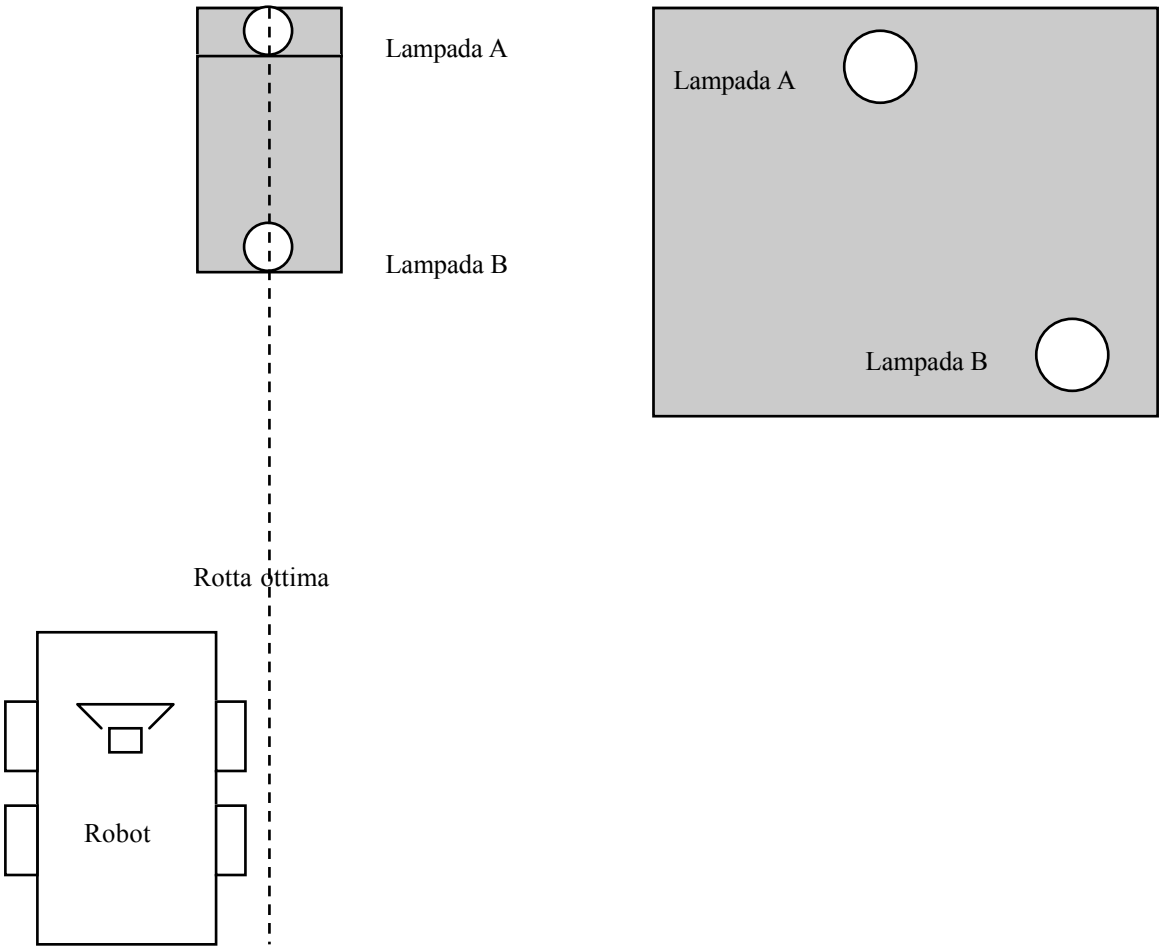

Figura 3

Se il robot si trova invece spostato a destra rispetto all'asse della rotta ottima i due marker saranno identificati da due entità luminose posizionate una sopra l'altra, in cui quella superiore si trova spostata a destra e quella inferiore spostata a sinistra rispetto al centro dello spazio di visione (Figura 4).

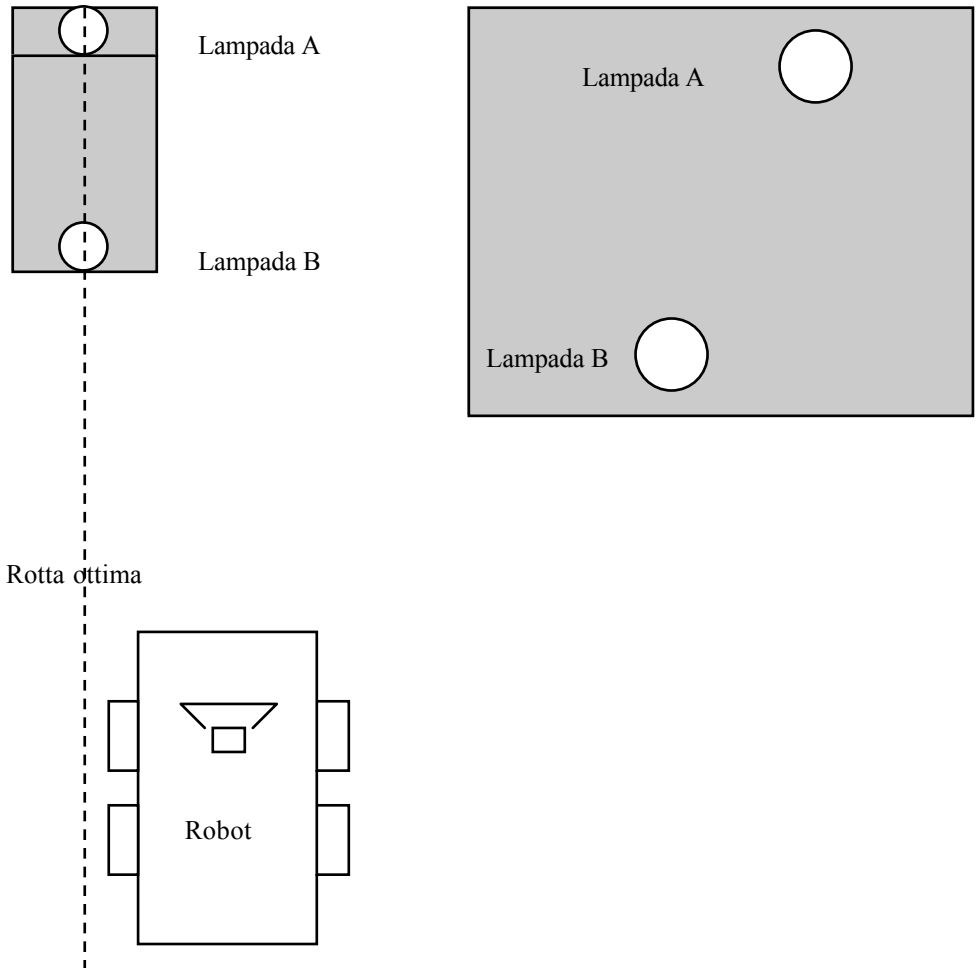

Figura 4

Rilevando la posizione in pixel assunta dai due marker nello spazio di visione sarà possibile determinare poi il disassamento e la distanza reali del robot rispetto alla docking station.

Più precisamente, misurando la distanza orizzontale in pixel tra i due marker è possibile stimare il disassamento con cui il robot si avvicina alla docking station; la distanza in pixel rilevata tra le coordinate verticali dei due marker consente invece di ottenere una stima della distanza a cui il robot si trova rispetto alla stazione di ricarica.

3.3. Specifiche per la realizzazione del supporto marker

Si è deciso di utilizzare come marker attivi due lampadine alogene da 10 W non colorate.

La realizzazione del supporto per i marker richiede che questi siano posizionati a distanze prestabilite e secondo una specifica configurazione per permettere poi all'algoritmo una corretta localizzazione.

In base alle misure sperimentali e alle strutture del robot e della docking station il supporto delle lampadine deve rispettare la seguente disposizione (Figura 5).

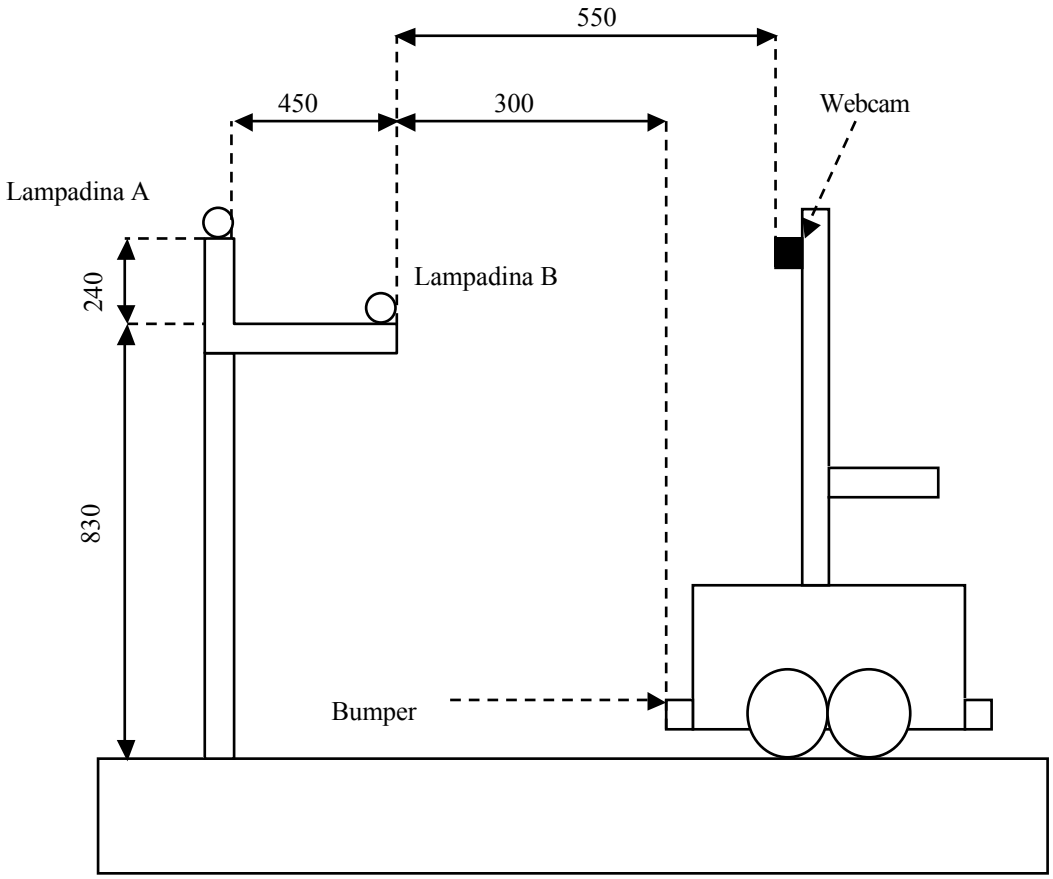

Figura 5

L'altezza da terra dell'asta è stata misurata verificando che entrambi i marker rimangano nello spazio di visione della webcam durante l'intero tragitto che porta il robot dalla distanza massima alla distanza minima consentite: mediante prove sperimentali abbiamo infatti verificato che l'angolo di visione orizzontale della webcam è pari circa a 34°, mentre il suo angolo di visione verticale è di circa 24°.

La distanza tra le lampadine sia sull'asse verticale che sull'asse orizzontale risulta ottimale per il calcolo della posizione dei marker: in questo modo essi risultano sempre ben distinguibili l'uno dall'altro anche quando il robot si trova alla distanza massima consentita dalla docking station.

La distanza minima tra il marker basso ed i bumper anteriori del robot consente di evitare la saturazione dell'immagine da parte della telecamera, problema che potrebbe rendere irriconoscibili o indistinguibili i due marker.

3.4. L'algoritmo per il riconoscimento dei marker

Su Morgul è presente un calcolatore portatile a cui è connessa, mediante porta USB, la webcam Philips per l'acquisizione delle immagini. Poiché il sistema operativo installato sul calcolatore è Linux (distribuzione Mandrake 9.2), l'accesso a device di tipo video viene fornito mediante speciali

file di dispositivo situati nella directory /dev: nel nostro caso la webcam è associata al file di dispositivo video /dev/video0 dal quale è possibile gestire lo stream per l'acquisizione video delle immagini.

Per gestire in modo più agevole e performante l'acquisizione delle immagini dalla webcam si è deciso di realizzare un programma pesantemente basato sul software vgrabbj, applicazione per il grabber d'immagini con tecnologia "Video4Linux" che, oltre ad una buona compatibilità con la maggior parte delle webcam in commercio, mette a disposizione numerose ed utili funzionalità per personalizzare secondo le proprie esigenze la fase di acquisizione: il nome del nuovo programma è blobFinder.

La tipologia delle informazioni riportate da vgrabbj varia a seconda del formato scelto per la rappresentazione delle immagini: in tal senso vgrabbj mette a disposizione la possibilità di utilizzare i formati jpeg, png e ppm.

Per il nostro progetto la scelta è ricaduta sul formato ppm poiché in prospettiva al lavoro di elaborazione che dovrà essere effettuato sull'immagine era quello che offriva la struttura più adeguata per estrapolare semplicemente le informazioni RGB associate ad ogni pixel.

Per quanto riguarda la risoluzione delle immagini, vgrabbj offre la possibilità di utilizzare alcuni degli standard grafici più comuni: per i nostri scopi si adatta bene lo standard VGA, che consente di ottenere immagini di dimensioni pari a 640x480 pixel.

Un altro aspetto da tenere in considerazione è la scelta di ricorrere ad immagini in bianco e nero: poiché l'informazione sul colore non è necessaria per l'individuazione ed il riconoscimento dei pixel relativi ai marker, si è deciso di operare su immagini rappresentate tramite scale di grigio. In questo modo,oltre a risparmiare sulle dimensioni dell'immagine, si semplifica la ricerca dei marker.

Dopo aver acquisito l'immagine nel formato e nella risoluzione voluti è necessario che questa sia fornita come input al nostro algoritmo di individuazione dei marker per calcolare le coordinate dei loro baricentri. Per fare questo abbiamo realizzato il nuovo software blobFinder che contiene il codice relativo all'algoritmo di ricerca dei blob e una versione modificata di vgrabbj: in questo modo l'acquisizione dell'immagine può essere effettuata leggendo direttamente nell'apposito buffer di memoria in cui sono memorizzate le informazioni relative all'immagine (così come viene effettuato da vgrabbj).

Prima di entrare nel dettaglio dell'applicazione per il riconoscimento dei marker è necessario porre l'attenzione su di un particolare importante: la webcam montata sul robot è al contrario, con la base rivolta verso l'alto. Tale scelta, effettuata per garantire una migliore protezione alla webcam durante gli spostamenti del robot, ha delle ripercussioni non trascurabili sulle immagini che vengono acquisite. Queste saranno infatti ruotate di 180° rispetto alla realtà, apparendo di fatto rovesciate.

Per ovviare a questo inconveniente si è reso necessario realizzare un programma capace di capovolgere le immagini affinché queste rispettassero l'effettiva disposizione degli oggetti del mondo reale.

L'immagine correttamente orientata sarà poi quella sui cui agirà il programma di riconoscimento dei marker.

Il nostro programma ha lo scopo di determinare, sulla base dell'immagine acquisita, il valore in pixel delle coordinate associate ai baricentri dei due marker attivi.

Il cuore dell'applicazione è costituito dall'algoritmo di ricerca dei marker e dall'algoritmo per la determinazione della soglia di luminosità minima che consente il riconoscimento di entrambi i marker. Questo algoritmo ha un duplice compito: verificare se l'immagine contiene effettivamente due marker distinti e calcolare le coordinate dei loro baricentri. Esso è in grado di identificare e distinguere le situazioni nelle quali risultano visibili o non visibili i due marker, restituendo i valori delle coordinate dei due baricentri nel primo caso ed un codice di errore nel secondo.

La tecnica sfruttata per determinare la presenza dei marker tiene conto della particolare disposizione con cui questi sono stati montati: trovandosi uno sopra l'altro è possibile realizzare una duplice scansione pixel per pixel.

Inizialmente l'immagine viene scandita a partire dal vertice alto sinistro spostandosi verso destra, e ricominciando dalla fila di pixel immediatamente inferiore ad ogni fine riga. Quando viene trovato un punto bianco avente un valore maggiore o uguale ad una certa soglia stabilita, si presuppone che esso coincida con l'inizio del marker superiore. Si procede poi con la scansione dei pixel successivi, incrementando, quando ne vengono individuati altri, il numero dei pixel associati al marker superiore

che rientrano nel taglio della soglia attuale. La scansione termina quando si trova un'intera riga che non contiene pixel che soddisfano la soglia attuale.

Successivamente si esegue una nuova scansione dell'immagine alla ricerca del marker basso. Per fare questo la ricerca viene effettuata a partire dal vertice basso destro spostandosi verso sinistra, e ricominciando dalla fila di pixel immediatamente superiore ad ogni fine riga. Quando viene trovato un punto bianco avente un valore maggiore o uguale ad una certa soglia stabilita, si presuppone che esso coincida con l'inizio del marker inferiore. Anche in questo caso, analogamente a prima, si procede con la scansione dei pixel successivi, incrementando, quando ne vengono individuati altri, il numero dei pixel associati al marker inferiore che rientrano nel taglio della soglia attuale. La scansione termina quando si trova un'intera riga che non contiene pixel che soddisfano la soglia attuale.

L'operazione che determina il valore della soglia riveste un ruolo fondamentale nell'algoritmo: la soglia definisce infatti il taglio sotto il quale i pixel non vengono considerati appartenenti ad un marker poiché sono caratterizzati da un valore RGB ritenuto troppo basso. E' chiaro che calcolare una soglia troppo permissiva porterebbe erroneamente a considerare appartenenti ad un marker anche gruppi di pixel che in realtà non vi hanno niente a che fare: in questo caso si rischia di falsare la corretta individuazione del marker e quindi di calcolare con pessima precisione il posizionamento del suo baricentro.

Al contrario, l'utilizzo di soglie troppo restrittive causerà il taglio di pixel che appartengono anche al marker. Questo problema sarà tanto più grave tanto più sarà elevata la distanza tra la webcam ed i marker. Allontanandosi dai marker, infatti, i pixel che li rappresentano diminuiscono in numero e sono caratterizzati da un valore RGB sempre minore: si corre il rischio che ad un certo punto la soglia tagli tutti i pixel associati al marker, impedendone di fatto l'individuazione.

Per l'individuazione dei marker effettueremo la scansione esclusivamente sul piano R del formato RGB, in quanto i valori numerici dei tre piani, essendo l'immagine rappresentata mediante scala di grigi, assumono lo stesso valore.

Quanto detto finora mostra che la webcam percepisce i pixel dei marker in maniera diversa a seconda della distanza e dell'orientamento a cui si trovano rispetto ad essa, quindi l'utilizzo di un valore fisso per la soglia non costituisce certo la soluzione ottimale al problema: tenendo conto di questa considerazione, si è ritenuto più opportuno implementare un calcolo dinamico della soglia mediante un algoritmo di ricerca binaria.

La ricerca viene effettuata in un intervallo che va da una soglia minima ad una soglia massima. Inizialmente l'intervallo è da 0 a 255, cioè comprende tutti i possibili valori che possono essere assunti dai pixel. Ad ogni iterazione questi valori vengono opportunamente aggiornati mediante un algoritmo binario fino a quando non si raggiunge il massimo valore di soglia ritenuto valido: sarà proprio questo valore ad essere utilizzato per la localizzazione finale dei marker.

Una volta conclusa la doppia scansione dell'immagine, l'algoritmo si incarica di calcolare il baricentro relativo al gruppo di pixel associati a ciascuno dei due marker. Per la coordinata x_B del baricentro di un marker si utilizza la seguente formula:

$$x_B = \frac{\sum_{i=1}^{N} 1 \cdot x_i}{N}$$

in cui si assegna ad ogni pixel il valore unitario, si esegue la sommatoria delle coordinate x_i di tutti i pixel appartenenti al gruppo associato al marker e la si divide per il numero totale N di tali pixel. Analogamente, per la coordinata y_B del baricentro di un marker si utilizza la formula:

$$y_B = \frac{\sum_{i=1}^{N} 1 \cdot y_i}{N}$$

dove si assegna ad ogni pixel il valore unitario, si esegue la sommatoria delle coordinate y_i di tutti i pixel appartenenti al gruppo associato al marker e la si divide per il numero totale N di tali pixel.

Specifici test su immagini campione hanno mostrato come la ricerca dei baricentri mediante questa tecnica sia efficace, in quanto le soluzioni che l'algoritmo è in grado di fornire sono caratterizzate da un ottimo livello di precisione, evidenziabile dalla minima approssimazione delle soluzioni rispetto alle posizioni reali dei baricentri.

3.5. Scelta del formato d'acquisizione dell'immagine

"vgrabbj" permette di acquisire le immagini in differenti formati:

- jpg
- png
- ppm

I primi due formati determinano la memorizzazione dell'immagine in forma compressa, mentre il terzo è di tipo bitmap. Risulta di fondamentale importanza la scelta del formato in cui acquisire l'immagine da elaborare. Dopo un'attenta valutazione si è scelto di operare con il formato ppm per le seguenti motivazioni:

- l'immagine deve essere disponibile solo per il tempo necessario all'elaborazione per la ricerca dei due marker e non è quindi necessario memorizzare l'immagine su disco; di conseguenza non si pongono problemi dovuti all'occupazione di memoria fisica delle immagini in formato non compresso;
- per elaborare l'immagine è necessario che sia disponibile in forma non compressa; altrimenti è necessario, mediante un opportuno algoritmo, decomprimerla per poter accedere all'informazione contenuta. Gli algoritmi di compressione e decompressione risultano onerosi in termini di risorse computazionali, in particolare il tempo di decompressione potrebbe risultare fortemente incidente rispetto al tempo di elaborazione dell'immagine. Nel nostro caso un fattore determinante per la buona riuscita del progetto è rappresentato dal tempo totale di elaborazione, che risulta essere uno dei principali vincoli da considerare.
- La compressione dell'immagine determina una perdita di risoluzione e nel nostro caso potrebbero risultare delle misurazioni imprecise. L'immagine che viene acquisita è già affetta da scarsa qualità a causa della bassa risoluzione della telecamera e dal fatto che le immagini vengono scattate in movimento; inoltre i parametri della telecamera sono impostati per poter effettuare una veloce elaborazione in grado di acquisire la posizione dei marker in condizioni ambientali con un elevato disturbo, e non per l'acquisizione dell'immagine con elevata qualità.

3.6. Il formato ppm

Il formato ppm prevede due versioni, molto simili tra loro:

- Plain
- Raw

"vgrabbj" utilizza il formato "raw", che risulta oggi essere l'unico utilizzato (tipicamente il formato ppm è associato per default alla versione raw). Di seguito si descrive il formato ppm "raw" accennando successivamente alla versione "plain":

Il formato è definito come segue:

- due caratteri in sequenza (detti "magic number") che descrivono la tipologia di formato, "P6"
- Whitespace (spazi bianchi, tabulazioni, newline, CRs).
- Larghezza in pixel dell'immagine (un numero intero positivo scritto con caratteri ASCII)
- Whitespace
- Altezza in pixel dell'immagine (un numero intero positivo scritto con caratteri ASCII)

- Whitespace (spazi bianchi, tabulazioni, newline, CRs).
- Il massimo valore della componente di colore (numero intero positivo scritto con caratteri ASCII, con valore massimo 255)
- Whitespace
- L'immagine viene rappresentata mediante una matrice di pixel (pari all'altezza per la larghezza), dove ad ognuno di essi viene associata una terna di 3 valori rappresentante le tre componenti fondamentali dei colori, rispettivamente rosso, verde e blu. La sequenza di valori inizia dal pixel posto nell'angolo in alto a sinistra, prosegue da sinistra a destra e dell'alto al basso, secondo il normale ordine di lettura occidentale. I valori di colore dei pixel sono memorizzati come una sequenza di byte
- Non sono ammessi whitespace nella sequenza dei byte dei pixel;
- I caratteri contenuti tra il carattere "#" e il carattere "end of line" sono ignorati, poiché rappresentano i commenti; non possono essere specificati all'interno della sequenza dei byte rappresentanti l'immagine

Di seguito si riporta un esempio di come si possa presentare un'immagine in formato ppm "raw":

P6	⇐ Tipologia di formato
#Questo è un file d'esempio del formato ppm raw	⇐ Commento
640 480	⇐ Larghezza e altezza immagine
255	⇐ Massimo valore di colore
$$$%%%$$$"""!!!%%% $$$!!!"""!!!	⇐ Inizio sequenza byte immagine
...	⇐ Sequenza byte immagine
"""######$$$$$$$$$%%%)))(((($$$!!!	⇐ Fine sequenza byte immagine

Il formato ppm "plain", raramente utilizzato differisce solo per pochi dettagli:

- I primi due caratteri che specificano il formato sono "P3"
- I valori di colore dei pixel sono memorizzati come una sequenza numeri in formato ASCII decimale; tali valori sono separati da almeno uno spazio bianco
- Ogni linea non può essere più lunga di 70 caratteri

4. Modalità operative

Nei seguenti paragrafi descriviamo i componenti necessari per il corretto funzionamento dell'applicazione e le sue modalità di uso e di installazione. Si descriveranno inoltre i metodi principali che costituiscono il programma.

4.1. Componenti necessari

L'applicazione è stata pensata per funzionare in ambiente Linux, quindi prerequisito fondamentale è che il calcolatore su cui la si vuole eseguire abbia installato una delle distribuzioni di questo sistema operativo: nel nostro caso si utilizza Mandrake 9.2.

Linux mette a disposizione già la maggior parte dei moduli e delle librerie necessarie, ad eccezione dei moduli aggiuntivi pwcx e pwc necessari per consentire il corretto funzionamento della webcam Philips e per settarne i parametri di acquisizione delle immagini.

L'immagine da elaborare è acquisita ruotata di 180° rispetto all'ambiente reale. Essa viene riportata nella configurazione coincidente a quella dell'ambiente grazie ad un apposita funzione inserita all'interno di blobFinder.

Il nostro programma di elaborazione blobFinder contiene al proprio interno vgrabbj e quest'ultimo viene eseguito in modalità demone: tale scelta è stata effettuata per consentire una più rapida acquisizione delle immagini senza avere la necessità di ricaricare ed avviare ad ogni iterazione il processo di acquisizione. In questo modo è possibile ottenere le informazioni RGB relative all'immagine direttamente dal buffer di memoria senza dover ricorrere alla scrittura dell'immagine su file o alla realizzazione di una FIFO per la lettura e la scrittura.

Inoltre questa soluzione consente di eliminare eventuali problemi di sincronizzazione tra l'acquisizione delle immagini e la relativa elaborazione.

All'interno del file vgrabbj.c, nel metodo main(), è presente la chiamata alla funzione elabora_char() che consente di avviare l'elaborazione delle immagini per la ricerca dei blob luminosi:

- *elabora_char(char* sequenzaPixel, int fd)* consente l'esecuzione dell'applicazione e richiama al suo interno i metodi per il caricamento dell'immagine dal buffer di memoria, per la determinazione della soglia valida e delle coordinate dei baricentri dei due marker. Una volta ottenute le coordinate, richiama il metodo che si occupa di scriverle nel formato [x_marker_alto, x_marker_basso, y_marker_alto, y_marker_basso] sull'apposita FIFO "fifo_coordinate";

Tutte le seguenti funzioni sono contenute all'interno del file v_utils.c e poi dichiarate nell'header file elabora_immagine.h:

- *carica_immagine(char* sequenzaPixel)* si occupa di caricare l'immagine da elaborare dal buffer di memoria utilizzato per la memorizzazione. Poiché, come abbiamo detto, le immagini da elaborare sono in bianco e nero, i tre valori RGB che caratterizzano un pixel saranno identici. Per questo motivo è inutile memorizzarli tutti e tre all'interno di una matrice di dimensioni 3x640x480: basterà memorizzarne uno per ciascun pixel, riducendo così la matrice associata all'immagine alle dimensioni di 640x480. Al termine dell'esecuzione di questo metodo avremo quindi a disposizione la matrice 640x480 contenente l'informazione RGB per ciascun pixel dell'immagine caricata.
- *getBaricentri(int sogliaMin,int sogliaMax)*: metodo ricorsivo che prima determina, mediante un algoritmo di ricerca binaria, la massima soglia valida compresa tra sogliaMin e sogliaMax da utilizzare per la determinazione delle coordinate dei baricentri e poi avvia il calcolo effettivo delle coordinate sulla base di tale soglia chiamando il metodo *getCoordBaricentri(valoreSoglia)*;
- *getCoordBaricentri(int valoreSoglia)*: metodo che effettua sulla matrice contenente i pixel dell'immagine la duplice scansione per l'individuazione dei marker attivi. Se la scansione è andata a buon fine ed ha individuato entrambi i marker, il metodo passa al calcolo delle coordinate dei loro baricentri. Una volta determinate esse saranno memorizzate in un apposito array nell'ordine [x_marker_alto, x_marker_basso , y_marker_alto, y_marker_basso]. In caso di mancata individuazione dei due marker verrà restituito un codice d'errore.
- *scrivi_coord_fifo(int fd)*: metodo che si occupa di scrivere sulla FIFO "fifo_coordinate" i valori delle coordinate (x,y) dei baricentri dei due marker secondo il formato [x_marker_alto, x_marker_basso , y_marker_alto, y_marker_basso].

Lo script di shell che avvia blobFinder si chiama *trovaBlob*, e contiene le seguenti istruzioni:

```
#!/bin/bash
setpwc -b
setpwc -g 0 -s 1
./blobFinder -i vga -o ppm -L 100000
setpwc -r
```

Inizialmente lo script memorizza le impostazioni attuali della webcam per permetterne il ripristino al termine dell'elaborazione. Successivamente setta i valori dell'AGV (Automatic Gain Control) a 0 e dello shutter a 1 (velocità massima): in questo modo la webcam acquisisce immagini in bianco e nero in cui sono visibili solo le fonti luminose a più alta intensità, che nel nostro caso corrispondono ai due blob da identificare.

Si esegue poi l'avvio di blobFinder in modalità demone (opzione –L, riportando il tempo di acquisizione delle immagini in microsecondi) e specificando la risoluzione (vga, opzione -i) ed il formato delle immagini (ppm, opzione -o).

Conclusa l'esecuzione di blobFinder in modalità demone, lo script ripristina le precedenti impostazioni della webcam per consentirne un eventuale corretto utilizzo anche da parte di altre applicazioni (es.Camstream).

Se si desidera poi testare il funzionamento dell'applicazione per la determinazione delle coordinate dei baricentri dei due marker attivi si può avviare il programma *leggi_fifo*, che ha lo scopo di leggere le coordinate individuate dall'applicazione ed inserite nella FIFO "fifo_coordinate".

4.2. Modalità di installazione

Il corretto funzionamento dell'applicazione richiede la realizzazione dei seguenti passi:

- Installare setpwc:
```
tar -xvf setpwc-0.6.tar
cd setpwc-0.6
make
make install
```
- Installare il blobFinder:
```
tar -xvf blobFinder.tar
cd blobFinder
make
```
- Per avviare il programma, nella stessa directory di blobFinder è presente lo script trovaBlob che dovrà essere avviato con:
```
./trovaBlob
```
- Per leggere le informazioni sulle coordinate dei baricentri dei due marker si può utilizzare come tester il programma leggi_fifo, contenuto nella stessa cartella di trovaBlob:
```
./leggi_fifo
```

Il programma visualizzerà sullo standard output (a video) le letture relative alle coordinate (x,y) dei baricentri nel formato [x_marker_alto, x_marker_basso , y_marker_alto, y_marker_basso].

Per un riconoscimento ottimale dei due blob, si consiglia di alimentare i marker luminosi con una tensione compresa tra i 6 e gli 8V.

5. Conclusioni

Il progetto risulta conforme a tutte le specifiche richieste raggiungendo totalmente gli obiettivi prefissati.

L'acquisizione e l'elaborazione dell'immagine avvengono in tempi brevi e sono in grado di fornire con un ritmo elevato (10 misurazioni al secondo) i risultati ottenuti corrispondenti alle coordinate dei baricentri dei due marker.

I tempi per l'elaborazione dell'immagine risultano trascurabili rispetto ai tempi di acquisizione, ma si consiglia, in base a prove sperimentali, di non aumentare il ritmo di acquisizione e di elaborazione delle immagini in quanto il carico medio del processore dovuto all'esecuzione del programma è circa del 20%.

La precisione delle misure risulta estremamente alta sia alla distanza massima (4m) che a quella minima (30cm dai bumper): l'approssimazione risulta essere dell'ordine di un pixel.

Bibliografia

[1] M. Cagno, M. Panteghini: Tesi di laurea;

[2] http://www-info2.informatik.uni-wuerzburg.de/mitarbeiter/wolfram/lehre/bildformate.html

[3] http://www.vanheusden.com/setpwc

[4] http://vgrabbj.gecius.de

Indice