POLITECNICO DI MILANO

Facoltà di ingegneria dell'informazione.

V Facoltà di Ingegneria
Sede di Milano Leonardo

Progettazione e implementazione di un robot mobile dotato di un sistema di visione laser per la percezione della profondità.

Relatore: Ing. Michele FOLGHERAITER

Tesina di laurea di:
Cardamone Luigi
Matr. 665405

Anno Accademico 2005-2006

A mio nonno Carmelo,

alla mia famiglia

e a tutti coloro

che mi vogliono bene.

All the limitative Theorems of metamathematics and the theory of computation suggest that once the ability to represent your own structure has reached a certain critical point, that is the kiss of death: it guarantees that you can never represent yourself totally. Godel's Incompleteness Theorem, Church's Undecidability Theorem, Turing's Halting Problem, Turski's Truth Theorem-- all have the flavour of some ancient fairy tale which warns you that "To seek self- knowledge is to embark on a journey which . . . will always be incomplete, cannot be charted on a map, will never halt, cannot be described."

Douglas R. Hofstadter

Indice

Introduzione……6

1. Sistemi di visione 3D ……...…..……9
   - Introduzione ……9
   - Le principali applicazioni...…….10
   - Le tecnologie usate ……11
   - Triangolazione attiva……..……..14
   - Analisi delle misurazioni ……..…….17
   - Stato dell'arte ..……..……...……18
2. Navigazione di robot mobili …….....22
   - Introduzione ……...22
   - Mapping …..……...……..……23
   - SLAM…..……..……...……25
   - Stato dell'arte …..……..……...……27
3. Progettazione ed implementazione di uno scanner 3D ……..……29
   - Introduzione ……..……29
   - Modello teorico ……..……..……..……30
   - Implementazione ……..……37
   - Meccanica ed elettronica del sistema ..……..……40
   - Acquisizione, filtraggio e analisi dei fotogrammi……..45
   - Software del Sistema ……..……..……48
   - Taratura del dispositivo ……..……..……53
4. Progettazione ed implementazione di un robot mobile ……55
   - Introduzione ……55
   - Cinematica ……57
   - Elettronica……58
   - La base mobile ……61
   - Il sistema di visione ……63
   - Funzionamento del robot ……65
   - Software del robot……67

5. Risultati e Conclusioni ……74
- Risultati scanner 3D……74
- Risultati robot mobile ……79
- Conclusioni ……82
- Sviluppo futuri ……83

Digressione filosofica sul futuro della robotica ……84

Ringraziamenti ……87

Bibliografia……88

Introduzione

In questi ultimi anni si sta assistendo al passaggio del robot da semplice automa da catena di montaggio industriale a elemento attivo presente nella vita di tutti i giorni e parte integrante in svariate attività umane.
I robot mobili stanno trovando applicazione in molteplici campi: dall'agricoltura alla telesorveglianza, dalle operazioni di soccorso alle missioni spaziali, con una potenziale estensione a tutti quei settori caratterizzati da compiti ripetitivi, operazioni che richiedono alta precisione o attività rischiose per l'uomo.
Il passaggio dalla cosiddetta robotica industriale alla robotica mobile non è diretto ma comporta nuove problematiche da affrontare che al momento sono argomento di ricerca in questo settore. Si passa da un ambiente strutturato e ben conosciuto nel quale il robot effettua delle azioni ben determinate, ad un ambiente molto più complesso, dinamico con il quale le interazioni possono essere molto complicate.
In questo nuovo scenario il robot deve innanzitutto essere dotato di sensori per percepire l'ambiente dinamico che lo circonda: solo dopo una corretta modellizzazione della spazio circostante si può operare in modo efficiente su di esso.
Questo lavoro di tesi, in cui si è implementato un robot mobile per il *mapping* di ambienti indoor, parte proprio dal problema di una corretta rilevazione dello spazio circostante. Per raggiungere tale scopo si è scelti di dotare il robot di un sistema di visione per la percezione della distanza, che rispetto ad altri tipi di sensori permette di ottenere informazioni più dettagliate sull'ambiente da analizzare. La complessità di tale sistema ha richiesto una fase di studio e di sviluppo separata da quella di progettazione del robot stesso. Durante questa fase si sono studiati diversi tipi di sensori 3D, valutandone vantaggi e svantaggi nell'utilizzo su un robot mobile. Si è quindi progettato un dispositivo in luce strutturata basato sulla triangolazione di un segmento laser, di cui in letteratura esistono pochi esempi applicati alla robotica mobile. Il dispositivo ottenuto è di fatto uno scanner 3D, cioè uno strumento in grado di acquisire

immagini tridimensionali, che può essere utilizzato per scansionare degli oggetti al fine di ottenerne il modello virtuale.
Nella fase successiva del lavoro si è sfruttato tale sistema di visione, opportunamente adattato, per implementare un vero è proprio robot mobile su ruote, in grado di muoversi tra gli ostacoli e costruire una mappa globale del spazio esplorato.
La progettazione ha richiesto lo studio di diversi modelli cinematici: si è scelto di basare il movimento del robot sul modello del differential drive. E' stata quindi implementata struttura del robot e l'elettronica necessaria per comandare l'intero sistema tramite computer.
L'architettura software del sistema si occupa di gestire i dati di input provenienti dal sensore di visione, di elaborarli e di determinare il movimento del robot inviando gli opportuni comandi agli attuatori che permettono la locomozione.
Per creare la mappa dell'ambiente esplorato è stato studiato lo *SLAM problem*, cioè il problema di costruire una mappa simultaneamente alla localizzazione della posizione del robot nel riferimento globale. Il problema dello SLAM (Simultaneous Localization And mapping) è uno degli argomenti più attuali nella ricerca in campo robotico.
Con questa tesi si è voluto creare un robot, che fosse in grado di percepire lo spazio circostante e muoversi al suo interno, e che rapprensenti il mattone base per un progetto più ampio che miri a raggiungere task più complessi.
Questo lavoro è strutturato in 6 capitoli:
Nel primo verrà presentato un quadro generale sui sistemi di visione 3D, e sulle loro applicazioni. Verrano illustrati le diverse tecniche usate per la misura delle distanze e verranno approfondite le tecniche di triangolazione attiva. Infine verrà esposto lo stato dell'arte.
Nel Capitolo 2, sarà spiegato cosa si intende per navigazione di un robot mobile. Verrano analizzate le principali problematiche della navigazione con particolare attenzione alla fase di mapping. Verrano infine presentate le soluzioni adottate in alcuni prototipi studiati nello stato dell'arte.
Nel Capitolo 3, verrà presentata la parte di progettazione ed implementazione dello scanner 3D. Dapprima verrà introdotto il modello teorico usato per il calcolo delle distanze e poi verrà affrontato in dettaglio l'aspetto implementativo. Verrà quindi analizzata la parte hardware e la parte software del dispositivo.
Nel Capitolo 4 verrà esposta la parte di progettazione ed implementazione del robot mobile. Ne verrà analizzata la cinematica, e la parte hardware con particolare attenzione

all'elettronica necessaria a comandare il robot. Saranno poi evidenziate le modifiche dello scanner 3D per poter funzionare da sensore per la navigazione e verrà analizzata l'architettura software che gestisce il movimento del robot e le operazioni necessarie per eseguire il mapping.
Nel Capitolo 5, verranno commentati i risultati ottenuti sia dal singolo sistema di visione, sia dal robot mobile. Dopo le conclusioni verrano presentati i possibili sviluppi futuri.

1. Sistemi di visione 3D

1.1. Introduzione.

La costruzione di rappresentazioni tridimensionali a partire da semplici immagini bidimensionali è stata, e continua ad essere una problematica molto attuale nel campo della robotica ed in particolare della computer vision. Tale problematica, che nel sistema visivo umano viene eseguita in modo estremamente efficiente, risulta particolarmente complessa quando in maniere algoritmica si cerca di ricostruire la profondità della scena catturata da un sensore ottico.
I diversi studi condotti per affrontare il problema hanno portato allo sviluppo sistemi di visione 3D, in grado di determinare la distanza dei punti che compongono la scena osservata. Tali dispositivi sono più complessi rispetto ad un semplice sensore ottico
in quanto nella maggior parte dei casi non si limitano ad acquisire immagini in modo passivo, ma agiscono in modo attivo proiettando dei pattern di luce o delle onde elettromagnetiche. Attraverso questi segnali si ottengono delle informazioni supplementari che permettono il calcolo della profondità. Successivamente il software strettamente correlato con questi sistemi di visione, partendo da un insieme di punti di cui è nota la distanza, permette di ottenere, tramite diversi metodi di interpolazione, delle rappresentazioni 3D e dei modelli virtuali estremamente fedeli.
Questi sensori 3D oltre ad acquisire il modello geometrico, sono in grado di acquisire importanti informazioni sulla superficie degli oggetti analizzati, come il colore, il tipo di materiale e la lucentezza. Quindi nelle scansioni più complete vengono usati più sensori differenti ognuno dei quali è progettato per rilevare un particolare aspetto della superficie dell'oggetto.

I progressi tecnologici e il miglioramento delle tecniche usate hanno permesso di ottenere dei dispositivi per la visione 3D sempre più precisi e affidabili, che hanno trovato notevoli applicazioni anche in campi diversi dalla robotica.
Nel corso del capitolo verrano descritte le principali applicazioni dei sistemi di visione 3D. Successivamente verrà analizzato il funzionamento, le tecniche utilizzate e i principi fisici sui cui si basano. In particolare verrà approfondita la triangolazione attiva. Verra fatta una panoramica sui metodi utilizzati per generare delle rappresentazioni tridimensionali e infine verranno presentati alcuni progetti che rappresentano lo stato dell'arte nel campo dei sistemi di visione 3D.

1.2. Le principali applicazioni.

Un'area di grande interesse per i sistemi di visione 3D è senza dubbio la robotica mobile, dove vengono impiegati per la navigazione di veicoli autonomi, soppiantando in parte l'utilizzo dei sensori di prossimità. Infatti, rispetto ad un sensore di prossimità che si limita a rilevare la presenza di un ostacolo, un sistema di visione 3D è in grado di individuarne la forma, la dimensione, e la sua distanza, permettendo di implementare degli efficienti algoritmi di *obstacle avoidance*. I sensori 3D, vengono utilizzati sia in robot che navigano in ambienti sconosciuti, effettuando il *mapping*, sia in robot che dispongono di una mappa: in questo caso il sensore viene usato per ottenere una corretta localizzazione all'interno nella mappa, e per individuare eventuali ostacoli imprevisti. L'uso dei sistemi di visione 3D nei robot mobili verrà ripreso in modo più approfondito nel Cap. 4 dove verrà descritto la progettazione e l'implementazione di un robot che utilizza proprio un sistema di visione per muoversi in un ambiente non noto a priori.
Nella robotica, i sensori 3D vengono anche utilizzati per l'*object detection*, cioè il riconoscimento di oggetti. In questo settore hanno praticamente sostituito tutte le tecniche basate su immagini bidimensionali. Infatti, con un sensore 3D è possibile acquisire delle informazioni sulla struttura geometrica e altre caratteristiche distintive degli oggetti esaminati. Tali informazioni sono indipendenti della condizioni di luce, e dalle caratteristiche della superficie. In questo modo, per esempio, un robot sarà in grado di riconoscere una sedia, indipendentemente dal colore o dal materiale di cui è fatta, perché ciò che conta saranno le sue caratteristiche distintive come la presenza di una base e quattro piedi. Per fare ciò si utilizzano algoritmi, che applicano in cascata

diversi classificatori, fino a raggiungere un corretto riconoscimento dell'oggetto, come è possibile vedere nel lavoro [3].
Come già anticipato, i sistemi di visione 3D, hanno trovato numerose applicazioni anche al di fuori del campo della robotica. Un esempio è il loro utilizzo come scanner 3D cioè per scansionare degli oggetti estraendone la struttura geometrica al fine di creare dei modelli virtuali molto precisi. Gli scanner 3D, sono particolarmente importanti per la cosiddetta reingegnerizzazione (reverse engineering). Sempre più spesso è necessario costruire modelli 3D (modelli CAD) di oggetti di cui non si ha il progetto e la documentazione originale. La necessità di modelli 3D si ha anche nell'industria e in generale in quelle attività nelle quali è più facile progettare un oggetto lavorandoci fisicamente utilizzando materiali modellabili, anziché progettarlo al computer: è quindi necessario un dispositivo che sia in grado di acquisire con la massima precisione la forma dell'oggetto creato.
Sempre in campo industriale tali sensori sono necessari nelle catene di montaggio, per l'assemblaggio automatico, e per il controllo e l'ispezione dei prodotti finiti.
I dispositivi per la percezione della distanza vengono utilizzati anche in ambito topografico per la creazione di DEM (digital elevation model) cioè rappresentazioni 3D della superficie terrestre acquisite solitamente con foto aeree. In questo modo è possibile ottenere modelli completi di strade, abitazioni, città. Alcune delle soluzioni adottate per costruire questi modelli sono state analizzate in [4].
Infine altre importati applicazioni sono rappresentate dalla video sorveglianza e dalla costruzione di modelli 3D destinati ad applicazioni di realtà virtuale.

1.3. Le tecnologie usate.

Un generico sistema di visione 3D è composto da un dispositivo in grado di emettere un segnale verso la scena da analizzare, e da un sensore in grado di rilevare il segnale di ritorno calcolando quindi la distanza dell'oggetto puntato.
La vasta gamma di sistemi per la percezione della distanza si basa solo su tre diversi principi:

- Triangolazione
- Time of flight (TOF)
- Interferometria

Questi tre principi sono diversi nel modo in cui varia l'errore di misura con la distanza degli oggetti analizzati. L'incertezza nelle misura può andare da pochi nanometri ad alcuni millimetri, in base al raggio d'azione dello strumento. Se indichiamo con z la distanza dell'oggetto misurato e con δ_z l'incertezza della misura, si ha che δ_z varia al variare di z nei seguenti modi:

- Triangolazione: $\delta_z \sim z^2$
- TOF: $\delta_z \sim z^0$
- Interferometria: $\delta_z \sim z^{-1}$

L'andamento dell'errore relativo δ_z / z che è il parametro più importante, viene riportato in Figura 1.1:

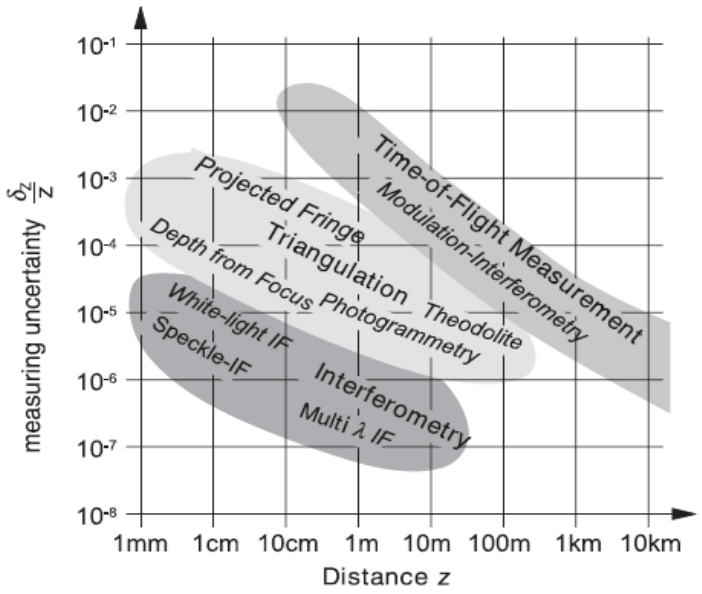


Figura 1.1: Andamento dell'incertezza relativa di misurazione nelle diverse tecniche per rilevare la distanza.

Triangolazione:

Nella triangolazione si determina la distanza di un punto, tramite il triangolo che si viene a formare con tale punto e la base dello strumento. Conoscendo l'angolo alla base e la dimensione della base stessa si ottiene il lato del triangolo che rappresenta la distanza cercata. La triangolazione verrà approfondita nel prossimo paragrafo.

Time of Flight:

In questa tecnica le distanza vengono determinate misurando il cosiddetto tempo di volo di un segnale luminoso. Tale segnale viene emesso da un emettitore, e dopo essere stato riflesso da uno ostacolo che rappresenta l'oggetto che vogliamo misurare, viene ricevuto da un rilevatore. Se chiamiamo τ il tempo necessario a compiere questo cammino la profondità *z* non è altro che:

$$z = c \cdot \tau / 2$$

Dove *c* è la velocità della luce. Anche per distanze elevate, la misura di un tempo τ cosi piccolo crea molte difficoltà, a causa dell'eccessiva velocità della luce: $3 \cdot 10^8 m/s$. E' quindi necessario modulare il segnale luminoso con onde a frequenza minore. Gli errori nella misura del tempo di volo, sono dovuti ai ritardi che si hanno nel circuito elettronico. Tali errori sono praticamente indipendenti dalla distanza misurata, e sono nell'ordine dei millimetri.

Come è possibile vedere dalla Figura 1.1, questa è la tecnica più efficiente e precisa per le grandi distanze.

L'interferometria classica:

Anche l'interferometria si basa sull'emissione e la ricezione di un segnale. L'emettitore divide il segnale in due parti: una parte servirà per misurare l'oggetto, l'altra per effettuare il confronto con il segnale ricevuto dal rilevatore.

Dalla Figura 1.1 è possibile vedere come il principale problema dell'interferometria è l'impossibilità ad essere utilizzata per distanze oltre ai cento metri. Per le distanza minori, invece, risulta la tecnica più precisa, raggiungendo risoluzioni inferiori al nanometro.

L'interferometria, come anche le tecniche basate sul tempo di volo, hanno il vantaggio che la sorgente del segnale e il ricevitore non hanno il vincolo di essere posti ad un certa distanza e con un certo angolo. Ciò evita la formazione di zone d'ombra, cioè zone colpite dal segnale ma non inquadrate dal sensore ottico, che sono invece un problema nella triangolazione.

1.4. Triangolazione attiva.

La triangolazione è la tecnica più ampiamente usata per misure ottiche 3D. In Figura 1.2 è possibile vedere le cinque principali varianti della triangolazione, che nonostante lo stesso principio di base, appaiono estremamente differenti.
Tra queste, le tecniche basate su luce strutturata (triangolazione attiva) sono quelle che hanno permesso di ottenere risultati migliori. In queste tecniche, il sensore ottico che viene utilizzato è un dispositivo CCD. I sensori CCD (coupled charge device), sono costituiti da un piccolo chip in silicio, formato da tante micro celle ognuna della quali trasforma il segnale luminoso in ingresso in un segnale digitale.

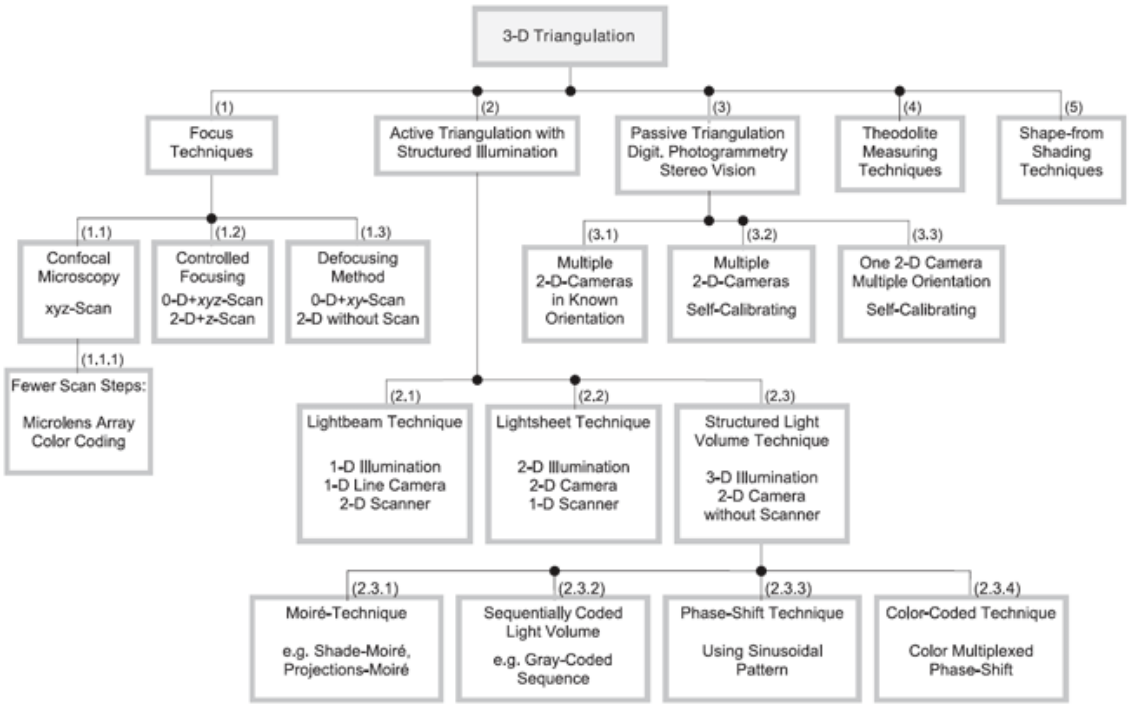


Figura 1.2: Le diverse tecniche di triangolazione.

La triangolazione attiva, a sua volta, si divide in tre categorie a seconda della forma luminosa che viene proiettata: può essere un piccolo punto luminoso, o un stretta linea o un reticolo.
Nel caso in cui viene proiettato un punto (spot) si ha un triangolo tra la sorgente laser, il punto illuminato e il sensore CCD. Essendo note *d*, distanza relativa tra proiettore e sensore CCD, e l'angolo α, inclinazione della sorgente, le coordinate dello spot si determinano tramite triangolazione misurando la posizione della luce sul sensore CCD.

Per misurare le distanze di tutti i punti della scena occorre una scansione bidimensionale.

Se invece di utilizzare uno spot singolo si proietta una striscia luminosa Figura 1.3 si può calcolare la distanza di più punti contemporaneamente ed è necessario scandire la scena solo in una direzione.

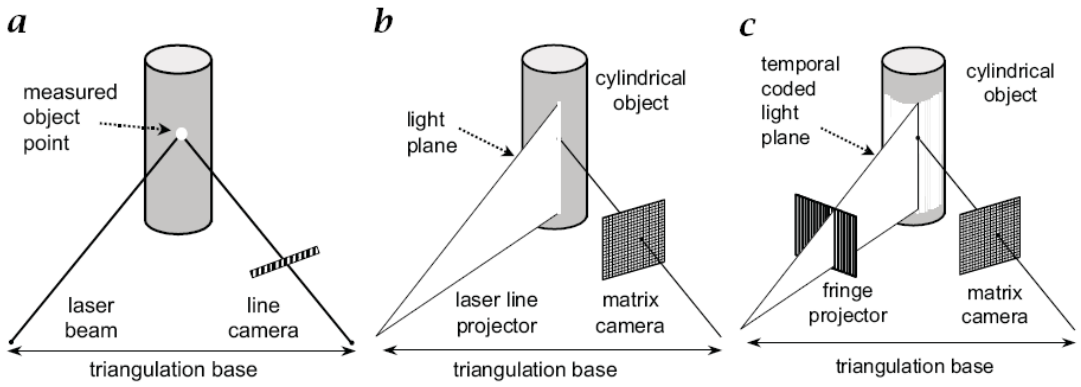


Figura 1.3: Tecniche basate su luce strutturata.

Inoltre, se si proiettano più strisce di luce contemporaneamente, si può istantaneamente acquisire tutta la scena. In quest'ultimo caso sorge la necessità di identificare e distinguere univocamente le varie strisce per poter determinare tutti i punti appartenenti a queste. Per risolvere tale problema si usano patterns di un particolare tipo di luce detta luce codificata, che investe completamente l'oggetto 3D. Ogni pattern, caratterizzato da una propria sequenza binaria, è costituito da strisce bianche e nere e permette di dividere l'area proiettata in zone chiare e scure che vengono infittite ad ogni passaggio, come è possibile vedere in Figura 1.4.

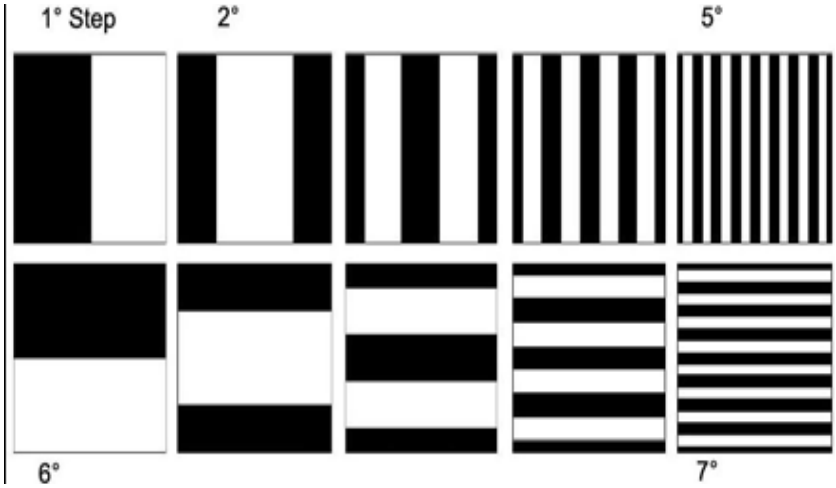


Figura 1.4: Sequenze di patterns di luce strutturata, i cui passi sono caratterizzati da un dimezzamento progressivo della lunghezza d'onda.

La sequenza di immagini relativa alla proiezione dei vari patterns consente di determinare univocamente la locazione dei vari pixel impressionati. Il sensore ottico CCD determina la presenza o meno della luce proiettata ad ogni step, generando una sequenza di bit, che decodificata, permette di individuare le coordinate *x* ed *y* relative alla posizione di ciascun pixel impressionato.

Nel metodo di Gray-Code, ciascun pattern, impiegato nella sequenza temporale, è caratterizzato da frange bianche e nere, la cui lunghezza d'onda si dimezza progressivamente ad ogni step successivo. Questa tecnica risulta vantaggiosa in quanto l'acquisizione dei dati 3D dall'oggetto reale, situato generalmente a distanze inferiori al metro, avviene sotto forma di densa nuvola di punti 3-D.

Sempre nel caso in cui il proiettore genera un volume di luce, i metodi Moiré permettono di risalire alla forma tridimensionale analizzando le immagini di alcune frange di interferenza proiettate sull'oggetto e modificate da esso. Questi metodi sono particolarmente adatti per acquisizioni molto precise di superfici di grandi curvature e senza discontinuità, anche se sono ancora piuttosto lenti e poco automatici.

I metodi di digitalizzazione basati sulla triangolazione sono caratterizzati da una incertezza della misura sul calcolo della profondità *z*. Si vede sperimentalmente che questa incertezza è inversamente proporzionale sia alla distanza relativa tra il sensore ottico e il proiettore, sia alla distanza focale della lente, ma è direttamente proporzionale alla radice quadrata della profondità *z*.

Sfortunatamente sia la distanza proiettore sensore che la distanza focale non possono essere prese molto grandi per ovviare a tale problema, difatti il set-up meccanico del sistema ottico dipende principalmente dalla distanza relativa tra la camera ed il proiettore, il cui valore troppo elevato potrebbe perturbarne la stabilità.
Inoltre l'aumento di tale valore facilita la formazione di effetti d'ombra (self-occlusion), poiché determinate zone dell'oggetto possono rendersi inaccessibili alla scansione ottica.
Durante l'acquisizione si possono presentare degli effetti indesiderati che vanno a pregiudicare la qualità dei dati rilevati, amplificando rumori e disturbi di tipo ottico.
Ciò può essere determinato da fenomeni d'interferenza della luce presente nell'ambiente, da assorbimento e dispersione eccessivi della luce riflessa da superficie ruvida, da misure poco accurate su superfici molto oblique.
Inoltre, in presenza di superfici molto riflettenti, durante la digitalizzazione potrebbero essere interpretati dati provenienti da altre posizioni dell'oggetto.
Per ridurre al minimo i fenomeni di disturbo e rumore è necessario opacizzare omogeneamente l'intera superficie da scansionare, in modo tale da rendere abbastanza chiara e delimitata la direzione di riflessione della luce dopo che questa abbia impattato la superficie dell'oggetto.

1.5. Analisi delle misurazioni.

Tramite un generico sistema di visione, indipendentemente dalla tecnologia usata per acquisire informazioni sulla scena antistante, si ottiene a fine scansione un insieme di punti di cui è nota la distanza. Tali informazioni vengono memorizzate in una griglia regolare chiamata mappa di profondità (depth map), che può essere espressa sia in coordinate cartesiane che in coordinate polari. Assieme alle informazioni sulla distanza possono essere presenti informazioni sull'intensità del segnale, nel caso di acquisizioni in bianco e nero, oppure i valori RGB del punto nel caso di acquisizioni a colori.
La depth map non è ancora un utile rappresentazione tridimensionale, ma è il punto di partenza per tutti quegli algoritmi che tramite interpolazione sono in grado di ricostruire struttura geometrica della scena osservata.
Una prima analisi della depth map può essere costituita da un filtraggio volto ad eliminare errori di misura. Successivamente si utilizzano degli algoritmi per il

riconoscimento delle linee. Sull'insieme delle linee trovate vengono applicati altri algoritmi che si occupano del riconoscimento delle superfici. Dopo questa fase si ha un modello geometrico della scena osservata. Nelle analisi più accurate si possono applicare dei classificatori volti a determinare quali oggetti sono presenti nella scena, sulla base di informazioni memorizzate a priori in un database.
Nel caso di scansioni multiple di una stessa scena, si pone il problema di unire i dati delle varie misure in un unica rappresentazione. Si usano quindi metodi di allineamento e di unione che ricostruiscono la scena unendo le varie scansioni nelle zone di massima verosimiglianza. Questo è un tipico problema che si ha nelle operazioni di mapping, dove una mappa viene ricavata dall'unione di tante scansioni.
Tutti questi algoritmi, che dai punti determinano linee e superfici, sono computazionalmente pesanti e spesso vengono usato off line, per ricostruire mappe 3D dettagliate. Nei sistemi di visione dedicati alla robotica mobile si applicano degli algoritmi più grossolani, ma veloci, che forniscono al robot una prima approssimazione dell'ambiente circostante in real time.

1.6. Stato dell'arte.

Verranno di seguito presentati alcune implementazioni pratiche di sistemi di visione 3D che rappresentano lo stato dell'arte nei diversi contesti presi in esame. In ognuno dei progetti che verranno presentati si mira a raggiungere risultati diversi: si va dalla ricerca di elevate precisioni, alla velocizzazione delle acquisizioni a secondo degli obiettivi per cui è stato realizzato il sistema.

Il primo esempio [1], è un sistema di visione utilizzato come scanner 3D per la scansione e la creazione di modelli virtuali delle statue di Michelangelo. In questo progetto si è puntato molto sulla precisione in modo da riuscire ad acquisire la superficie delle statue ad un livello di dettaglio che permettesse di riconoscere i colpi di scalpello. Per raggiungere questo scopo si è realizzato uno scanner laser con un precisione pari ad un quarto di millimetro. Inoltre per creare dei modelli perfettamente fedeli all'originale si è cercato di acquisire il più informazioni possibile sulla superficie: come il colore, il tipo di materiale e la sua attitudine a riflettere. Tramite questa ultima

informazione è stato possibile durante le operazioni di rendering osservare le statue in modo molto realistico al variare delle condizioni di luce riprodotte virtualmente.
Il sistema utilizzato per le acquisizioni è composto principalmente da un scanner laser montato su una struttura motorizzata di supporto. Tra la vasta gamma di sensori 3D, si è scelto uno scanner basato sulla triangolazione di una striscia laser proiettata orizzontalmente, in quanto offre la migliore combinazione di accuratezza, volume di lavoro, e robustezza. Lo scanner utilizza un diodo laser da 5 mW e 660 nanometri, e un sensore ottico CCD 512 x 480. Per quanto riguarda la risoluzione verticale, cioè la distanza tra la proiezione di una striscia e l'altra, è stata impostata ad 1/4 di mm mentre per la profondità zeta si è scelto una risoluzione di 1/8 di mm. Con questi valori il campo visivo è ampio 14 cm, e profondo sempre 14 cm.
La scelta della risoluzione di scansione è risultata dal trade off tra una giusta precisione e una dimensione non eccessiva dei dati raccolti durante ogni scansione. Infatti con questo valore di precisione, per una singola statua sono stati raggiunti fino a 2 gigabyte di dati, e una risoluzione maggiore avrebbe appesantito pesantemente i database e gli algoritmi che successivamente avrebbero dovuto lavorare su di essi. Quindi, una risoluzione maggiore avrebbe causato il dover gestire per ogni statua una mole estremamente grande di dati, mentre una risoluzione minore avrebbe perso il dettaglio necessario a rendere visibili i colpi di scalpello.
Si è preferito acquisire il colore in una scansione separata, utilizzando un CCD a maggiore risoluzione. Per avere il più possibile un'illuminazione uniforme sulla superficie delle statua si è scelto di illuminare da vicino la parte da scansionare con un lampada alogena.
Le enormi quantità di dati acquisite durante le singole scansioni non hanno permesso nessun tipo di elaborazione real time. Tutta la gran mole di dati è stata analizzata in una fase di post processing nella quale sono stati creati dei modelli virtuali altamente fedeli agli originali. Per la creazione di tali modelli sono stati utilizzati complessi algoritmi di allineamento (aligning) e unione (merging). Infatti per ogni statua, per ottenere la massima precisione sono state scansionati porzioni di superficie nell'ordine della decina di centimetri. Ciò provoca moltissime diverse scansioni che devo essere allineate e unite per ricostruire il modello originale di un intera statua.
L'obiettivo dei questo progetto è stato di spingere un passo in avanti lo stato dell'arte nel settore degli scanner 3D. Si sono quindi ottenuti dei modelli tridimensionali, due volte più precisi rispetto a quelli precedenti. Tale elevata precisione è stata ottenuta a

discapito del tempo: le scansioni più lunghe possono richiedere anche un ora, e l'analisi dei dati molto più.

Il secondo esempio [6], tratta di un sistema di visione 3D progettato per la robotica mobile. In questo progetto si propone di usare scansioni 3D, rispetto alle normali 2D, in quanto durante la navigazione del robot mobile, permettono di evitare le collisioni con oggetti bassi o dal profilo non omogeneo, come sedie e tavoli.
Per realizzare il sensore 3D, è stato montato un telemetro laser 2D su un motore servo. Il telemetro laser 2D, è dispositivo di rilevamento delle distanza basato sul time of flight, che acquisisce informazioni solo lungo una sezione dello spazio 3D. Si usa il motore servo per fare ruotare il telemetro attorno all'asse orizzontale e ottenere scansioni tridimensionali. In questo modo è possibile fare un intera scansione di 150° orizzontali per 90° orizzontali in 4 secondi, o in 12 se si scegli una risoluzione maggiore. La risoluzione standard è di 5 cm, mentre l'errore sulla distanza misurata dipende dal telemetro ed è nell'ordine del centimetro. Sopra il motore servo è stato montato un sensore CCD per acquisire le texture, che è stato interfacciato con il computer tramite porta USB. Il motore servo invece si interfaccia tramite parallela, e il telemetro tramite porta seriale. I dati catturati dal telemetro vengono elaborati sia in real time che successivamente off line per ottenere una mappa globale. Gli algoritmi real time, lavorano solo sui dati appena acquisiti, senza utilizzare le scansioni precedenti. I dati vengono inizialmente preprocessati, in modo da applicare velocemente un algoritmo lineare di *line detection*. Le linee ottenute con un determinata angolazione del telemetro, vengono confrontate con quelle delle altre sezioni al fine di trovare una superficie che passi per esse. In particolare se nelle diverse sezioni un certo numero di linee mantiene la stessa posizione, si è trovata una superficie verticale, che rappresenta un ostacolo per il robot. L'analisi real time dei dati, non prosegue oltre. Nella fase off line, si parte dalle superfici individuate, per raggrupparle in oggetti: un particolare algoritmo intrappola le superfici sufficientemente vicine all'interno di blocchi. Non viene implementato però nessun meccanismo di object detection. Successivamente si utilizzano tutti i dati per ottenere un completa mappa 3D dell'ambiente esplorato. L'obiettivo raggiunto dal progetto è stato la realizzazione di un sensore 3D molto veloce e affidabile che grazie anche agli algoritmi real time sviluppati, è adatto per la navigazione di robot mobili.

Nel terzo e ultimo esempio vengono studiati dei metodi per realizzare dei modelli virtuali della superficie terrestre, a partire da misurazione aeree. I dati utilizzati sono stati ottenuti attraverso un dispositivo per il rilevamento delle distanza montato su un aereo. Questo dispositivo è un telemetro laser, che scansiona la superficie terrestre con una frequenza di 7kHz, e una densità di campionamento pari a 0.25-5 metri volando ad un altezza tra i 50 e 1000 metri. In questo modo viene prodotta una matrice che per ogni punto *x,y* scansionato, contiene un valore che rappresenta la distanza dall'aereo. Le misurazioni di profondità vengono effettuate con una precisione di 0.1-0.3 metri. Mentre la precisione nel misurare *x,y* dipende dall'accuratezza del GPS utilizzato per determinare la posizione dell'aereo.

I dati ottenuti, non sono altro che una nube di punti, che giacciono su una superficie 3D. Su questi punti, prima di tutto, vengono applicati dei potenti filtraggi, per eliminare il rumore che in questo tipo di misurazione è molto maggiore rispetto agli esempi analizzati in precedenza. La causa principale del rumore, è l'elevata distanza dalla quale vengono acquisite le misurazioni.

Successivamente questi dati vengono interpolati. Particolari algoritmi vengono impiegati per distinguere la vegetazione e le costruzioni, dalla superficie terrestre. Degli algoritmi di object detection si occupano di classificare le costruzioni individuate: strade, ponti, case. Degli algoritmi più avanzai creano dei modelli 3D dell'intera superficie terrestre scansionata.

Le tecniche usate per elaborare i dati del telemetro, hanno permesso di ottenere dei risultati molto accurati.

2. Tecniche di navigazione per un robot mobile

2.1. Introduzione.

Un robot mobile è un robot dotato di un sistema di locomozione, di un sistema sensoriale e di un unità elaborativa. Tale unità elaborativa deve essere in grado di utilizzare i sensori per modellare l'ambiente, e in base a queste informazioni deve comandare il sistema di locomozione per permettere al robot di navigare nello spazio circostante.
Un robot mobile può essere utilizzato per eseguire diversi compiti, i quali si riconducono al raggiungimento di un punto di *goal* all'interno dell'ambiente.
Quindi, la navigazione consiste nel dirigere il cammino di un robot mobile affinché raggiunga la destinazione, non si perda e non si schianti contro ostacoli fissi o in movimento che siano.
Il problema della navigazione può essere scomposto in tre sotto problemi:

- mapping
- planning
- driving

Il mapping (mappatura), è la fase in cui il robot utilizza i suoi sensori per esplorare l'ambiente circostante al fine di ricavare tutte le informazioni necessarie per interagire con esso. Questa fase precede le altre, in quanto senza una minima conoscenza dello spazio antistante al robot non è possibile effettuare alcuna operazione.
Il planning (pianificazione), è la fase in cui vengono calcolare le traiettorie e tutte le operazioni necessarie per raggiungere un determinato punto di *goal*, che può essere un sotto problema di un task più complesso. L'attività di planning è fortemente legata al tipo di informazioni ricavate durante la fase di mapping, e a come tali informazioni sono

rappresentate. Il planning deve inoltre tenere conto di alcuni vincoli con cui si vuole raggiungere la destinazione: distanza di sicurezza dagli ostacoli, percorso ottimo, ecc.
Il problema del driving invece consiste nel far eseguire il cammino creato nella fase di planning, con opportune caratteristiche di velocità, accelerazione e accuratezza. Per fare ciò vengono inviati comandi a livello dei singoli attuatori.
Le attività di mapping, planning, e driving possono non essere sequenziali ma essere iterate come per esempio quando si incontra un ostacolo improvviso. In tal caso potrebbe essere necessario rifare il mapping e pianificare una nuova traiettoria.
Nel corso del capitolo verranno approfondite le operazioni effettuate durante la fase di mapping, il tipo di mappe che vengono costruite per la navigazione e verranno introdotte le tecniche di SLAM. Infine verranno presentati e analizzati alcuni prototipi dello stato dell'arte.

2.2. Mapping.

Come già anticipato il problema del mapping è quello di raccogliere informazioni sull'ambiente e fornirne una giusta rappresentazione.
Nel caso dei robot mobili basati su ruote, in grado quindi di navigare in uno spazio 2D, il problema del mapping si riduce a quello di determinare la struttura geometrica dell'ambiente circostante e crearne una mappa bidimensionale.
La creazione della mappa, inizia con la scelta di un sistema di riferimento assoluto in cui rappresentare l'ambiente. Successivamente in questo sistema assoluto, verranno "incollate" le informazioni locali che via via il robot misura durante il suo percorso di esplorazione. La mappa quindi è inizialmente vuota, e viene aggiornata durante il cammino del robot, che deve muoversi in modo da coprire tutto lo spazio circostante che può raggiungere.
In alcuni casi la creazione della mappa, viene fatta a partire da una mappa già caricata nella memoria del robot. In questo caso le operazioni mapping consistono nell'aggiornare la mappa originaria con gli ostacoli dinamici.
In entrambi gli scenari, tutte le operazioni di mapping sono fortemente dipendenti dal tipo di mappa che si sceglie per rappresentare l'ambiente circostante.

Si hanno tre tipi di mappe:

- metriche
- topologiche
- ibride

Nelle mappe metriche vengono memorizzate le informazioni sulla distanza degli oggetti misurati e presentano una struttura che non è altro che un rimpicciolimento della mappa reale vista in 2D. Per costruire una mappa metrica, i dati dei sensori non subiscono grosse elaborazioni. Per contro, tali tipi di mappe contengono molte informazioni, che appesantiscono gli algoritmi che lavorano su di esse.
Le mappe topologiche sono invece un formalismo che rappresentano l'ambiente tramite una struttura a grafo. Ogni nodo del grafo rappresenta un punto di particolare interesse, come landmark artificiali o naturali. La struttura del grafo invece, permette di capire quali punti sono raggiungibili da una determinata posizione.
La mappe topologiche, hanno una rappresentazione compatta però contengono troppe poche informazioni per permettere una corretta navigazione. Invece quelle metriche, danno un sufficiente dettaglio a scapito però di un enorme consumo di memoria.
Le mappe ibride uniscono i vantaggi dei due approcci. Possono presentare una struttura a grafo, dove per ognuno dei nodi esiste un mappa metrica.
La scelta tra le varie mappe viene influenzata dal tipo di ambiente esplorato: ambienti molto strutturati e di grandi dimensioni prediligono rappresentazioni topologiche, mentre per ambienti piccoli e complessi è meglio usare mappe di tipo metrico.
Una volta stabilita la mappa da usare si possono analizzare le operazioni per elaborarla.
Il mapping può essere scomposto in tre sotto problemi:

- costruzione di una mappa locale con le misurazioni direttamente disponibili dai sensori
- aggiornamento della mappa globale in base alle nuove informazioni locali
- calcolo della nuova posizione di esplorazione

Tra questi il secondo sotto problema è spesso molto complicato in quanto l'aggiornamento della mappa viene eseguito simultaneamente alla localizzazione del robot all'interno della mappa globale. Tale problema è noto con il termine di SLAM e varrà affrontato nel paragrafo 2.3.

Per quanto riguarda il problema del calcolo della nuova posizione, ci sono diversi approcci. Negli approcci più semplici si spinge il robot verso le aree della mappa inesplorate senza tenere conto di ulteriori accorgimenti. In tale modo non è sempre garantita la copertura totale della mappa.
Se invece diventa importante trovare il cammino minimo per esplorare l'intero ambiente si usano dei metodi più complicati basati sull "Art gallery Problem". Questo problema consiste nel trovare il numero minimo di telecamere che riescano ad osservare l'intera area della galleria d'arte, che viene rappresentata da un poligono. Il numero minimo di telecamere, corrisponde al numero minimo di scansioni. Nell'ambito del mapping però c'è qualche complicazione in più in quanto tale poligono, non è inizialmente noto, ma viene ottenuto per approssimazioni successive, via via che aumento le scansioni dello spazio circostante. Con queste tecniche è possibile quindi ottenere un cammino ottimo e un copertura totale dell'ambiente che si vuole esplorare.

2.3. SLAM.

SLAM, cioè simultaneous localization and mapping, consiste nel problema di aggiornare la mappa globale con le misure locali contemporaneamente a stimare la sua posizione. Se il robot conoscesse con esattezza la sua posizione, cioè la terna (x, y, θ), potrebbe facilmente aggiornare la mappa: basta sovrapporre le misurazioni della mappa locale a quella globale nella posizione (x,y) con l'orientamento θ. Viceversa se si fosse a conoscenza della mappa precisa di quella regione il robot potrebbe facilmente localizzarsi.
Le difficoltà si presentano quando si tenta di risolvere i problemi di aggiornamento della mappa e di localizzazione, contemporaneamente. Ci sono diversi approcci: ci si può basare direttamente sull'odometria, o utilizzare metodi più precisi come lo *scan matching* e gli approcci probabilistici.
Nelle tecniche basate su odometria gli spostamenti relativi, e quindi la posizione del robot vengono calcolati solo in base al movimento delle ruote. Tale movimento è misurato con dei sensori interni detti encoder, che valutano l'entità dello spostamento angolare e la velocità di questo spostamento. Le misure eseguite con questo metodo si possono ritenere precise solo per piccoli spostamenti, mentre soffrono di notevole imprecisione sulle lunghe distanze. Infatti l'errore si accumula nel tempo all'aumentare

della distanza percorsa. La misura dei sensori odometrici soffre di errori sistematici dovuti a diversi fattori come la risoluzione finita degli encoder, piccole differenze nel diametro delle ruote o il disallineamento degli assi su cui sono montati le ruote. Gli errori non sistematici sono invece dovuti allo scivolamento delle ruote nelle accelerazione, o alle asperità del terreno. E' difficile contenere questi errori, in quanto non sono statisticamente indipendenti nelle diverse misurazioni, ma si accumulano nel tempo.

Per ottenere una maggiore precisione si utilizza il metodo dello scan matching. Questa tecnica consiste nel trovare la regione della mappa globale con cui la mappa locale ha il maggior numero di punti in comune. E' come avere un piccolo rettangolo di un immagine più grande, è bisogna cercare il punto in cui si ha una perfetta sovrapposizione. Tale algoritmi, utilizzano la posizione misurata dai sensori odometrici come stima iniziale, e poi per approssimazioni successive si rototrasla la mappa locale fino a trovare l'incastro perfetto. Questi algoritmi terminano quando si trova il punto che minimizza una certa cifra di merito, basata sul principio della massima verisimiglianza (maximum likelihood).

Una delle maggiori limitazioni di un approccio del genere è che non permette di correggere eventuali errori introdotti in precedenza. Una conseguenza di tale limitazione è che sebbene le mappe prodotte siano localmente consistenti, talvolta non è possibile preservare la consistenza globale.

Un approccio diverso è quello probabilistico. Le tecniche di questo tipo, solitamente basate sul filtro di Kalman, hanno conosciuto un notevole successo a partire dagli anni novanta e vengono impiegate in maniera assai diffusa. Il filtro di Kalman è uno stimatore ottimo di stato che si basa su un modello di errore di tipo gaussiano. L'introduzione del filtro all'interno di un problema di SLAM, porta a inserire nel vettore di stato l'insieme delle posizioni del robot e degli oggetti rilevati attraverso i sensori. Mentre il robot si aggira nell'ambiente effettuando le rilevazioni, il vettore di stato viene via via aggiornato, calcolando di volta in volta la nuova stima dello stato del robot, cioè la posizione del robot e la mappa. Attraverso il filtro di Kalman si mira ad ottenere una mappa precisa che abbia una consistenza globale. Questa precisione si paga con un consumo di risorse computazionali, che diventa molto elevato al crescere alle dimensioni dell'ambiente e del numero di oggetti rilevabili dai sensori.

2.4. Stato dell'arte.

Vengono di seguito analizzati alcuni prototipi dello stato dell'arte. I robot descritti in questi progetti, sono stati realizzati per funzioni diverse: si possono quindi analizzare le varie soluzioni adottate per la navigazione in relazione ai compiti per cui il robot è stato progettato.

Nel primo esempio [12], è stato progettato un robot per il mapping tridimensionale degli ambienti 3D. Tra gli obiettivi di questo progetto vi è la creazione di mappe 3D consistenti, e la ricerca di un cammino minimo per effetture l'esplorazione completa dell'ambiente. Il robot implementato, può muoversi ad una velocità di 0.8 m/s e trasportare un carico di 200 Kg. Esso equipaggiato di due telemetri 2D usati come sensori di prossimità e di un sensore di visione 3D. L'unità elaborativa è un pentium III, dotato di un sistema operativo realtime.
Il sistema di visione usato, è basato su un telemetro 2D esteso tramite un motore servo e riesce a scandire l'intera scena antistante in 3.4 secondi. Il software che gestisce il sistema di visione si occupa di analizzare i punti trovati durante la scansione, per ottenere le superfici e i blocchi che compongono l'ambiente. Per affrontare lo SLAM, viene utilizzato un algoritmo di massima verosimiglianza, in quanto utilizzando solo l'odometria sono stati riscontrati diversi errori. L'algoritmo di massima verisimiglianza adottato è detto ICP, iterative closest point. Si è notato il notevole consumo di risorse computazionali quando tale algoritmo viene applicato a dati in ingresso non preprocessati. Si è quindi deciso di applicare ai punti ottenuti dalle scansioni uno dei vari algoritmi di riduzione: tali tecniche riducono il numero di punti, applicando diversi criteri, come il clustering. In questo modo si è passato da un elaborazione di quasi 4 ore a qualche minuto. L'algoritmo ICP ha complessità quadratica, e una diminuzione dei punti da analizzare comporta un notevole speed up.
Per quanto riguarda la copertura della mappa, si è utilizzato un metodo basato sull' “art gallery problem”, studiando delle soluzioni efficienti applicati ad una mappa parzialmente nota.
I metodi utilizzati per affrontare i problemi di SLAM e di copertura della mappa si sono dimostrati particolarmente efficienti e performanti, mantenendo comunque un consumo di risorse computazionali sostenibile da un computer di media potenza.

Nel secondo e ultimo esempio [11], viene presentato il robot RHINO utilizzato per guidare i visitatori all'interno del Deutsches Museum Bonn. Il software di controllo di RHINO integra ragionamenti probabilistici di basso livello con risolutori di problemi basati sulla logica del primo ordine.
Questo progetto si pone l'interessante sfida di navigare senza pericolo e in modo affidabile in mezzo a dense folle di persone. Per fare ciò il robot è dotato di un software avanzato e di 4 sensori allo stato dell'arte: laser, sonar, infrarosso e tattile.
Il software effettua in real time compiti come l'obstacle avoidance, il mapping, la localizzazione e il path planning. Questi compiti vengono gestiti da 25 processi che vengono eseguiti in parallelo su tre computer al bordo del robot e su tre workstation esterne collegate in wireless. Il software distribuito, che gira su questi sei computer è in grado di adattarsi dinamicamente alle risorse computazionali disponibili.
Il robot RHINO, diversamente dell'esempio precedente [12], ha una completa mappa della porzione di museo in cui dovrà muoversi. Quindi, a partire dai dati ricevuti dai sensori, il robot deve essere in grado di determinare in quale punto della mappa si trova. Questo problema è noto con il nome di localizzazione. RHINO, per localizzarsi, utilizza la versione metrica dell'approccio probabilistico di Markov. Secondo tale approccio, il robot ha una stima probabilistica della sua posizione. All'accensione del robot, si ha un distribuzione uniforme: ogni punto della mappa ha uguale probabilità di essere la posizione iniziale. Tale distribuzione di probabilità viene aggiornata via via con le misure che provengono dai sensori. Il metodo di localizzazione sperimentato è risultato molto efficiente e con un errore massimo inferiore a 15 centimetri.
Per quanto riguarda il mapping, è necessario individuare i cambiamenti del museo rispetto alla mappa precaricata nel robot. Infatti la configurazione del museo, cambia frequentemente, a causa dello spostamento dei numerosi sgabelli utilizzati dai visitatori. Quindi il calcolo delle traiettorie non può basarsi unicamente sulla mappa memorizzata, in quanto ci potrebbero essere degli ostacoli non previsti, che devono essere rilevati tramite mapping. Per il mapping viene usato un approccio molto efficiente basato sulle occupancy grid: in queste griglie ogni cella contiene un valore che indica la probabilità che quella cella contenga un ostacolo.
Nel complesso, i diversi meccanismi adottati per controllare RHINO, hanno permesso di raggiungere ottimi risultati, per quanto riguarda una navigazione sicura in mezzo a numerose persone, e il raggiungimento della destinazione voluta con la massima precisione.

3. Progettazione e implementazione di uno scanner 3D

3.1. Introduzione.

La prima parte di questo lavoro di tesi è rappresentata dalla progettazione e l'implementazione di un sistema di visione che funzioni come un scanner 3D.
L'idea è partita dal volere progettare un robot mobile in grado di effetture il mapping di piccoli e medi ambienti domestici con l'ausilio di un sistema per la percezione delle profondità.
La complessità di tale sistema ha richiesto una fase di studio e di sviluppo separata da quella di progettazione del robot stesso.
Si è quindi creato un dispositivo in grado di effetture scansioni 3D con sufficiente precisione, mantenendo comunque delle piccole dimensioni e un penso non elevato, in vista del suo futuro uso su un robot mobile.
Nella progettazione del dispositivo, tra i vari metodi per la determinazione delle distanze si è scelta la triangolazione basata su luce strutturata. In base anche ai risultati ottenuti in [1] si è deciso di usare un emettitore laser in grado di proiettare una striscia laser orizzontale. In questo modo, è necessario effettuare una scansione monodimensionale lungo l'asse verticale, diversamente da quanto accade con l'utilizzo di un punto laser come in [7] in cui è necessaria una scansione bidimensionale. E' possibile cosi raggiungere una risoluzione orizzontale maggiore e ottenere una scansione più veloce. Quest'ultima risulta essere una caratteristica molto importante soprattutto in vista del successivo uso sul robot mobile.
Per effettuare la scansione verticale si è scelto di montare l'emettitore su un attuatore meccanico in grado di ruotare la striscia laser su tutta la scena da analizzare.

Seguendo l'esempio dei lavori analizzati [1,2,7], è stato utilizzato un sensore ottico di tipo CCD (charge coupled device), cioè un dispositivo che per acquisire il segnale luminoso usa un piccolo chip in silicio nel quale l'ingresso analogico viene trasformato in un'uscita digitale pronta per l'elaborazione tramite computer.
La risoluzione orizzontale della scansione quindi sarà limitata solamente dalla più piccola distanza rilevabile dal dispositivo CCD che è stato usato. Mentre per quanto riguarda la risoluzione verticale dipenderà dall'ampiezza dell'angolo di rotazione dell'attuatore meccanico.
Nell'implementazione del sistema si sono scelti componenti di media qualità, facilmente reperibili in commercio a costo contenuto, come è stato fatto in [2], in modo da paragonare i risultati ottenuti con [1] nel quale invece sono stati utilizzati componenti di elevata qualità e maggiore precisione.
Per calcolare la distanza dei punti della scena osservata si sfrutta la triangolazione: infatti la proiezione del laser genererà diverse geometrie a seconda di come sono disposti gli oggetti osservati. Tale procedimento verrà esposto dettagliatamente nel secondo paragrafo dove verrà presentato il modello teorico su cui ci si è basati per calcolare le distanze. Nel paragrafo 3.3 viene descritto il passaggio dal modello all'implementazione reale del sistema. Quindi i componenti usati, l'interfacciamento con il computer e il funzionamento dell'intero sistema. In particolare nel paragrafo 3.4 verrà esaminata l'elettronica e la meccanica del sistema, e nel paragrafo 3.5 verrano presentate le tecniche adottate per analizzare le immagini e rilevare il profilo del laser. La parte software che gestisce il sistema, viene esposta nel paragrafo 3.6. Infine i risultati ottenuti con il sistema di visione verrano poi analizzati e commentati nel capitolo 5.

3.2. Modello teorico del sistema di visione.

In questo paragrafo viene spiegato dettagliatamente il modello teorico alla base del funzionamento del sistema di visione. Come già anticipato in precedenza, i componenti del sistema sono:

- un attuatore meccanico
- una sorgente laser (emettitore)
- un sensore ottico CCD

Il sistema di riferimento, nel quale saranno espresse tutte le distanze, è posto nel centro ottico del sensore CCD. In particolare gli assi *X* e *Y* giacciono sul piano immagine, mentre l'asse *Z* è uscente. Accanto al sensore ottico, è posizionato l'attuatore meccanico. Questo attuatore meccanico comanda la rotazione di un asse al quale verrà ancorata la sorgente laser: in questo modo sarà possibile ruotare a piacimento l'emettitore. L'attuatore meccanico viene posizionato in modo che il suo asse sia parallelo all'asse *X* del sistema di riferimento. Tale accorgimento permette di risparmiare un parametro in quella che successivamente sarà la rappresentazione vettoriale del sistema.

La sorgente laser proietta nella scena antistante un segmento luminoso. Unendo gli estremi del segmento proiettato con l'origine dell'emettitore si trova un triangolo i cui vertici sono contenuti in un piano che verrà indicato con il simbolo *P*. La sorgente laser è posizionata in modo che il piano *P* passi per l'asse dell'attuatore meccanico, per ogni valore dell'angolo di rotazione. Ciò permette di semplificare il calcolo dell'angolo tra il piano *P* e il piano Z*X*. Quando il piano *P* è parallelo al piano Z*X*, si assume che l'angolo θ comandato dall'attuatore è zero.

Nel sensore CCD, i punti della scena vengono proiettati nel suo piano immagine. La coordinata 3D di un punto diventa una coordinata 2D. La trasformazione prospettica di un punto della scena nel suo punto immagine rilevato dal sensore si ottiene facendo l'intersezione tra il piano immagine e la retta passante per il punto 3D e il centro ottico (Figura 3.1). Tale trasformazione prospettica è valida in assenza di distorsioni, cioè basandosi sul modello ottico del sensore CCD, e assumendo che gli oggetti rilevati siano ad una distanza maggiore della distanza focale. E' possibile notare come tale trasformazione non sia biunivoca: diversi punti nella scena hanno lo stesso punto immagine. Ciò accade in quanto vengono perse le informazioni sulla distanza.

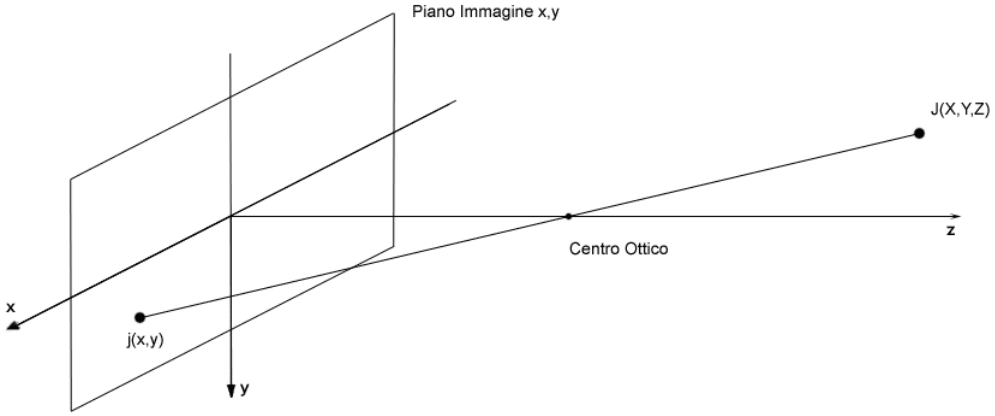


Figura 3.1: Trasformazione prospettica di un punto 3D, nella sua immagine

Nel sensore CCD, viene effettuata una discretizzazione: il piano immagine è diviso in tante piccole celle, detti pixel. Due punti nella spazio 3D che si trovano tra loro ad una distanza sufficientemente piccola vengono interpretati come un unico punto:quindi un pixel è in corrispondenza con un insieme di punti della scena osservata. Nel piano immagine di un sensore CCD ogni pixel viene indicato da una coppia di numeri naturali che rappresentano le loro coordinate discrete: moltiplicando tali coordinate per le dimensioni del pixel è possibile risalire alle coordinate reali rispetto allo stesso sistema di riferimento.

Ad ogni pixel è possibile associare un vettore $\vec{C}_n$. Tale vettore ha origine nel centro ottico del sensore, e termina nel centro del pixel. Per determinare le coordinate di $\vec{C}_n$ è utile fare riferimento ad una rappresentazione particolare del modello ottico del sensore CCD. Tale rappresentazione si ottiene anteponendo il piano immagine al centro ottico, ad una distanza pari alla distanza focale (Figura 3.2).

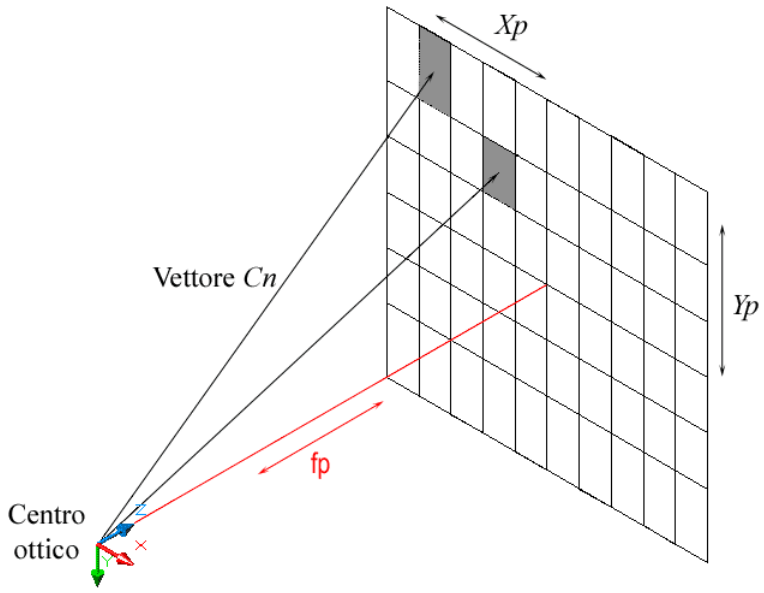


Figura 3.2: Rappresentazione alternativa del modello ottico del CCD

Utilizzando tale rappresentazione si vede facilmente che per ogni pixel il suo vettore ha coordinate: $\vec{C}_n = \begin{pmatrix} X_p \\ Y_p \\ f_p \end{pmatrix}$

dove f_p è la distanza focale espressa in pixel, e X_p e Y_p sono le coordinate del pixel nel sistema di riferimento *R*, definito in Figura 3.3. In genere il sensore ottico restituisce le coordinate di ogni pixel in base al sistema di riferimento standard posizionato nell'angolo in alto a sinistra del piano immagine. Il sistema di riferimento *R*, è invece posto al centro del piano immagine.

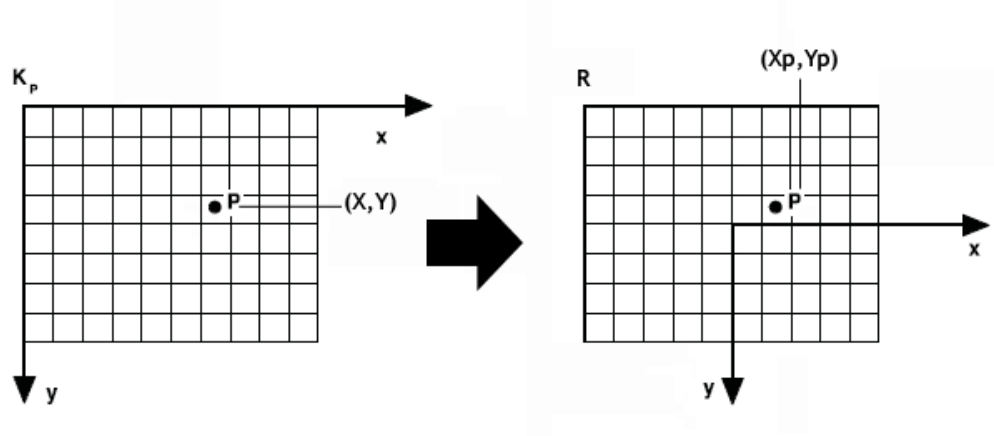


Figura 3.3: Sistema di riferimento standard e sistema di riferimento R

Date le coordinate *x* e *y* di un pixel per ottenere X_p e Y_p in *R* basterà calcolare:

$$X_P = x - \frac{wcam}{2} - 1/2$$

$$Y_P = y - \frac{hcam}{2} - 1/2$$

Dove *wcam* e *hcam* sono rispettivamente la lunghezza e l'altezza in pixel del piano immagine. Il fattore 1/2 viene sottratto per far puntare i vettori al centro del pixel. I vettori $\vec{C}_n$ sono solidali con il sistema di riferimento. Anche il vettore lungo l'asse di rotazione dell'attuatore è solidale con il sistema di riferimento.

Per calcolare la distanza degli oggetti dal sensore ottico è necessario introdurre un altro vettore: $\vec{n}$. Questo vettore è sempre perpendicolare all'asse di rotazione ed è contenuto nel piano *P* definito in precedenza. A differenza degli altri vettori definiti in precedenza, $\vec{n}$ varia al variare di θ. Indicando con $\vec{k}$ il versore dell'asse *Z*, le coordinate di $\vec{n}$ si ottengono come:

$$\vec{n} = rot(X,\theta)\cdot\vec{k} \qquad \vec{n} = \begin{bmatrix} 1 & 0 & 0 \\ 0 & \cos\theta & -\sin\theta \\ 0 & \sin\theta & \cos\theta \end{bmatrix} \cdot \begin{bmatrix} 0 \\ 0 \\ 1 \end{bmatrix} = \begin{bmatrix} 0 \\ -\sin\theta \\ \cos\theta \end{bmatrix}$$

Infatti per $\theta=0$, $\vec{n}$ è parallelo all'asse *Z*. Le successive rotazioni vengono eseguite attorno all'asse *X* parallelo all'asse di rotazione dell'attuatore meccanico.
Infine definiamo con $\vec{d}$ il vettore che congiunge il centro ottico del sensore con l'asse di rotazione nel punto in cui vi è ancorata la sorgente laser.
I vettori $\vec{C}_n, \vec{n}, \vec{d}$ e gli assi del sistema di riferimento posto nel centro ottico del sensore, definisco in modo completo la geometria del sistema. Si può quindi esprimere la relazione che permette di trovare la distanza di un punto 3D illuminato dal laser:

$$\gamma \vec{C}_n = \vec{d} + \alpha \vec{i} + \beta \vec{n}$$

Dove (α, β, γ) sono degli opportuni coefficienti. Questa relazione si ricava da semplici considerazioni geometriche sul sistema. Si consideri un punto *J* sulla scena 3D. Si supponga che tale punto sia illuminato dal laser per un certo angolo che indicheremo con θ. Dato θ è immediato esprimere $\vec{n}$. Si supponga altresì che questo punto *J* sia inquadrato dal sensore ottico e siano (*x*,*y*) le sue coordinate dalle quale è immediato ricavare X_p , Y_p e quindi $\vec{C}_n$.

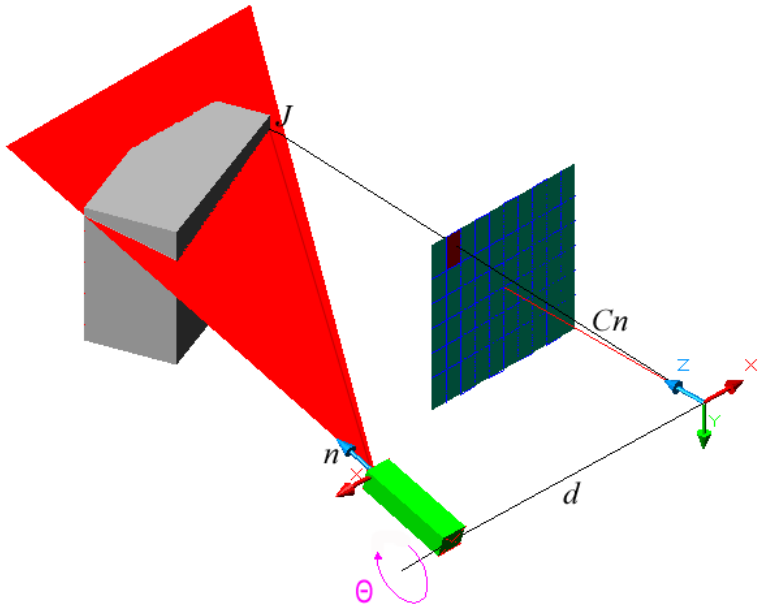


Figura 3.4: Modello geometrico del sistema di visione

Conoscere $\vec{C}_n$ significa conoscere la direzione del raggio che dal punto 3D giunge nel centro ottico: quindi il vettore $\vec{C}_n$ moltiplicato per γ rappresenta il vettore che congiunge centro ottico e il punto *J*. Essendo f_p la componente lungo *Z* di $\vec{C}_n$, si

ha che $f_p \cdot \gamma$ rappresenta proprio la distanza di *J* dal piano immagine. Per impostare il sistema lineare, si eguaglia $\gamma \vec{C}_n$ ad un altro vettore. Il fatto che *J* è illuminato dal laser, implica che *J* si trovi sul piano *P*. Tutti i punti del piano *P* possono essere espressi come combinazione lineare dei vettori $\vec{n}$ e $\vec{i}$, che per come sono stati definiti sono giacciono su *P*. Dati due coefficienti *α e β* si ottiene il vettore che congiunge l'emettitore con il punto *J*. Sommando a quest'ultimo vettore $\vec{d}$ si riottiene il vettore $\gamma \vec{C}_n$.
Si è cosi ottenuto un sistema di 3 equazioni in 3 incognite (*α, β, γ)* facilmente risolubile con il metodo di Cramer.

$$-\alpha \vec{x} - \beta \vec{n} + \gamma C_n = \vec{d}$$

$$-\alpha \cdot \begin{bmatrix} 1 \\ 0 \\ 0 \end{bmatrix} - \beta \cdot \begin{bmatrix} 0 \\ -\sin\theta \\ \cos\theta \end{bmatrix} + \gamma \begin{bmatrix} X_p \\ Y_p \\ f_p \end{bmatrix} = \begin{bmatrix} d_x \\ d_y \\ d_z \end{bmatrix}$$

$$\begin{bmatrix} -1 & 0 & X_p \\ 0 & \sin\theta & Y_p \\ 0 & -\cos\theta & f_p \end{bmatrix} \cdot \begin{bmatrix} \alpha \\ \beta \\ \gamma \end{bmatrix} = \begin{bmatrix} d_x \\ d_y \\ d_z \end{bmatrix}$$

$$\Delta = det \begin{pmatrix} -1 & 0 & X_p \\ 0 & \sin\theta & Y_p \\ 0 & -\cos\theta & f_p \end{pmatrix} = -(f_p \sin\theta + Y_p \cos\theta)$$

$$\begin{bmatrix} \alpha \\ \beta \\ \gamma \end{bmatrix} = \frac{1}{\Delta} \cdot \begin{bmatrix} -\Delta & -X_p \cos\theta & -X_p sint\,\theta \\ 0 & f_p & Y_p \\ 0 & -\cos\theta & -\sin\theta \end{bmatrix} \cdot \begin{bmatrix} d_x \\ d_y \\ d_z \end{bmatrix}$$

$$\gamma = \frac{(-d_y \cos\theta - d_z \sin\theta)}{\Delta} = \frac{(-d_y \cos\theta - d_z \sin\theta)}{-(f_p \sin\theta + Y_p \cos\theta)}$$

Infine la distanza del punto 3D dal piano immagine è:

$$dist = (\vec{C}_n)_z \cdot \gamma = f_p \frac{(-d_y \cos\theta - d_z \sin\theta)}{-(f_p \sin\theta + Y_p \cos\theta)}$$

Se *dz* e *dy* vengono espressi in centimetri, e *fp* e *Yp* in pixel, si trova che anche *dist* è espresso in centimetri ed è quindi indipendente dalla dimensione dei pixel del sensore CCD:

$$dist = pixel \frac{cm}{pixel} = cm$$

Per trovare *fp*, distanza focale del sensore ottico, non è possibile un misura diretta: ma è necessario trovare tale parametro mediante triangolazione. Basta infatti porre un oggetto parallelamente al piano immagine come in Figura 3.5.

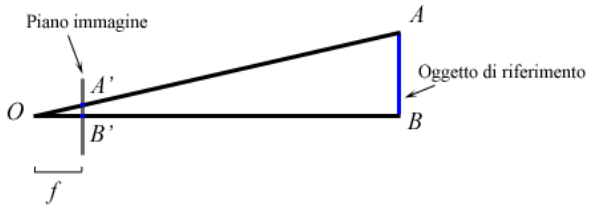


Figura 3.5: Modello geometrico del sistema di visione

Dati la distanza dell'oggetto dal piano immagine e la distanza tra due vertici A e B disposti orizzontalmente lungo l'oggetto è possibile impostare la proporzione dati i triangoli simili:

A'B' : OB' = AB : OB

dove A' e B' sono rispettivamente le proiezioni di A e B sul piano immagine, e OB' è uguale a *fp*.

*fp= A'B' * OB/ AB =valore di fp espresso in pixel.*

Come si può vedere il valore di *fp* trovato è espresso in pixel: ciò non comporta nessun problema, in quanto nel calcolo della distanza il valore di pixel si semplifica con le altre grandezze espresse nella medesima unità di misura.

E' importante notare che la precedente formula introdotta per il calcolo della funzione *dist*, di un punto *J* è valida se e solo se il determinante Δ è diverso da zero.

Se $\Delta = 0$ si ha che: $\begin{array}{c} -(f_p \sin\theta + Y_p \cos\theta) = 0 \\ f_p \sin\theta = -Y_p \cos\theta \end{array}$

Si può dividere ambo i membri dell'uguaglianza per $\cos\theta$ sicuramente diverso da 0: infatti quando $\cos\theta = 0$ si ha che $\Delta = \pm f_p$.

Si ottiene che $tg\theta = \frac{-Y_p}{f_p}$. Mentre si ha che $\frac{-Y_p}{f_p} = tg\alpha$ dove α è l'angolo del vettore $\vec{C}_n$ sul piano *XZ*. Praticamente il determinate è uguale a zero quando il vettore $\vec{C}_n$ è sul piano *P* del laser. Infatti in questa configurazione l'oggetto potrebbe essere ad una distanza qualsiasi dal centro ottico: esistono infinite soluzioni a problema.

3.3. Implementazione.

L'implementazione del sistema di visione si basa sul modello teorico sopracitato. Come sensore CCD è stata utilizzata una normalissima webcam USB, in grado di acquisire fotogrammi a colori ad una risoluzione di 352*288 pixel.

Come attuatore meccanico si è scelto di utilizzare un motore passo passo. Il motore passo passo, è un motore comandabile in anello aperto, che tramite un sequenza di impulsi produce una rotazione sempre di uno stesso angolo. Il motore passo passo verrà descritto con maggiore dettaglio nel paragrafo 3.4.

La sorgente laser è un piccolo fotodiodo, la cui luce passa attraverso un ottica capace di generare un segmento. Il fotodiodo è realizzato tramite semiconduttore ed emette una luce rossa con lunghezza d'onda pari a 650 *nm*. Il segmento laser viene proiettato con ampiezza di 90°, e le condizioni operative per un corretto funzionamento sono di 3 *V*, con una corrente di 45 *mA*.

L'intero dispositivo viene interfacciato al computer tramite porta parallela: l'utilizzo di una porta di tipo parallelo rispetto ad una seriale permette di semplificare moltissimo l'elettronica presente sul dispositivo.

Per comandare il sistema si è scelto di utilizzare il software Matlab, sia per la sua versatilità, sia per le notevoli funzioni che mette a disposizione per l'analisi e i filtraggi delle immagini.
Quindi tramite Matlab, vengono comandati i bit sulla parallela che tramite l'elettronica presente sul dispositivo fanno ruotare il motore passo passo e sempre tramite Matlab vengono acquisite le immagini dalla webcam USB, come è possibile vedere nello schema in Figura 3.6.

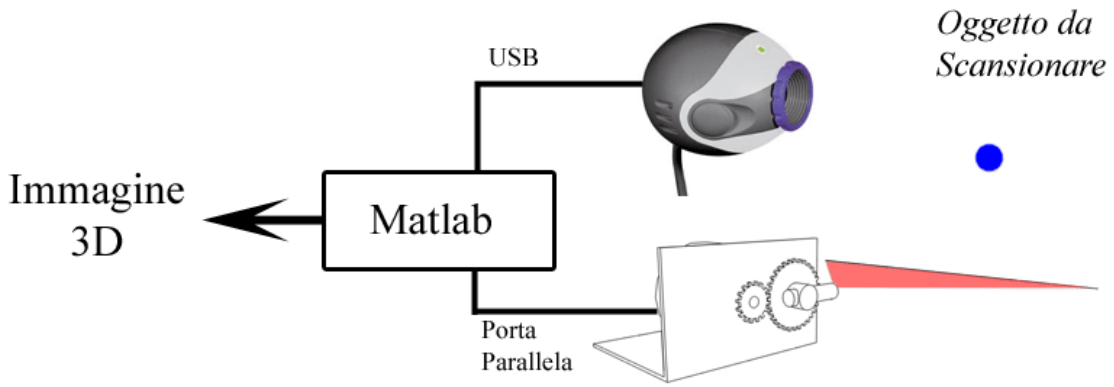


Figura 3.6: Schema generale del sistema di visione

Più in dettaglio, i bit utilizzati per comandare il motore, vengono mappati sui pin che vanno dal 2 al 5, come è possibile vedere nello schema della porta parallela riportato in Figura 3.7.

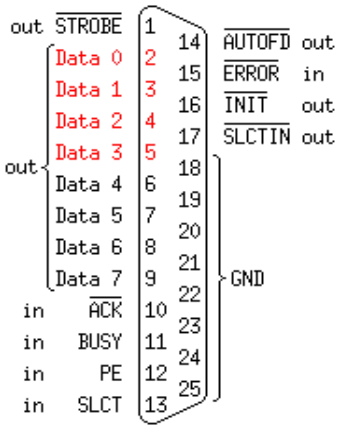


Figura 3.7: Piedinatura della porta parallela. In rosso i piedini usati per comandare il motore passo passo

Le operazioni Matlab necessarie per elaborare i dati in Input, comandare la rotazione del laser e creare l'immagine 3D, sono contenute nello script “main.m”.

```
MAIN( )
Inizializzazione variabili e dichiarazione costanti;
Inizializzazione della comunicazione sulla porta parallela;
For (numero di passi della scansione) {
      Rotazione del motore passo passo;
      Calcolo del nuovo angolo teta;
      Acquisizione di un fotogramma delle scena;
      For (per ogni colonna del fotogramma){
            Ricerca dei pixel illuminati dal laser;
            Calcolo della distanza del pixel;
            Aggiornamento della matrice delle distanze;
      }
}
Stampa a video della matrice delle distanze;
END.
```

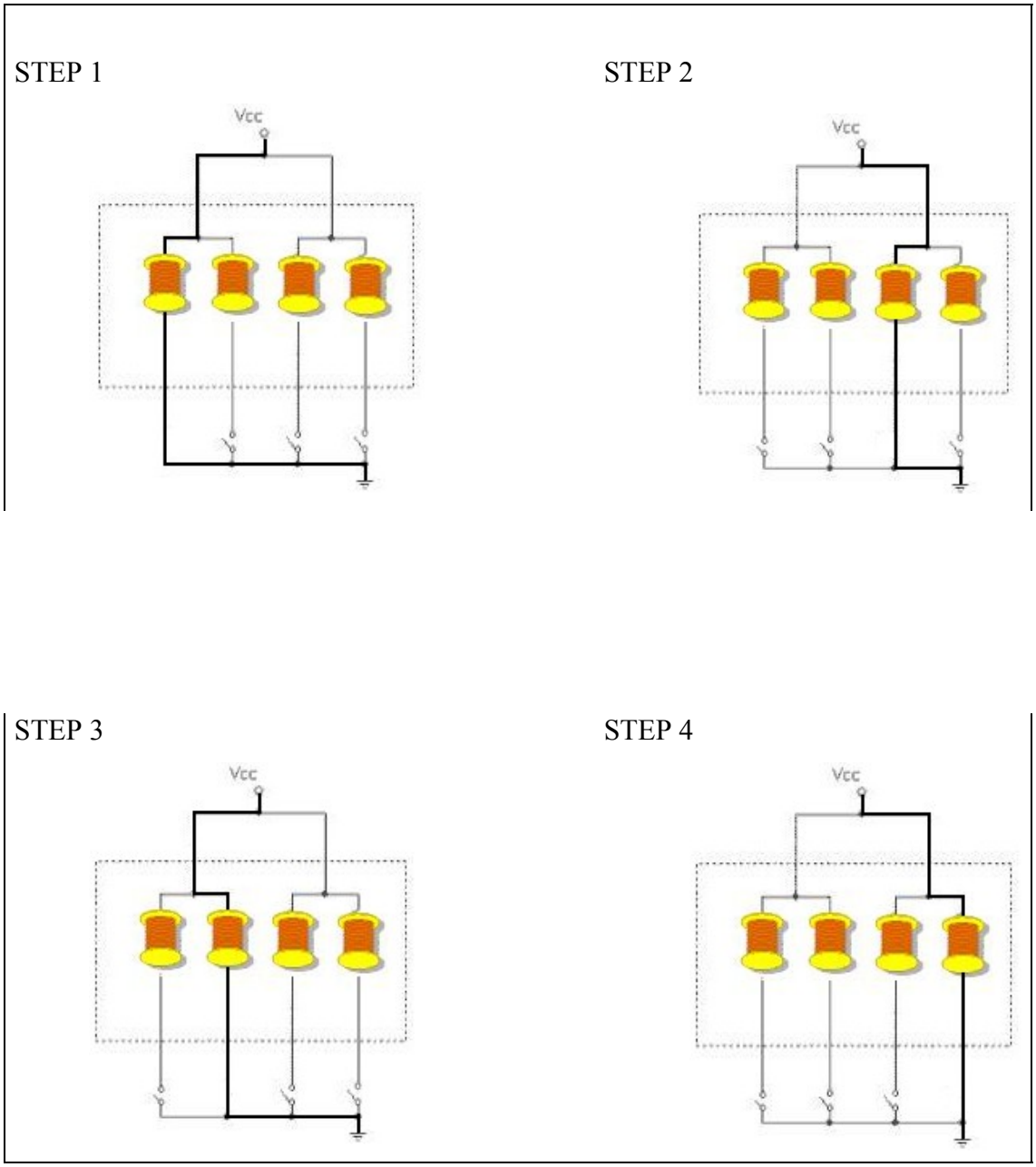
Come si può vedere, l'algoritmo è un semplice ciclo *For*, nel quale per ogni passo della scansione, viene analizzata l'immagine alla ricerca dei punti illuminati dal laser. L'analisi dell'immagine verrà descritta con maggiore dettaglio successivamente. Dopo aver individuato il pixel illuminato se ne determina la distanza con la formula ricavata nel paragrafo 3.2 dove viene esposto il modello teorico.
Successivamente viene aggiornata la matrice delle distanze (depth map). Tale matrice ha le dimensioni del fotogramma: 352*288, dove ogni cella contiene -1, se il pixel corrispondente non è mai stato illuminato dal laser durante l'intera scansione, altrimenti contiene un valore che rappresenta la distanza in centimetri.
L'insieme di celle con valori pari a -1, rappresenta le zone tra le diverse proiezioni del laser: sono delle zone d'ombra sulle quali non si hanno informazioni sulla distanza. La grandezza di tali zone diminuisce al diminuire dell'angolo di avanzamento del motore. Per un angolo abbastanza piccolo si può supporre che la distanza di quelle zone sia simile ai pixel adiacenti e si può quindi interpolare.
Alla fine dell'interpolazione si ottiene un vero e proprio modello geometrico della scena osservata o dell'oggetto scansionato. Se ci interessa utilizzare principalmente questo dispositivo come scanner 3D, cioè per ottenere modelli geometrici dettagliati di oggetti si potrebbe migliorare il software di acquisizione. Infatti per ottenere un modello

completo di un oggetto è necessario scansionarlo da più angolazioni (retro, altro, basso,..) utilizzando degli algoritmi di merging, tutt'altro che banali, che permetterebbero di unire tutte le scansioni in unico insieme di poligoni e ottenere un modello CAD dell'oggetto.
Per evitare di spostare l'oggetto in modo da ottenere scansioni da diverse angolazioni, si potrebbe posizionarlo su un supporto rotante e tenere laser e webcam fermi, come realizzato in [2].

3.4. Meccanica ed elettronica del sistema.

Il motore utilizzato nel sistema di visione è un motore di tipo passo passo. Questo tipo di motore oltre a essere caratterizzato da un elevata robustezza sia meccanica che elettrica, ha il grosso vantaggio di poter essere controllato in anello aperto. Il controllo in anello aperto è vantaggioso in quanto permette di risparmiare potenza di calcolo e i sensori di posizione o di velocità sul motore.
Tale motore è composto da due avvolgimenti dotati di presa centrale alla quale viene collegata l'alimentazione Vcc. Una volta alimentato, l'asse del motore viene tenuto fermo in una condizione di equilibrio. Per ottenere il movimento del motore bisogna alternare il passaggio della corrente negli avvolgimenti. Ogni movimento, detto passo, l'asse del motore ruota di un angolo fissato. Per ottenere un sequenza di passi in senso antiorario, si devono alimentare le fasi in una sequenza ben precisa (Figura 3.8).

Figura 3.8: Principio di funzionamento del motore passo passo (in grassetto il circuito chiuso per ogni step)



Nella pratica gli interruttori in Figura 3.8 sono stati sostituiti con dei transistor mos, i quali verranno comandati dalla porta parallela del computer attraverso dei fotoaccoppiatori, che impediranno eventuali danneggiamenti alla scheda madre del computer.

Nella Figura 3.9 è riportato lo schema circuitale per il pilotaggio del motore passo passo:

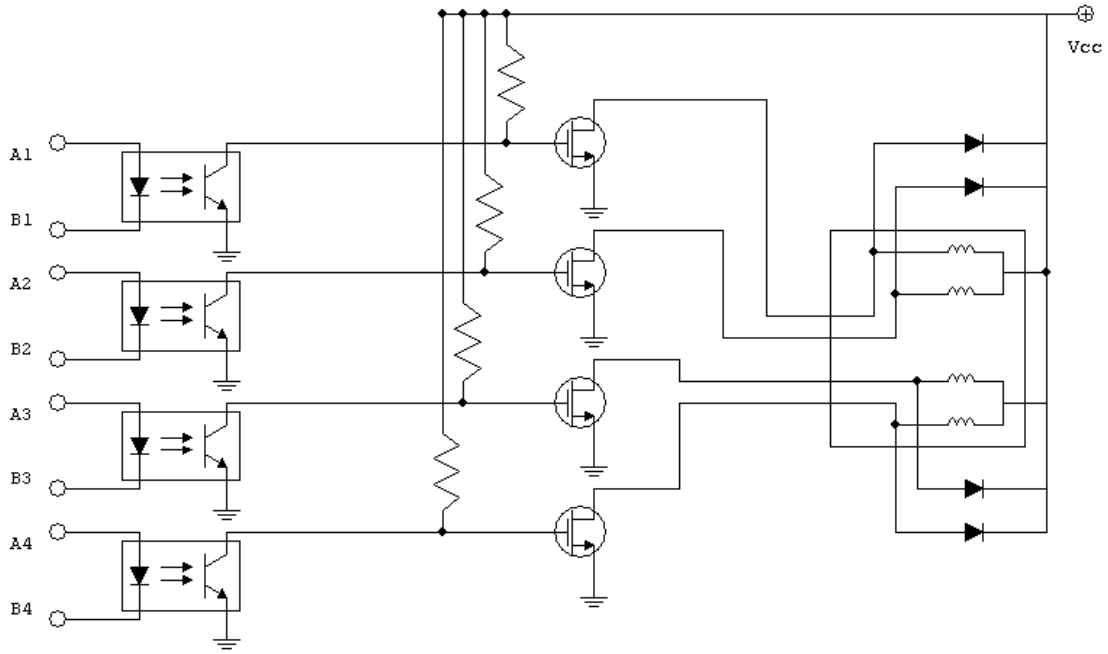


Figura 3.9: Circuito per il pilotaggio del motore passo passo

La presenza di un "segnale alto" al gate del transistor, lo fa accendere e in termini pratici chiude il circuito alimentando l'avvolgimento corrispondente.
Nello specifico sono stati utilizzati i fotoaccoppiatori 6N135 della Fairchild Semiconductor, i power mosfet IRF540 della Sgs-Thomson Microelectronics, resistenze da 6 kΩ e diodi 1N4001, sulla base del circuito creato in [7].
I pin A1, A2, A3, A4, vengono mappati direttamente con i segnali Data0, Data1, Data2, Data3 della parallela. Mentre i pin B1, B2, B3, B4 devono essere corto circuitati con il GND (massa) sempre della porta parallela.
È importante ricordare che un circuito così costruito e cioè un segnale digitale in ingresso a fotoaccoppiatori che pilotano il gate di un mosfet di potenza che a sua volta funge da interruttore ad un carico come un motore passo-passo, non è in grado di funzionare correttamente senza i diodi che sono stati inseriti in parallelo agli induttori.
Infatti quando si pilotano carichi induttivi, come appunto lo sono questi motori, è sempre necessario inserire il cosiddetto "diodo di ricircolo", pena la repentina distruzione del transistor a causa delle elevate tensioni generate dal carico pilotato.

Per essere più precisi ciascun avvolgimento del motore è sostanzialmente un induttore, perciò tende a mantenere costante la corrente che in esso scorre.
Quando un transistor si apre l'induttore tende a compensare la repentina diminuzione di corrente e per questo la tensione sul collettore del mosfet sale pericolosamente fino a danneggiare il componente (tensione di fly-back).
Per evitare questo problema è stato inserito in parallelo alla bobina del motore un diodo che fornisce alla corrente una via alternativa a quella del transistor nel momento in cui questo si apre.
Il circuito durante i test iniziale è stato sempre alimentato a 7,5 volt. Successivamente per evitare l'ingombro dovuto all'utilizzo di un alimentatore esterno, si è scelto di alimentare il circuito con una batteria alcalina da 9 volt.
Il circuito di potenza per il motore passo passo, non presenta nessun tipo di problema quando si passa da una Vcc di 7,5 volt, a una di 9 volt.
L'unico componente problematico è il diodo laser: una corrente superiore ai 45 *mA* potrebbe causarne la rottura. E' quindi necessario ridimensionare la resistenza dai 100 Ω iniziali a 140 Ω.

La tabella di verità del circuito vista dai gate dei transistor deve quindi essere:

Tab. 1

	Gate Mos 1	Gate Mos 2	Gate Mos 3	Gate Mos 4
Stato 1	1	0	0	0
Stato 2	0	0	1	0
Stato 3	0	1	0	0
Stato4	0	0	0	1

Il simbolo 1 corrisponde a Vcc e il simbolo 0 corrisponde agli 0V.

Per ottenere una rotazione continua del motore basta semplicemente mettere in ciclo gli stati sempre rispettando la tabella di verità sopra indicata.
Con questa sequenza di stati, l'albero del motore girerà in senso antiorario. Per ottenere una rotazione oraria, basterà percorrere la tabella di verità dal basso verso l'alto cioè dallo stato 4 allo stato 1.
Una semplice analisi circuitale (Figura 3.9) mette in evidenza che la presenza di un "livello alto" in uscita dalla parallela accende il transistor all'interno del

fotoaccoppiatore, mandando la tensione del gate del mos a 0 (più precisamente alla tensione tra collettore e emettitore del transistor a monte).
Viceversa un "livello basso" sulla parallela non accende il transistor del fotoaccoppiatore e quindi la tensione sul gate del mos diventa Vcc.
Dunque si nota che la presenza del fotoaccoppiatore, a livello logico, funziona da inverter. Quindi per pilotare il motore come si voleva in precedenza, bisogna ricavare il complemento della Tab.1.
La sequenza di bit da inviare sulla parallela a questo punto diventa:

Tab.2

	Bit 1	Bit 2	Bit 3	Bit 4
Stato 1	0	1	1	1
Stato 2	1	1	0	1
Stato 3	1	0	1	1
Stato4	1	1	1	0

Attraverso delle riduzioni interne, la ruota dentata esterna al motore ha una risoluzione di 7,5°. A questo punto, sorge un problema: una risoluzione così ampia, anche a distanze non troppo elevate, potrebbe portare alla perdita di informazioni importanti relative all'oggetto che si sta scansionando o addirittura alla "perdita" (nella scansione) dell'oggetto stesso.
Questi motivi ci ha portato a cercare delle soluzioni per aumentare la risoluzione, diminuendo l'angolo di passo.
Un primo modo è quello di inserire nella tabella di verità degli stati intermedi tra quelli già presenti in modo da realizzare con ogni passaggio di stato, un mezzo passo al posto del passo intero.
Il mezzo passo si realizza alimentando più parti di avvolgimento in contemporanea.
Qui sotto viene riportata una tabella che serve d'esempio:

Tab.3

	Fase 1	Fase 2	Fase 3	Fase 4
Stato 1	1	0	0	0
Stato 1-2	**1**	0	**1**	0
Stato 2	0	0	1	0

Per convenzione, i bit a "1" equivalgono alle fasi alimentate

Alternativamente si può applicare una riduzione alla ruota dentata esterna (cioè un'altra ruota dentata ma di diametro più grande) che permette di diminuire l'angolo di passo in modo proporzionale al diametro della riduzione applicata.

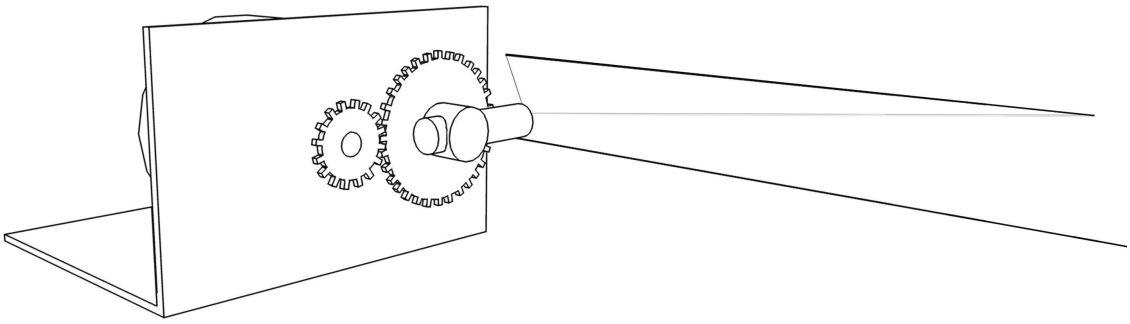


Figura 3.10: Sistema di riduzione per aumentare la risoluzione angolare del motore

Tra le due soluzioni è stata utilizzata la più efficiente, cioè quella di applicare una riduzione (Figura 3.10). Ciò ci ha permesso di ottenere una risoluzione di circa 1,5° molto inferiore a quella di partenza.

3.5. Acquisizione, filtraggio e analisi dei fotogrammi.

L'acquisizione, il filtraggio e l'analisi delle immagini acquisite dalla webcam, vengono effettuate tramite Matlab. Matlab supporta l'acquisizione di immagine da webcam USB, tramite la libreria DLL (vfm.dll) scritta da:
Farzad Pezeshkpour, School of Information Systems, University of East Anglia.
L'acquisizione di un singolo fotogramma della webcam avviene semplicemente tramite il comando:

```
img0=double(VFM('grab',1));
```

In questo modo viene creata una matrice 352*288*3 dove per ogni cella vi è una tripla di interi da 0 a 255 che rappresentano il valore RGB del pixel corrispondente.
Una volta acquisita l'immagine si può lavorare su di essa come su una comunissima matrice.

Per alleggerire le operazioni, è preferibile lavorare con il solo livello rosso:

imgR=img0(:,:,1);

Dopo l'acquisizione dell'immagine il problema principale è quello di analizzarla alla ricerca dei punti illuminati dal laser. Un modo di procedere potrebbe essere quello di cercare i bordi tracciati dal segmento laser, tramite algoritmi di edge-detection che permettono di evidenziare la variazione di luminosità tra un punto ed un altro. Tali algoritmi sono però influenzati dalla presenza di rumore sull'immagine acquisita. Il rumore ha molteplici origini: principalmente è dovuto al rumore termico che si ha in tutta l'elettronica del CCD, al quale si sommano gli errori di quantizzazione nel trasformare un segnale reale in uno discreto e altri disturbi (nell'ambiente durante l'acquisizione). Tenendo conto del rumore, l'immagine misurata J può quindi essere scomposta in due componenti:

$$J(x,y)=I(x,y)+N(x,y)$$

dove $N(x,y) \sim WN(0,\sigma^2)$ e I(x,y) è l'immagine reale.

N(x,y) si assume sia un rumore bianco, a media nulla e varianza sigma.
Per ridurre l'influenza del rumore sull'analisi delle immagini è necessario utilizzare degli opportuni filtri detti di "smoothness". Tali filtri permettono di eliminare il rumore dalle immagini ma come effetto collaterale schiariscono i bordi, creando problemi per l'edge detection.
Quindi è necessario un giusto trade off tra il filtro da applicare per pulire l'immagine e l'algoritmo necessario per individuare i pixel illuminati dal laser. Infatti tarando male i parametri del filtro di smoothness si potrebbe avere il caso in cui proprio i pixel illuminati vengano interpretati come rumore.
Un esempio di filtro di smoothness è il filtro media, che sostituisce il valore di ogni pixel con la media dei valori degli 8 pixel attorno.
Uno dei principali problemi del filtro media è l'attenuazione di bordi e altri dettagli delle forme. Questi problemi possono essere risolti significativamente con il filtro Mediana. In questo caso l'uso di Matlab è risultato molto comodo in quanto presenta al suo interno gli algoritmi di molti filtri.

Per il filtro mediana:

```
imgR=medfilt2(imgR);
```

Per quanto riguarda l'analisi delle immagini al fine di trovare i pixel illuminati dal laser sono state effettuate diverse prove sperimentali. Alla fine si è ottenuto un semplice metodo, che anche non utilizzando algoritmi di edge-detection ha dato ottimi risultati nei casi di test in cui è stato utilizzato. Il metodo consiste nel catturare un immagine (img0) della scena a laser spento, o con il laser inclinato tanto da non essere inquadrato dalla webcam. Dopo aver spostato il laser sulla scena viene acquisita una nuova immagine (img1). Entrambe le immagini vengono filtrate con il filtro mediano. Dopo aver tolto il rumore si è notato che nelle zone non illuminate dal laser le due immagini sono praticamente identiche con fluttuazioni intorno al valore 20, nella scala RGB da 0 a 255. Nelle zone illuminate dal laser, invece, si ha una notevole differenza tra i valori di img0 e img1. Per estrarre i pixel illuminati dal laser si calcola quindi la matrice Delta, che non è altro che la differenza tra img1 e img0, e si considerano buoni solo quei punti con valore maggiori di una certa soglia. Tale soglia dopo diversi tentativi è stata stimata a 40 unità. Infine sui punti trovati viene applicata un ulteriore selezione basandosi sul fatto che il segmento proiettato ha direzione orizzontale. Quindi scandendo l'immagine per colonne si prendono in considerazione i gruppetti adiacenti composti da almeno 2 pixel e tra questi si prende il più grande. Il fatto di prendere in considerazione solo i gruppi di almeno 2 pixel è un ulteriore selezione sul rumore. Infatti il rumore è un fenomeno puntiforme è la probabilità che due pixel adiacenti abbiano un elevato valore di rosso a causa del rumore è bassissima.
Per quanto riguarda il costo computazionale per analizzare l'immagine si può dire che la maggior parte è dovuto all'applicazione dei filtri mediani alle 2 immagini, in quanto l'algoritmo per trovare i pixel illuminati è lineare. Nonostante tutto il costo non è notevole in quanto il filtro di Matlab è ottimizzato per buone prestazioni.
Infine è importante aggiungere che un sistema di visione basato su luce laser può essere affetto da diversi problemi.
Si ha difficoltà a rilevare il segmento laser su alcuni materiali, che riflettono poco e male la luce del laser. Diversi problemi si hanno con gli oggetti trasparenti che oltre a riflettere debolmente la luce del laser proiettano il fascio sul resto della scena in modo diverso dal normale alterando notevolmente il calcolo delle distanze.

E preferibile scandire la scena in condizioni di luce diffusa: i raggi solari diretti possono essere confusi con la luce laser. In tal caso è necessario analizzare anche le componenti blu e verde. Infatti la luce del laser presenta dei picchi solo nel livello del rosso, mentre la luce solare presente dei picchi in tutti e tre i livelli. Analizzando i tre livelli è possibile discriminare da quale luce è illuminato il pixel. Ciò non risolve i problemi dovuti alla luce solare, in quando una diretta esposizione presenta notevoli fluttuazioni tra un immagine e un'altra.
Le condizioni di scarsa illuminazione si sono mostrate adatte, per queste scansioni, mentre il buio totale al contrario di quanto si possa pensare non è la situazione ideale. Infatti in condizioni di buio anche una semplice scatola riflette la luce laser verso altri oggetti, fenomeno che in condizioni di luce diffusa e meno visibile.

3.6. Software del Sistema.

Come già anticipato, il sistema è gestito interamente tramite Matlab. Lo script principale è il file "main.m" che permette di eseguire un intera scansione della scena antistante la webcam, e di stampare a video il grafico con le distanze ottenute.
Lo script *main*, segue le operazioni dell'algoritmo presentato nel paragrafo 3.3. Tali operazioni e gli script chiamati da *main* per le operazioni più complesse vengono analizzati in dettaglio in questo paragrafo:

Dichiarazione Costanti:
Vengono dichiarate le costanti che verrano utilizzate nello script: le dimensioni del fotogramma (*wcam*, *hcam*), la distanza focale (*fcam*), i parametri interni del sistema di visione (Dy, Dz).

Inizializzazione variabili:
Vengono dichiarate le variabili quali lo stato d'avanzamento del motore (*statolaser*), e la matrice delle distanze (*imgdist*). E' importante ricordare che lo stato d'avanzamento del motore, non viene misurato tramite sensori, ma deve essere opportunamente incrementato in base ai comandi inviati, essendo un controllo in anello aperto.

Inizializzazione della comunicazione sulla porta parallela:
L'inizializzazione della porta parallela in Matlab, si esegue semplicemente tramite due righe di codice:

```
port =digitalio('parallel','LPT1');
hwlines=addline(port,0:3,'out');
```

La comunicazione sulla porta parallela è cosi inizializzata, e per scrivere dei bit sui corrispondenti pin della porta basta scrivere:

```
putvalue(port,[1 1 1 1]);
```

Come si può vedere la comunicazione su porta parallela è stata inizializzata solo su 4 bit degli 8 disponibili. Infatti per comandare, il motore passo passo basta un bit per fase, cioè 4. I rimanenti 4 bit verrano utilizzati quando il sistema di visione sarà montato sul robot mobile: in quel caso gli ultimi bit serviranno per comandare i motori delle ruote.
Prima di iniziare la scansione viene catturata un immagine, quanto ancora il segmento laser è proiettato fuori dalla scena inquadrata dalla webcam.
Tale immagine, come spiegato in precedenza, verrà sottratta ai successivi fotogrammi per una più facile determinazione dei pixel illuminati dal laser.

Scansione:
Finita la fase di inizializzazione, inizia la scansione vera è propria: si deve fare scorrere il segmento laser sulla scena osservata. Per ogni passo della scansione vengono ripetute quindi le stesse operazioni.
Prima di tutto è necessario comandare il motore passo passo, in modo da ruotare il laser. Tale rotazione viene eseguita tramite uno script esterno "AvanzaLaser.m", che viene invocato nel seguente modo:

```
statolaser=AvanzaLaser(statolaser,port,R);
```

Lo script *AvanzaLaser*, si occupa quindi di ruotare il laser verso il basso. Per fare ciò scrive sulla parallela, identificata dal riferimento "port", un'opportuna riga della matrice di bit necessaria per comandare il motore passo passo. Le righe della matrice devono

essere applicate nella giusta sequenza, e per questo è necessario passare allo script lo stato di avanzamento del motore (*statolaser*). Lo script dopo aver inviato i bit sulla parallela, restituisce a *main* il nuovo valore di *statolaser*. Il parametro "R" che sta per reverse, permette di avere una rotazione nel verso opposto, in questo caso verso l'alto. Ciò permette di utilizzare lo script *AvanzaLaser* sia nella fase di scansione, sia per riportare il laser nella sua posizione iniziale mettendo R uguale a -1.
Dopo che il laser è stato ruotato, bisogna calcolare la sua nuova inclinazione:

```
teta=teta0 + K * statolaser;
```

Dove teta0 è l'angolo iniziale di partenza e K è una costante che rappresenta l'angolo di cui viene ruotato l'asse in un singolo passo. L'angolo teta è fondamentale per poi calcolare la distanza dei punti, quindi è importante conoscerlo con sufficiente precisione.
A questo punto si può acquisire una nuova immagine e passare all'analisi della stessa, per individuare i punti illuminati dal laser. L'immagine appena catturata viene analizzata per colonne: per ogni colonna viene quindi chiamato lo script "TrovaLaser" nel seguente modo:

```
ylaser=TrovaLaser(img0(:,x,1),img1(:,x,1),hcam);
```

Questo script, riceve una colonna dell'immagine a laser spento, la corrispettiva colonna dell'immagine con il laser, e l'altezza dell'immagine. Quindi viene seguito l'algoritmo spiegato nel paragrafo 3.5, che restituisce l'ordinata del pixel illuminato, altrimenti -1, se non è stato possibile trovare niente.
Se *ylaser* è diverso da -1 allora è possibile calcolare la distanza dal sistema, attraverso lo script "TrovaDist" nel modo seguente:

```
imgdist(ylaser,x)=TrovaDist(ylaser,fcam,teta,Dy,Dz,hcam);
```

Questo script ricevi tutti i parametri e le costanti necessarie per applicare la formula della distanza:

$$dist = f_p \frac{(-d_y \cos\theta - d_z \sin\theta)}{-(f_p \sin\theta + Y_p \cos\theta)}$$

ricavata dal modello teorico nel paragrafo 3.2. *TrovaDist* può a sua volta restituire -1, se il denominatore della formula è 0. La matrice delle distanze viene aggiornata con i nuovi valori calcolati.

Infine, al termine dello script *main* viene stampata a video la rappresentazione tridimensionale (Figura 3.11) della matrice delle distanza tramite il comando *mesh*:

mesh(imgdist);

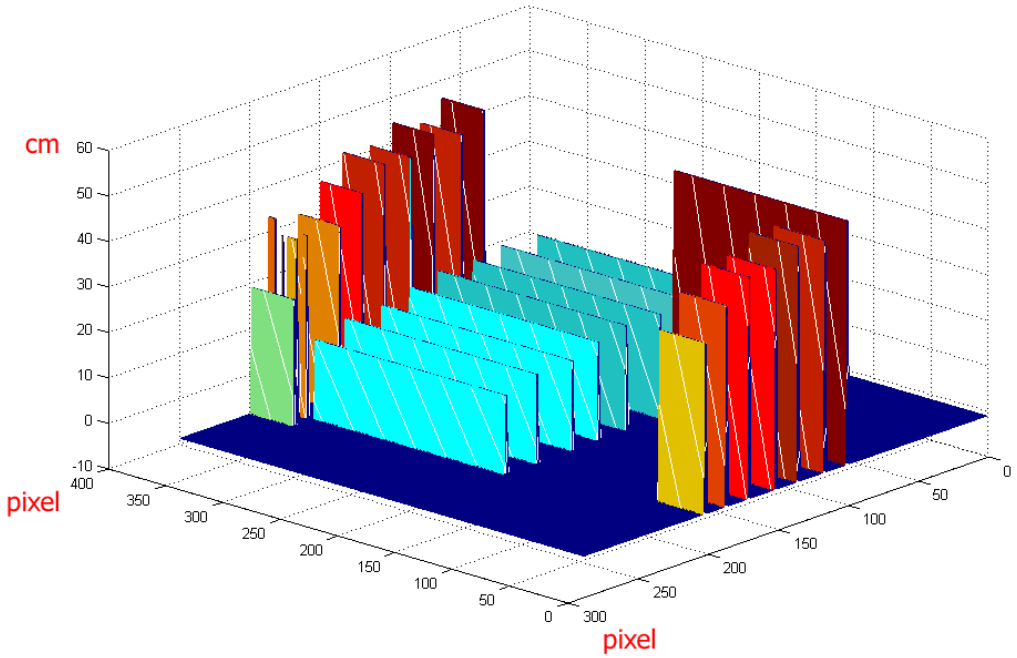


Figura 3.11: Depth map

La matrice delle distanze ha le dimensioni del fotogramma acquisito dalla webcam: 352 x 288. E' possibile vedere che per ogni pixel vi è una colonna verticale che rappresenta la distanza in cm, oppure il valore -1 se non è stato possibile determinare la distanza di tale punto.

L'ultimo script usato, è "scena3d", che in maniera grossolana ricostruisce l'immagine 3D osservata. Per fare ciò, sono necessari più passi. Prima di tutto viene effettuata un interpolazione lungo l'asse delle ordinate, mentre lungo l'asse ascisse non è necessario. Ciò è dovuto alla diversa risoluzione orizzontale rispetto a quella verticale. La risoluzione orizzontale dipende solo dalla webcam: infatti a meno di errori, lungo il

segmento del laser tutti i pixel sono illuminati. Per scansionare l'intera scena si è visto come il laser venga spostato non in modo continuo, ma bensì a scatti. Quindi un oggetto sarà composto da una serie di strisce orizzontali. Essendo l'angolo di passo, molto piccolo, si può supporre con buona approssimazione, che i pixel intermedi abbiano distanza simile ai pixel vicini illuminati. Si interpola quindi lungo le colonne della matrice delle distanze.

Per semplicità si è implementata un interpolazione lineare. Dopo l'interpolazione si ottengono una serie di blocchi, che non rappresentano ancora la scena 3D che è stata misurata. Infatti, nella matrice delle distanze gli le colonne più alte rappresentano gli oggetti più lontani. Quindi è necessario invertire i valori sulle distanze e mettere gli oggetti più lontani a zero e successivamente sovrapporre gli oggetti via via più vicini. In Figura 3.12 viene riportata l'interpolazione della matrice delle distanze di Figura 3.11.

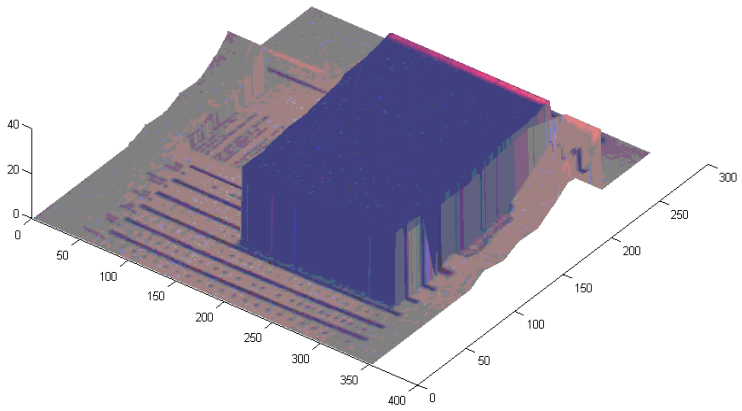


Figura 3.12: Interpolazione della depth map

Come si può vedere, il blocco blu emerge dal piano che è situato ad una distanza maggiore.

3.7. Taratura del dispositivo.

Una fase importante per utilizzare il sistema di visione, è la sua taratura, cioè la misurazione dei parametri che ci servono per il calcolo della distanza. Se si prende in considerazione la formula per la distanza:

$$dist = f_p \frac{(-d_y \cos\theta - d_z \sin\theta)}{-(f_p \sin\theta + Y_p \cos\theta)}$$

Si vede che i parametri del sistema che servono sono *dy*, *dz*, e θ. Tra questi la misura di θ è la più semplice e precisa da ottenere. Infatti per trovare θ basta sapere l'angolo iniziale dal quale si avvia il sistema, e sommargli per ogni passo l'angolo caratteristico del motore. L'angolo iniziale si trova semplicemente proiettando il laser su un foglio rigido posto trasversalmente, e marcando i puni di intersezione. Tramite il triangolo rettangolo che si viene a formare è facile trovare l'angolo cercato.
Più difficile è la misura degli altri due parametri: *dy e dz* sono rispettivamente la componente *y* e *z* del vettore che congiunge il centro ottico della webcam con il punto attorno al quale fa perno il laser. Il centro ottico, essendo all'interno della webcam, complica la misura e ci si è accorti che era facile sbagliare anche qualche centimetro. Si è quindi pensato di determinare i parametri in modo inverso: puntare il laser del sistema con una angolo noto, verso oggetti a distanza nota, misurare *Yp* dalle immagini e ottenere una serie di equazioni con incognite *dy e dz*. Per determinare questi parametri in modo accurato è stato necessario fare più misure e quindi fare una media dei vari valori ricavati.
Durante tale misurazioni si è notato che per avere un calcolo delle distanze abbastanza preciso, sia *dy* che *dz* debbano essere abbastanza grandi: almeno maggiori di 5 cm.
Sfortunatamente non si può aumentare *dy e dz* a piacimento, per ottenere sempre maggiore precisione. Infatti all'aumentare di questi due parametri aumentano le zone di occlusione, cioè zone inquadrate dalla webcam, ma in cui non ci sono pixel illuminati dal laser a causa di ostacoli intermedi.
Si è verificato che mettendo a distanza di pochi centimetri laser e webcam, le zone di occlusione erano praticamente inesistenti, però la determinazione delle distanze ha prodotto dei valori poco significativi.

Per concludere, tale sistema di visione può presentare diverse configurazioni al variare dei suoi parametri interni. Non esiste una configurazione buona in generale, ma dipende dal tipo di scena osservata, dalla distanza degli oggetti, e da come varia la distanza tra un oggetto e un altro.

4. Progettazione ed implementazione di un robot mobile

4.1. Introduzione.

Un robot mobile, diversamente da un robot industriale, deve avere la capacità di muoversi in un ambiente dinamico e non strutturato al fine di interagire con esso. Quindi due punti chiave nella progettazione di un robot mobile sono la scelta di un adeguato sistema di locomozione e di un insieme di sensori che permetta di rilevare l'ambiente circostante. Entrambi questi punti vanno affrontati tenendo conto del tipo di task (compiti) che il robot dovrà eseguire, e delle caratteristiche dell'ambiente in cui dovrà operare.

Per quanto riguarda la locomozione, si può scegliere se dotare il robot di zampe, di ruote o di cingoli. Le zampe e i cingoli vengono usati per robot operanti in ambienti dal terreno sconnesso. Rispetto ai cingoli, le zampe permettono anche di scavalcare ostacoli, e salire scalini a discapito però di un controllo molto più complicato che deve preoccuparsi di mantenere il robot sempre in equilibrio.

Per la navigazione indoor, si è ormai consolidato l'uso di robot basati su ruote, come giusto trade off tra velocità, efficienza e semplicità di controllo.

Per quanto riguarda i sensori, la scelta dipende da quali informazioni sono necessarie per i compiti del robot. Ci sono delle caratteristiche dell'ambiente meno importanti, come la temperatura dell'aria, e altre più importanti come la presenza di ostacoli. Un generico robot mobile è di solito equipaggiato di sensori di prossimità e di un sistema di visione, tramite i quali riesce a percepire la struttura geometrica dell'ambiente antistante. Una volta definiti un sistema di locomozione e un insieme di sensori, il robot ha tutto il necessario per navigare correttamente nell'ambiente. La navigazione è eseguita da un calcolatore che fa da ponte tra il sistema sensoriale e il sistema di locomozione.

In questo lavoro di tesi è stato progettato ed implementato un robot mobile per il mapping di ambienti indoor. La fase di progettazione ha richiesto di analizzare con cura le problematiche di cui si è appena parlato. In particolare si è scelto un sistema di locomozione basato su ruote secondo il modello del differential drive. Si è quindi dotato il robot del sistema di visione 3D opportunamente adattato, e sviluppato in una fase precedente come spiegato nel Cap. 3.

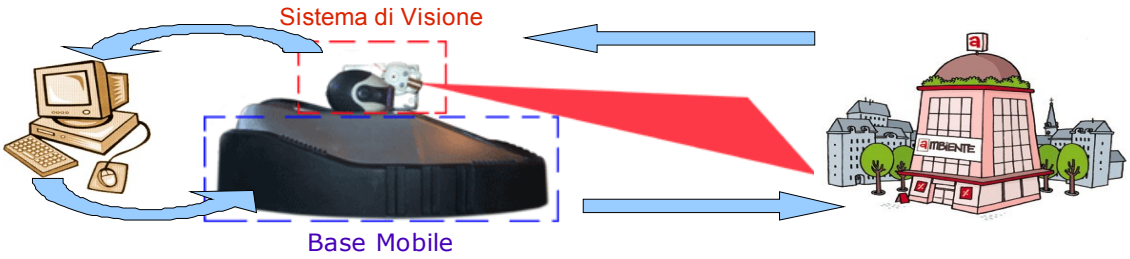


Figura 4.1: Schema generale del sistema

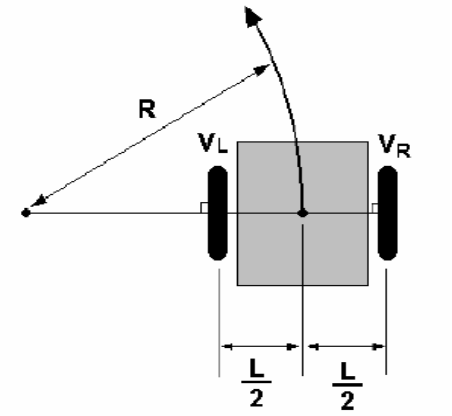
Tutto il sistema viene controllato tramite computer, che si occupa di acquisire i dati dal sistema di visione e dopo aver creato un modello virtuale della scena antistante, comanda opportunamente il sistema di locomozione in modo da eseguire un particolare task. In particolare si è progettato il robot per eseguire operazioni di mapping e utilizzare meccanismi di obstacle avoidance.
E' possibile schematizzare il robot che è stato progettato, in tre componenti principali che verrano approfondite nel resto del capitolo:

- una base mobile
- un sistema di visione
- un calcolatore

Nei paragrafi 4.2 e 4.3, verrà analizzata dettagliatamente la struttura della base mobile: la parte meccanica e la parte elettronica necessaria per comandarne il movimento. Nel paragrafo 4.4 verrano spiegate le modifiche effettuate sul sistema di visione sviluppato nel Cap. 3, per adattarlo al robot mobile. Infine nel paragrafo 4.5 verrà presentato il funzionamento del robot e i passi necessari per effettuare il mapping, mentre nel paragrafo 4.6 verrà descritta l'architettura software che gestisce l'intero sistema.

4.2. Cinematica.

Nella progettazione del robot mobile, si è scelto di seguire il modello del differential drive, che si è dimostrato molto semplice.
Il differential drive è un modello nel quale si hanno due ruote motrici e una ruota passiva omnidirezionale. Le due ruote motrici sono indipendenti, cioè possono avere velocità differenti, da qui il nome differential drive. Quindi le due ruote non sono collegate ad un unico albero motore, ma bensì sono collegate ognuna all'asse di un diverso attuatore meccanico. E' importante che gli assi dei due motori, su cui sono montate le ruote motrici, siano sulla stessa retta.
Queste due ruote stanno da una parte della base mobile, solitamente in quella posteriore, mentre nell'altra è montata una ruota omnidirezionale che ha sono una funzione di supporto, e quindi deve assecondare la direzione imposta dalle ruote motrici.



La cinematica diretta dipende dalla differenza di velocità tra le ruote motrici, e naturalmente dal tempo durante il quale tali velocità vengono applicate.
Da un semplice di studio, di questo modello, e sotto l'ipotesi di puro rotolamento, si ricavano le seguenti relazioni:

$$V = \frac{(V_r + V_l)}{2} \qquad \omega = \frac{(V_r - V_l)}{l}$$

In fase di implementazione si è deciso di far muovere le ruote sempre alla stessa velocità, in modo da evitare il problema di comandare velocità variabili. Quindi fissato *v*, le ruote possono ruotare ad un velocità *v* oppure *-v* (senso opposto). Ciò ha permesso di semplificare ulteriormente il controllo del robot, senza inficiare le funzionalità.

Dalle relazioni precedenti ponendo *Vl*=*Vr*, si ottiene che $\omega = 0$ e *V*=*Vr*=*Vl*: in questo caso si ottiene un movimento rettilineo senza nessuna rotazione. Invece quando si ha *Vr*=-*VL* si ricava che $\omega = 2\mathrm{Vr}/l$ mentre *V*=0, cioè una rotazione sul posto. Tramite questi due semplici movimenti si può facilmente muovere il robot per esplorare l'ambiente.

Per implementare il differential drive si sono scelte due motoriduttori sui quali sono state montate due ruote di 6 cm di diametro. I motoriduttori utilizzati, lavorano in corrente continua, con tensioni da 3 a 12V. Naturalmente con le tensioni negative, da -3 a -12 V si ottengono le stesse velocità di rotazione ma in senso opposto. Ruote e motori sono stati montati sulla parte posteriore di un supporto in plastica di forma rettangolare. Le ruote sporgono dalla base tramite due fenditure. Sulla parte anteriore si è costruito un supporto metallico in grado di trattenere una sferetta che fungesse da ruota omnidirezionale. Questa è la parte meccanica necessaria a far muovere il robot.

4.3. Elettronica.

Per comandare gli attuatori meccanici da computer è necessaria la relativa elettronica. Come per il sistema di visione presentato in precedenza, per comandare il movimento si utilizzeranno i bit inviati sulla porta parallela. Il sistema di locomozione e il sistema di visione dovranno funzionare contemporaneamente. Quindi degli 8 bit utilizzabili sulla porta parallela 4 vengono impiegati per comandare il passo passo, e 4 rimangono per comandare la locomozione: 2 bit per comandare il motore sinistro, e 2 bit per comandare il motore destro.

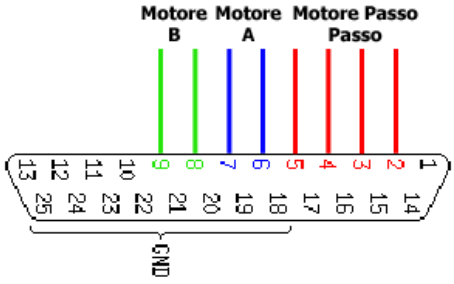


Figura 4.2: Mappatura dei pin della porta parallela

Come già accennato in precedenza si è deciso di far muovere le ruote sempre ad una velocità costante, impostata a priori. Quindi tramite i due bit è necessario comandare solo i seguenti movimenti: rotazione in avanti, rotazione indietro, motore fermo.
Nel realizzare l'elettronica si è scelta un opportuna velocità di rotazione dei motori, e la corrispondente tensione di alimentazione. Per far muovere il motore in avanti è necessario applicare al motore Vcc, mentre con -Vcc si ottiene la rotazione in senso opposto alla stessa velocità. Per tenere fermo il motore basta non alimentarlo.
Quindi si è realizzato un circuito di tipo ponte H, il cui schema concettuale è riportato in Figura 4.3.

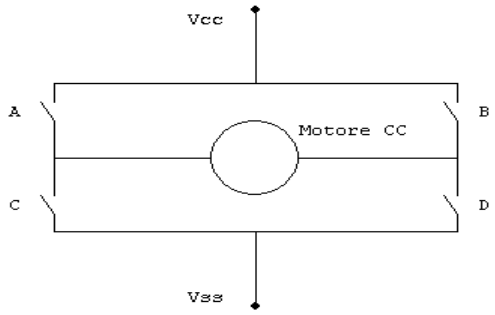


Figura 4.3: Schema di un circuito ponte H

Le combinazioni degli interruttori interessanti sono:

A e D chiusi	Il motore gira in avanti
B e C chiusi	Il motore gira all'indietro
Tutti gli interruttori aperti	Il motore non è alimentato

Nel costruire il circuito vero e proprio basta sostituire gli interruttori con dei transistor, e aggiungere i fotoaccoppiatori in modo da separare l'elettronica dei motori dalla porta parallela, la quale è direttamente collegata alla scheda madre, che potrebbe seriamente danneggiarsi in caso di malfunzionamenti del circuito. Lo schema elettrico è qui riportato:

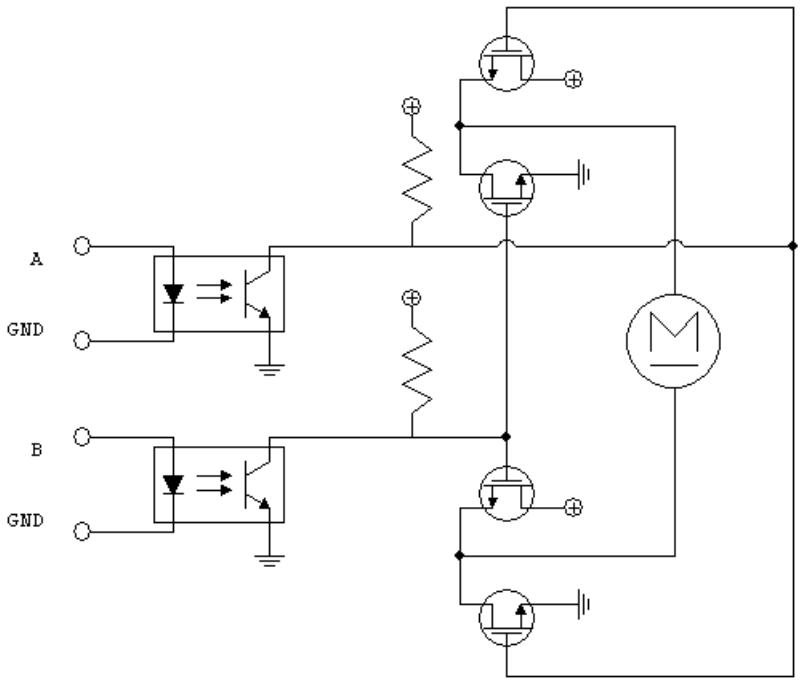


Figura 4.4: Schema generale del sistema

Per realizzare il circuito sono stati utilizzati i fotoaccoppiatori 6N135 della Fairchild Semiconductor, i power mosfet IRF540 della Sgs-Thomson Microelectronics e resistenze da 6,8 kΩ .

Ai pin A e B dei fotoaccoppiatori (Figura 4.4) possono essere applicate le sequenze 11, 01, 10 ma è da evitare la configurazione 00, che provoca un eccessivo assorbimento di corrente da parte del circuito. Infatti con la sequenza 00, le due coppie di transistor sono entrambe accese e si crea un corto circuito dall'alimentazione a massa. Se si utilizza un alimentatore poco potente, quest'ultimo potrebbe bruciarsi. Se invece l'alimentatore riesce ad erogare tutta la corrente necessaria, si potrebbero bruciare i transistor. Quando viene collegato il circuito al computer, tramite cavo parallelo, non è noto quale sia il valore degli 8 bit della porta: probabilmente potrebbe presentarsi proprio sui bit che comandano i motori la configurazione 00, danneggiando cosi circuito e l'alimentatore.

Per ovviare al problema, prima di inserire il cavo parallelo e accendere il circuito, bisogna aprire Matlab, inizializzare la comunicazione su parallela e scrivere sugli 8 pin la configurazione 11111111:

putvalue(port,[1 1 1 1 1 1 1 1]);

Tale sequenza di bit è la configurazione iniziale del sistema.

In Figura 4.5 è invece possibile vedere tutta l'elettronica del robot: il circuito per pilotare il passo passo, il circuito che alimenta il laser e i due circuiti uguali che comandano i motori A e B. I circuiti sono stati realizzati su due breadboard.

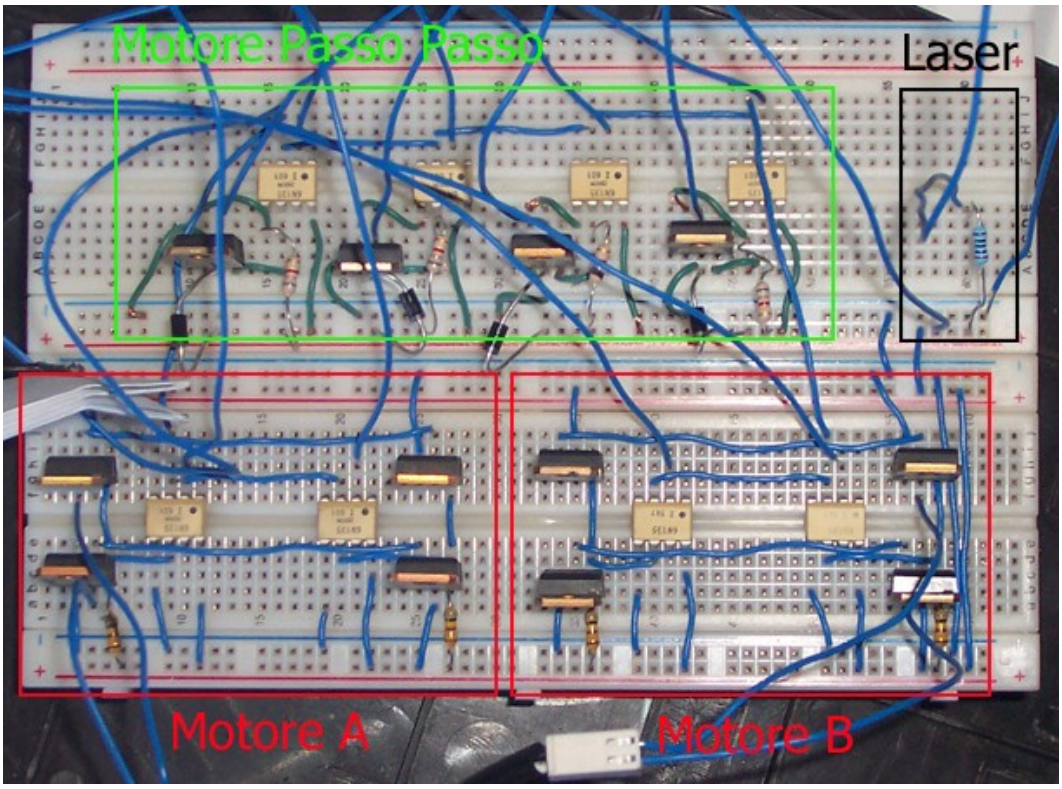


Figura 4.5: Circuito elettronico dell'intero sistema

Durante la fase di test del circuito si è lavorato con una tensione di 7,5 volt, erogata da un alimentatore. Successivamente, per avere meno intralci per il movimento del robot, l'alimentatore è stato sostituito da una comune batteria alcalina da 9 V. Il circuito per comandare i motori, non presenta problemi con una tensione maggiore: l'unico effetto è che le ruote avranno una maggiore velocità di rotazione.

4.4. La base mobile.

Il circuito elettrico viene montato sulla base, assieme alla batteria alcalina. Sulla base viene montata una parte superiore in plastica, che serve sia per proteggere i circuiti e i

motori, sia come supporto per il sistema di visione (Figura 4.6). Un lungo cavo parallelo esce dal robot, permettendo di comandarlo.

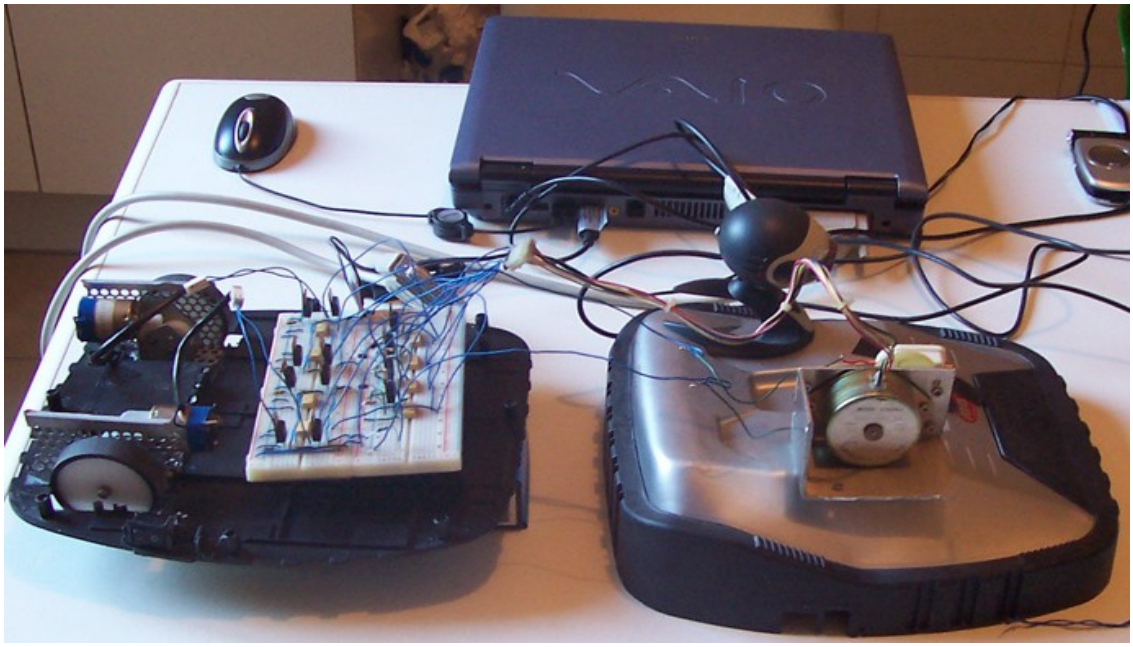


Figura 4.6: La base mobile

Prima di montare il sistema di visione, si è testato la locomozione del robot, in modo dal controllare che entrambe le ruote fossero in asse, e il movimento avvenisse senza problemi. Per muovere il robot si è quindi collegato il computer e dopo aver avviato MatLab si è inizializzata la comunicazione su parallela su tutti gli 8 bit:

```
port =digitalio('parallel','LPT1');
hwlines=addline(port,0:7,'out');
```

I 4 movimenti possibili si ottengono semplicemente scrivendo sui bit della parallela le opportuna sequenze di bit:

Comando Matlab	**Movimento**	**Rotazione Motori**
putvalue(port,[1 1 1 1 1 0 1 0]);	Movimento in avanti	motori entrambi in avanti
putvalue(port,[1 1 1 1 0 1 0 1]);	Movimento indietro	motori entrambi indietro
putvalue(port,[1 1 1 1 1 0 0 1]);	Rotazione oraria	motore A in avanti, B indietro
putvalue(port,[1 1 1 1 0 1 1 0]);	Rotazione antioraria	motore A indietro, B avanti
putvalue(port,[1 1 1 1 0 1 1 0]);	Nessun movimento	motori fermi

Le altre altre sequenze di bit possibili non sono di particolare interesse: alcune, come quelle che contengono 00 su un singolo motore, sono da evitare. Per evitare di avere 00, anche per qualche istante bisogna tra un movimento è il successivo passare sempre per la sequenza [11111111]. Infatti a causa di ritardi di propagazione si potrebbero avere problemi: si prenda in considerazione sul motore A la transizione 01->10, cioè la transizione da rotazione oraria a rotazione antioraria. Se il secondo bit viene modificato più velocemente, per pochi istanti sui pin dei fotoaccoppiatori si ha il valore 00. Quindi prima di cambiare il senso di rotazione di un motore, bisogna fermarlo.

4.5. Il sistema di visione.

Sulla base mobile costruita, si è poi montato il sistema di visione progettato e implementato in precedenza, come spiegato nel Capitolo 3. Si è quindi posizionata la webcam e il motore passo passo, con annesso laser, sulla base mobile. L'elettronica che gestisce il motore e il laser, sono stati posizionati all'interno, con il circuito dei motori, alimentando tutto sempre con la batteria da 9 V.
Il sistema di visione è stato adattato al contesto: non deve più funzionare come uno scanner 3D, ma come sensore per supportare il movimento del robot. Per fare ciò il dispositivo deve essere più veloce. Per raggiungere un maggiore velocità si è ridotto il numero di passi di ogni singola scansione, fino al caso limite di non muovere il laser, fisso su una determinato angolo. Si è in questo modo riusciti ad acquisire una scena in meno di 2 secondi. La riduzione del numero di scansioni, è stata possibile grazie alla particolare struttura di un ambiente indoor: i principali ostacoli sono sedie, tavoli, pareti, porte. Tali ostacoli vengono visti tutti come dei parallelepipedi o dei cilindri, la cui base può essere misurata senza ruotare il laser, è ulteriori scansioni risultano superflue. Inoltre, per evitare gli ostacoli, al robot serve scansionare un volume pari a poco più la sua altezza: non importa se sopra di esso vi è un tavolo o il tetto. Naturalmente il numero di passi per singolo scansione può essere aumentato ogni volta che l'ambiente presenti degli ostacoli non strutturati di cui ci interessa la forma, oppure in quei casi in cui ci serva un mappatura tridimensionale dell'ambiente circostante.
In questo lavoro si è posta maggiore attenzione al problema della costruzione di una mappa 2D, dell'ambiente, come se fosse osservato dall'alto. Si è quindi eseguito un passo per scansione, o al massimo 2, o 3 per avere una maggiore precisione nella

misura. Un problema che si riscontra nell'eseguire pochi passi consiste nell'interpretare una scatola alta una decina di centimetri come una parete. D'altra parte è importante non puntare il laser troppo in alto, perché si potrebbero perdere informazioni su ostacoli bassi ed avere delle collisioni. Inoltre quando si è esegue un solo passo, l'analisi dell'immagine è più difficoltosa: non si ha infatti un immagine con il laser fuori dalla scena. Tale immagine, come spiegato nel Cap. 3 è importante per trovare i punti illuminati dal laser. Si sono quindi dovute studiare delle tecniche diverse.

Un altra problematica da affrontare è il posizionamento della webcam rispetto al laser. Come si è visto nel capitolo precedente la precisione delle distanze calcolate aumenta all'aumentare della distanza tra webcam e il laser. Durante i primi test, con un distanza minore di 2 cm, si è osservata una scarsa precisione. Si è poi notato che in quelle condizioni punti distanti tra loro per più di 20 cm, differivano nell'immagine di un solo pixel. Ciò implicava una bassissima sensibilità, e un errore nel riconoscimento della posizione del laser anche di un solo pixel, poteva causare un errore di oltre 20 cm sulla distanza calcolata. Si è quindi cercato di massimizzare la distanza in modo che *dy* e *dz* fossero abbastanza grandi. Queste nuove condizioni presentavano una migliore triangolazione per il calcolo delle distanze. Per raggiungere una maggiore grado di precisione si è evitato di misurare *dy* e *dz* direttamente, ma indirettamente come spiegato nel paragrafo 3.7.

Per quanto riguarda la precisione c'è da prendere in considerazione un altro fattore. Nella formula ricavata nel precedente capitolo:

$$dist = f_p \frac{(-d_y \cos\theta - d_z \sin\theta)}{-(f_p \sin\theta + Y_p \cos\theta)}$$

Si può vedere che il denominatore non è sempre diverso da zero. I casi in cui si annulla corrispondono al caso in cui il determinante della matrice che rappresenta la triangolazione è nullo. Esiste quindi fissati tutti gli altri parametri un valore di *Yp* che annulla il denominatore. Si è fatto in modo che sui pixel che vanno da 1 a 288, nessun valore annulli il determinante. Però il problema si presenta ugualmente con tutti i valori vicini a quel *Yp*, singolare. Si analizzi il seguente esempio dove la formula precedente è stata semplificata imponendo un angolo di zero gradi, e dove al posto di *dy*, sono stati sostituiti dei numeri.

$$dist = \frac{1500}{Y_p}$$

Si analizzino quei pixel vicino al valore singolare 0. Per *Yp*=1,2,3,4 si ha che *dist*=1500,750,500,375.

Si può vedere che i valori di *dist* sono elevati, e variando anche un solo pixel si ottengono errori nell'ordine delle centinaia di centimetri. Diverso è se si lavora con i pixel attorno al 50, 51. Per capire quale sia il campo visivo del robot si deve scegliere un intervallo di valori di *Yp,* cioè un intervallo di distanze all'interno del quale si ha un precisione affidabile. Le distanze al di fuori di tale intervallo sono poco attendibili e quindi da scartare. In pratica dire che il robot ha un campo visivo di 70 cm, vuol di re che tutte le distanze inferiori ai 70 cm hanno un'adeguata precisione, mentre quelle esterne hanno elevata probabilità di essere sbagliate. Successivamente è importante muovere il robot di una distanza inferiore al campo visivo. Dopo che il robot sarà avanzato si potranno misurare gli eventuali oggetti che prima erano fuori dal suo campo visivo.

Quindi in fase di test sono state provate diverse configurazioni, e per ognuna è stato valutato il campo visivo, scegliendo la configurazione più adatta.

4.6. Funzionamento del robot.

Il robot mobile implementato in questo lavoro, in grado di muoversi in ambienti indoor, può essere utilizzato per varie applicazioni impiegando diversi algoritmi di mapping, o di path planning affrontando problematiche come la localizzazione o lo SLAM. In particolare il path planning può essere applicato precaricando un mappa dell'ambiente, oppure dopo che è stata effettuata una completa operazione di mapping.

Questo lavoro è stato centrato sul problema dell'esplorazione di un ambiente sconosciuto implementando semplici meccanismi di obstacle avoidance, al fine di costruirne una mappa 2D, che nelle future estensioni potrà essere utilizzata per il path planning definendo punti di start e di goal. Per il mapping si è utilizzato un algoritmo semi casuale, che non garantisce l'intera copertura della mappa. L'utilizzo di diversi algoritmi, come per esempio quelli proposti in [8], è immediata. In questa sede non sono stati implementati, in quanto si è voluto studiare come tramite un sistema di visione su

robot mobile fosse possibile acquisire informazioni sull'ambiente circostante, senza preoccuparsi del problema di una copertura totale e quindi senza fare un confronto tra i diversi algoritmo di esplorazione.
L'esplorazione inizia con l'inizializzazione di una mappa globale dell'ambiente, di forma rettangolare, la cui grandezza viene decisa a priori in base alla grandezza dell'ambiente da mappare. All'interno della mappa, il robot viene posto per convenzione in posizione centrale. E' importante sottolineare che ciò non implica che il robot si realmente al centro dell'ambiente da esplorare: la posizione centrale è giustificata dal fatto che lo spazio libero attorno al robot potrebbe trovarsi in una qualsiasi direzione con la stessa probabilità. Ponendo come posizione iniziale uno degli angoli si dovrebbe fare l'ipotesi che i lati dell'angolo corrispondano a delle pareti, in quanto non potrebbero essere aggiunte informazioni sull'ambiente fuori dalla mappa. Invece scegliendo la posizione centrale e delle dimensioni della mappa globale pari almeno al doppio delle dimensioni reali dell'ambiente non si hanno problemi qualsiasi sia la posizione reale del robot.
La mappa globale, nel calcolatore viene rappresentata con una matrice, dove ogni cella rappresenta una area predefinita, a seconda della precisione con la quale vogliamo memorizzare le informazioni sull'ambiente. Questo tipo di mappa è detta metrica. Ogni cella contiene un valore intero, in base alle informazioni raccolte: -1 se è un'area inesplorata, 0 se è un area libera, 100 se è un area occupata da un ostacolo. Nelle simulazioni eseguite si è utilizzata una mappa con celle da un centimetro quadrato, e grandezza variabile da 2m per 2m, al massimo di 6m x 6m.
Tale mappa è inizialmente formata da tutti -1, e verrà riempita con le successive misurazioni. Ogni misurazione ha una forma triangolare: è un triangolo con un vertice nella posizione del robot. L'angolo al vertice corrispondente al robot è pari all'apertura dell'ottica della webcam. La mappa globale dopo un certo numero di passa è costituita dalla sovrapposizione di tanti triangoli, uno per rilevazione effettuata. Ogni misurazione non viene incollata nella mappa globale alla stesso modo, ma tenendo conto della posizione del robot nella mappa, e il suo angolo di rotazione rispetto agli assi della mappa. Quindi ogni mappa locale acquisita dal robot deve essere traslata e ruotata quando viene aggiunta nel sistema di riferimento assoluto, il quale dipende esclusivamente dalla posizione del robot all'inizio dell'esplorazione. Questo rappresenta grosso modo il problema dello SLAM, cioè la localizzazione del robot nella mappa eseguita simultaneamente alla sua mappatura. Lo SLAM è un problematica non semplice da affrontare: ogni volta che si inserisce una porzione di mappa, nella mappa

globale si dovrebbero applicare algoritmi di massima verosimiglianza che garantiscano un incastro perfetto, o addirittura metodi più avanzati che fanno uso dei filtri Bayesiani e di Kalman. In questo contesto si è scelto di semplificare il problema basandosi esclusivamente sulla distanza percorsa dalle ruote in un dato intervallo di tempo. Questo controllo in anello aperto, purtroppo da buoni risultati solo per pochi pochi passi di movimento: successivamente la posizione reale del robot inizia a discostarsi dalla posizione stimata. Questo problema può essere facilmente superato utilizzando dei sensori odometrici. Nel Cap.5 sono presenti alcuni casi di test, effettuati su piccoli ambienti indoor, eseguendo solo poche iterazioni.

4.7 Software del robot.

L'intero robot, come già accaduto per il sistema di visione è gestito su computer tramite il software Matlab. Matlab riceve come input, tramite porta USB, i fotogrammi catturati dalla webcam, e in output tramite gli 8 bit della parallela comanda il movimento del laser e i due motori che servono per la locomozione.
Tutta l'esplorazione è gestita da uno script "main.m" il cui algoritmo è il seguente:

```
Dichiarazione di costanti e variabili;
Inizializzazione della comunicazione su porta parallela;
Inizializzazione della mappa globale e della posizione iniziale del
robot;
Inizializzazione della matrice delle distanze;
Do {
      acquisizione di un fotogramma dalla webcam;
      ricerca dei punti illuminati dal laser;
      calcolo delle distanze;
      interpolazione dei punti e creazione di un mappa locale 2D;
      scalatura della mappa locale;
      aggiornamento della mappa globale;
      spostamento del robot e calcolo della nuova  posizione;
} Loop (Finché la mappa globale non subisce più modifiche)

end.
```

I passi più importanti e complessi verranno analizzati in dettaglio:

Inizializzazione della mappa globale e della posizione iniziale del robot:
In questa fase dello script vengono inizializzate alcune costati e viene creata la mappa globale:

```
wmap=400;
hmap=400;
mappa_ambiente(1:wmap,1:hmap)=-1;
```

Le costanti *wmap e hmap* sono rispettivamente la larghezza e l'altezza della mappa, e sono da intendersi in centimetri. Viene quindi creata una matrice di 4 metri per 4 metri, in cui tutte le celle sono inizializzate a -1. Infatti nella fase iniziale, la mappa è completamente ignota.
Successivamente viene inizializzata la posizione del robot all'interno della mappa:

```
posx=wmap/2;
posy=hmap/2;
direzione=0;
```

Come è possibile vedere dal codice Matlab, la posizione iniziale del robot viene posta al centro della mappa. La variabile *direzione* rappresenta invece l'orientamento del robot. Tale variabile può assumere solo quattro valori: da 0 a 3. Il valore 0 indica che il robot sta puntando verso la parte alta della mappa. I valori 1, 2, 3 indicano rispettivamente un direzione verso sinistra, in basso, e verso destra.
Il fatto che la variabile *direzione* possa assumere sono 4 valori, indica che il robot muovendosi nel piano non può assumere una direzione qualsiasi. Si è scelto di permettere solo 4 posizioni per semplificare l'operazione di aggiornamento della mappa globale, come verrà spiegato dettagliatamente più avanti.
Ogni volta che il robot ruota, esegue sempre una rotazione di 90°. In questo modo la variabile *direzione* può rappresentare correttamente la direzione del robot, in quanto non vengono effettuate rotazioni intermedie, a meno di errori.

Infine si deve notare, che l'orientamento e la posizione del sistema di riferimento assoluto della mappa globale dipendono solo dall'orientamento e dalla posizione iniziale del robot rispetto all'ambiente circostante. Infatti la mappa è costruita attorno al robot.

Analisi dell'immagine:

L'analisi dell'immagine quando la scansione è eseguita attraverso più passi del laser, è identica a quanto descritto nel Cap.3, quando il sistema di visione viene operato come scanner 3D. L'analisi dell'immagine cambia nel caso in cui la scansione è eseguita senza spostare la striscia laser: in questo caso non si ha un immagine della scena senza laser. Per trovare il laser, si procede direttamente fissando una soglia per i pixel che possono essere illuminati dal laser. Poi l'immagine viene analizzata per colonne scegliendo per ogni colonna il gruppetto di pixel adiacenti più grande. Imponendo una soglia abbastanza elevata e che la porzione verticale sia formata da almeno 2 pixel, si ottiene che tutti i punti selezionati dall'algoritmo fanno parte della striscia laser proiettata sulla scena. Non è vero il viceversa: non tutti i punti che appartengono al laser passano il valore di soglia. Nei test effettuati, solo pochi punti appartenenti al laser non venivano riconosciuti come tali, e tramite interpolazione si è sempre potuto ricostruire il corretto profilo del laser sull'immagine.

Per determinare con maggiore precisione i pixel illuminati, si è impostato la webcam, in modo da rilevare solo i punti della scena con più luminosità. In questo modo non è possibile acquisire le texture delle parti della scena non illuminate, che vengono rilevate dalla webcam come zone di colore nero. Ciò non rappresenta un problema in quanto le texture sono inutili in una scansione 2D.

Dopo la determinazione dei punti, si procede al calcolo della loro distanza dal robot.

Interpolazione dei punti, creazione e scalatura della mappa locale 2D:

I punti ottenuti nella fase precedente, dopo essere stati interpolati, vengono utilizzati per costruire una mappa locale di ciò che sta davanti al robot. La mappa locale è rappresenta da una matrice che ha un numero di colonne pari al numero di colonne del fotogramma acquisito dalla webcam. Per ogni colonna di tale fotogramma, se c'è un pixel illuminato dal laser vuol dire che c'è un ostacolo. Quindi nella corrispondente colonna della mappa locale si segna l'ostacolo alla corrispondente distanza. Quindi la mappa locale ha un numero di colonne pari ai pixel dell'immagine acquisite e un numero di righe pari al massimo valore di distanza rilevato. Il numero di righe non può mai essere superiore

alla profondità del campo visivo. Si vede che la matrice di questa mappa non è omogenea: ad ogni riga è associato un valore in centimetri, mentre ad ogni colonna un valore in pixel. Si ha quindi il problema di capire a quanti centimetri corrispondono i 352 pixel che rappresentano le colonne dell'immagine. E' evidente che ciò dipende dalla scena osservata. Infatti se davanti alla webcam si metto un oggetto molto vicino, i 352 pixel potrebbero corrispondere ad una decina di centimetri. Se invece non c'è nessun ostacolo davanti al sensore, si può inquadrare una scena ampia anche qualche metro.
Per risolvere il problema si analizza la mappa locale a alla ricerca dell'ostacolo più distante. Le righe dopo tale distanza vengono eliminate. A questo punto, a partire dall'angolo di apertura della webcam, e la distanza dell'oggetto si risale a quanti cm corrispondono i 352 pixel. La mappa locale può essere quindi scalata, e dopo aver annerito le zone al di fuori dell'angolo d'apertura della webcam si ottiene un risultato come in figura:

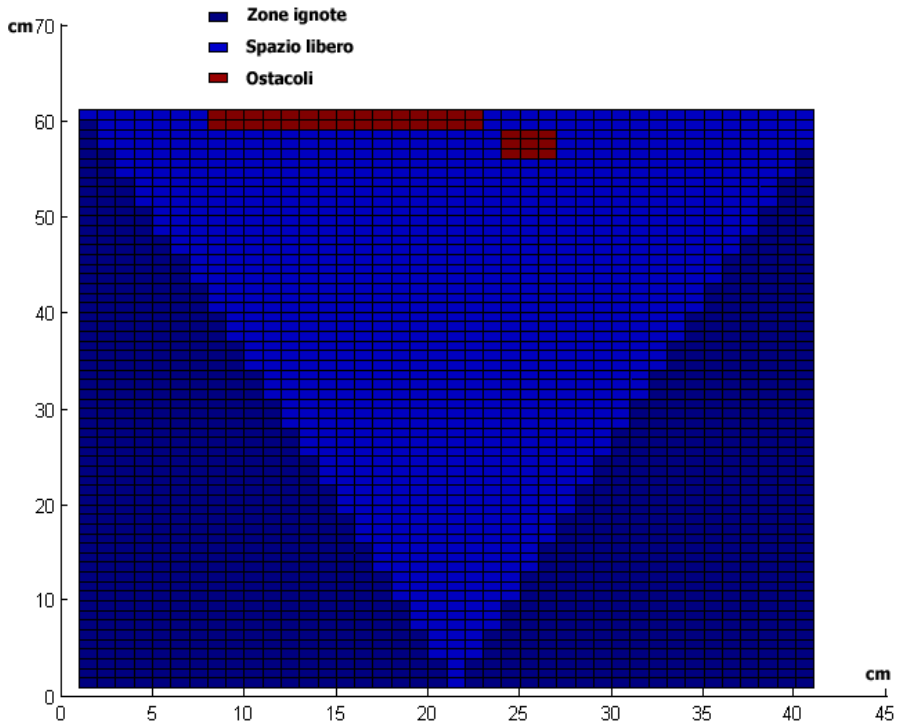


Figura 4.7: Un esempio di mappa locale

In Figura 4.7 è visibile la mappa già scalata. A partire dell'ostacolo posto circa a 58 cm, e da un angolo di apertura della webcam di 37,4° si è ricavato che l'ampiezza dell'immagine è di circa 40 cm, ragionando sul triangolo visibile in figura.

Aggiornamento della mappa globale:

Ogni volta che viene fatta una scansione locale, deve essere aggiornata la mappa globale dell'ambiente. Questo in generale è un operazione complicata. Il problema di incollare la mappa locale in quella globale deve tenere conto sia della posizione del robot, sia che i punti che si stanno aggiungendo alla mappa combacino con i precedenti. In questo lavoro si è adottata una soluzione semplice: non si cerca di far combaciare le mappa locale con quella globale, ma si effettua una semplice sovrapposizione.

Tale sovrapposizione deve essere fatta a partire dalla posizione di partenza del robot, e ruotando la mappa locale in modo che il suo orientamento coincida quello del sistema di riferimento assoluto. Qui l'ipotesi iniziale, di poter avere solo 4 direzioni, semplifica il problema. Infatti, ruotare la matrice di un angolo arbitrario è abbastanza complicato: e ancora più complicata e la sua sovrapposizione. Infatti fin quando le celle della mappa sono ortogonali tra loro, un cella nella mappa locale corrisponde ad una cella nella mappa globale. Se invece le celle non sono tra loro ortogonali, un cella della mappa locale ne può occupare quattro in quella globale: ne risulta un ingrandimento degli ostacoli.

Nel caso in cui tutte le rotazioni sono di 90° la matrice globale viene facilmente ruotata tramite il comando Matlab:

```
mappa=rot90(mappa);
```

Spostamento del robot e calcolo della nuova posizione:

Per effettuare lo spostamento del robot viene chiamato lo script “algo_movimento.m”. Si è deciso di inserire tutte le operazioni necessarie allo spostamento e all'aggiornamento della posizione in unico script in modo da facilitare eventuali modifiche al software. Nelle simulazioni, per muovere il robot è stato utilizzato un algoritmo semicasuale. Qualora si volessero testare nuovi algoritmi per il mapping, come quelli proposti in [8], basta scrivere un nuovo m-file e sostituirlo allo script precedente in modo totalmente trasparente al resto del programma. Lo script che si occupa del movimento riceve tutte le informazioni importanti: riceve la mappa locale, la mappa globale, e la posizione del robot in quest'ultima. Tramite queste informazioni si può riprodurre qualsiasi algoritmo di mapping.

Nelle simulazioni è stato utilizzato il seguente algoritmo:

```
Se ( c'è sufficiente spazio libero )
      vai avanti;
Altrimenti
      ruota in senso antiorario di 90°;
end.
```

Questo algoritmo se pur semplice si è dimostrato molto efficace, soprattutto per l'obstacle avoidance. Seguendo l'algoritmo, la prima operazione eseguita dal robot è controllare quali ostacoli, ci sono di fronte. Se in una distanza pari alla metà del campo visivo è tutto libero, il robot si muove in avanti, di un distanza fissata. Questa distanza fissata è sempre pari alla metà del campo visivo del robot. Se invece ci sono degli ostacoli il robot viene fatto ruotare di 90°. Naturalmente se dopo la rotazione il cammino sarà ancora ostruito, alla prossima iterazione dell'algoritmo ci sarà nuovamente una rotazione. In caso di vicolo cieco si possono avere anche 3 rotazioni consecutive.
Come si può vedere, questo script utilizza solo informazioni locali, e non utilizza la mappa globale per scegliere il movimento da fare. Invece per un mapping più approfondito bisogna analizzare la mappa globale in modo da dirigersi verso le zone con più punti ignoti. L'algoritmo invia quindi i segnali ai motori, e successivamente aggiorna la posizione del robot nella mappa globale, restituendo il controllo allo script principale *main*.
Per far giungere il robot nella nuova posizione calcolata dall'algoritmo di movimento, si è sfruttata l'alta ripetibilità dei motori e l'ipotesi di puro rotolamento delle ruote. Infatti in queste condizioni, applicando ai motori un comando per un certo intervallo di tempo, si è ottenuto sempre lo stesso spostamento, con errori nell'ordine dei millimetri. Basandosi sul fatto che l'algoritmo usato comanda al robot sempre rotazioni di 90° o avanzamenti di una distanza prefissata e bastato calcolare i rispettivi intervalli di tempo. Tale metodo non comporta particolari problemi, se i tempi sono tarati bene e il pavimento non presenta grosse variazioni di attrito. Una maggiore precisione si potrebbe ottenere montando sulle ruote degli encoder: in tal modo anche variazioni di peso del robot, o il passaggio tra diversi pavimenti con caratteristiche diverse non comporterebbero errori sulla posizione raggiunta.

Terminazione delle operazioni di mapping:

Nell'algoritmo precedentemente esposto si è scritto che le operazioni di mapping continuano fin quando, non si rilevano più modifiche alla mappa globale: ciò potrebbe verificarsi o perché la mappa globale è acquisita completamente e non viene quindi modificata con nuovi punti, o perché il robot è rimasto bloccato in una certa zona della mappa senza scoprire più nuove zone.

Nella pratica questo meccanismo di terminazione, non può essere applicato: infatti anche nel caso migliore, in cui il robot abbia mappato tutto l'ambiente, la mappa globale verrà continuamente modificata per errori di misura. Un ostacolo già mappato che in un secondo momento viene rilevato anche un centimetro più vicino, implica una nuova modifica alla mappa. Quindi nella pratica l'algoritmo viene fatto eseguire all'interno di un ciclo *For*, con un numero di passi sufficientemente elevati.

5. Risultati e conclusioni

5.1. Risultati Scanner 3D.

La versione finale dello scanner 3D implementato in questo lavoro, è stato sottoposto a diversi casi di test. Prima di procedere tale fase si è eseguita la taratura del dispositivo: sono stati determinati i parametri *dy*, *dz,*e l'inclinazione iniziale del laser. Si è deciso di eseguire 9 passi per scansione: in queste condizioni sono necessari 12 secondi per scandire la scena e 5 per riportare il laser nella posizione iniziale. Nelle simulazioni eseguite, si è utilizzato il sistema di visione per scandire delle scene formate da un piano verticale, davanti al quale sono stati posti degli oggetti in diverse configurazioni.
Nel primo caso di test è stato inserito nella scena un parallelepipedo blu (Figura 5.1).

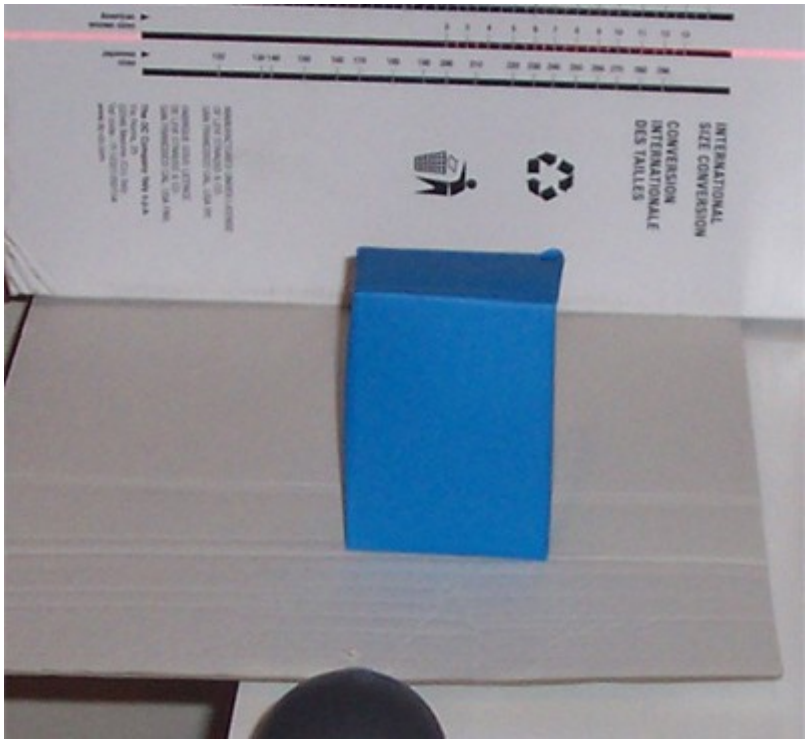


Figura 5.1: Primo test

Dopo la scansione si è ottenuta la matrice delle distanze (depth map) visibile in Figura 5.2 . Nella matrice delle distanze, per ogni punto è presente un valore che ne rappresenta la distanza. Se tale valore è pari a -1, vuol dire che per quel punto non è stato possibile ottenere informazioni sulla distanza. In figura si vedono chiaramente alcuni rettangoli azzurri che rappresentano il parallelepipedo analizzato. I rettangoli più alti, rappresentano le porzioni del piano verticale visibile dalla webcam: come si può notare hanno un distanza maggiore rispetto ai rettangoli che identificano il parallelepipedo.

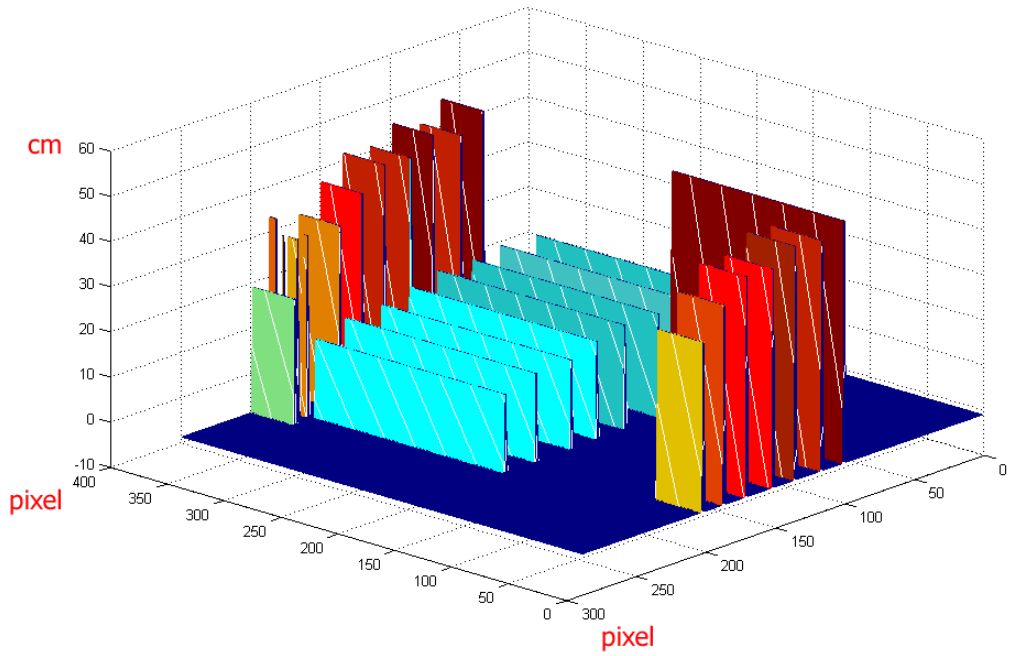


Figura 5.2: Depth map ricavata nel test 1

In Figura 5.2 è possibile vedere due di zone d'ombra (zone formate dai punti con valore -1). Il primo tipo è dovuto al movimento a scatti del laser, e il secondo ad un fenomeno d'occlusione: tra i rettangoli azzurri e rossi si può vedere una zona d'ombra più ampia. In tale zona la proiezione del laser è ostacolata dalla presenza del del parallelepipedo.
Nella Figura 5.3, alla matrice delle distanze sono state sovrapposte le texture acquisite dalla webcam: si può vedere con chiarezza che i rettangoli più bassi corrispondono all'oggetto blu delle Figura 5.1 .

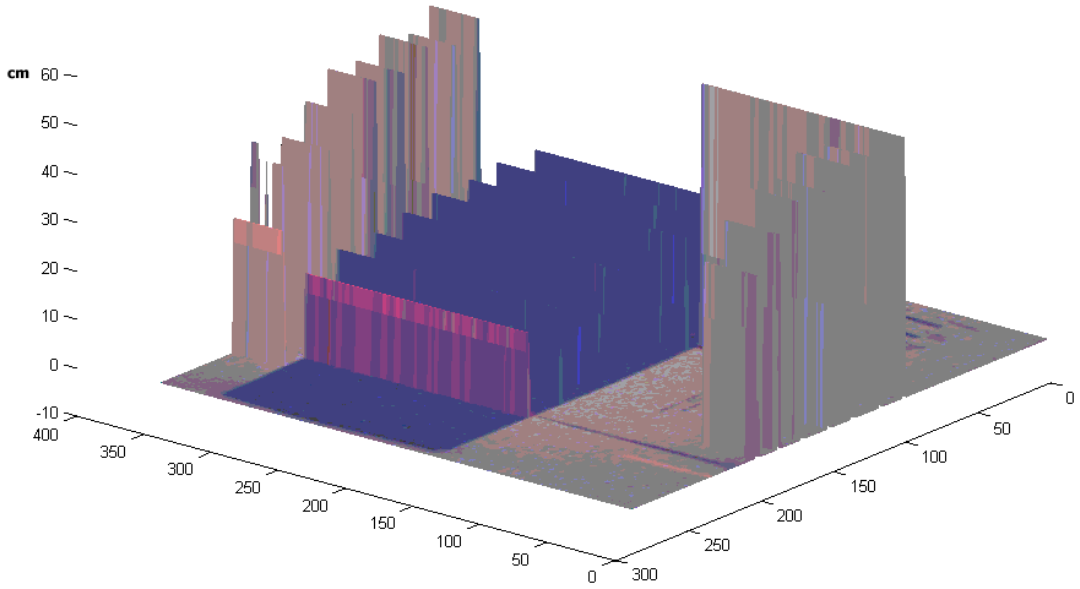


Figura 5.3: Depth map del test 1 con sovrapposizione di texture

Nella Figura 5.4 sono state utilizzate le informazioni della depth map per creare una rappresentazione 3D della scena osserva:

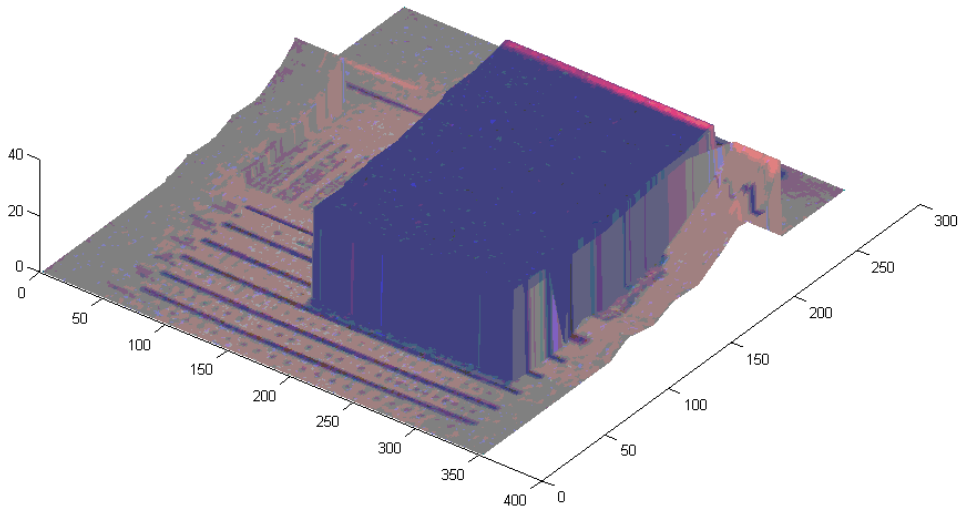


Figura 5.4: Rappresentazione 3D del test 1

Si può notare l'oggetto blu, che emerge rispetto al piano verticale. Nella rappresentazione creata non è presente nessuno spazio, tra il piano e l'oggetto, che invece esiste nella scena reale. Ciò succede perché il sistema di visione non può osservare il volume dietro l'ostacolo. In assenza di queste informazioni l'algoritmo di interpolazione unisce direttamente l'oggetto al piano retrostante.

Nel secondo test, si è utilizzata la scena precedente, dove a destra e un po più avanti dell'oggetto blu è stato posizione un cubo nero. I risultati sono visibili in Figura 5.5, dove si vede chiaramente l'oggetto precedente, e il cubo nero rappresento dalle colonne più basse: è infatti più vicino rispetto al parallelepipedo.

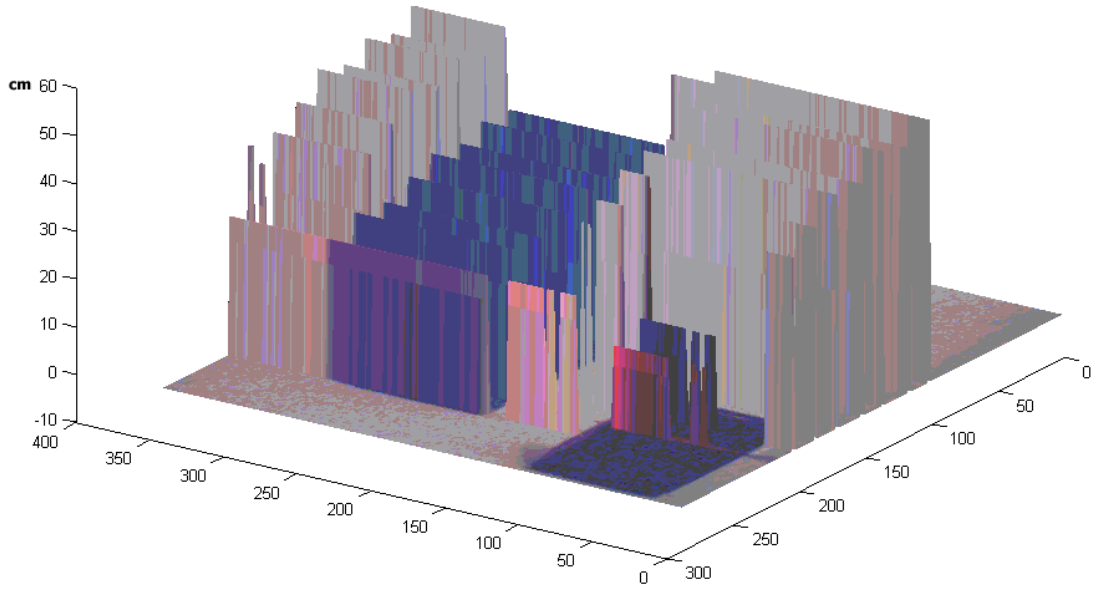


Figura 5.5: depth map con texture del test 2

Nel terzo Test, l'oggetto blu è stato posizionato in trasversalmente, e su di esso è stato poggiato il cubo nero. In Figura 5.6 si può vedere la rappresentazione 3D della scena analizzata. Si vede distintamente il cubo nero sull'oggetto blu.

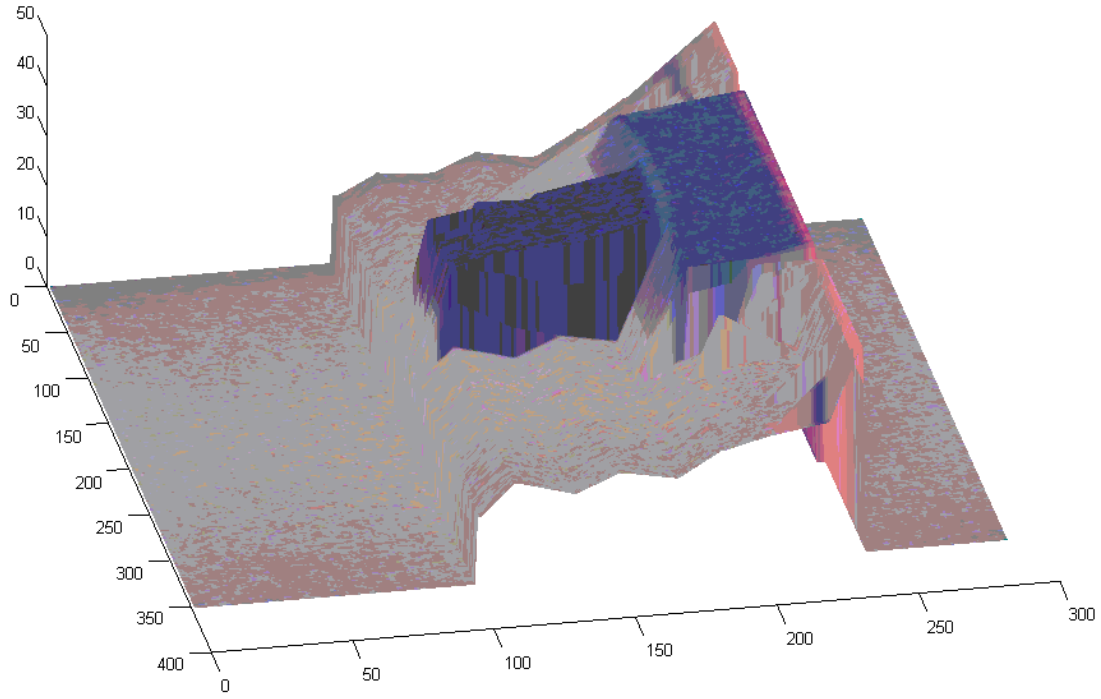


Figura 5.6: Rappresentazione 3D del test 3

Nei diversi casi di test si sono stimati errori nell'ordine del centimetro. Questi errori di misura hanno varie cause. Una parte degli errori è dovuta alla qualità della webcam utilizzata: si otterrebbero risultati migliori con un webcam molto più costose ma senza distorsione radiale e con una maggiore risoluzione. Un'altra causa di errori, è da ricercare nella costruzione dell'intero dispositivo: non si è avuta la logistica necessaria per far si che i componenti del sistema rispettassero i vincoli geometrici del modello come che la rotazione del laser fosse in asse con il motore, e che l'asse del motore fosse perpendicolare al piano immagine della webcam.

Infine sarebbe stato possibile raggiungere un maggiore grado di precisione testando un maggior numero di configurazioni, e quindi spostando la webcam rispetto alla sorgente laser. Non è sempre intuitivo capire quale configurazione offre una maggiore triangolazione. L'obiettivo e fare in modo che per le distanze analizzate, il laser possa illuminare qualsiasi pixel del sensore ottico: in tal modo già con una webcam di basso livello si ottengono 288 distanze diverse. Se si scansiona un scena in cui la distanza massima è un metro, con 288 distanze diverse si ha una sensibilità media minore al mezzo centimetro. Il problema è che non è facile individuare la configurazione migliore e fare in modo che il laser sfrutti tutto i pixel. Fare dei test per ogni configurazione

richiede un certo quantitativo di tempo, tanto che prima di eseguire una scansione sono state provate 2 o 3 configurazioni al massimo.
Comunque, un errore sulla misura di circa un centimetro, è un ottimo risultato. Infatti utilizzando questo dispositivo su un robot mobile, un centimetro è un errore più che accettabile rispetto agli spazi ampi in cui il robot dovrà muoversi.

5.2. Risultati mapping.

Nel primo test viene mostrato, come il sistema di visione opportunamente adattato sul robot mobile, riesce a riconoscere con sufficiente precisione i punti illuminati dal laser.
Le immagini della webcam sono state acquisite con una bassa luminosità al fine di riconoscere più facilmente i punti rilevati dal laser: il resto della scena viene in gran parte rilevato come una regione nera. Si può notare come a causa di diversi fenomeni non viene proiettata una retta perfetta: ciò complica l'analisi dell'immagine

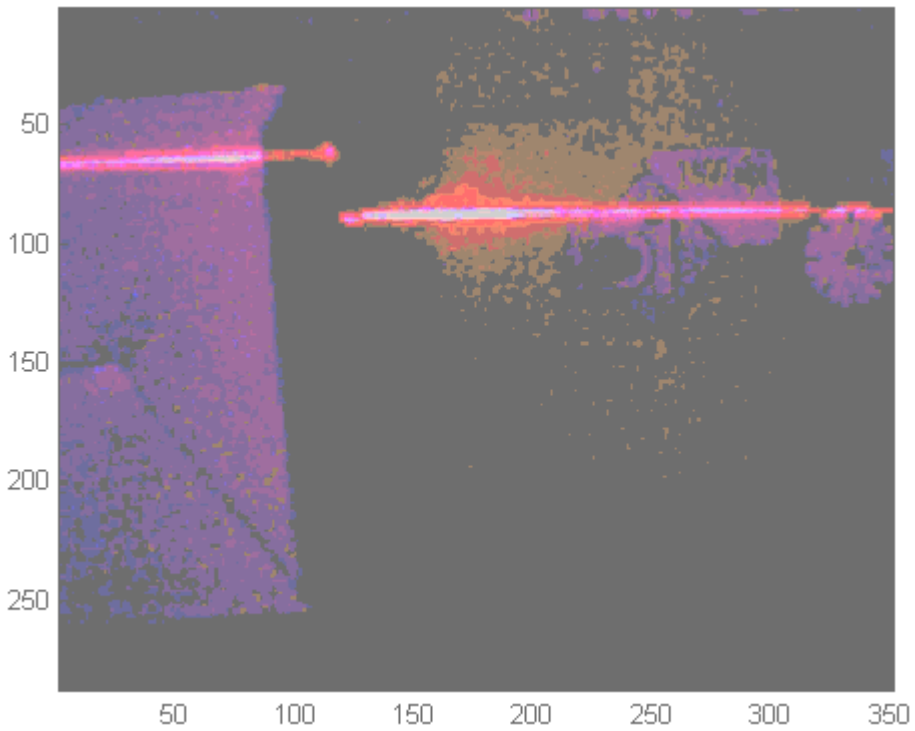


Figura 5.7: Laser proiettato sulla scena

In Figura 5.8 sono visibili i punti individuati, applicando l'algoritmo di analisi esposto nel paragrafo 4.7. Tale algoritmo si è dimostrato molto efficace.

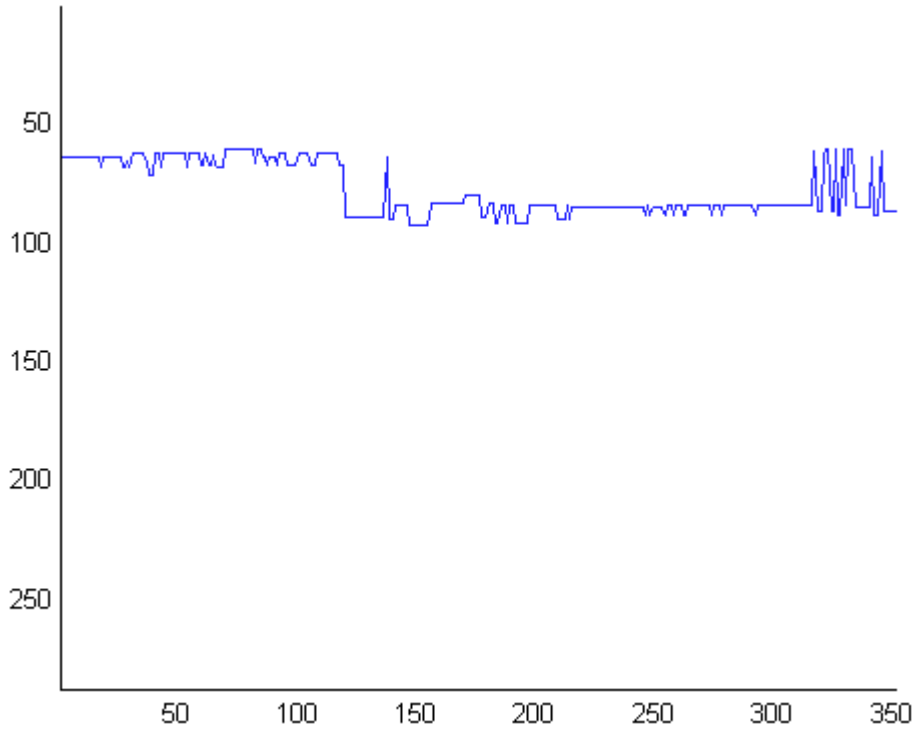


Figura 5.8: Punti del laser individuati

Nel successivo caso di test, si è usato il robot per mappare una porzione di un normale ambiente domestico. Dalla Figura 5.9 è possibile vedere il percorso seguito dal robot: ogni volta che si presenta un ostacolo al di sotto della distanza di sicurezza, viene effettuata una rotazione di 90°.

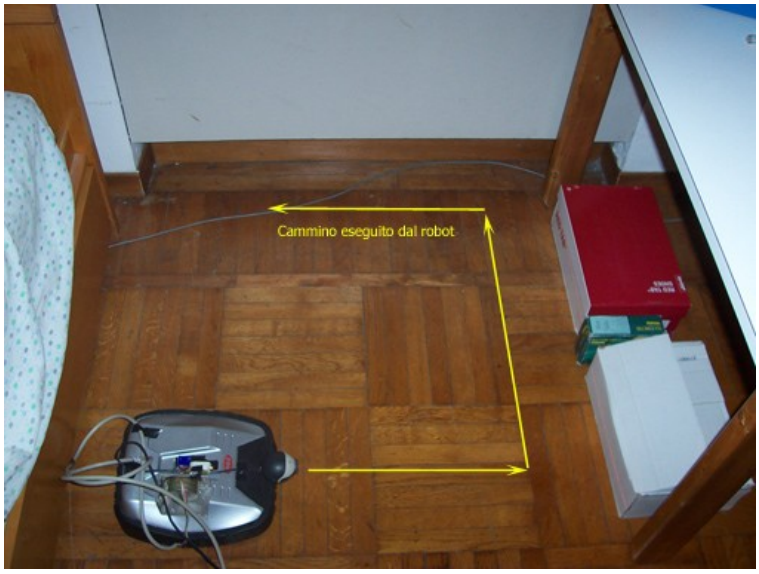


Figura 5.9: Percorso di mapping

I risultati acquisiti nelle singole misurazioni sono stati utilizzati per creare la mappa globale in Figura 5.10.

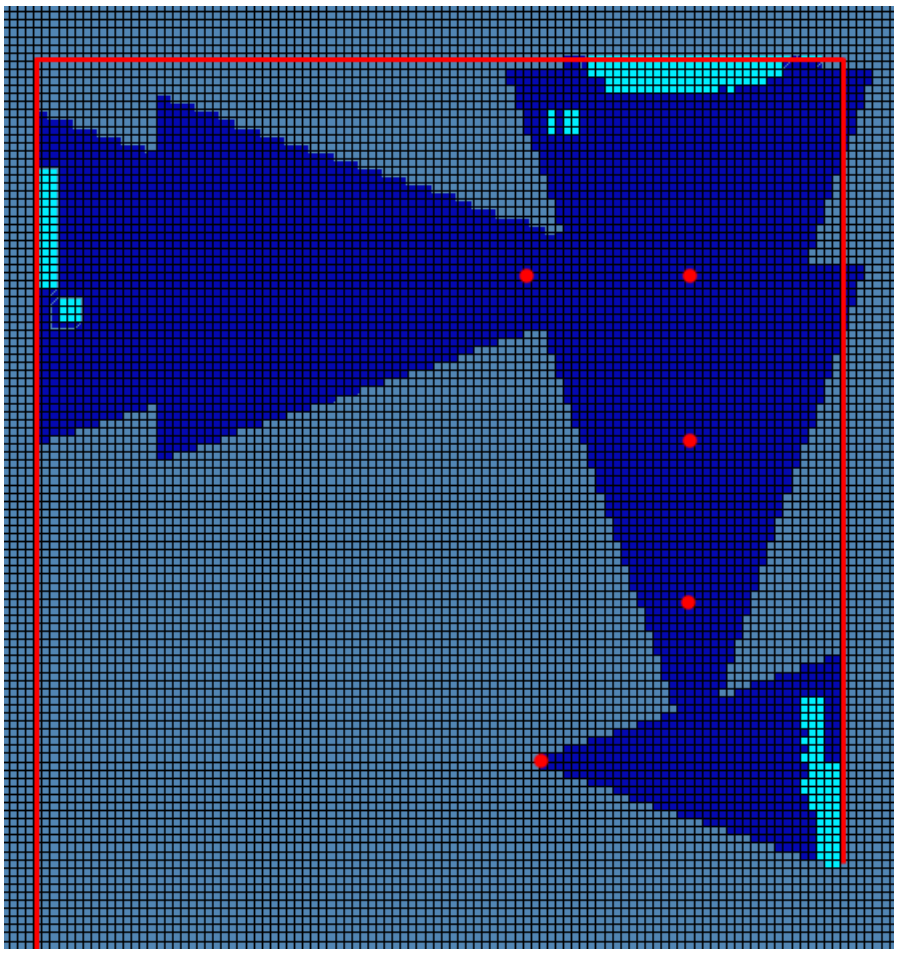


Figura 5.10: Mappa globale rilevata durante il mapping

In Figura 5.10, le zone più scure rappresentano le porzioni dell'ambiente che sono state esplorate e sono state rilevate come zone libere. Invece le zone più chiare, rappresentano gli ostacoli, incontrati. Se si interpolano gli ostacoli rilevati si può ricostruire il profilo della porzione di stanza esplorata: linea rosso in figura 5.10.

Le altre zone rappresentano la porzione di spazio inesplorata, della quale il robot non ha nessuna informazione. I puntini rossi, indicano le diverse posizioni attraversate dal robot durante il suo cammino.

Si vede che la mappa non è completa: infatti per non complicare il test e limitare gli errori si sono eseguito solo 7 passi di esplorazione. I risultati ottenuti sono comunque molto precisi.

5.3. Conclusioni.

Nella prima fase del lavoro si è quindi ottenuto uno scanner 3D, molto veloce e in grado in real time di calcolare le distanze con una buona precisione e creare una grossolana approssimazione della scena antistante. Nella configurazione attuale questo dispositivo non permette di ottenere modelli 3D con elevato dettaglio: ciò può essere ovviato diminuendo il passo del laser tra una scansione ed un altra e utilizzando degli algoritmi di interpolazione più sofisticati.
La configurazione attuale del sistema di visione risulta invece particolarmente adatta alla robotica mobile, sia per la velocità di scansione che per il peso e le dimensioni ridotte che lo rendo facilmente installabile anche su robot di piccole dimensioni.
Nella seconda fase del lavoro si è sperimento, con ottimi risultati, l'utilizzo del sistema di visione realizzato precedentemente su un robot mobile.
Si è quindi implementato un robot mobile in grado di effettuare il mapping di ambienti indoor. Le semplificazioni adottate hanno permesso di raggiungere un ottimo trade off tra precisione e velocità di elaborazione. Infatti, in meno di un secondo si è riusciti a scandire la scena antistante, aggiornare la mappa globale e calcolare il movimento successivo.
Si è cosi aggiunto in letteratura un esempio completo, di robot mobile in grado di esplorare l'ambiente circostante mediante un sistema di visione in luce strutturata rispetto all'utilizzo di un telemetro 2D. I risultati ottenuti con un dispositivo in luce strutturata non sono lontani, da quelli ottenuti con i telemetri 2D che per contro utilizzano un tecnologia molto più costosa.
Infine i diversi test hanno messo in luce, la necessità di utilizzare dei sensori odometrici, in modo da poter eseguire esplorazioni molto più accurate. Ciò è in pieno accordo con la filosofia di questo lavoro che non si pone come un progetto chiuso in se stesso, ma come il mattone base di un progetto più ampio che porti alla realizzazione di un robot in grado di navigare autonomamente nell'ambiente circostante eseguendo anche task di una certa complessità.

5.4. Sviluppi futuri.

Per la parte di scanner 3D, delle possibili estensioni sono:

- Scandire in modo continuo la scena utilizzando al posto del motore passo passo, un normale motore, in modo da ottenere una scansione densa di tutta la scena.
- Studiare ed implementare degli algoritmi più avanzati per il riconoscimenti di segmenti e superfici, in modo da effettuare un rendering più accurato.
- Studiare l'utilizzo di un supporto rotante, che permetta di scandire a 360° l'oggetto sul supporto

Invece per quanto riguarda la parte di robot mobile, le estensioni sono numerose. Infatti la costruzione di un robot mobile è un progetto che richiede di analizzare molti aspetti, di cui solo alcuni in questo lavoro sono stati studiati con il giusto dettaglio. Alcune possibili estensioni sono:

- Utilizzare due encoder per migliorare la stima della posizione del robot tramite odometria. In questo modo si otterrebbe un metodo per costruire una mappa più precisa.
- Implementare diversi algoritmi di mapping, e in particolare algoritmi per una copertura totale della mappa con un percorso minimo. Effettuare quindi un confronto e selezione l'algoritmo con il migliore compromesso tra velocità e copertura della mappa.
- Studiare il problema del mapping 3D, rispetto a quello 2D realizzato in questo progetto.
- Affrontare lo SLAM problem con metodi più avanzati come lo scan matching o gli approcci probabilistici tipo il filtro di Kalman.
- Utilizzare la mappa creata in fase di mapping, per eseguire operazione di path planning al fine di raggiungere dei punti di goal.
- Implementare meccanismi per un obstacle avoidance dinamica.

Digressione filosofica sul futuro della robotica

La robotica ed in particolare la robotica umanoide, persegue l'obiettivo di simulare il modo di agire di un essere umano. Questo obiettivo può essere diviso in tre parti. In primo luogo è necessario simulare il sistema sensoriale di un uomo nella sua notevole complessità ed efficienza. Successivamente la parte centrale consiste nel passare dai dati sensoriali ad una percezione della realtà, sulla quale prendere decisioni. L'ultima parte consiste nel nel tradurre le decisioni in azioni motorie, e simulare quindi la miriade di movimenti che riesce a fare un uomo.

A prima vista appare subito evidente, come la vera sfida sia simulare la capacità umana di percepire la realtà e decidere le proprie azioni in base ad un obiettivo e all'ambiente circostante. Ad una vista più attenta si può realizzare che anche la simulazione del sistema sensoriale e del sistema motorio umano so due ardue sfide: basti pensare alla complessità di un occhio umano, alle innumerevoli informazioni che si possono ricavare dal sistema tattile di una mano, o ai movimenti di un braccio. Naturalmente, per quanto complessi, nessuno di questi organi è comparabile in complessità al cervello umano, che si occupa di integrare percezione e azione.

E' necessario sottolineare come le difficoltà che sta affrontando la robotica umanoide nel perseguire i suoi obiettivi e gli scarsi risultati ottenuti finora sono del tutto normali. Si sta tentando, in poche decine di anni, di fare ciò che la natura ha fatto in quasi 4 miliardi di anni: il processo evolutivo culminato con la comparsa dell'uomo. E' importante a questo punto aprire una parentesi, sulla complessità della natura e vedere cosa l'uomo rappresenta in questo contesto. Proseguendo verrà più volte utilizzato il termine “natura” fortemente legato col concetto di evoluzione: è ovvio che questo termine può essere sostituito dai termini “universo”, “creato”, “Dio” in base alla concezione filosofica che ognuno ha di ciò che ci circonda. Prima di parlare della complessità della natura è bene capire cosa si intende per complessità. Secondo i matematici un sistema raggiunge un buon grado di complessità quando riesce a “parlare di se stesso” cioè riesce ad essere ricorsivo. Questo concetto è molto importante, e si ricava dalle importanti conseguenze sollevate dal teorema di Godel. Un esempio di sistema in grado di parlare di se stesso è la logica del primo ordine, arricchita degli assiomi per raggiungere la potenza dell'aritmetica. In questo sistema formale, è possibile scrivere teoremi sul sistema stesso e sulle sue proprietà: cioè il sistema è in grado di

uscire da se stesso, per guardarsi dall'esterno dimostrando teoremi sul suo funzionamento. Questa definizione che può sembrare prettamente matematica si può applicare bene alla realtà. In particolare possiamo dire che senza l'uomo, la natura non era sufficientemente complessa. Solo con l'uomo la natura riesce ad acquisire la capacità di parlare di se stessa. L'uomo ha studiato la natura di cui fa parte, ne ha scoperto il funzionamento, interagendo con esso. A questo punto, ci si riallaccia alla robotica, proprio quando il discorso sembrava aver preso tutt'altra direzione. L'uomo nel cercare di simulare un essere umano intelligente sta cercando, forse implicitamente, di creare la sua ricorsione di primo livello, cioè di creare qualcosa in grado di parlare di se stesso, di studiarsi, e di migliorarsi. Solo quando si riuscirà a raggiungere in un robot tale livello di complessità, si potrà dire che la robotica ha raggiunto l'obiettivo di simulare un essere umano. Fino ad un livello prima si avranno solo dei robot che sapranno fare niente di più di ciò che ci si aspetti che possano fare. Naturalmente Godel ci insegna che la complessità ha un prezzo: in un sistema formale significa rinunciare alla completezza. Senza troppa ironia possiamo dire che anche la natura, per mano dell'uomo, ha pagato il prezzo della sua complessità. Ci si domanda quindi a che punto l'uomo sarà in grado di gestire l'eventuale complessità del robot, che come si è intuito complessità è sinonimo di intelligenza. E' importante sottolineare che non ci sono limiti teorici che ci dimostrino l'impossibilità di creare un essere intelligente. Il problema è che non si può dire nulla alla domanda “quando ciò succederà” e soprattutto alla domanda “se succederà”.

A questo punto si potrebbe pensare che sia folle, fare ciò che la natura ha fatto in 4 miliardi di anni, in un tempo comparabile con la generazione umana. C'è una piccola grossa differenza. Per coloro che credono in Darwin, la natura è andata avanti a caso. L'uomo ha scelto tutt'altra via che quindi dovrebbe tagliare in partenza le soluzioni sbagliate. L'uomo nel creare il suo “essere complesso” sta copiano la natura nel risultato, ma non nel come si è arrivati al risultato. La natura o il caso, ha scelto una stessa unità base per ogni parte dell'uomo: la cellula. Una stessa unità che si differenzia a secondo delle funzioni che dovrà fare. Come una classe Java dalla quale si ereditano altre classi. Nella robotica il processo evolutivo sembra totalmente diverso: a secondo della funzione che si vuole ottenere si usano meccanismi e tecnologie completamente differenti. Sarebbe interessante provare a creare un essere intelligente senza perseguire la sua somiglianza con l'uomo, anche se si potrebbe avere difficoltà ad interagire con esso e rendersi conto della sua eventuale intelligenza.

Il tempo alla fine ci darà tutte le risposte: ci dirà se il metodo seguito dall'uomo per creare un essere intelligente sarà stato più efficiente o no di quello usato dalla natura.

Ringraziamenti

Nel lungo percorso che mi ha portato fin qui sono molti i ringraziamenti da fare. Questi tre anni a Milano, al Politecnico di Milano, sono stati molto intensi ma il mio percorso per diventare ingegnere inizia molto prima, nel lontano 7 novembre del 1984. Il primo ringraziamento va quindi ai miei genitori, Giovanna e Enzo, che voglio tanto bene e che mi hanno sempre motivato, stimolato e aiutato in questo lungo percorso.

Un ringraziamento particolare va a mio nonno Leone Carmelo, che tanto avrebbe desiderato essere qui in questo momento e vedermi laureato. Lo ringrazio per tutto: per la sua saggezza, per avermi insegnato ad essere forte, per essere stato mio modello in tante cose. Dedicare questa tesi a mio nonno, è il minimo che potessi fare rispetto a quanto lui ha fatto per me.

Ringrazio gli altri miei nonni, e gli altri parenti. E ringrazio la mia sorellina Laura.

La mia avventura universitaria a Milano, è iniziata esattamente 3 anni fa. In questi anni, alle persone che mi stavano accanto e che volevo bene se ne sono aggiunte di nuove. Una delle prime è Nicola, collega, coinquilino, compagno di avventure straordinario con cui ho condiviso tantissimi momenti in questi anni. E come dimenticare gli altri coinquilini di via Ampere, 95: Carlo, Siro e Marco. E le ragazze di fronte: Oriana, Marzia e Maria Lucia. E i paesani che studiano qui a Milano: Martino, Maurilio e Giorgio. E tutti i colleghi di università.

E un grazie particolare va ai miei amici paesani che studiano a Pisa: Salvo, Fabio e Marco. Ed Emanuele grande amico, e come dice lui, compagno di merende. Ringrazio i ragazzi di via pratale per avermi ospitato sia nei momenti più difficili in cui avevo bisogno di loro che nei momenti più belli.

E come dimenticare tutti gli altri amici: Tony, Concetto, mpa Ferla, Corallo. E le amiche: Elisa, Ludovica, Serena, Chiara. E Francesca.

Infine un grazie speciale a Valentina.

Tutte queste persone, ognuna in modo diverso, ognuna con la sua simpatia, ognuna con il suo particolare modo di essere, mi hanno aiutato a superare questi tre anni pieni di sacrifici, ma anche di traguardi, con dei momenti difficili ma soprattutto con tante gioie, tante nuove esperienze, e nuove amicizie.

Grazie.

Bibliografia

1. “The Digital Michelangelo Project: 3D Scanning of Large Statues”, Marc Levoy, Kari Pulli, Brian Curless, Szymon Rusinkiewicz, David Koller, Lucas Pereira, Matt Ginzton, Sean Anderson, James Davis, Jeremy Ginsberg, Jonathan Shade, Duane Fulk.
2. “A Desktop 3D Scanner Exploiting Rotation and Visual Rectification of Laser Profiles”, Carlo Colombo, Dario Comanducci, and Alberto Del Bimbo, Dipartimento di Sistemi ed Informatica - Via S. Marta 3, I-50139 Florence, Italy.
3. “Automatic Classification of Objects in 3D Laser Range Scans”, Andreas N¨uchter, Hartmut Surmann, Joachim Hertzberg, Fraunhofer Institute for Autonomous Intelligent Systems (AIS) - Schloss Birlinghoven ,D-53754 Sankt Augustin, Germany.
4. “Processing of laser scanner data—algorithms and applications” Peter Axelsson, Department of Geodesy and Photogrammetry, Royal Institute of Technology, 100 44 Stockholm, Sweden 2 December 1998.
5. “Digital Reunification of the Parthenon and its Sculptures”, Jessi Stumpfel, Christopher Tchou, Nathan Yun, Philippe Martinez, Timothy Hawkins,Andrew Jones, Brian Emerson, Paul Debevec - University of Southern California, Institute for Creative Technologies - 4th International Symposium on Virtual Reality, Archaeology and Intelligent Cultural Heritage (2003).
6. “A 3D laser range finder for autonomous mobile robots”, Hartmut Surmann, Kai Lingemann, Andreas N¨uchter and Joachim Hertzberg - Proceedings of the 32nd ISR(International Symposium on Robotics),19-21 April 2001; GMD - German National Research Center for Information Technology; AiS - Institute for Autonomous intelligent Systems, 53754 Sankt Augustin, Germany
7. “Sistemi di visione per la percezione della profondità e progettazione di un prototipo in luce strutturata” - Giordano Tamburrelli - Politecnico di Milano – 2005
8. “SIMULATORE DI UN ROBOT PER LA PULIZIA DI AMBIENTI BASATO SU MAPPE” - Roberto BRUSA,Paolo NICORA - Politecnico di Milano (2005)

9. “An Alternative Interpretation of Structured Light System Data” - Dr Alan M. McIvor – Industrial Research Limited – POP Box 2225, Auckland – New Zealand
10. “Grasping Unknown Objects with a Humanoid Robot” - Geoffrey Taylor and Lindsay Kleeman – (2002)
11. 2The Interactive Museum Tour-Guide Robot” - Wolfram Burgard, Armin B. Cremers, Dieter Fox, Dirk H¨ahnel, Gerhard Lakemeyer, Dirk Schulz, Walter Steiner, and Sebastian Thrun
12. “An autonomous mobile robot with a 3D laser range finder for 3D exploration and digitalization of indoor environments” - Hartmut Surmann∗, Andreas Nüchter, Joachim Hertzberg - Fraunhofer Institute for Autonomous Intelligent Systems (AIS), Schloss Birlinghoven, D-53754 Sankt Augustin, Germany (2003)
13. “Automatic 3D underground mine mapping” - Daniel F. Huber, Nicolas Vandapel - The Robotics Institute Carnegie - Mellon University - Pittsburgh, Pennsylvania 15213
14. “three-Dimensional Computer Vision” - Olivier Faugeras
15. “Computer Vision and Applications - A Guide for Students and Practitioners” - Bernd Jähne, Horst Haußecker
16. “Using 3D Laser Range Data for SLAM in Outdoor Environments” - Christian Brenneke, Oliver Wulf, Bernardo Wagner - Institute for Systems Engineering, University of Hannover, Germany
17. “A Very Low Cost Distributed Localization and Navigation System for a Mobile Robot” - Riccardo Cassinis, Paolo Meriggi, Maria Panteghini - Department of Electronics for Automation - University of Brescia
18. “Shape Reconstruction Incorporating Multiple Non-linear Geometric Constraints” - Naoufel Werghi, Robert Fisher, Anthony Ashbrook and Craig Robertson - Division of Informatics, University of Edinburgh
19. “An Efficient and Accurate Structured Light Algorithm for Direct Measurement of Cylindrical Surface Parameters” - Wolfgang Sorgel and Robert Schalkoff - Electrical and Computer Engineering - Clemson University
20. “Vision-based Mobile Robot Localization And Mapping using Scale-Invariant Features” - Stephen Se, David Lowe, Jim Little - Department of Computer Science - University of British Columbia - Vancouver, Canada

21. “Accurate 3D measurement using a Structured Light System” - R.J. Valkenburg, A.M. Mc Ivor - Industrial Research Limited - Auckland, New Zealand
22. “MICROROVER RESEARCH FOR EXPLORATION OF MARS” - Samad Hayati, Richard Volpe, Paul Backes, J. Balaram, and Richard Welch - Jet Propulsion Laboratory, California Institute of Technology - Pasadena, California 91109
23. “Mars Rover Navigation Results Using Sun Sensor Heading Determination” - Richard Volpe - Jet Propulsion Laboratory, California Institute of Technology - Pasadena, California 91109
24. “Modelli Approssimati per Localizzazione e Mapping mediante Robot Mobili” - Gian Diego Tipaldi - Università degli studi di Roma, Facoltà di Ingegneria – 2005
25. “Un criterio entropico per l'esplorazione con robot mobili” - Umberto Galtarossa - Facoltà di Ingegneria dell'Informazione, Dipartimento di Elettronica e Informazione del Politecnico di Milano - 2003
26. “homemade 3d scanner – the tutorial” - Roger Mueller - http://www.muellerr.ch/engineering/laserscanner/tutorial/the_tutorial.html