UNIVERSITÀ DEGLI STUDI DI ROMA
TOR VERGATA

FACOLTÀ DI INGEGNERIA

CORSO DI LAUREA IN INGEGNERIA DELL'AUTOMAZIONE

A.A. 2008/2009

Tesi di Laurea

REALIZZAZIONE DI UNA CELLA AUTOMATIZZATA PER IL TRACCIAMENTO DI ROBOT MOBILI

RELATORE

Ing. Francesco Martinelli

LAUREANDO

Loris Cerroni

a Debora, Mario e Luigina

la mia ragazza ed i miei genitori.

Indice

Ringraziamenti

Il lavoro di una tesi nasce e riflette il cammino affrontato in tutti gli anni di università. Desidero ringraziare la mia famiglia. I miei genitori per il supporto che mi hanno dato in tutti questi anni, per avermi permesso di studiare ed in particolar modo mio padre Mario. Ringrazio la mia ragazza Debora, la quale mi è stata accanto in ogni momento e mi ha sopportato in tutti questi anni ed infine mio zio Sebastiano per le conversazioni costruttive. Ringrazio il professor Francesco Martinelli per avermi dato l'opportunità di intraprendere questo particolare tipo di tesi, per la sua disponibilità e per l'avermi dato la possibilità di conoscere meglio il campo della Robotica e della Visione Artificiale. Ringrazio tutti i Professori del dipartimento che mi hanno insegnato in questi anni poiché ognuno di loro ha contribuito alla mia formazione e ha consentito il mio avvicinamento al mondo della Robotica e dell'Automazione. Infine ringrazio l'Università di Tor Vergata, questo mondo magnifico che mi ha reso in parte quello che sono, sperando che possa, col passare degli anni, essere ancora un punto di riferimento culturale e non un semplice momento di passaggio.

Introduzione

L'obiettivo di questa tesi è stato la realizzazione di una cella automatizzata per l'individuazione ed il tracciamento della posizione di robot mobili, nel caso specifico dell' OCTAGON in dotazione al laboratorio di"Hardware Pesante" dell'università di Tor Vergata (Figura 1).

Iniziamo col definire il concetto di cella, inteso come l'ambiente di lavoro nel quale si andrà ad operare, comprendente anche l'insieme delle strutture sia fisiche che software necessarie al funzionamento globale del sistema realizzato. Numerose problematiche sono state risolte con soluzioni originali. Il sistema realizzato deve essere in grado complessivamente di richiedere immagini ad una webcam, immagazzinarle in un'apposita directory, elaborarle e trarne informazioni riguardanti lo stato del robot, risulta dunque di fondamentale importanza ottenere un'integrazione tra "Spazio percepito" e "Spazio reale", in quanto risulta fondamentale associare alle immagini acquisite, lo stato del robot. È stata richiesta una struttura fisica facilmente assemblabile, in grado di poter essere richiusa agevolmente per esigenze di spazio, un sistema veloce e semplice da calibrare e naturalmente di costo contenu-

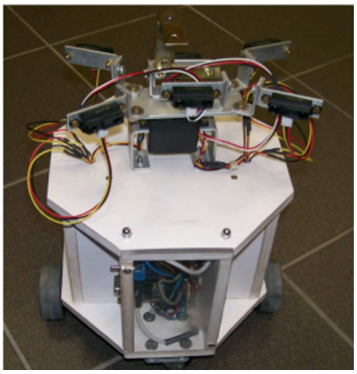


Figura 1: Robot Octagon

Figura 2: Webcam Philips SPC1300 NC e le sue caratteristiche tecniche

to. La realizzazione complessiva del progetto ha presentato numerose problematiche che andremo a sviluppare nel corso del testo. Il lavoro ha richiesto numerose fasi, delle quali sicuramente cinque risultano essere di maggiore interesse:

- fase di progettazione della struttura portante necessaria al sostenimento della webcam
- fase di realizzazione fisica di tale struttura
- studio ed eliminazione delle diverse distorsioni ottiche presenti nel sistema
- fase di elaborazione e stesura dell'algoritmo di calibrazione assi
- fase di sviluppo dell'algoritmo di ricerca ed individuazione del robot mobile nell'ambiente di lavoro.

Per l'acquisizione delle immagini è stata utilizzata un comunissima webcam da pc, Philips SPC1300NC (Figura 2) dotata delle caratteristiche tecniche riportate in figura 2.

Nel corso del testo verranno affrontati i seguenti argomenti: nel primo capitolo verrà riportata una descrizione accurata delle fasi di progettazione e realizzazione della struttura portante, nel secondo capitolo verrà trattata la visione artificiale tramite webcam e la risoluzione dei problemi riscontrati sia dal punto di vista hardware che dal punto di vista software per la realizzazione dell' algoritmo di calibrazione, nel terzo capitolo affronteremo il problema della ricerca e del tracciamento del robot ed infine il quarto ed ultimo capitolo sarà dedicato interamente all'analisi di tre esempi di funzionamento dell'applicazione realizzata. I contesti applicativi in cui è possibile utilizzare le tecniche discusse in questa tesi sono molteplici. Fondamentalmente l'applicazione è stata concepita come uno strumento che permetta di misurare in modo accurato la posizione di un oggetto nel piano di lavoro, anche in funzione del tempo, con lo scopo di ottenere un "ground true", ovvero un riscontro che risulta di fondamentale importanza nelle fasi di test di algoritmi di localizzazione. Presso la nostra università sono stati numerosi gli algoritmi di localizzazione realizzati, ad esempio tramite sensori installati sul robot stesso, ma non era stato ancora realizzato uno strumento che permettesse di effettuare un confronto tra i dati stimati dai sensori e quelli reali, senza il bisogno di effettuare fastidiose misurazioni manuali. Il lavoro di questa tesi mira inoltre alla realizzazione di uno dei mattoni fondamentali per la costruzione di sistemi robotici più avanzati che verranno poi trattati nel capitolo conclusivo.

Capitolo 1

Progettazione e Realizzazione della struttura

1.1 Progettazione

La struttura portante assolve il compito di sostenere il dispositivo ottico ad una determinata altezza da terra, in modo tale da poter inquadrare un piano di lavoro sufficientemente ampio. La costruzione di tale struttura è stata di fondamentale importanza per la realizzazione complessiva del progetto. La fase di progettazione si è rivelata molto utile, per la modellazione della struttura è stato usato un importante software di modellazione 3d chiamato "Cinema4d" (Fig. 1.1) della MAXON, non ancora molto affermato in ambito industriale ma con ottimi strumenti di cinematica diretta, inversa e con una capacità di rendering di gran lunga superiore ad altri software concorrenti [4], quali "Autocad","Archicad" e "3D StudioMax". Grazie a tale strumento è stato possibile progettare ogni parte della telaio nei minimi dettagli, ciò ha permesso la risoluzione di numerosi problemi, che verranno nel corso del testo esaminati, ben prima della realizzazione materiale, risparmiando in tale modo sia in termini di tempo che di costo della realizzazione.

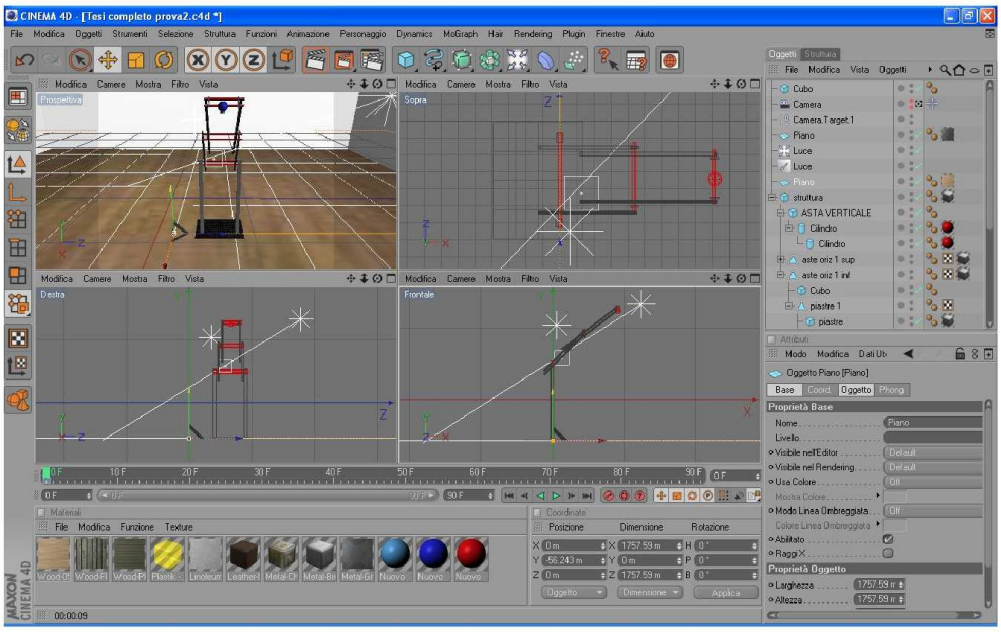


Figura 1.1: Cinema 4d, il software usato per la progettazione

Figura 1.2: Prima struttura progettata

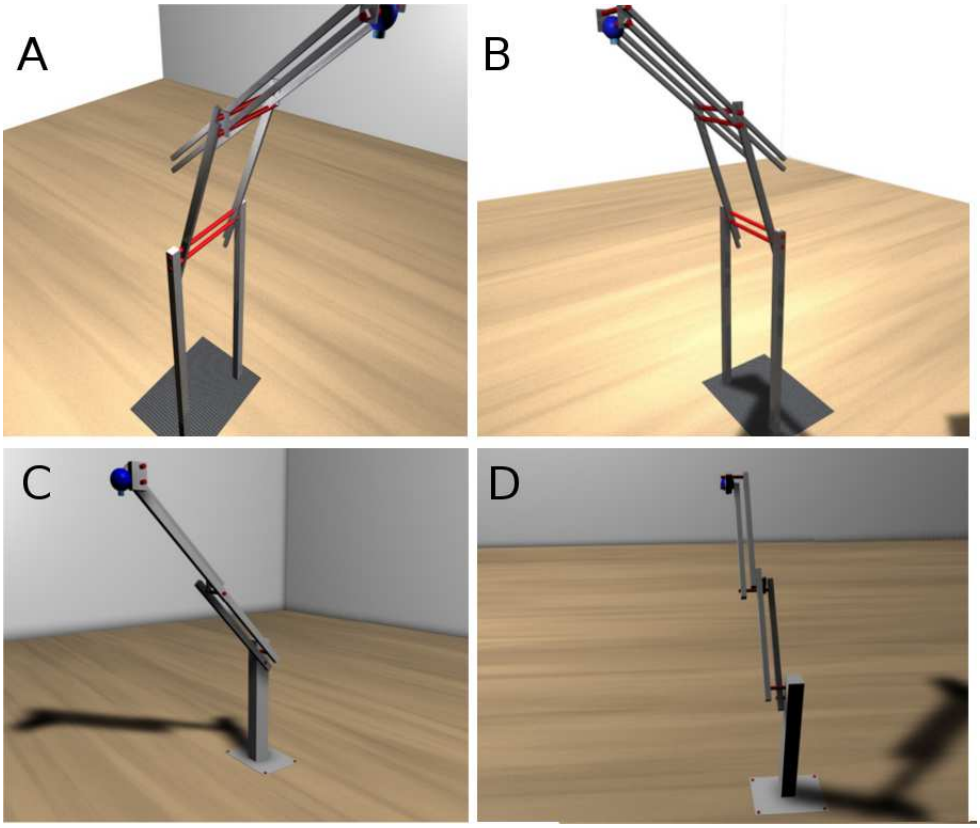


Figura 1.3: Le varie strutture progettate

Il primo modello ipotizzato (Fig. 1.2) è stato fin da subito scartato in quanto, anche se molto stabile, risultava essere poco funzionale a causa dell'impossibilità di modificare l'altezza della webcam da terra, inoltre il suo ingombro risultava notevole, poiché anche chiuso avrebbe occupato uno spazio pari all'altezza necessaria al monitoraggio del robot. La seconda soluzione (Fig. 1.3 A e B) riduce notevolmente il peso della struttura, inoltre permette il posizionamento della webcam all'altezza desiderata, mantenendo costantemente la perpendicolarità dell'inquadratura rispetto al terreno.

Infatti il particolare meccanismo usato, detto parallelogramma articolato (Fig. 1.4) , è costitutito da due coppie di bracci paralleli (2) tra loro collegati in serie. La prima coppia (Fig. 1.4, 2) di aste è collegata da un lato alla colonna portante (Fig. 1.4, 1) tramite due coppie rotoidali, dall'altro ad una piastra (Fig. 1.4, 3) necessaria a mantenere costante la distanza tra i centri di rotazione dei bracci stessi. In ogni istante del moto il segmento che congiunge i due assi di rotazione della piastra

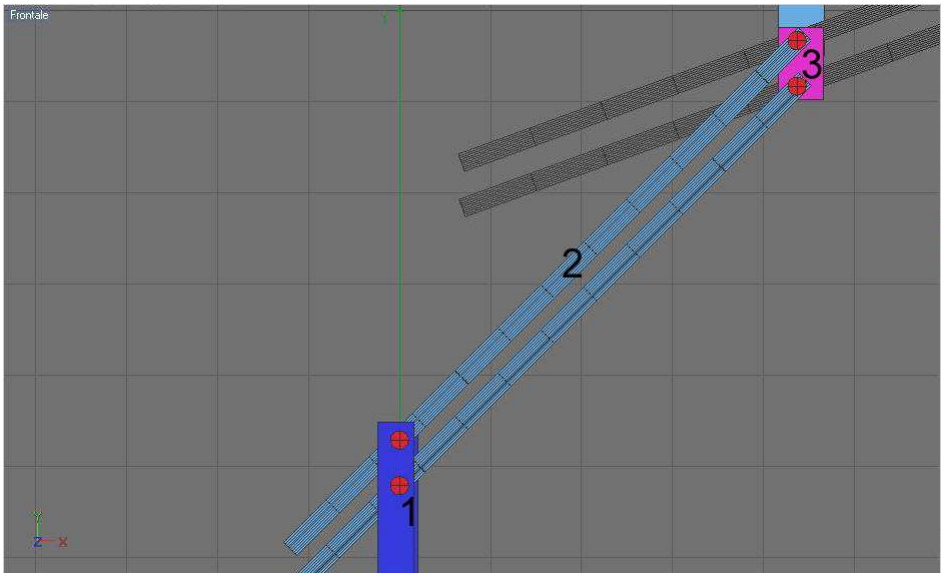


Figura 1.4: Meccanismo utilizzato

mobile (3) resta parallelo al segmento che congiunge i due assi della colonna (1), dunque dato che la distanza tra tali assi è fissa e dato che la posizione degli assi della colonna portante non varia rispetto al terreno, se la retta congiungente tali assi sarà perpendicolare a terra anche la piastra mobile risulterà essere perpendicolare al piano terra. Il meccanismo della seconda coppia di bracci è equivalente a quello appena descritto, infatti se è vero che la prima piastra mobile (3) sarà, per ogni posizione dei bracci, perpedicolare a terra[1], anche la seconda piastra mobile lo sarà a sua volta. La struttura poteva anche essere realizzata tramite un solo snodo, ma si è preferito adottare questa seconda soluzione in quanto, avendo più gradi di libertà, consente al meccanismo maggiore flessibilità d'uso ed un miglior rapporto sbraccio[2]/altezza, consentendo inoltre l'occupazione di minor spazio a struttura chiusa.

Posizionando il primo snodo a 45 gradi si ottiene il miglior compromesso tra distanza dalla base ed altezza, infatti in tale posizione la prima piastra può raggiungere i 2 metri da terra (Fig. 1.5) ed il secondo snodo può essere sfruttato, a secondo delle esigenze, per spostarsi sia lungo l'asse X che

[1] Tale affermazione risulta vera per angoli di apertura dei bracci superiori a 5 gradi, in quanto per angoli inferiori, a causa di qualche inevitabile gioco meccanico dovuto alla costruzione, la condizione di perpendicolarità non è mantenuta.

[2] Con il termine "sbraccio" si intende la distanza orizzontale tra il piede della struttura e la posione del dispositivo ottico.

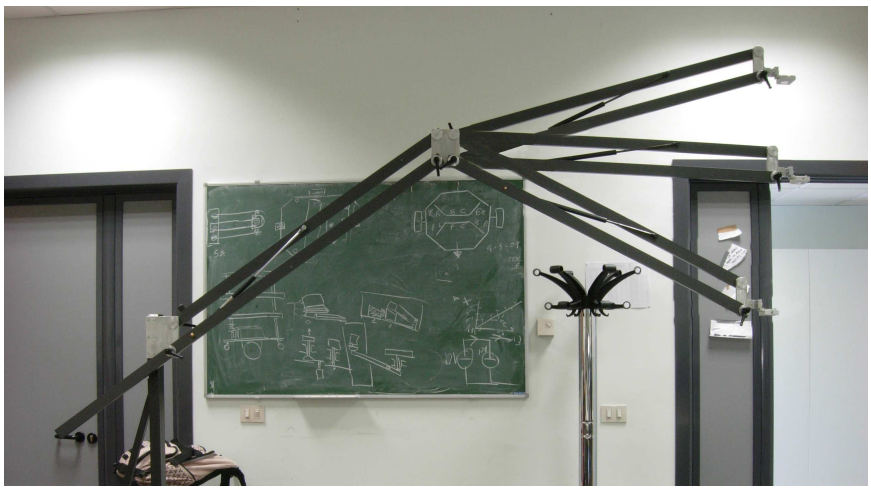


Figura 1.5: Diverse posizioni della struttura

lungo l' asse Y.

La terza struttura (Fig. 1.3 C e D) , cioè quella fisicamente realizzata, è stata derivata direttamente dalla seconda usando alcuni accorgimenti atti ad ottimizzarla, infatti il compito dell'ingegnere è quello di progettare uno strumento non solo efficace ma in particolar modo efficiente, in termini di spazio, peso, costo ed altri fattori che variano di caso in caso. In tale modello (C,D) sono stati eliminati i doppi bracci e la doppia colonna portante in quanto ritenuti eccessivi in rapporto al carico che realmente avrebbero dovuto sostenere, inoltre sono stati aggiunti alla base dei piedini di equilibratura regolabili in altezza, in modo tale che la struttura possa adattarsi ad un qualsiasi tipo di terreno, anche sconnesso.

Passiamo ora alla fase realizzativa di tale progetto, la quale ha richiesto molte ore di lavoro.

1.2 Realizzazione fisica

La scelta del materiale da utilizzare per la struttura è ricaduta sull'alluminio, il quale è sembrato fin da subito il giusto compromesso tra costo, resistenza, peso e facilità di lavorazione.

La prima struttura realizzata (Fig. 1.6)[3] con barre di alluminio di 1 cm di spessore risultava

[3]Come si può notare in tale figura la struttura una volta alleggerita tendeva a flettersi in quanto eccessivamente fragile.

Figura 1.6: Prima struttura realizzata

Figura 1.7: Struttura finale

troppo pesante da manovrare, per questo è stata alleggerita tramite asportazione di materiale sui due lati delle barre, ma tale alleggerimento ha portato ad una eccessiva fragilità della struttura.

La struttura definitiva invece è stata realizzata con tubolari 120x6x4 cm e con spessore di 1 mm che hanno dato alla struttura maggiore resistenza a parità di peso. Come si può osservare (Fig. 1.7) la struttura chiusa occupa poco più di 140 cm in altezza e 35 cm in larghezza ed a differenza del modello è stata dotata di una barra diagonale posteriore di stabilizzazione, che riduce sensibilmente le oscillazioni, ed una coppia di pesi da 5 kg che evitano il ribaltamento della struttura in condizioni

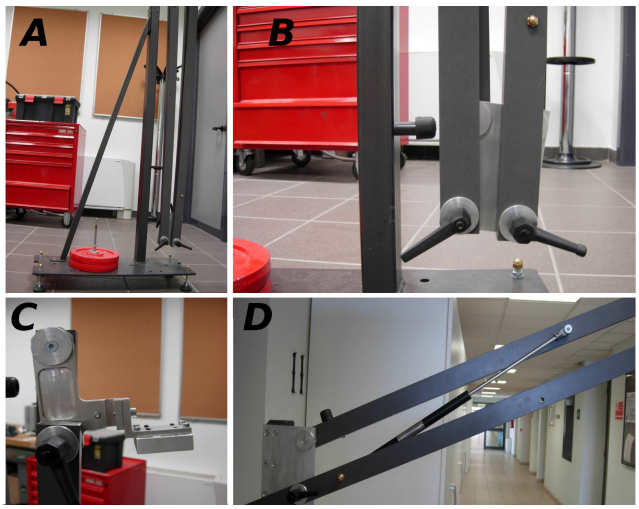


Figura 1.8: Dettagli della struttura finale

di completa apertura. Solo tre dei quattro bracci hanno una lunghezza pari a 120 cm mentre il quarto è stato di proposito lasciato più lungo in modo da consentire l'apertura della struttura con minor sforzo. Come già detto la struttura è stata completamente realizzata in alluminio tranne il blocco base-colonna realizzato in ferro al fine di assicurare maggiore stabilità.

Sulla colonna portante (Fig. 1.8 A e B) sono stati installati dei tamponi di battuta in gomma in modo tale da non consentire ai bracci paralleli una configurazione negativa[4]. Le maniglie di ripresa (Fig. 1.8 B) consentono il bloccaggio della struttura nella posizione desiderata, inoltre data l'elevata coppia che agisce su tali maniglie in fase di apertura completa, sono stati installati due pistoncini (Fig. 1.8 D) che compensano gran parte del peso gravante su di esse e aiutano l'apertura dei bracci. Infine, sulla seconda piastra, è installato un supporto per la webcam (Fig. 1.8 C) regolabile secondo due assi di rotazione (uno orizzontale, l'altro verticale) per consentire una migliore taratura della stessa. La telecamera può essere fissata al suo supporto (Fig. 1.9) inserendola nell'apposita sede e bloccandola tramite l'utilizzo di una chiave a brugola, inoltre nella parte posteriore del supporto sono presenti quattro "spinette" per l'eliminazione di un eventuale gioco meccanico residuo.

[4]Superata una determinata posizione, il parallelogramma formato dai due bracci assume una configurazione negativa nella quale i due lati più lunghi si incrociano. Questa configurazione non consente il mantenimento della condizione di perpendicolarità della piastra con la terra.

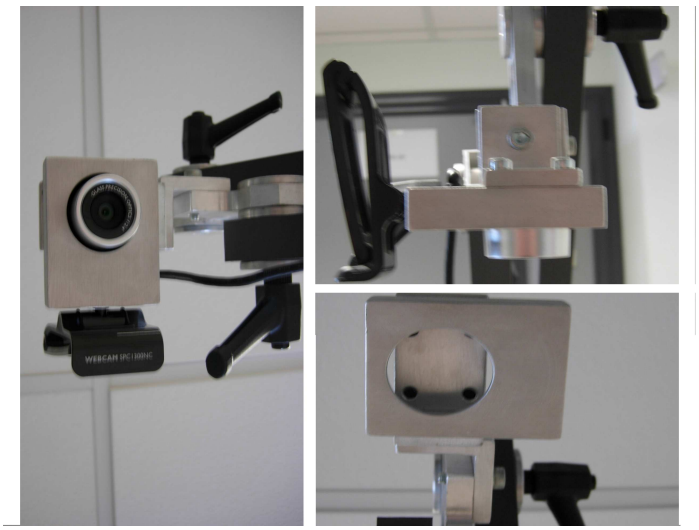


Figura 1.9: Supporto della webcam

Con il supporto realizzato (Fig. 1.10) la webcam potrà raggiungere senza alcuna difficoltà 3,5 m di altezza, ma cosa più importante, la fase di preparazione sarà semplice e rapida, in quanto una volta regolati i piedini della base in modo da livellare il dispositivo ottico, sarà sufficiente spostare la webcam nella posizione desiderata e bloccare il tutto con le maniglie di ripresa, senza ulteriori regolazioni.

Figura 1.10: Varie posizioni della struttura

Capitolo 2

Calibrazione assi della webcam

2.1 Acquisizione immagini

Le immagini neccessarie al nostro studio verranno acquisite dalla webcam installata sulla struttura descritta nel capitolo precedente, tramite CaptureMax 2.5[1], un sofware in grado di catturare in tempo reale le immagini ad un determinato intervallo temporale. Sembra opportuno prima di proseguire nello studio del problema dare un breve accenno riguardante la formazione delle immagini digitali, in quanto tali concetti saranno indispensabili per una piena comprensione dei problemi affrontati in seguito.

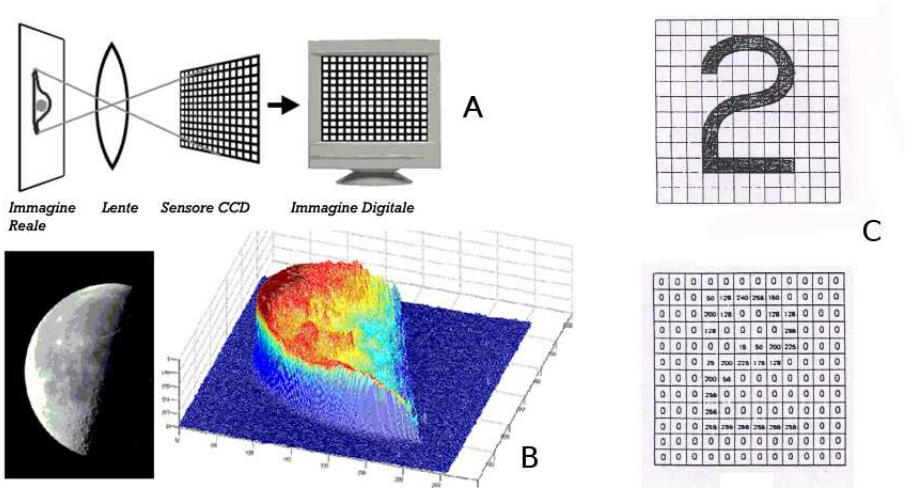


Figura 2.1: Formazione di un'immagine digitale

[1]Per una descrizione dettagliata sui settaggi e l'uso del programma vedi Appendice B.

L'immagine reale attraversa la lente del dispositivo ottico la quale convoglia tutti i raggi sul sensore CCD della fotocamera (Fig. 2.1 A), il quale registra l'intensità del fascio luminoso che colpisce ciascuna cella del sensore generando così una sorta di matrice di valori numerici (Fig. 2.1 B e C).

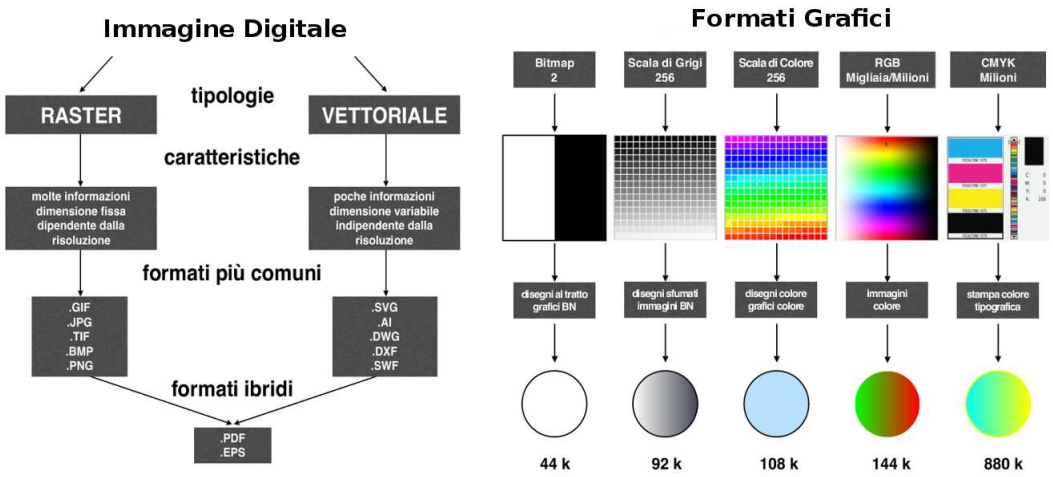


Figura 2.2: Diversi formati di un'immagine digitale

Le immagini digitali formate sono fondalmentalmente di due tipi: una matrice di punti (o pixel) nelle immagini raster o, nelle immagini vettoriali, un insieme di punti (o nodi) uniti da linee o altre primitive grafiche che compongono l'immagine, insieme ad eventuali colori e sfumature (Fig. 2.2 A). In questo tipo di immagini, i valori memorizzati indicano le caratteristiche di ogni punto dell'immagine da rappresentare (pixel): nelle immagini a colori, viene memorizzato solitamente il livello di intensità dei colori fondamentali (nel modello di colore RGB, uno dei più usati, sono tre: rosso, verde e blu). Un altro esempio è CMYK, usato per la stampa, basato su quattro colori fondamentali: ciano, magenta, giallo e nero (Fig. 2.2 B), mentre nelle immagini monocromatiche in scala di grigio (dette impropriamente bianco e nero) il valore indica l'intensità del grigio, che varia dal nero al bianco. Il numero (detto anche "profondità") di colori o di livelli di grigio possibili dipende dal massimo numero di combinazioni permesse dalla quantità di bit utilizzata per ognuno di questi dati:

un'immagine con 1 bit per pixel avrà al massimo due combinazioni possibili (0 e 1) e quindi potrà rappresentare solo due colori o solo bianco e nero, ad esempio nelle immagini a 4 bit per pixel, si possono rappresentare al massimo 16 colori o 16 livelli di grigio mentre in un'immagine a 8 bit per pixel, 256 e così via. Oltre a questi dati, è solitamente presente un header, che contiene diverse informazioni sull'immagine, a partire dal numero di righe e colonne di pixel: le dimensioni sono necessarie per poter dividere e disporre la sequenza di pixel in linee, in modo da formare una griglia rettangolare di punti, simile ad un mosaico, in cui ogni riga è formata da un numero preciso (indicato appunto dal valore larghezza) di tessere.

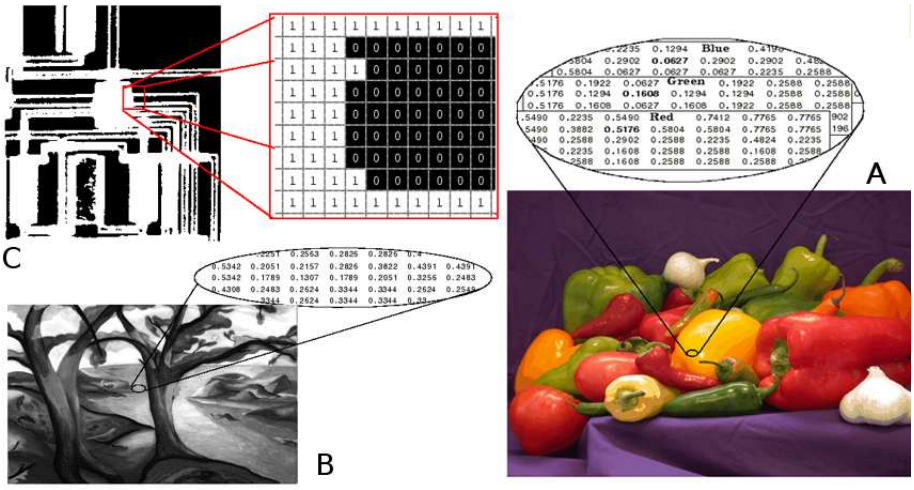


Figura 2.3: Formati utilizzati per il trattamento dei dati

Tra le varie estensioni esistenti per l'archiviazione e l'elaborazione delle immagini, si è scelto però di usare i seguenti formati in quanto ritenuti i più idonei al trattamento dei nostri dati. L'immagine iniziale sarà di tipo RGB la quale può essere rappresentata tramite tre matrici ciascuna relativa ad uno dei tre colori fondamentali rosso, verde e blue (Fig. 2.3 A). Essendo l'immagine acquisita ad 8 bit ciascuna matrice avrà dei valori compresi tra 0 e 255 che indicano l'intensità del colore presente in ciascun pixel dell'immagine. Dato però, che si sta sviluppando un sistema real-time questo tipo di immagine risulta essere troppo complessa da elaborare per l'individuazione degli oggetti presenti nell'immagine e per l'applicazione di filtri, cosa che richiederebbe molto tempo, dunque le immagini

verranno trasformate in altri due formati: "grayscale"[2] per l'applicazione dei filtri e "binario" per l'individuazione degli oggetti (Fig. 2.3 B e C). Le immagini grayscale sono caratterizzate da una sola matrice nella quale ciascun valore rappresenta l'intensità di grigio di ciascun pixel dell'immagine, al contrario nelle immagini binarie la matrice che descrive l'immagine è un array logico, ovvero sarà formata solo da valori binari 0 bianco, 1 nero.

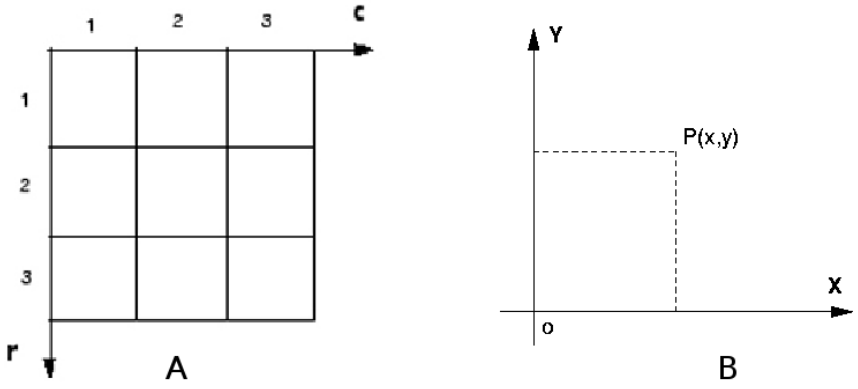


Figura 2.4: Differenza tra assi Pixel e assi Cartesiani

Altra importante caratteristica delle immagini digitali che non può essere trascurata è la disposizione del sistema di riferimento pixel, infatti l'origine delle coordinate degli assi (r e c) si trova in alto a sinistra (Fig. 2.4 A), diversamente dalla solita convenzione in basso a destra degli assi cartesiani (Fig. 2.4 B), questo fattore risulta molto importante nella fase di trasformazione tra coordinate dell'immagine e quelle del mondo reale.

2.2 Calibrazione assi

Generalmente con il termine calibrazione si intende quell'insieme di operazioni attraverso le quali uno strumento di misura viene tarato in modo da migliorarne l'accuratezza, tale operazione richiede il confronto con delle misure di riferimento prodotte utilizzando uno strumento campione, ovvero il nostro "calibratore". Una prima calibrazione dell'obiettivo è stata necessaria per l'eliminazione della distorsione radiale prodotta dal sistema ottico della webcam, infatti a causa dell'utilizzo di lenti

[2]Detta anche "gray scale".

di bassa qualità e dall'esigenza di avere una lente grandangolare, l'effetto di tale distorsione risulta rilevante in particolar modo ai bordi dell'immagine, tale problema è esaurientemente affrontato nell'Appendice A del presente testo. La fase di calibrazione degli assi risulta essere altrettanto importante in quanto, proprio in tale fase viene definito il sistema di riferimento del mondo reale, rispetto al quale verranno calcolate tutte le coordinate. Inoltre proprio tramite tale procedimento è possibile calcolare i legami che consentono di passare dalle coordinate dei punti calcolate in pixel, come centri degli oggetti presenti nella scena, a quelle riferite ad un sistema di riferimento esterno definito nel piano di lavoro del robot. Generalmente, anche nei sistemi industriali, i due assi del sistema di riferimento, X e Y, vengono associati al bordo inferiore e a quello sinistro dell'immagine stessa, tale consuetudine non è però molto conveniente in termini pratici, infatti nel caso in cui si volesse fare un confronto tra le coordinate stimate e quelle reali sarebbe necessario perdere molto tempo nel sistemare la webcam in modo tale da far coincidere i bordi dell'immagine con i riferimenti desiderati nel piano di lavoro. Sarebbe dunque, di gran lunga più pratico fare in modo che il software riconosca autonomamente gli assi di riferimento posizionati in maniera del tutto indipendente dalla webcam.

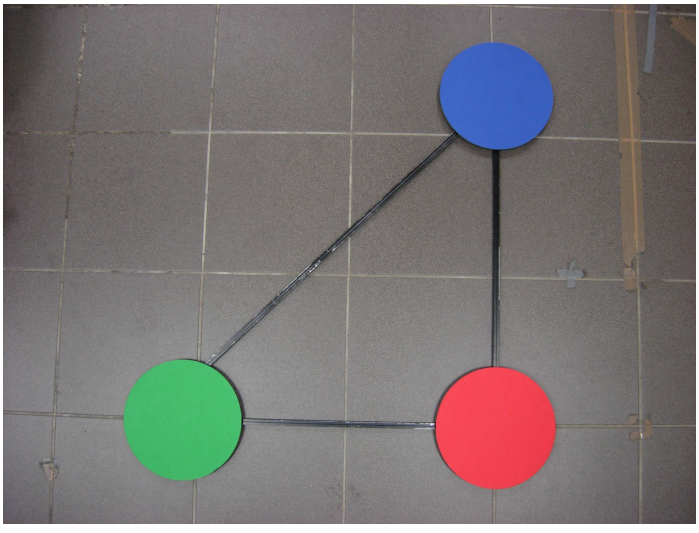


Figura 2.5: Il calibratore realizzato

Per rendere possibile tale soluzione è stato fisicamente realizzata una struttura di calibrazione (Fig. 2.5) che può essere posizionata a piacimento nel piano di lavoro del robot. Le piastre circolari dei vari colori sono quelle necessarie all'individuazione dei due assi X e Y, mentre la struttura triangolare in alluminio è necessaria solamente a mantenere in posizione i tre riferimenti, infatti un loro posizionamento errato nella fase di calibrazione comporterebbe un errore non trascurabile durante la fase di trasformazione delle coordinate da quelle immagine a quelle reali[3].

2.3 Filtraggio adattativo

Il primo passo da effettuare dopo l'acquisizione dell'immagine è sicuramente quello di eseguire un filtraggio della stessa, infatti senza un buon filtraggio sarebbe impossibile effettuare qualsiasi tipo di operazione. La fase più complessa di questo passo è stata quella di realizzare un algoritmo robusto in grado di riconoscere autonomamente il grado di filtraggio ottimo per ogni immagine, infatti come si può facilmente immaginare, esiste un'inevitabile variabilità delle condizioni operative dovuta ad esempio ad una diversa illuminazione della stanza, ciò comporta uno spostamento della soglia di equalizzazione dell'immagine e di conseguenza necessita l'uso di filtri con diversa intensità.

L'apparato software realizzato, al contrario della maggior parte dei programmi di questo tipo presenti anche in ambito industriale nei quali si utilizza uno sfondo neutro realizzato con appositi pannelli monocromatici e con l'ausilio di luci, può essere utilizzato anche in ambienti ricchi di rumore[4]. L'idea di fondo è quella di sfondare un'immagine iniziale e lasciare inalterati solo i nuovi oggetti, un po' come avviene negli effetti speciali dei film.

[3] Vedi Appendice A-Modello di Pinhole.

[4] Il termine rumore è qui inteso come disturbo, come immagine ricca di oggetti che potrebbero disturbare il normale funzionamento del software.

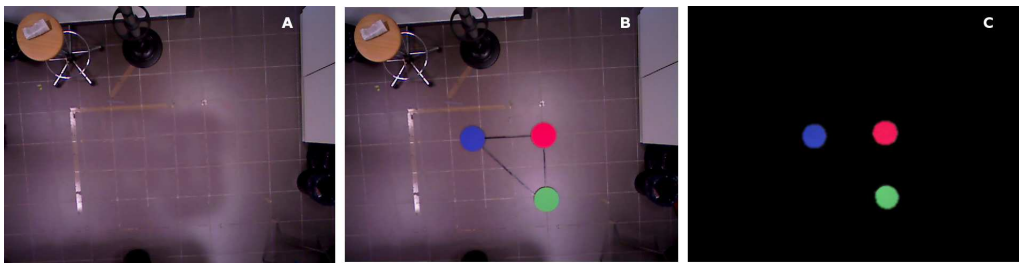


Figura 2.6: Dimostrazione pratica di come lavora il filtro rimuovi sfondo

Verrà scattata un'immagine iniziale di riferimento nella quale non vi siano presenti né robot né apparati per la calibrazione (Fig. 2.6 A), tale immagine verrà archiviata in memoria e confrontata con tutte le immagini scattate in seguito. Il software effettuerà una sottrazione tra l'immagine di riferimento e le successive lasciando così nello scatto solo le differenze tra le due foto (Fig. 2.6 B e C). Inizialmente questo primo tipo di filtraggio era stato realizzato direttamente dentro MatLab tramite il toolbox "Image Acquisition", ma dato che MatLab è un "interprete[5]" il programma realizzato era in grado di analizzare solo 1 fotogramma ogni 12 secondi, per questo motivo si è pensato di sfruttare una funzionalità offerta dalla webcam stessa, che lavora nello stesso modo ma risulta essere molto più veloce nell'esecuzione raggiungendo i 30 fotogrammi al secondo[6]. L'immagine acquisita sarà dunque pre-filtrata con tali accorgimenti, ma ciò non è sufficiente all'elaborazione dei dati necessari dunque è necessario effettuare ulteriori filtraggi.

[5] MatLab interpreta il proprio codice, ovvero la compilazione del programma va di pari passo con l'esecuzione dello stesso, dunque in generale un programma interpretato è di gran lunga più lento di uno compilato, in quanto cicli e ricorsioni sono ricompilate ogni volta.

[6] L'utilizzo di tale funzionalità è spiegata nell'Appendice B.

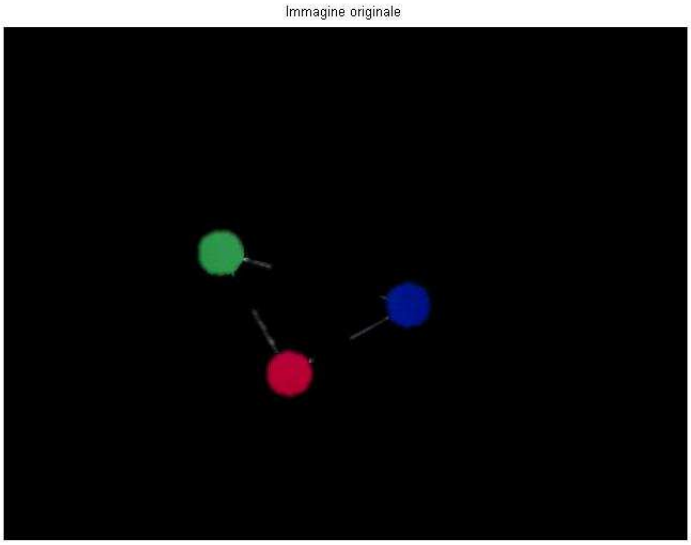


Figura 2.7: Immagine pre-filtrata

Nella maggior parte dei casi l'immagine acquisita è ancora ricca di rumore (Fig. 2.7), tale immagine viene aperta tramite il comando "*imread*(*nomeimmagine.estensione*)"[7] e assegnata ad una variabile I[8].

[7]Ogni volta che di seguito verrà spiegata la funzione di un comando MatLab si farà sempre riferimento alla nota bibliografica [5].

[8]Per il codice si faccia riferimento al paragrafo "Realizzazione software" del presente capitolo.

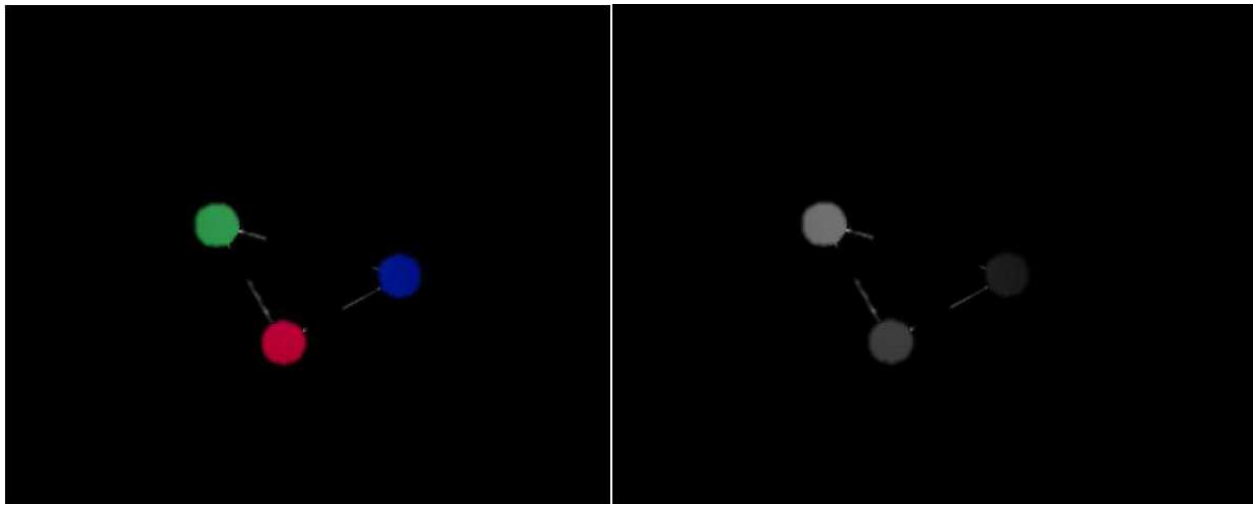


Figura 2.8: L'immagine Rgb viene trasformata in formato gray scale

Tale immagine, come già accennato è inizialmente in formato RGB, dunque prima di essere trattata con particolari filtri essa verrà trasformata in formato gray scale tramite il comando "$rgb2gray(I)$" (Fig. 2.8) e assegnata alla nuova variabile G. Tramite il comando "$J = imadjust(G)$" viene effettuata una prima sogliatura dell'immagine,

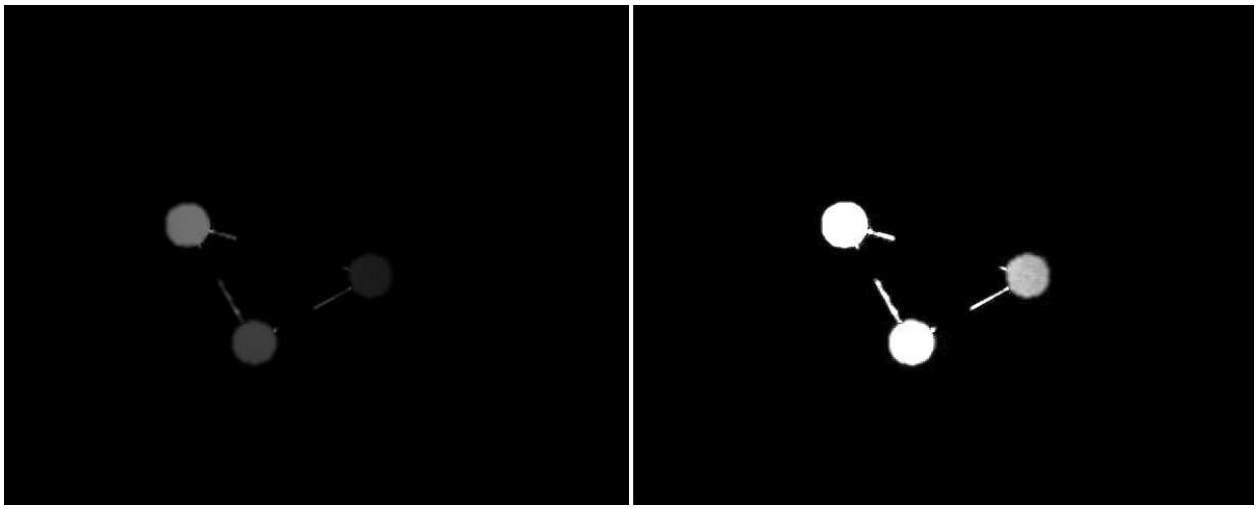


Figura 2.9: Risultato del comando imadjust sull'immagine

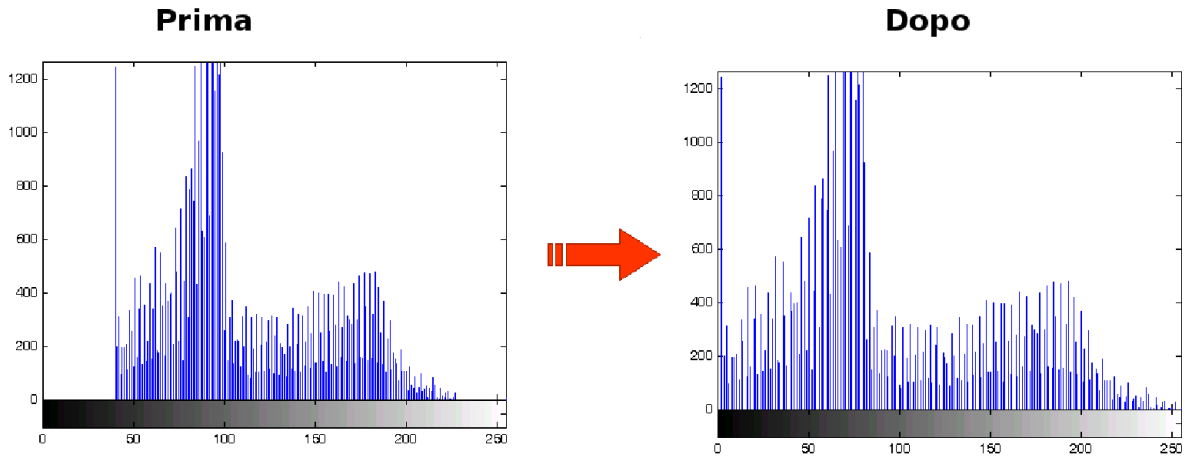


Figura 2.10: Istogramma dell'immagine prima e dopo l'applicazione del comando "imadjust(G)"

tale comando calcola il livello di contrasto limite dell'immagine (Fig. 2.9) modificandone i livelli di grigio, come è possibile notare anche nell'istogramma dell'immagine (Fig. 2.10) ottenibile tramite il comando "$imhist(G)$" ed assegna la nuova immagine calcolata alla variabile "J". Adesso "J" è pronta per essere filtrata, ed il primo filtro che verrà applicato sarà quello di Wiener.

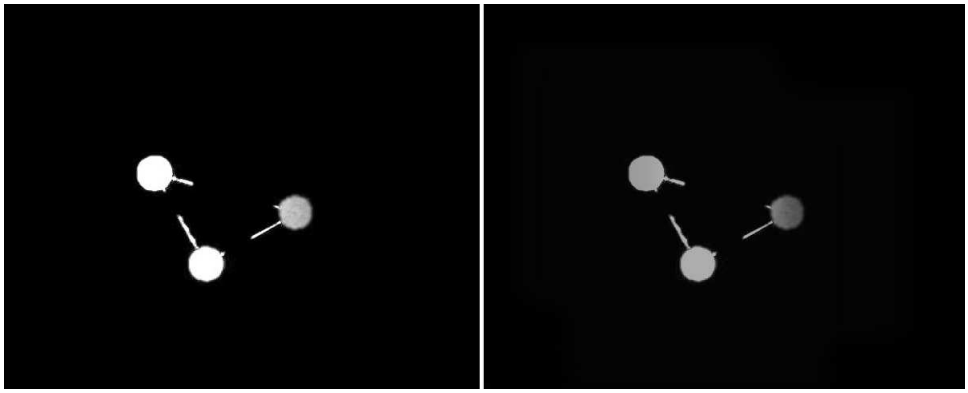


Figura 2.11: Risultato dell'applicazione del filtro di Wiener

Tale filtro passa-basso è di tipo adattativo, il suo scopo è quello di eliminare il rumore presente nell'immagine (Fig. 2.11), attraverso il comando "$J = wiener2(J, [\text{intensità X}][\text{intensità}Y])$" il filtro di Wiener, basato sulla stima statistica, calcola la media e la varianza relativa all'intensità di grigio nell'intorno di ogni pixel

$\mu = \frac{1}{M*N} \sum_{n1,n2 \in \eta} a(n_1, n_2)$

$\sigma^2 = \frac{1}{M*N} (\sum_{n1,n2 \in \eta} a^2(n_1, n_2)) - \mu^2$

dove MxN è la dimensione dell'intorno del pixel analizzato, η è l'insieme contenente tutti i pixel di tale intorno e n_1, n_2 rappresentano le coordinate di tali pixel, a questo punto il filtro crea una stima del disturbo [6]

$b(n_1, n_2) = \mu + \frac{\sigma^2 - \nu^2}{\sigma^2}(a(n_1, n_2) - \mu)$

dove b sarà la correzione effettuata su ciascun pixel dell'intorno e ν^2 è la varianza del disturbo, dopodiché usando tali dati stimati corregge l'immagine (Fig. 2.11).

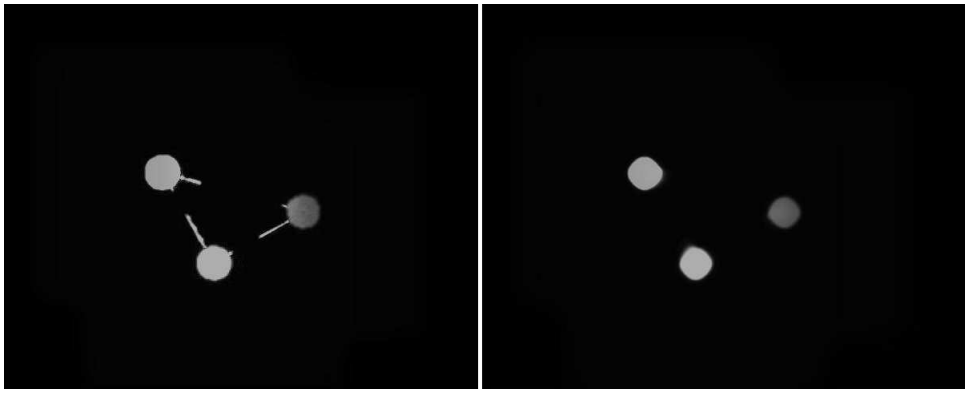


Figura 2.12: Effetto del filtro mediano sull'immagine

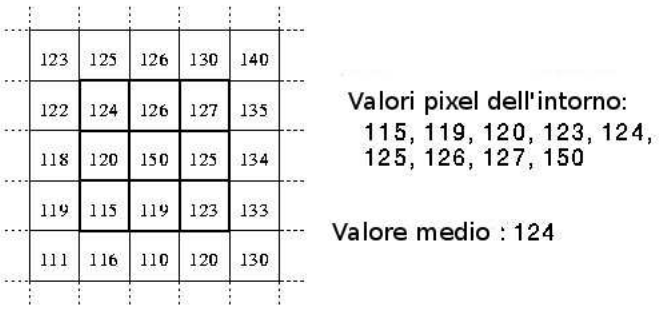


Figura 2.13: Come lavora il filtro mediano

Il secondo tipo di filtro utilizzato è il filtro mediano, applicandolo all'immagine tramite il comando "$J = medfilt2(J, dimensioneintorno)$" otteniamo una sensibile riduzione dei disturbi (Fig. 2.12) in quanto tale filtro esamina per ciascun pixel l'intorno della dimensione desiderata, ad esempio 3x3 (Fig. 2.13), raccoglie tutti i valori di intensità del grigio, li ordina in ordine crescente e ne trova il valore medio, dopodiché sostituisce il valore del pixel centrale con tale valore medio eliminando così gran parte dei disturbi.

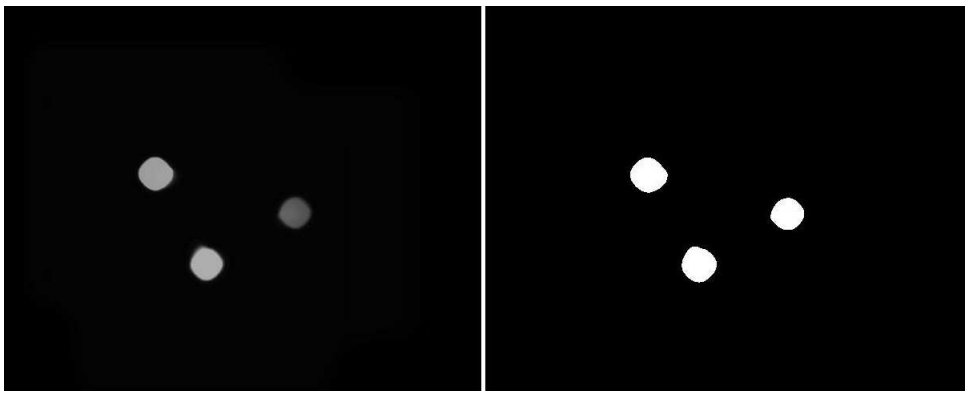


Figura 2.14: L'immagine gray scale viene trasformata in immagine binaria

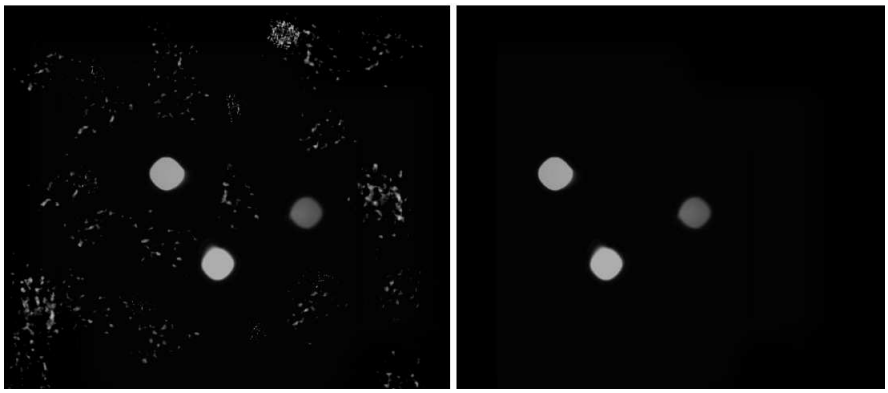


Figura 2.15: Applicazione del comando "bwareaopen"

A questo punto l'immagine è pronta per essere trasformata in immagine binaria (Fig. 2.14) attraverso il comando "$BW = im2bw(J)$". Fatto ciò, tramite il comando "$BWN = bwareaopen(BW, DimensioneMinim$ è possibile eseguire un ulteriore filtraggio (Fig. 2.15) che individua ed elimina tutti gli oggetti di dimensione minore a quella da noi indicata, infatti nel caso in cui fossero rimasti ancora dei disturbi residui tale filtro assicura la loro rimozione.

È ora possibile lanciare il comando "$[Z, numero] = bwlabel(BWN, 8)$" attraverso il quale MatLab è in grado di individuare il numero di oggetti di dimensione maggiore di 8 pixel presenti nella scena. Il risultato di tale operazione sarà la creazione di una matrice Z di questo tipo

$$Z = \begin{bmatrix} 0 & 0 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 0 & 0 & 0 & 0 & 0 & 0 \\ 1 & 1 & 1 & 1 & 0 & 0 & 3 & 0 \\ 1 & 1 & 0 & 0 & 0 & 3 & 3 & 3 \\ 0 & 0 & 0 & 0 & 0 & 0 & 3 & 0 \\ 2 & 2 & 2 & 0 & 0 & 0 & 0 & 0 \end{bmatrix}$$

della dimensione pari alla dimensione dell'immagine[9], costituita da numeri che individuano gli oggetti della scena, inoltre alla variabile "*numero*" viene assegnato il numero complessivo di oggetti presenti nel piano di lavoro.

La combinazione di questi tre filtri, unita ad un sistema di controllo intelligente che regola la loro intensità, porta ad ottimi risultati. La funzione complessiva di filtraggio è inserita all'interno di un

[9]Nel caso reale sarebbe di 640x480 ma nell'esempio è stata riportata una matrice minore.

ciclo "*while*" che prosegue fin quando nella scena non vengono individuati tre oggetti, ovvero fin quando la variabile "*numero*" risulta essere pari a tre[10], ovvero i riferimenti del nostro calibratore. Per realizzare ciò si è fatto uso di semplici operatori condizionali quali "if" ed "$elseif$", i quali nel caso in cui ci siano meno di tre figure presenti nella scena diminuiscono l'intensità del filtro di 10 unità, viceversa la incrementano.

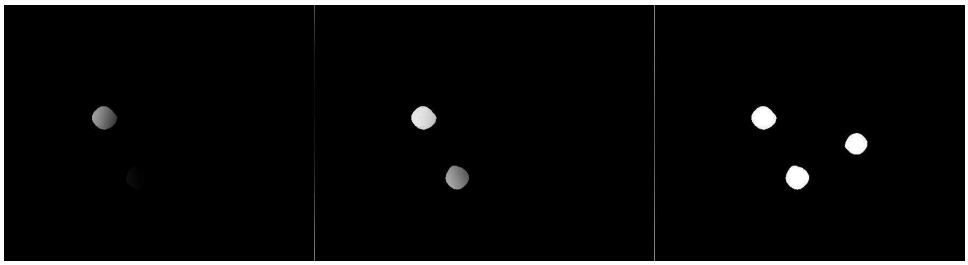


Figura 2.16: Risultati del filtraggio intelligente che modificando i valori dei filtri individua tre figure

Una volta individuati esattamente tre oggetti (Fig. 2.16) prima di terminare definitivamente il ciclo "*while*", viene effettuata un'ulteriore verifica per assicurarsi che i tre corpi individuati siano effettivamente i tre riferimenti desiderati. Il programma entra in un secondo ciclo "*while*" che ha il compito di verificare che le area delle tre figure siano uguali[11]. Tale operazione viene effettuata usando il comando

$[r, c] = find(Z == 1)$

$rc1 = [rc]$

$[r, c] = find(Z == 2)$

$rc2 = [rc]$

$[r, c] = find(Z == 3)$

$rc3 = [rc]$

in questo modo vengono individuati i tre oggetti e costruite le tre matrici "$rc1$", "$rc2$" e "$rc3$" le quali contengono le coordinate pixel espresse secondo gli assi r e c, come visto precedentemente, delle

[10] Per evitare un ciclo infinito è stata opportunamente inserita nel codice una funzione di controllo dei tentativi che blocca l'esecuzione del programma dopo 10 tentativi falliti tramite il comando "*break*".

[11] A meno di uno scarto in pixel da noi impostato.

tre figure. Una volta ottenute tali coordinate è possibile richiamando le funzioni

$area1 = bwarea(rc1)$

$area2 = bwarea(rc2)$

$area3 = bwarea(rc3)$

calcolare le aree dei tre oggetti e dunque eseguire un confronto per differenze tra di esse, nel caso reale se tale differenza risulta inferiore a 500 pixel, viene impostato il flag "$areaottima = 1$" che permette la terminazione del ciclo. Al contrario invece, in caso di esito negativo viene modificato il valore del filtro mediano di meno 5 unità per volta fino ad ottenere aree simili.
Tale operazione oltre a controllare che la dimensione delle tre figure sia simile, risulta essere molto importante per altre due ragioni:

- individua il valore di filtraggio ottimo, che verrà usato anche nella fase di individuazione del robot
- rendendo le figure della stessa dimensione, si avrà la quasi assoluta certezza che esse siano dei cerchi perfetti e dunque anche l'individuazione dei loro centri sarà più accurata.

2.4 Riconoscimento figure e selezione assi

Una volta riconosciuti i tre oggetti sul piano di lavoro è possibile passare alla fase di calibrazione vera e propria, cioè al calcolo dei loro centri e all'individuazione dei due assi X e Y. MatLab possiede già un proprio algoritmo per il calcolo dei centri delle figure tramite il comando "$regionprops(BW N,'centroid')$", ma tale comando risulta essere poco adatto ai nostri scopi in quanto sarebbe necessario estrarre l'oggetto di cui si vuole calcolare il centro dall'immagine, salvarlo come uno scatto a sé stante e solo dopo sarebbe possibile applicarvi tale comando. Per ragioni di praticità ed in particolar modo in quanto tali calcoli contribuivano ulteriormente a rallentare l'esecuzione del programma, si è deciso di realizzare un algoritmo di calcolo dei centri apposito.

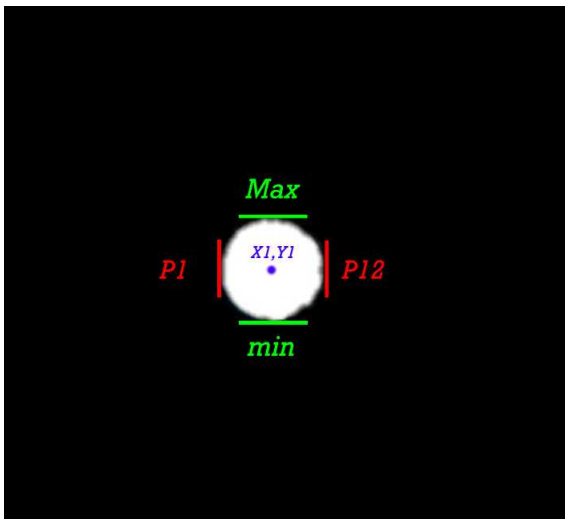


Figura 2.17: Calcolo del centro degli oggetti

Come visto nel paragrafo precedente le varibili "$rc1$", "$rc2$" e "$rc3$" sono matrici in cui sono contenute le coordinate pixel che ciascun oggetto occupa all'interno dell'immagine, proprio da esse è possibile calcolare il centro di ciascun corpo (Fig. 2.17) nel seguente modo:

- salviamo nella variabile $P1$ la prima coordinata delle ascisse, cioè la posizione più a sinistra occupata dall'oggetto
- calcoliamo tramite il comando $S1 = size(rc1)$ le dimensioni di tale matrice salvandole nella variabile $S1$
- la posizione $NROW = S1(1,1)$ conterrà il numero di righe delle matrice $rc1$
- nella variabile $P12 = rc1(NROW, 2)$ verrà salvato il valore limite di destra occupato dalla figura
- è ora possibile calcolare il valore medio delle X occupate $META = \frac{(P12-P1)}{2}$ e dunque quella che sarà effettivamente l'ascissa del centro dell'oggetto[12]

[12]Nel caso in cui $P12 - P1$ sia un numero dispari la sua metà viene approssimata per eccesso.

- la coordinata X del centro sarà dunque pari a $X1 = META + P1$
- per il calcolo della coordinata Y del centro salviamo invece in vettore $RC11 = rc1(:, 1)$ ovvero tutte le righe della prima colonna di $rc1$
- troviamo il valore minimo e quello massimo tramite i due comandi $m = min(RC11)$ e $M = max(RC11)$
- è ora possibile trovare il valore medio come fatto in precedenza per le ascisse $Y1$
- abbiamo a questo punto le coordinate del centro dell'oggetto $[X1\ ,\ Y1]$
- ripetendo tali operazioni per le altre due figure possiamo calcolare anche i loro centri.

Il calcolo delle coordinate di tali centri risulta essere alquanto precisa, infatti l'errore medio, appurato durante le fasi di test del programma, non supera il valore di un pixel, ovvero alle normali altezze di lavoro l'errore risulta essere inferiore allo $0,02\%$.

Una volta ottenute le coordinate dei centri delle tre figure il software è in grado di riconoscere autonomamente i due assi in base ai colori dei tre riferimenti. Il "tag" rosso individua l'origine degli assi mentre i restanti due, verde e blue, sono posti rispettivamente lungo l'asse Y ed X. Ricordiamo che ciascun pixel dell'immagine originale RGB è descritto da tre valori numerici compresi tra 0 e 255 che indicano i livelli di intensità di ciascun colore rispettivamente rosso, verde e blue, proprio sfruttando tale caratteristica delle immagini RGB risulta possibile distinguere i tre centri.

Inizialmente tramite i comandi

$COLORE1 = impixel(I, X1, Y1)$

$COLORE2 = impixel(I, X2, Y2)$

$COLORE3 = impixel(I, X3, Y3)$

verranno interrogati i pixel corrispondenti ai tre centri precedentemente individuati, e a ciascuna delle variabili "$COLORE$" sarà assegnato un vettore caratterizzato dai tre indici numerici precedentemente descritti. Dopodiché tramite una serie di operatori condizionali "if" verranno assegnati

i valori delle generiche coordinate "$X1$", "$Y1$", "$X2$" ecc.. che non contengono alcuna informazione riguardante il colore, a delle variabili specifiche, come "rx" e "ry", che indicano appunto l'appartenenza di tale coordinata al colore rosso, stessa cosa avviene per le altre coordinate. L'appartenenza di un pixel ad un colore oppure ad un'altro è determinata da alcuni valori di soglia del vettore "$COLORE$", è bene tenere presente infatti che a ciascun colore corrisponde esclusivamente una combinazione dei tre livelli di rosso, verde e blue e che tale colore può essere formato solo se i valori dei tre colori restano in un determinato intervallo, facciamo un'esempio: nel caso del colore rosso i tre livelli di intensità da rispettare saranno

- $rosso > 170$
- $verde < 100$
- $blue < 170$

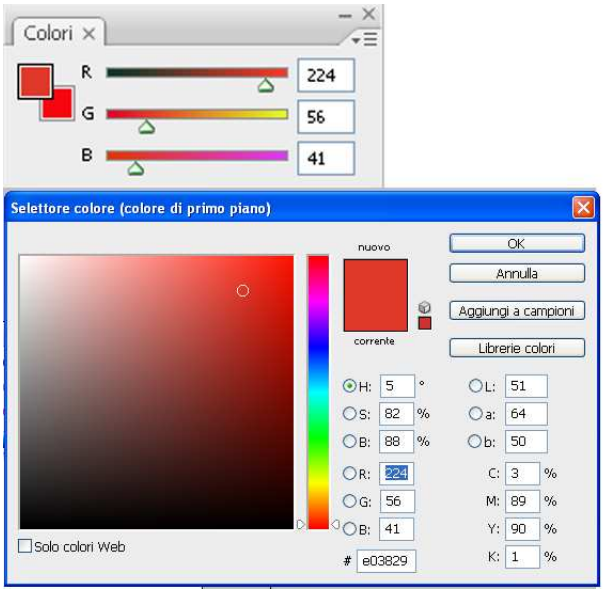


Figura 2.18: Verifica pratica della formazione di un colore

é semplice verificare l'esattezza di questi livelli, basta avere un qualsiasi software di fotoritocco,

andare sulla tavolozza dei colori (Fig. 2.18) e provare a modificare i livelli dei tre colori nei range indicati, ciò che si otterrà, modificando tali valori, sarà una vasta gamma di gradazioni di rosso. La scelta di tale range è molto importante per ciò che riguarda la robustezza del programma, infatti anche piccole modifiche nell'illuminazione della stanza, comportano grandi variazioni nelle tonalità dei colori dei riferimenti. Conoscendo ora con esattezza le tre coordinate $[rx,\ ry]$, $[gx,\ gy]$ e $[bx,\ by]$ sappiamo che l'asse delle ascisse sarà orientato nella direzione rosso-blue e a sua volta l'asse delle ordinate lungo la direzione rosso-verde. Sfruttando tale conoscenza è possibile ricavari alcuni dati molto importanti per il prosieguo del lavoro, come l'orientamento di tali assi rispetto ai bordi dell'immagine ed il rapporto pixel-cm.

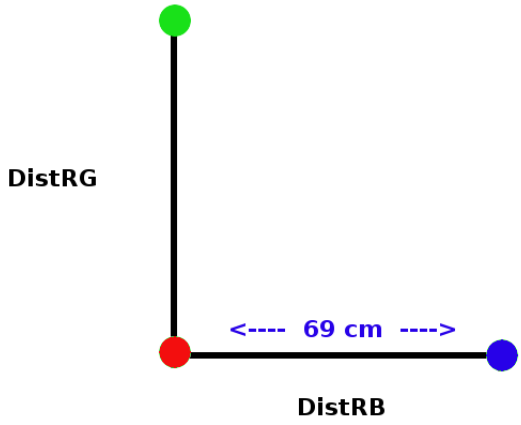


Figura 2.19: Confronto tra le diagonali

È importante in fase di calibrazione effettuare un confronto tra la misura in pixel della diagonale reale e quella ideale (Fig. 2.19), con lo scopo di verificare se gli assi stimati sono realmente perpendicolari tra loro, ciò è vero se e solo se le due diagonali calcolate coincidono[13].

[13] A meno di un'errore che non deve superare lo 0.99%.

$DistRG = \sqrt{(rx - gx)^2 + (ry - gy)^2}$

$DistRB = \sqrt{(rx - bx)^2 + (ry - by)^2}$

$DiagonaleIdeale = \sqrt{DistRB^2 + DistRG^2}$

$DiagonaleReale = \sqrt{(bx - gx)^2 + (by - gy)^2}$

$Distmedia = \frac{(DistRG + DistRB)}{2}$

$scartodiagonale = \frac{((DiagonaleIdeale - DiagonaleReale) * 100)}{DiagonaleIdeale}$

Infatti se tali assi non risultano essere a 90 gradi tra loro la successiva trasformazione da coordinate rispetto ai bordi dell'immagine a coordinate rispetto all'asse mobile da noi calibrato risulterà errata. Sfruttando la conoscenza della distanza tra i riferimenti del calibratore, che risulta essere di 69 cm lungo gli assi, è possibile calcolare il rapporto pixel-cm attraverso una semplice trasformazione lineare

$Distanza = 69cm$

$cm = \frac{(Distanza * PosPixelRob)}{Distmedia}$

dove "$PosPixelRob$" contiene le coordinate in pixel del robot non ancora calcolate e dunque per il momento impostate a zero.

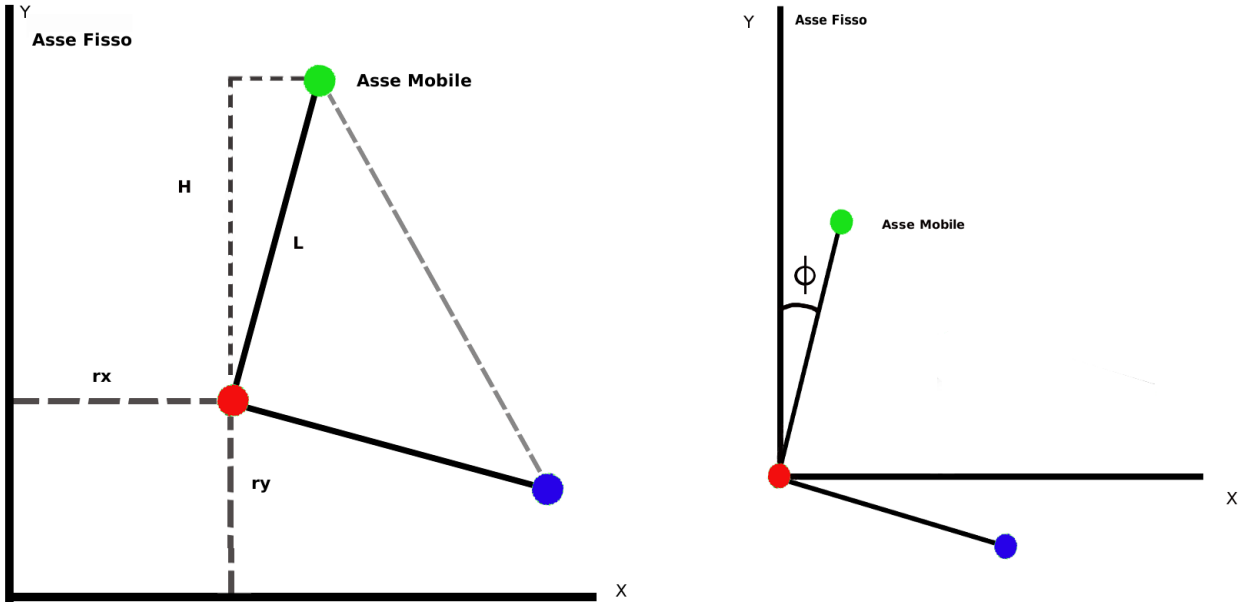


Figura 2.20: Movimenti necessari al passaggio di coordinate

Sempre durante la fase di calibrazione viene effettuato il calcolo riguardante i valori della matrice di rototraslazione necessaria al passaggio tra le coordinate dell'asse fisso[14] a quelle dell'asse mobile (Fig. 2.20).

$$M = \begin{bmatrix} cos\phi & -sen\phi & -rx * cos\phi + (altezzaim - ry) * sen\phi \\ sen\phi & cos\phi & -rx * sen\phi - (altezzaim - ry) * cos\phi \\ 0 & 0 & 1 \end{bmatrix}$$

dove (Fig. 2.20)

$H = \sqrt{(ry - gy)^2}$

$L = DistRG$

$\phi = arccos(\frac{L}{H})$

essa deriva dal fatto che

[14]Con tale termine si indicheranno da adesso in poi le coordinate riferite ai bordi dell'immagine, mentre si userà il termine asse mobile per indicare le coordinate rispetto all'asse del calibratore da noi posizionato.

$$\begin{bmatrix} Xmobile \\ Ymobile \\ 1 \end{bmatrix} = \begin{bmatrix} cos\phi & -sen\phi & 0 \\ sen\phi & cos\phi & 0 \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} 1 & 0 & -rx \\ 0 & 1 & -ry \\ 0 & 0 & 1 \end{bmatrix} * \begin{bmatrix} Xfisso \\ Yfisso \\ 1 \end{bmatrix}$$

è necessaria prima una traslazione che porti il centro di istantanea rotazione (Fig. 2.20), ovvero l'origine degli assi mobili, a coincidere con l'origine del riferimento fisso, l'angolo in basso a sinistra dell'immagine, in seguito sarà necessaria una rotazione pari ad un angolo ϕ, opportunamente calcolato, in senso antiorario che porti a far coincidere i due sistemi di riferimento.

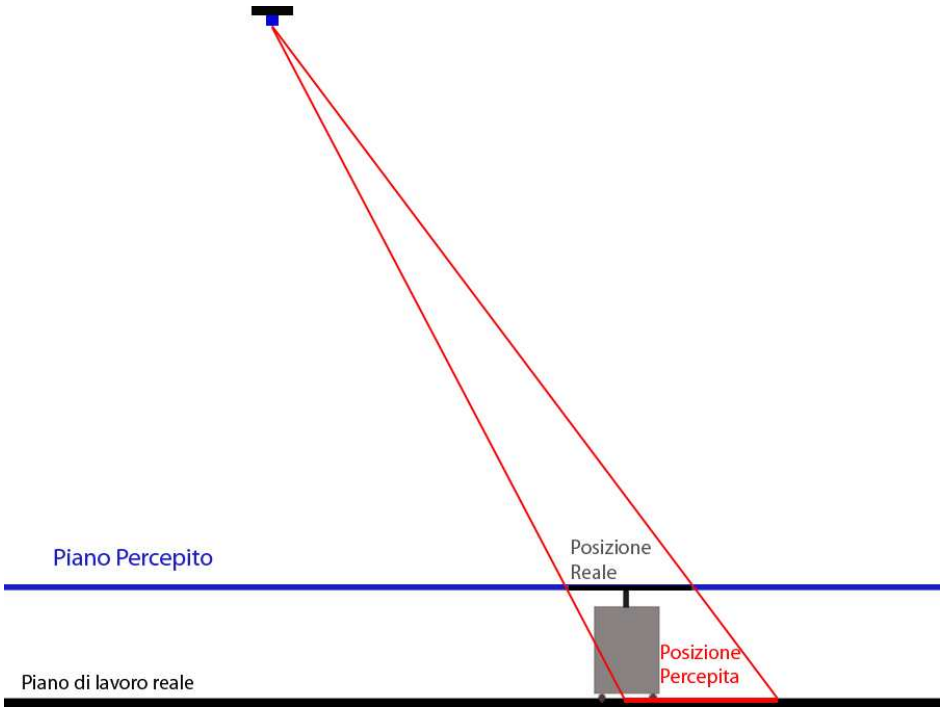


Figura 2.21: La distorsione prospettica

Un secondo tipo di distorsione da tenere bene presente è quella dovuta alla distorsione prospettica (Fig. 2.21) , la quale anche se può essere considerata trascurabile nella fase di calibrazione, in quanto i riferimenti del calibratore sono a circa 1,5 cm dal piano di lavoro, al contrario comporta un'errore notevole in fase di tracciamento della posizione del robot. Tale tipo di distorsione è dovuta al fatto che l'immagine del robot risulta traslata rispetto alla sua posizione reale a causa della sua altezza.

Infatti il riferimento del robot si trova ad una altezza di 35 cm da terra (Fig. 2.21), dunque la sua immagine viene proiettata sul piano di lavoro reale e la sua posizione risulta essere errata. Tale errore aumenta allontanandosi dalla webcam e risulta essere dipendente oltre che da tale distanza anche dall'altezza da terra, per questo motivo risulta di fondamentale importanza anche effettuare una stima dell'altezza del dispositivo ottico.

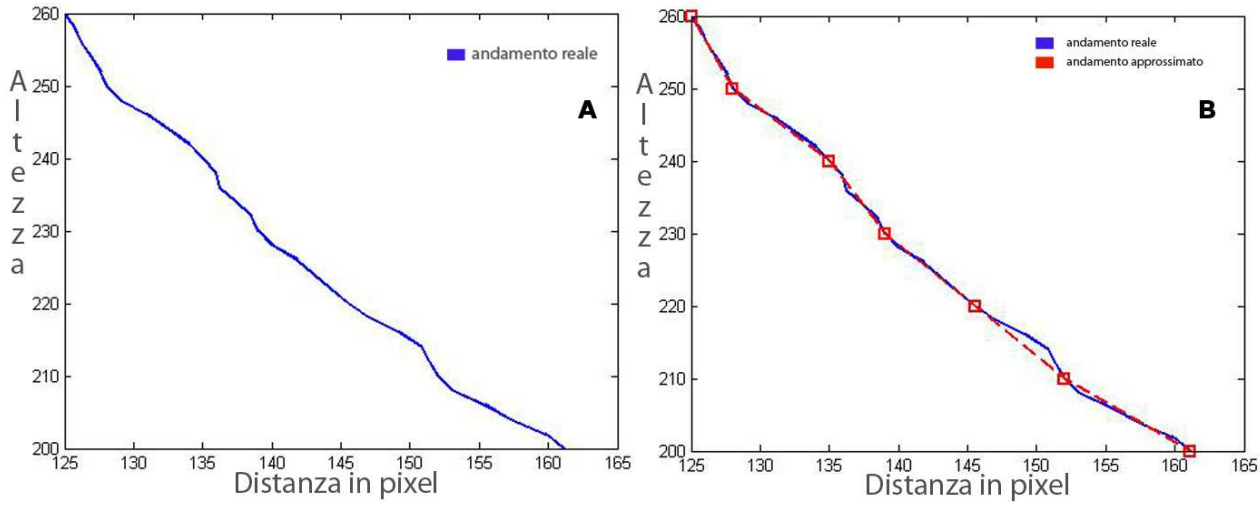


Figura 2.22: Andamento della distanza in pixel tra i riferimenti del calibratore in funzione dell'altezza e la sua approssimazione linearizzata

Essendo la distanza tra i riferimenti del calibratore fissa e pari a 69 cm è possibile, esaminando la dimensione di tale distanza in pixel ricavare una stima molto accurata dell'altezza della webcam. È stato possibile ricavare delle ottime approssimazioni di stima dell'altezza raccogliendo numerosi dati sperimentali, in tale modo si è potuto tracciare l'andamento della curva (Fig. 2.22 A) che descrive in che modo varia la distanza in pixel tra due riferimenti del calibratore, ad esempio tra rosso e blue, in funzione dell'altezza[15]. Tale curva non presenta, come si vede, un andamento lineare ma è stato possibile approssimarla discretamente (Fig. 2.22 B) tramite una serie di rette spezzate passanti per due punti:

[15]Per una miglior stima viene effettuata una media delle distanze in pixel calcolate tra i riferimenti rosso-blue, rosso verde.

$$y = mx + q$$

$$y - y_A = \frac{y_B - y_A}{y_B - y_A} * (x - x_A)$$

dove "y_A" e "y_B" risultano le due altezze della webcam prese in considerazione e "x_A", "x_B" risultano essere i valori relativi della distanza media tra i due riferimenti del calibratore calcolate in pixel.

Le equazioni ricavate sono:

$$stimaaltezza = (-1.9 * Distmedia + 485)$$

$$stimaaltezza = (-1.8174 * Distmedia + 482.6478)$$

$$stimaaltezza = (-1.5386 * Distmedia + 443.8913)$$

$$stimaaltezza = (-1.5338 * Distmedia + 443.1942)$$

$$stimaaltezza = (-1.1036 * Distmedia + 377.7850)$$

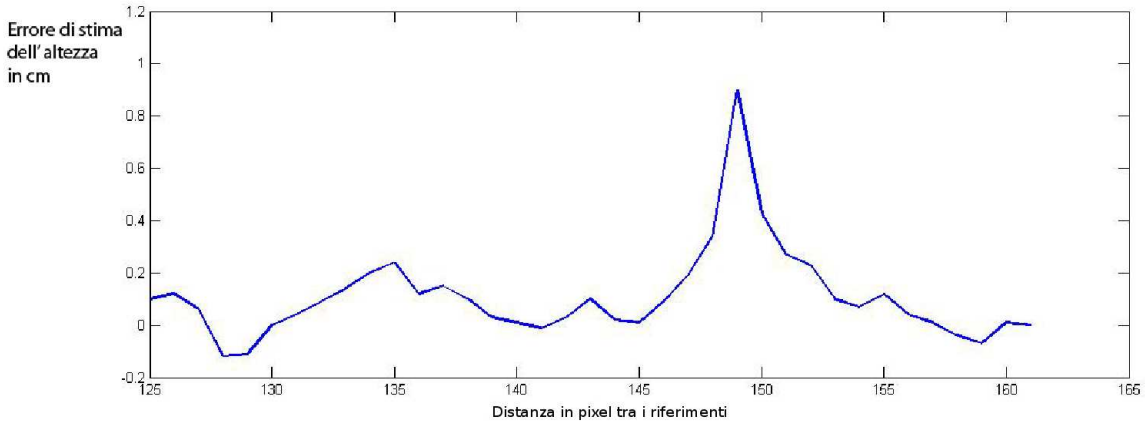


Figura 2.23: Andamento dell'errore di stima dell'altezza

una volta effettuta la misurazione della variabile "*Distmedia*", in base a tale valore, il software sceglie quale tra le equazioni calcolate approssima in maniera migliore la curva precedentemente

calcolata, in tale modo l'errore di stima dell'altezza non sale oltre il valore di 1 cm (Fig. 2.23). Dopo aver ottenuto una buona approssimazione dell'altezza possiamo tornare ad affrontare il problema precedentemente accennato riguardante la correzione della distorsione prospettica.

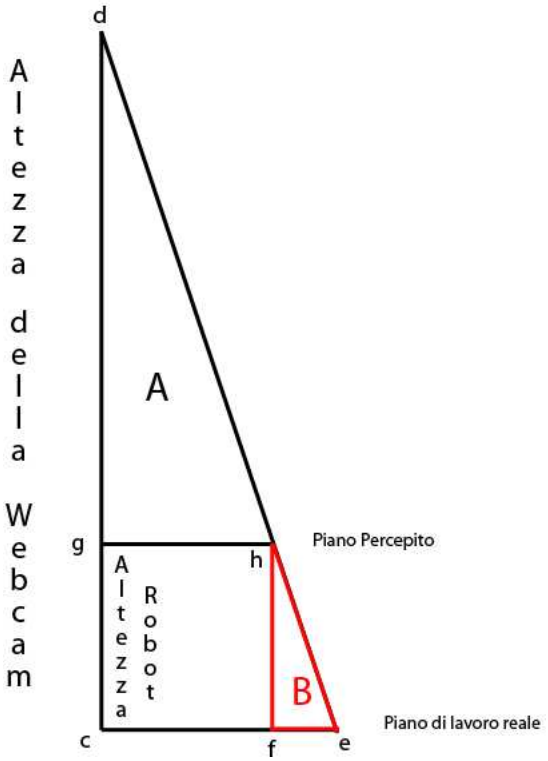


Figura 2.24: Analisi della distorsione prospettica

Tale errore presenta un andamento relativamente lineare, infatti è possibile notare un omotetia diretta[16] tra i due triangoli dei raggi prospettici (Fig. 2.24) A e B, dunque risulta valida la seguente relazione:

$\overline{(cd - cg)} : x = \overline{fh} : 10$

dove il valore 10 è necessario per il calcolo della distanza del robot dalla webcam $\overline{gh}$, alla quale, sul piano di lavoro, si ha un errore di posizione pari a 10 cm, mentre "x" risulta essere la nostra incognita, ovvero la posizione reale del robot $\overline{gh}$, che in codice si traduce in:

$erroredieci = \frac{(stimaaltezza - altezzarobot) * 10}{altezzarobot}$

[16]L'omotetia è un particolare tipo di similitudine, essa comporta una particolare trasformazione geometrica, che dilata o contrae gli oggetti, mantenendo invariati gli angoli e la forma.

tale distanza risulta essere in cm dunque per essere usata agevolmente dovrà essere poi calcolata in pixel tramite la trasformazione lineare

$erroredieci2pixel = \frac{erroredieci * Distmedia}{distanza}$

ora sappiamo che per tale distanza in pixel dall'asse della webcam, si ha un'errore di dieci centimetri, perciò tradotto in pixel l'errore, e dunque la correzione da effettuare, sarà pari a

$distanza : Distmedia = 10 : correzionedieci$

$correzionedieci = \frac{10 * Distmedia}{distanza}$

dove "$distanza = 69cm$" mentre "$Distmedia$" è la distanza tra i riferimenti misurata in pixel e "*correzionedieci*" risulta essere la correzione in pixel pari a dieci centimetri che risulta essere strettamente dipendente dall'altezza.

Per l'applicazione di tale correzione al calcolo della posizione il fattore correttivo per ciascun pixel sarà pari a

$correzioneognipixel = \frac{correzionedieci}{erroredieci2pixel}$

In tale modo,essendo già stati calcolati tutti i valori correttivi, durante la fase di tracciamento del robot risulta immediato intervenire e correggere l'errore di posizione senza gravare in modo rilevante sui tempi di elaborazione.

Per semplificare il lavoro dell'utente al terminale il software è stato dotato di un'interfaccia grafica semplice ed intuitiva[17] che consente l'esecuzione delle operazioni di calibrazione e la visione dei dati sensibili rilevati durante tale operazione, senza bisogno di dover digitare alcun comando da terminale. Nel prossimo paragrafo sarà presentato l'intero codice sorgente riguardante l'operazione di calibrazione automatizzata con alcuni commenti utili alla comprensione del codice.

[17] Per il manuale di uso del programma fare riferimento all'Appendice B del presente testo.

2.5 Realizzazione software

```
 clear
altezzaim=479;
altezzarobot=33;     %espressa in cm
intensitawiener=50   %intensit\‘a del filtro adattativo di wiener
intensitamed=30       % intensit\’a del filtro mediano
DimensioneMinima=500  %le immagini con un numero di pixel inferiori a questo vengono scartate
numero=0;   %%numero oggetti trovati nel piano di lavoro , deve essere uguale a 3
tentativi=0;
loading_calib;
%correggisemplificato
I=imread(’last.jpg’);
while(numero~=3) % I= imread(’last_rect.jpg’);  %quella esatta
                %trova il valore minimo del filtro necessario
                   a mostrare tutte e tre le immagini
I=imread(’last.jpg’);
sfondotracking=I;
figure(1);
imshow(I);
title(’Immagine originale’)
G= rgb2gray(I);
figure(2);
imshow(G);title(’Immagine Bianco e nero’);
hgram=imhist(G)
J=imadjust(G);  %IMPOSTA L’IMMAGINE AL VALORE DI CONTRASTO LIMITI
```

```
J=wiener2(J,[intensitawiener intensitawiener]); %FILTRO ADATTATIVO
                                                 DI WIENER CHE ELIMINA I DISTURBI
J=medfilt2(J,[intensitamed intensitamed]);    %%FILTRO MEDIANO
J=imadjust(J);                              %modifica il conrasto dell'immagine
J=imadjust(J,[0.2;0.4],[0;1]);
figure(3);
imshow(J);title('Immagine con pi\'u contrasto e senza disturbi');
BW=im2bw(J);
figure(4);
hgram=imhist(J);title('Istogramma'); %crea un istogramma dell'immagine
figure(5);
imshow(BW);title('Immagine binaria')
figure(6)
BWN=BW;
if(tentativi==10)
    break
    end
BWN=bwareaopen(BWN,DimensioneMinima);
imshow(BWN);title('Immagine invertita e senza disturbi')
[Z,numero]=bwlabel(BWN,8);              %trova gruppi separati da 8 caratteri
     if(numero<3)
         intensitawiener=intensitawiener-10
     end
    if(numero>3)
         intensitawiener=intensitawiener+10
```

```
    end
tentativi=tentativi+1
     if(numero==3)
         areaottima=0
         while(areaottima==0)
             if(intensitawiener==0) %questo controllo \‘e necessario
                           affinchÃ¨ non venga assegnato un valore
                           inferiore a zero al filtro di wiener,
                           in questo caso si andrebbe a generare
                            un errore di matlab
                 break
             end
                 figure(1);
                  imshow(I);
                   title('Immagine originale')
               G=rgb2gray(I);
               figure(2);
               imshow(G);title('Immagine Bianco e nero');
             hgram=imhist(G);
               J=imadjust(G);  %IMPOSTA L'IMMAGINE AL VALORE DI CONTRASTO LIMITE
               J=wiener2(J,[intensitawiener intensitawiener]); %FILTRO ADATTATIVO DI WIENER
               J=medfilt2(J,[intensitamed intensitamed]);      %FILTRO MEDIANO
               J=imadjust(J);  %modifica il contrasto dell'immagine
               J=imadjust(J,[0.2;0.4],[0;1]); %soglia il contrasto tra 0.2 e 0.4
               figure(3);
```

```
                imshow(J);title('Immagine con piu contrasto e senza disturbi');
                BW=im2bw(J);
                figure(4);
                %hgram=imhist(J);title('Istogramma');
                figure(5)
                imshow(BW);title('Immagine binaria')
                figure(6)
                BWN=BW;


             BWN=bwareaopen(BWN,DimensioneMinima);
imshow(BWN);title('Immagine invertita e senza disturbi')


             [Z,numero]=bwlabel(BWN,8);              %trova gruppi separati da 8 caratteri


             intensitawiener=intensitawiener-5
           [r,c] = find(Z==1);  %restituisce coordinate dei pixel del gruppo 1
           rc1 = [r c];
          %Z=bwlabel(BWN,8)
           [r,c] = find(Z==2); %restituisce coordinate dei pixel del gruppo 2
           rc2 = [r c];


           %Z=bwlabel(BWN,8)
           [r,c] = find(Z==3); %restituisce coordinate dei pixel del gruppo 3
           rc3 = [r c];
```

```
          area1=bwarea(rc1)

          area2=bwarea(rc2)

          area3=bwarea(rc3)

          if(sqrt((area1-area2)^2)<500)

            if(sqrt((area1-area3)^2)<500)

                if(sqrt((area2-area3)^2)<500)

                  areaottima=1

                end

            end

          end

         end

     end

end

if(numero~=3)                %Controllo sul numero di figure trovate

   ERROREFILTRAGGIO       %Avvia la GUI relativa all'errore

end

[r,c] = find(Z==1);   %restituisce coordinate dei pixel del gruppo 1

rc1 = [r c];

%Z=bwlabel(BWN,8)

[r,c] = find(Z==2);   %restituisce coordinate dei pixel del gruppo 2

rc2 = [r c];

%Z=bwlabel(BWN,8)

[r,c] = find(Z==3);     %restituisce coordinate dei pixel del gruppo 3

rc3 = [r c];

%Z=bwlabel(BWN,8)
```

```
[r,c] = find(Z==4);    %questo nella calibrazione deve essere vuoto

rc4 = [r c]

P1=rc1(1,2)

S1=size(rc1);

NROW=S1(1,1)        %memorizza numero righe

P12=rc1(NROW,2)      %prende il valore dell'ultima riga

META=(P12-P1)

a=mod(META,2)                %

if(a==1)                     %

    META=META+1              %

end                          %   TROVO LA X DEL CENTRO

META=META/2                  %

X1=META

X1=X1+P1

RC11=rc1(:,1);

m=min(RC11)                 % TROVA LE Y DEL CENTRO

M=max(RC11)

Y1=(M-m)

a=mod(Y1,2)

if(a==1)

    Y1=Y1+1

end

Y1=Y1/2

Y1=Y1+m

%-------------------------------OGGETTO2
```

```
P2=rc2(1,2)

S2=size(rc2)

NROW=S2(1,1)        %memorizza numero righe

P22=rc2(NROW,2)      %prende il valore dell'ultima riga

META=(P22-P2)

a=mod(META,2)                %

if(a==1)                     %

    META=META+1              %

end                          % TROVO LA X DEL CENTRO

META=META/2

X2=META

X2=X2+P2

RC22=rc2(:,1);

m=min(RC22)

M=max(RC22)

Y2=(M-m)

a=mod(Y2,2)

if(a==1)

    Y2=Y2+1

end

Y2=Y2/2

Y2=Y2+m

%-----------------------OGGETTO 3

P3=rc3(1,2)

S3=size(rc3)
```

```
NROW=S3(1,1)        %memorizza numero righe

P33=rc3(NROW,2)      %prende il valore dell'ultima riga

META=(P33-P3)

a=mod(META,2)                %

if(a==1)                     %

    META=META+1              %

end                          %  TROVO LA X DEL CENTRO

META=META/2                  %

X3=META

X3=X3+P3

RC33=rc3(:,1);

m=min(RC33)

M=max(RC33)

Y3=(M-m)

a=mod(Y3,2)

if(a==1)

    Y3=Y3+1

end

Y3=Y3/2

Y3=Y3+m

%RICONOSCIMENTO COLORE   M=[Red   Green   Blue]

COLORE1=impixel(I,X1,Y1)

COLORE2=impixel(I,X2,Y2)

COLORE3=impixel(I,X3,Y3)

%Riconoscimento colore
```

```
if(COLORE1(1,1)>170 && COLORE1(1,2)<100 && COLORE1(1,3)<170)
rx=X1
ry=Y1
elseif(COLORE1(1,1)<130 && COLORE1(1,2)>150 && COLORE1(1,3)<160)
gx=X1
gy=Y1
elseif(COLORE1(1,1)<100 && COLORE1(1,2)<130 && COLORE1(1,3)>100)
bx=X1
by=Y1
end
if(COLORE2(1,1)>170 && COLORE2(1,2)<100 && COLORE2(1,3)<170)
rx=X2
ry=Y2
elseif(COLORE2(1,1)<130 && COLORE2(1,2)>150 && COLORE2(1,3)<160)
gx=X2
gy=Y2
elseif(COLORE2(1,1)<100 && COLORE2(1,2)<130 && COLORE2(1,3)>100)
bx=X2
by=Y2
end
if(COLORE3(1,1)>170 && COLORE3(1,2)<100 && COLORE3(1,3)<170)
rx=X3
ry=Y3
elseif(COLORE3(1,1)<130 && COLORE3(1,2)>150 && COLORE3(1,3)<160)
gx=X3
```

```
gy=Y3

elseif(COLORE3(1,1)<100 && COLORE3(1,2)<130 && COLORE3(1,3)>100)

bx=X3

by=Y3

end

%CALCOLO RAPPORTO PIXEL-METRI

distanza=69  %cm  distanza tra i punti nella realta(deve essere uguale tra RB e RG)

DistRG=sqrt((rx-gx)^2+(ry-gy)^2)    %distanza in pixel tra il centro e la fine di Y

DistRB=sqrt((rx-bx)^2+(ry-by)^2) %distanza in pixel tra il centro e la fine di X

DiagonaleIdeale=sqrt(DistRB^2+DistRG^2)   %misura diagonale ideale tramite teorema di Pitagora

DiagonaleReale=sqrt(sqrt((bx-gx)^2)^2+sqrt((by-gy)^2)^2) %Misura effettiva della diagonali.

                                                          Le due diagonali coincidono solo se

                                                           %l'angolo \'e retto.

scartodiagonale=sqrt(DiagonaleIdeale-DiagonaleReale)^2  %come differenza

scartodiagonale=(scartodiagonale*100)/DiagonaleIdeale   %come rapporto

Distmedia=(DistRG+DistRB)/2

PosPixelRob=0;

cm=(distanza*PosPixelRob)/Distmedia

%TRASFORMAZIONE COORDINATE TRA ASSE FISSO E ASSE MOBILE

H=sqrt((ry-gy)^2)

L=DistRG

PHI=acos(L/H)

Ry=altezzaim-ry

Gy=altezzaim-gy

By=altezzaim-by
```

```
if(By>Ry)

    PHI=-PHI

end

M=[cos(PHI) -sin(PHI) -rx*cos(PHI)+(altezzaim-ry)*sin(PHI);

sin(PHI) cos(PHI) -rx*sin(PHI)-(altezzaim-ry)*cos(PHI);

0 0 1];

angolo=sqrt((real(PHI))^2+(imag(PHI))^2)

angolo=angolo*180/pi

if(gx<=rx)

angolo=angolo*(-1)

end

DATICALIBRAZIONE=zeros(2,4);  %esportazione valori nella tabella

DATICALIBRAZIONE(1,1)=rx

DATICALIBRAZIONE(2,1)=ry

DATICALIBRAZIONE(1,2)=gx

DATICALIBRAZIONE(2,2)=gy

DATICALIBRAZIONE(1,3)=bx

DATICALIBRAZIONE(2,3)=by

DATICALIBRAZIONE(1,4)=angolo

DATICALIBRAZIONE(2,4)=scartodiagonale

%%STIMA ALTEZZA WEBCAM

correzioneognipixel=0;

if(altezzarobot~=0)

pixeladuemetriemezzo=127.5598;

DifferenzaPixel=pixeladuemetriemezzo-Distmedia
```

```
if(Distmedia<128)
    stimaaltezza=(-1.9*Distmedia+485) %%rifai conto
elseif(Distmedia<139)
stimaaltezza=(-1.8174*Distmedia+482.6478)
elseif(Distmedia<145)
    stimaaltezza=(-1.5386*Distmedia+443.8913)
elseif(Distmedia<160)
    stimaaltezza=(-1.5338*Distmedia+443.1942)
else
    stimaaltezza=(-1.1036*Distmedia+377.7850)
end
%STIMA ERRORE PROSPETTICO MEDIO DOVUTO ALL'ALTEZZA DEL TROBOT
erroredieci=(((stimaaltezza-altezzarobot)*10)/altezzarobot);  %errore dieci risulta
                      essere la distanza dalla webcam alla quale ho un errore di 10 cm
erroredieci2pixel=(erroredieci*Distmedia)/distanza;
correzionedieci=(10*Distmedia)/distanza;
correzioneognipixel=correzionedieci/erroredieci2pixel
end
DATICALIBRAZIONE(1,5)=stimaaltezza     %esportazione valori nella tabella
DATICALIBRAZIONE(2,5)=correzioneognipixel
```

Capitolo 3

Ricerca ed inseguimento del Robot

Nella robotica mobile, il problema della localizzazione consiste nell'individuare la posizione del robot e il suo orientamento all'interno dell'ambiente di lavoro, effettuando questo in funzione di una variabile temporale è possibile ottenere anche una stima della velocità del robot ed inoltre registrando in memoria tutte le posizoni occupate sarà possibile tracciare la sua posizione graficamente. Esattamente questo è il problema affrontato nel presente capitolo. Il robot a nostra disposizione, come già accennato nell'introduzione, è un Octagon realizzato presso l'università di Tor Vergata, a causa delle sue ridotte dimensioni è stato necessario apportare delle modifiche strutturali.

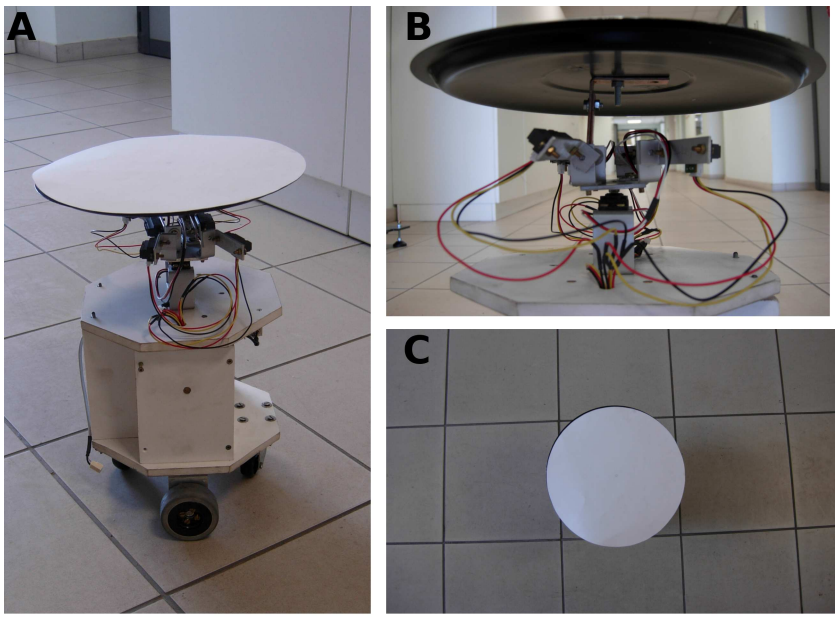


Figura 3.1: Modifiche strutturali apportate all'Octagon

È stato installato alla sua sommità un "*Tag*" di riferimento circolare (Fig. 3.1 A) di diametro pari a 26 cm, fissato opportunamente alla struttura tramite un doppio snodo a 90 gradi (Fig. 3.1 B). Tutto ciò è stato effettuato per facilitare l'individuazione del robot sul piano di lavoro, fondamentalmente per due ragioni:

- eliminare una possibile fonte di errore di individuazione, in quanto, escluso il caso in cui il robot si trovi esattamente sotto la webcam, in tutti gli altri casi l'immagine catturata avrebbe compreso non solo la base superiore ma anche parte della fiancata dell'Octagon, comportando errori nel calcolo della posizione.

- inoltre la dimensione ridotta della base superiore non consentiva l'uso di filtraggi tali da poter eliminare comodamente l'ombra del robot, la quale avrebbe comportato errori notevoli nella stima della posizione.

Il risultato finale di tali modifiche è osservabile nella figura 3.1 C, il robot risulta completamente coperto dal riferimento circolare bianco, in tale modo è possibile applicare filtri anche molto forti

senza intaccare i bordi del riferimento, ma eliminando tutti i disturbi circostanti. Riguardo la scelta del materiale, di riferimento circolare bianco del robot, come allo stesso modo quelli colorati del calibratore, hanno la struttura in alluminio ma il rivestimento colorato è stato realizzato in cartoncino, in quanto anche l'utilizzo di vernici opache si è rivelato inutile all'eliminazione dei bagliori e dei riflessi di luce che si formavano sul piano dei riferimenti. Inoltre il cartoncino oltre ad essere sufficientemente opaco risulta essere anche un materiale di facile sostituzione in caso di danneggiamento. Per il controllo della procedura di tracciamento del robot è stato creato un apposito oggetto "*cronometro*", non presente in MatLab. Il cronometro, all'avvio del programma di individuazione, richiede l'immissione di un valore numerico indicante il numero di minuti per i quali si desidera tracciare gli spostamenti del robot e al termine dei quali blocca l'esecuzione del programma. Tale oggetto funziona nel seguente modo[1]:

- riceve dall'utente il numero di minuti per cui il programma deve restare attivo
- richiede al sistema operativo l'ora corrente salvando tali valori in un vettore "*orologio*" contenente data, ore, minuti e secondi
- crea una copia dei valori ore e minuti chiamati "*orastop*" e "*minstop*" andando a modificare il valore dei minuti, ai quali verranno aggiunti i minuti di attività richiesti dall'utente, dunque in tali variabili sarà indicato l'orario di terminazione del programma
- dopodiché il cronometro effettua un confronto tra l'ora e i minuti correnti e "*orastop*" e "*minstop*", quando tali valori saranno identici il programma terminerà l'esecuzione.

All' interno di tale cronometro è contenuta tutta la procedura di tracking del robot che verrà trattata nei prossimi paragrafi.

[1]Per approfondimenti sul codice fare riferimento al Paragrafo "Realizzazione software" del presente capitolo.

3.1 Filtraggio

Per ciò che riguarda le fasi di filtraggio dell'immagine, durante la ricerca del robot, si può dire che esse non variano molto dalla fase di filtraggio del calibratore[2].

Figura 3.2: Risultati del prefiltraggio iniziale

Infatti oltre al prefiltraggo iniziale (Fig. 3.2) che "$sfonda$" l'ambiente di lavoro togliendo la maggior parte dei disturbi, i filtri usati restano il filtro di Wiener e quello mediano, che ereditano dalla fase di calibrazione l'intensità dei filtri.

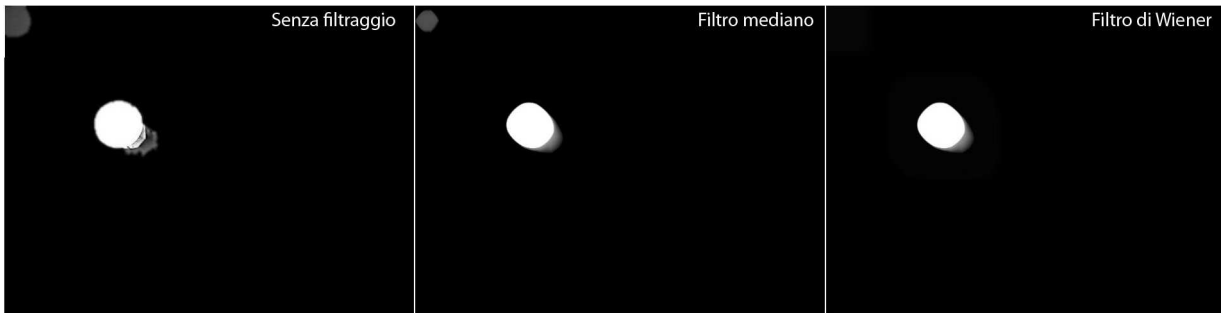


Figura 3.3: Risultati dell'applicazione dei vari filtri all'immagine

Infatti essendo pressoché identiche le condizioni ambientali tra la fase di calibrazione e la fase di tracciamento, usando gli stessi filtri oltre ad ottenere ottimi risultati di filtraggio (Fig. 3.3) si riduce

[2]Per ulteriori dettagli vedi capitolo precedente.

sensibilmente il tempo di calcolo, non dovendo calcolare nuovamente l'intensità ottima.

Figura 3.4: Confronto tra l'immagine originale e quella priva di distorsione radiale

Il primo passo compiuto è stato quello di eliminare la distorsione radiale presente nell'immagine (Fig. 3.4) a causata dalla lente stessa della webcam[3], la quale comporta come già accennato una distorsione dell'immagine reale e dunque errori nella valutazione della posizione. A differrenza del caso precedentemente trattato riguardante l'eliminazione della distorsione prospettica che comporta solo una correzione dei parametri calcolati[4], in questo caso è l'immagine vera e propria ad essere modificata, per questo motivo tale operazione richiede un tempo di calcolo oneroso, che contribuisce notevolmente al rallentamento del programma, ma che però non può essere assolutamente eliminato. Tramite il comando "$correggisemplificato$" l'immagine originale viene trasformata graficamente utilizzando i valori dei parametri intrinseci dell'obiettivo precedentemente calcolati grazie alla procedura descritta nell'Appendice A di questo testo. Il comando "$correggisemplificato$" non risulta altro che il comando "*undistort*" già presente nel toolbox "$CameraCalibration$" di MatLab, opportunamente modificato e semplificato per i nostri scopi, dunque non verrà spiegato nei dettagli il suo funzionamento.

[3] Vedi Appendice A.
[4] Vedi prossimo paragrafo.

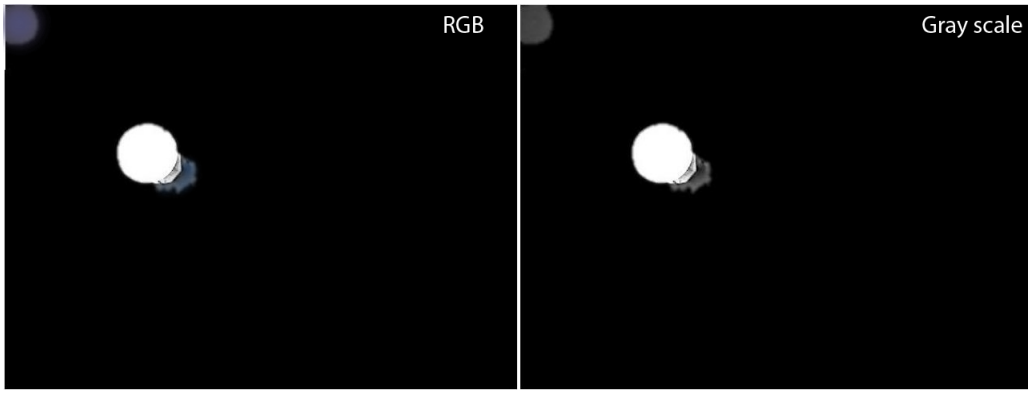


Figura 3.5: L'immagine originale RGB viene trasformata in grayscale

Una volta che l'immagine è stata trattata verrà salvata con il nome di "*last_rect*", dopodiché l'immagine originariamente RGB viene trasformata in grayscale (Fig. 3.5) prima dell'applicazione dei filtri.

Dopo aver eseguito un filtraggio sufficiente è possibile passare alla fase di individuazione del robot.

3.2 Individuazione del Robot

Figura 3.6: L'immagine grayscale viene trasformata in binaria

Prima di procedere all'individuazione del robot vera e propria, l'immagine "*grayscale*" dovrà essere trasformata in formato binario (Fig. 3.6), in tale modo sarà possibile applicarvi il comando "$BWareaopen(BWRobot, 1000)$" che elimina tutti gli oggetti con area inferiore a 1000 pixel, e dunque è in grado di rimuovere i disturbi quali ombre e riflessi ed il comando "$[Z, numero] =$

bwlabel$(BWN, 8)$" che riconosce tutti gli oggetti presenti in figura con un'area almeno di 8 pixel. In questo caso la variabile "*numero*" dovrà essere uguale ad uno, per essere sicuri che vi sia solo il robot all'interno dell'immagine. L'algoritmo di calcolo delle coordinate del robot è esattamente lo stesso usato per il calcolo delle coordinate dei riferimenti del calibratore[5] come riportato nel seguente codice MatLab:

```
[r,c] = find(ROBOT==1); %restituisce coordinate dei pixel
robotpixel = [r c];      del gruppo 1
R1=robotpixel(1,2);
S1=size(robotpixel);
NROW=S1(1,1);        %memorizza numero righe
R12=robotpixel(NROW,2); %prende il valore dell'ultima riga
META=(R12-R1);
a=mod(META,2);
if(a==1)
    META=META+1;
end                % TROVO LA X DEL CENTRO
META=META/2;
Xr=META;
xrobot=Xr+R1; %coordinata X del robot
R11=robotpixel(:,1);%------------
m=min(R11);   %TROVO LA Y DEL CENTRO
M=max(R11);
Yr1=(M-m);
a=mod(Y1,2);
```

[5] Vedi paragrafo "*Riconoscimento figure e selezione assi*" del capitolo precedente.

```
if(a==1)

    Y1=Y1+1;

end

Yr1=Yr1/2;

yrobot=Yr1+m  %coordinata Y del robot

yrobotnonrovesciata=yrobot;
```

Una volta calcolata la posizione del robot, le coordinate rispetto all'asse "Y", devono essere ribaltate, prima di effettuare il cambio di coordinate tra "$assefisso$" ed "$assemobile$", infatti come spiegato nel primo capitolo l'origine degli "$assipixel$" è posizionato in alto a sinistra, al contrario l'origine degli assi cartesiani si trova in basso a sinistra, per questo motivo le coordinate "Y" del robot risultano essere ribaltate. L'operazione di "ribaltamento" viene effettuata con la semplice operazione

$yrobot = altezzaim - yrobot$

dove "$yrobot$" è inizialmente la coordinata "y" stimata dal calcolo, ed "$altezzaim = 479$" essendo l'immagine di dimensione 640x480 pixel. Il cambio di coordinate viene però effettuato solo dopo l'eliminazione della distorsione prospettica.

3.3 Eliminazione della distorsione prospettica

L'eliminazione di tal distorsione, come già accennato in precedenza, avviene solo dal punto di vista numerico sulle coordinate calcolate, e non sull'immagine vera e propria. Dato che tale distorsione varia in funzione della distanza dall'asse della webcam, essendo tale asse corrispondente con il centro dell'immagine[6], essa sarà funzione della distanza del robot dal centro dell'immagine. Per questo motivo a seconda di quale sia la distanza della coordinata dal centro "O" della foto, varierà la correzione apportata a tale valore come segue:

[6]Questa affermazione risulta valida se e solo se la "$camera$" è perfettamente perpendicolare a terra.

```
if(xrobot<320)
    dcorrezione=320-xrobot;
    correzionepixel=dcorrezione*correzioneognipixel;
    xrobot=xrobot+correzionepixel;
elseif(xrobot>=320)
   dcorrezione=xrobot-320;
    correzionepixel=dcorrezione*correzioneognipixel;
    xrobot=xrobot-correzionepixel;
end
if(yrobot<240)
    dcorrezione=240-yrobot;
    correzionepixel=dcorrezione*correzioneognipixel;
    yrobot=yrobot+correzionepixel;
    yrobotnonrovesciata=yrobotnonrovesciata-correzionepixel
elseif(yrobot>=240)
   dcorrezione=yrobot-240;
    correzionepixel=dcorrezione*correzioneognipixel;
    yrobot=yrobot-correzionepixel;
    yrobotnonrovesciata=yrobotnonrovesciata+correzionepixel
end
```

Dove la variabile "*dcorrezione*" rappresenta la distanza della coordinata dal centro immagine, "*correzionepixel*" è appunto la correzione da effettuare su tale coordinata e "*xrobot*" sarà la coordinata finale stimata, senza l'effetto della distorsione prospettica. Tale operazione viene eseguita per entrambe le coordinate del robot.

Una volta effettuata tale correzione viene compiuto il cambio di coordinate tramite la matrice "M"

calcolata nella fase di calibrazione assi.

$$PosRealeRobot = [M] * \begin{bmatrix} xrobot \\ yrobot \\ 1 \end{bmatrix}$$

3.4 Tracking del Robot

Le coordinate del robot che vengono di volta in volta calcolate, ed altri valori sensibili, sono archiviati in un vettore "*registro*" che ha una lunghezza dinamica variabile in funzione dei minuti di monitoraggio, questo per ottimizzare lo sfruttamento della memoria di sistema, infatti nel caso in cui si scelga di monitorare il robot per 2 minuti la lungezza di tale registro sara pari a 120, al contrario se il tempo di tracking viene impostato a 20 minuti la lunghezza di tale vettore sarà di 1200. Durante tale fase di archiviazione non tutti i valori di coordinate vengono archiviati, infatti vengono scartati e dunque non scritti nel registro se le coordinate calcolate:

- sono le stesse coordinate calcolate al passo precedente, ciò vuole dire che il robot non si è mosso
- si discostano di un valore inferiore a 2 pixel, questo poiché anche se il robot é immobile potrebbe avvenire piccoli errori e dunque potrebbe accadere che le coordinate calcolate siano leggermente diverse, dunque per evitare tali errori si preferisce non scrivere affatto il valore
- si discostano più di 50 pixel dalla posizione precedente, infatti essendo la velocità massima dell'Octagon pari a 9 cm al secondo alle normali altezze di lavoro non potrà mai percorrere tale distanza tra uno scatto e l'altro, dunque nel caso si verificasse tale condizione, vorrà dire che la posizione calcolata risulta errata, frutto di qualche disturbo nell'immagine.

Tramite tali dati viene poi effettuata una stima dell'orientamento del robot, della sua velocità e vengono usati per il tracciamento del percorso compiuto, oltre ad essere esportati in una tabella visualizzabile alla fine dell'operazione di tracking tramite il comando"*Visualizza Dati Tracking*" dal menù grafico.

La stima dell'orientamento è effettuata usando la funzione "*atan*2" presente in MatLab, ovvero l'ar-

cotangente a 4 quadranti. Passando a tale funzione le coordinate "δY" ed "δX", cioè la differenza tra la posizione attuale del robot e quella precedente, è possibile effettuare una stima dell'orientamento dell'oggetto in movimento, che sarà tanto più accurata, quanto saranno frequenti i fotogrammi al secondo analizzati.

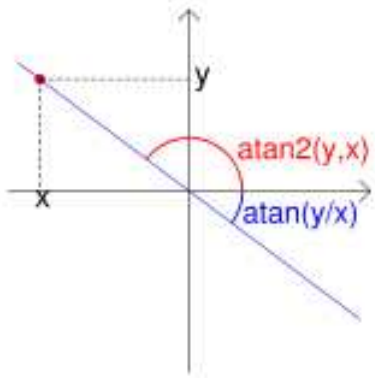


Figura 3.7: L'arcotangente a 4 quadranti

L'uso dell'arcotangente a 4 quadranti evita i possibili casi ambigui (Fig. 3.7) che si avrebbero invece con l'uso dell'arcotangente normale.

La stima dela velocità del robot viene effettuata usando i dati presenti nel vettore "*registro*", infatti:

```
Spazio=sqrt((posx1-posx2)^2+(posy1-posy2)^2)

secdif=sqrt(sec2-sec1)^2

if(min1==min2)

spaziosecondi=secdif;

else

mindif=min2-min1;

spaziosecondi=secdif+(mindif*60)

end
```

viene calcolato lo spazio percorso tra la posizione attuale "$posx2$", "$posy2$" e la posizione precedente "$posx1$", "$posy1$", dunque "$Spazio$" sarà lo spazio percorso nell'intervallo di tempo pari a "$secdif$". Dunque la stima della velocità verrà semplicemente calcolata come

$velocita = \frac{Spazio}{spaziosecondi}$

Una volta effettuata tale procedura fino allo scadere del tempo stabilito, il programma traccerà il percorso compiuto dal robot in tale lasso di tempo proprio grazie ai dati registrati nel "$registro$" con tale risultato finale.

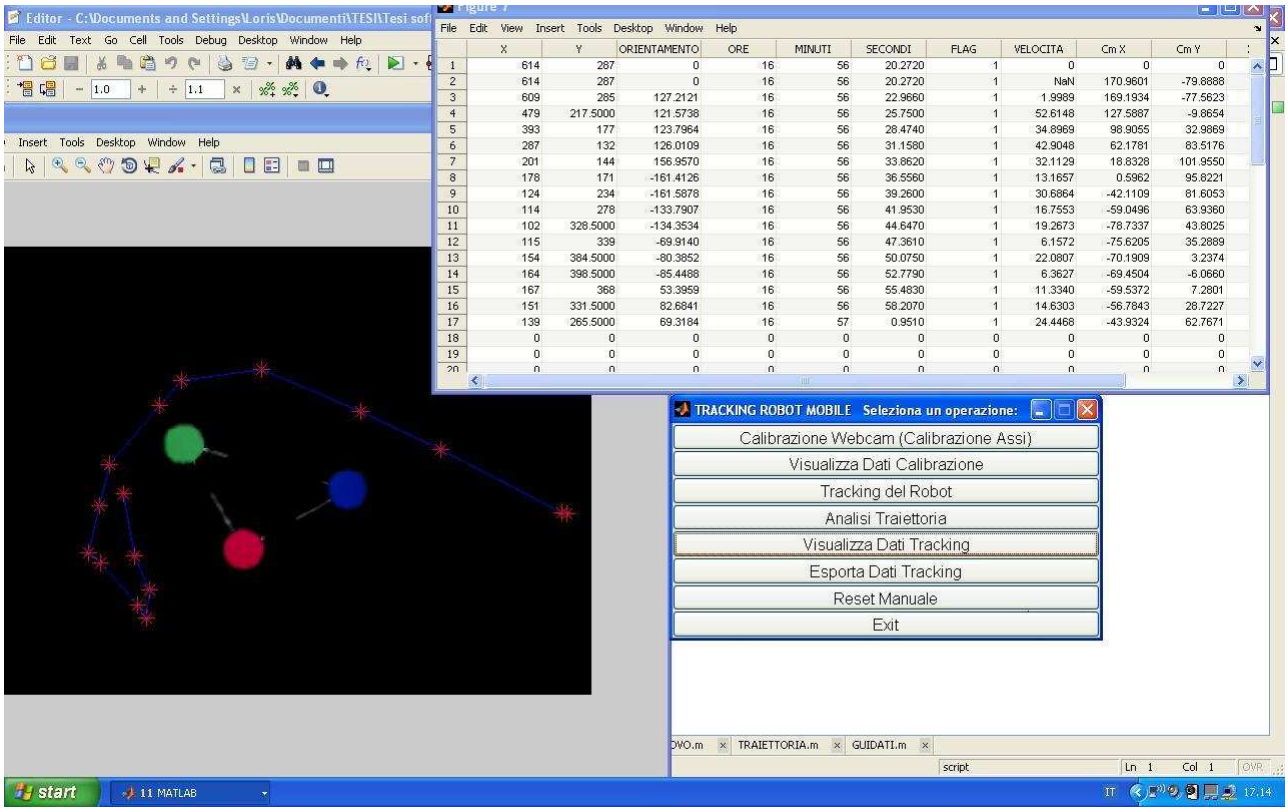


Figura 3.8: Esempio di lavoro

3.5 Realizzazione software

```
 ----------CRONOMETRO-----------------------

traccia=2;  %tiene conto della posizione nel registro

contavolte=0; %tiene conto del numero di valori

                scritti nel registro

clc; %pulisce la shell

tempo = input('Per quanto tempo vuoi effettuare il campionamento?

inserisci il numero di minuti:');

lunghezzaregistro=tempo*60;

registro=zeros(lunghezzaregistro,13);

%|_X_|_Y_|_ANGOLO_|_ORE_|_MINUTI_|_SECONDI_|_FLAG_|STIMA VELOCITA|

|posizione in Cm X |Posizione in Cm Y|X ASSI|Y ASSI|istante analizzato|


orologio=clock

ore=orologio(1,4)       %Ore

minuti=orologio(1,5)   v%Minuti

secondi=orologio(1,6)   %Secondi

minstop=minuti+tempo    %istante di arresto

if(minstop<60)

  orastop=ore;

  secstop=secondi;

else(minstop>=60)

        orastop=ore+1;

        secstop=secondi;

        minstop=minstop-60;
```

```
    end
controllo=0;
while ((controllo==0)&&(minuti~=minstop))
crono=clock;
ore=crono(1,4);
minuti=crono(1,5);
secondi=crono(1,6);
 if(ore==orastop)&&(secondi==secstop)&&(minuti==minstop)
controllo=1
 end
%-----------TRACKING del ROBOT-------------------
%la stampa delle figure e' disbilitata per non
%rallentare il programma
altezzaim=479;
%R=imread('last.jpg');
%imshow(R);
correggisemplificato
%figure(4)
R=imread('last_rect.jpg');
%figure(1);
%imshow(R);title('Immagine originale senza distorsione')
RGRAY=rgb2gray(R);
%figure(3);
%imshow(RGRAY);title('Immagine Gray scale');
J=imadjust(RGRAY,[0.1 0.6],[0 1]);
```

```
J=medfilt2(J,[intensitamed intensitamed]);

J=wiener2(J,[100 100]);

%figure(4);

%imshow(J);title('Immagine con piu contrasto e senza disturbi');

BWRobot=im2bw(J);

J=imadjust(J);

%figure(5)

%imshow(BWRobot);

BWN=bwareaopen(BWRobot,1000);

[Z,numero]=bwlabel(BWN,8);

if(numero~=1)   %Controllo sul numero di figure trovate

   ERROREFILTRAGGIO  %Avvia la GUI relativa all'errore

end

%imshow(BWN);

if(numero==1)

ROBOT=bwlabel(BWN,8); %trova gruppi di almeno 8 pixel

[r,c] = find(ROBOT==1); %restituisce coordinate dei pixel

                        del gruppo 1

robotpixel = [r c];

R1=robotpixel(1,2);

S1=size(robotpixel);

NROW=S1(1,1);      %memorizza numero righe

R12=robotpixel(NROW,2); %prende il valore dell'ultima riga

META=(R12-R1);

a=mod(META,2);
```

```
if(a==1)

    META=META+1;

end                 % TROVO LA X DEL CENTRO

META=META/2;

Xr=META;

xrobot=Xr+R1;

R11=robotpixel(:,1);

m=min(R11);

M=max(R11);

Yr1=(M-m);

a=mod(Y1,2);

if(a==1)

    Y1=Y1+1;

end

Yr1=Yr1/2;

yrobot=Yr1+m

yrobotnonrovesciata=yrobot;

yrobot=altezzaim-yrobot;    %rovescio le coordinate della y

PosPixelRob=[xrobot yrobot]

%-----correzione prospettica applicato sulla yrobotnonrovesciata e sulla x------

if(min1==min2)

spaziosecondi=secdif;

else

mindif=min2-min1;

spaziosecondi=secdif+(mindif*60)
```

```
end
if(xrobot<320)
    dcorrezione=320-xrobot;
    correzionepixel=dcorrezione*correzioneognipixel;
    xrobot=xrobot+correzionepixel;
elseif(xrobot>=320)
   dcorrezione=xrobot-320;
    correzionepixel=dcorrezione*correzioneognipixel;
    xrobot=xrobot-correzionepixel;
end
if(yrobot<240)
    dcorrezione=240-yrobot;
    correzionepixel=dcorrezione*correzioneognipixel;
    yrobot=yrobot+correzionepixel;
    yrobotnonrovesciata=yrobotnonrovesciata-correzionepixel
elseif(yrobot>=240)
   dcorrezione=yrobot-240;
    correzionepixel=dcorrezione*correzioneognipixel;
    yrobot=yrobot-correzionepixel;
    yrobotnonrovesciata=yrobotnonrovesciata+correzionepixel
end
%-------------------------------------------------------------------------

M=[cos(PHI) -sin(PHI) -rx*cos(PHI)+(altezzaim-ry)*sin(PHI);
sin(PHI) cos(PHI) -rx*sin(PHI)-(altezzaim-ry)*cos(PHI);
```

```
0 0 1];

PosRealeRobot=[xrobot;yrobot;1];

PosRealeRobot=M*[xrobot;yrobot;1];

XXX=PosRealeRobot(1,1)

YYY=PosRealeRobot(2,1)

PosRealeRobot(3,1)

centimetri=(distanza*PosRealeRobot)/Distmedia

-----Scrittura dati nel registro-----

scritto=0

xprecedente=registro(traccia-1,1)

yprecedente=registro(traccia-1,2)

%---controllo che elimina stima errata della posizione---

if(xrobot~=xprecedente)

    if(sqrt((xrobot-xprecedente)^2)<500 && sqrt((xrobot-xprecedente)^2)>1 )

registro(traccia,1)=xrobot;

registro(traccia,2)=yrobotnonrovesciata;

registro(traccia,4)=ore;

registro(traccia,5)=minuti;

registro(traccia,6)=secondi;

registro(traccia,7)=1;

registro(traccia,9)=centimetri(1,1);

registro(traccia,10)=centimetri(2,1);

registro(traccia,11)=XXX;

registro(traccia,12)=YYY;

registro(1,11)=registro(2,11);
```

```
registro(1,12)=registro(2,12);

-----Calcolo angolo--------------

deltaX=XXX-registro(traccia-1,11);

deltaY=YYY-registro(traccia-1,12);

registro(traccia,3)=atan2(deltaY,deltaX)*180/pi;%Stima angolo

contavolte=contavolte+1

scritto=1;

    end

elseif(yrobotnonrovesciata~=yprecedente)

--controllo che elimina stima errata della posizione--

if(sqrt((yrobotnonrovesciata-yprecedente)^2)<500

   && sqrt((xrobot-xprecedente)^2)>1 )

-------------------------------------------

registro(traccia,1)=xrobot;

registro(traccia,2)=yrobotnonrovesciata;

registro(traccia,4)=ore;

registro(traccia,5)=minuti;

registro(traccia,6)=secondi;

registro(traccia,7)=1;

registro(traccia,9)=centimetri(1,1);

registro(traccia,10)=centimetri(2,1);

registro(traccia,11)=XXX;

registro(traccia,12)=YYY;

registro(1,11)=registro(2,11);

registro(1,12)=registro(2,12);
```

```
deltaX=XXX-registro(traccia-1,11);

deltaY=YYY-registro(traccia-1,12);

registro(traccia,3)=atan2(deltaY,deltaX)*180/pi;

contavolte=contavolte+1

scritto=1

  end

end

if(scritto==1)

traccia=traccia+1;

  end

 end

end

figure(24)

imshow(I);

hold on

riga=2

%============CALCOLO VELOCITA MEDIA======

registro(1,1)=registro(2,1)

registro(1,2)=registro(2,2)

registro(1,3)=registro(2,3)

registro(1,4)=registro(2,4)

registro(1,5)=registro(2,5)

registro(1,6)=registro(2,6)

registro(1,7)=registro(2,7)

flag=1
```

```
riga=2

spaziosecondi tot=0

while(flag==1)

flag=registro(riga+1,7)

posx1=registro(riga-1,1)

posy1=registro(riga-1,2)

sec1=registro(riga-1,6)

min1=registro(riga-1,5)

posx2=registro(riga,1)

posy2=registro(riga,2)

sec2=registro(riga,6)

min2=registro(riga,5)


Spazio=sqrt((posx1-posx2)^2+(posy1-posy2)^2)

secdif=sqrt(sec2-sec1)^2

if(min1==min2)

spaziosecondi=secdif;

else

mindif=min2-min1;

spaziosecondi=secdif+(mindif*60)

end

velocita=Spazio/spaziosecondi   %IN PIXEL AL SECONDO

registro(riga,8)=velocita;

spaziosecondi tot=spaziosecondi tot+spaziosecondi

registro(riga,13)=spaziosecondi tot;
```

```
riga=riga+1

end

%=======TRACCIAMENTO TRAIETTORIA===============

registro2=zeros(1,5);  %COPIA REGISTRO SPOSTAMENTI

conto=1;               IN REGISTRO2

while(conto<=contavolte)

registro2(conto,1)=registro(conto+1,1);

registro2(conto,2)=registro(conto+1,2);

registro2(conto,3)=registro(conto+1,3);

conto=conto+1;

end

while(riga<lunghezzaregistro)

x=registro(riga,1);

y=registro(riga,2);

plot(x,y,'r*','LineWidth',2,'MarkerEdgeColor','r','MarkerFaceColor',

                          [.49 1 .63],'MarkerSize',12)

hold on

riga=riga+1;

end

xx=registro2(:,1);

yy=registro2(:,2);

line(xx,yy);
```

Per non appesantire ulteriormente il testo sono state volontariamente tralasciate le sezioni di codice realizzate per la costruzione dell'interfaccia grafica, delle tabelle riguardanti i valori sia di calibrazione che di tracking calcolati ed infine il codice necessario all'esportazione dei dati in un

documento .rtf esterno.

Capitolo 4

Risultati dei Test eseguiti sul programma

In questa sezione del testo verranno illustrati 3 esempi di utilizzo dell'applicazione realizzata con i relativi risultati. Nel primo paragrafo sarà analizzato il caso in cui il robot esegua un percorso rettilineo, nel secondo paragrafo il caso di un percorso generico, ed infine nell'ultimo paragrafo verrà eseguito un test realizzato tramite due webcam montate su due computer diversi che contemporaneamente analizzano il percorso compiuto dal robot. Lo scopo di quest'ultimo test è quello di mostrare la precisione del software nel calcolo delle posizioni del robot.

4.1 Esempio 1: Percorso Rettilineo

Una volta regolate tutte le impostazioni di CaptureMax e collocato il calibratore nel piano di lavoro è stata effettuata la procedura calibrazione assi.

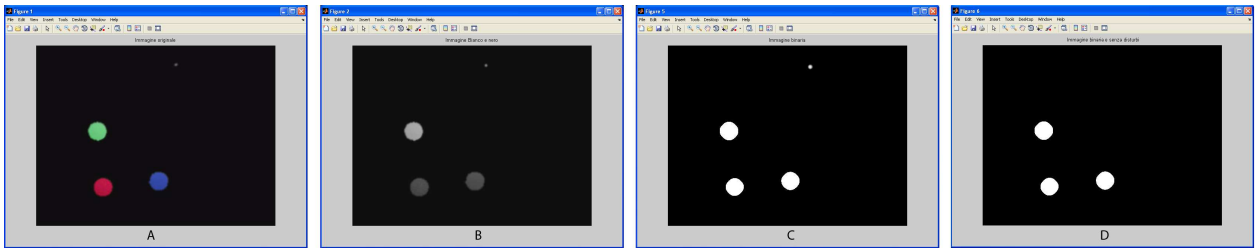


Figura 4.1: Procedura di calibrazione assi esempio 1

Come è possibile notare (Fig. 4.1) l'immagine inizialmente RGB (A) è stata prima trasformata

in grayscale (B), poi filtrata e trasformata in immagine binaria (C) e infine sono stati rimossi gli oggetti di area inferiore a quella desiderata (D).

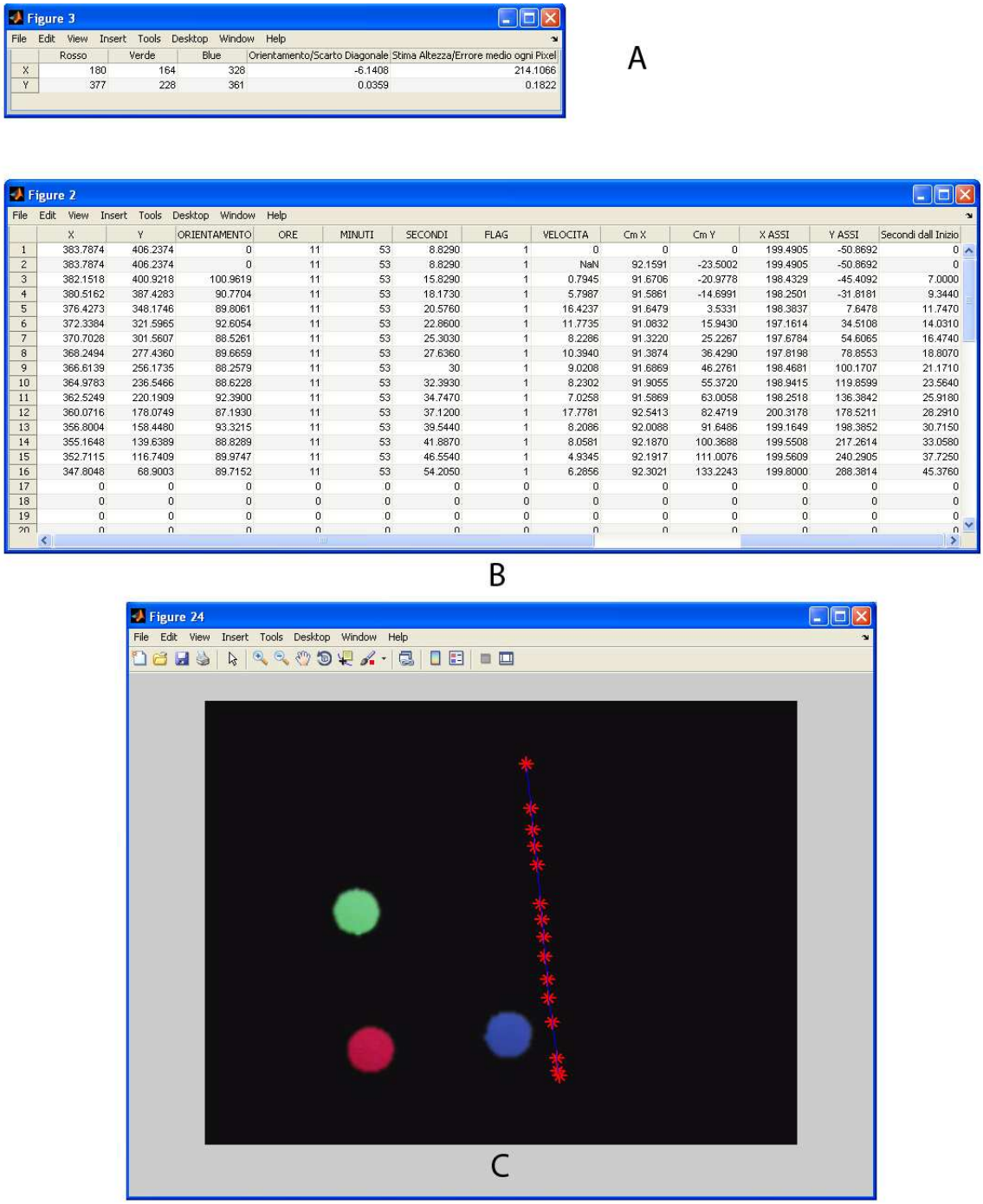
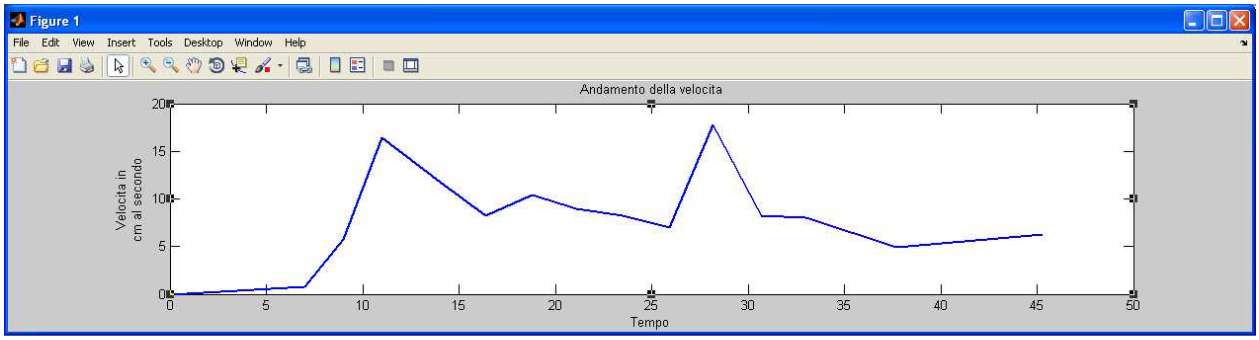


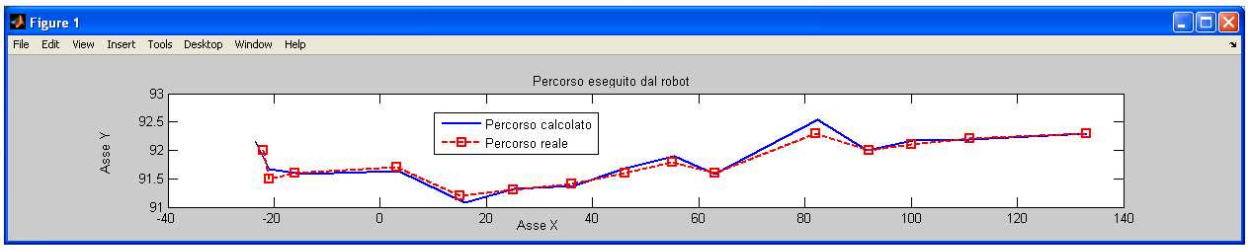
Figura 4.2: Risultati della calibrazione e del tracking

I risultati di tale procedura di calibrazione sono stati riportati in figura 4.2 A, è possibile osservare nelle prime tre colonne della tabella le coordinate pixel dei tre riferimenti rosso, verde e blue, seguono poi nelle successive colonne l'orientamento del calibratore rispetto ai bordi dell'immagine il quale risulta ruotato di 6,1408 gradi in senso antiorario. Sotto tale misura troviamo lo scarto percentuale tra la diagonale ideale e quella reale che risulta essere di circa 0.036%, dunque la calibrazione è perfettamente riuscita. Infine nell'ultima colonna troviamo la stima dell'altezza in cm calcolata dal software, che risulta essere di 214 cm. Infatti l'altezza reale era di 2.15 m circa, e l'errore medio dovuto alla distorsione prospettica, è di 0.1822 cm per ogni pixel di distanza dal centro dell'immagine.

Dopodiché è stato effettuato un monitoraggio del robot di 1 minuto, durante il quale è stato compiuto un percorso rettilineo. Nella figura 4.2 B,troviamo la tabella riportante tutti i valori notevoli calcolati,troviamo le coordinate pixel X e Y del robot rispetto ai bordi dell'immagine, l'orientamento stimato istante per istante del robot, ore, minuti e secondi dello scatto analizzato, un flag necessario per la conferma dell'avvenuta scrittura della riga, la velocità stimata punto a punto, la posizione espressa in centimetri rispetto al riferimento mobile, cioè rispetto agli assi decisi dall'utente, le coordinate pixel rispetto agli assi mobili ed infine a quanti secondi dall'inizio del monitoraggio si trova lo scatto analizzato. Troviamo invece nella figura 4.2 la traiettoria effettuata dal robot rispetto agli assi mobili, come si nota tale percorso risulta essere pressoché rettilineo a meno di qualche piccola oscillazione causata sia da un reale spostamento del robot dovuta alla giunzione del mattonato, sia da piccoli errori di individuazione generati dal programma a causa delle distorsioni residue e delle varie approssimazioni effettuate.



A



B

Figura 4.3: Andamento della velocita in funzione del tempo e analisi dell'errore di posizionamento

È stato realizzato un grafico che mostra l'andamento della velocità stimata del robot in funzione del tempo (Fig. 4.3 A), inoltre è stato possibile effettuare un confronto tra le misure effettuate dal software e le reali posizioni assunte dal robot (Fig. 4.3 B) posizionando un pennarello sotto di esso che ha tracciato al suolo il percorso effettuato, in tale modo è stato possibile misurare le varie posizione assunte da esso nel percorso. Come si può notare l'errore tra la posizione reale e quella calcolata, in questo caso, non supera il valore di 0.5 cm, dunque l'individuazione risulta essere molto buona.

4.2 Esempio 2: Percorso Generico

Questo secondo esempio mostra l'analisi di un generico percorso eseguito dall'Octagon.

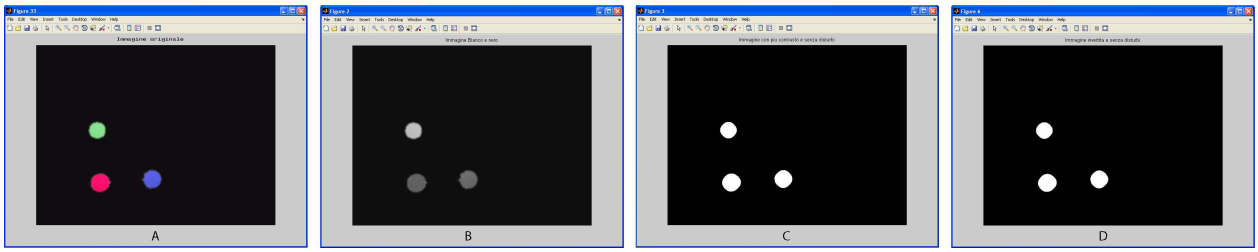


Figura 4.4: Procedura di calibrazione assi esempio 2

La fase di calibrazione, come si può notare risulta equivalente a quella dell'esempio precedente. Dopo aver effettuato i diversi filtraggi l'immagine risulta priva di disturbi (Fig. 4.4), risulta dunque possibile andare ad analizzare i risultati ottenuti in tale fase di calibrazione.

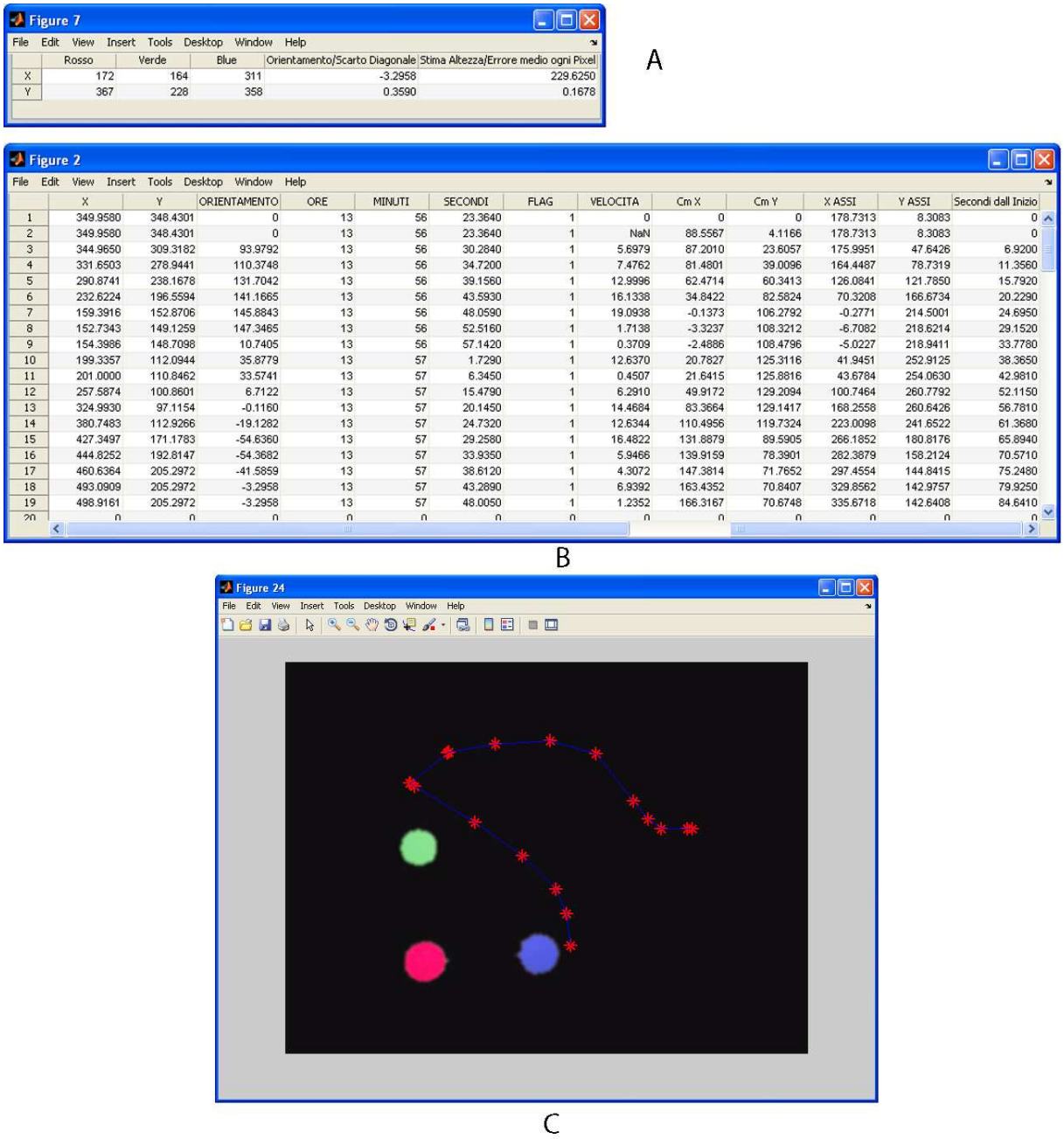


Figura 4.5: Risultati della calibrazione e del tracking esempio 2

Per ciò che riguarda i dati di calibrazione (Fig. 4.5 A) osserviamo un orientamento degli assi mobili di -3.2958 gradi e un'errore percentuale tra le diagonali dello 0.3590%, dunque ancora una volta molto buono. Questa volta l'altezza è variata ed è pari a circa 229.7[1] e con essa è variato

[1]Infatti l'altezza reale risulta essere di 230 cm.

anche l'errore della distorsione prospettica pari a 0.1678 ogni pixel. Con l'aumento dell'altezza c'è da aspettarsi anche un aumento nell'errore di individuazione del robot essendo maggiore il rapporto pixel-cm ad una stessa quantità di pixel corrisponderà un maggior numero di cm. Come è possibile notare (Fig. 4.5 C) in questo caso la traiettoria non risulta essere più rettilinea, ma il robot compie un percorso generico, è inoltre possibile notare che nella tabella di tracking (Fig. 4.5 B) anche se il tempo computazionale di ogni fotogramma si attesta intorno ai 3-4 secondi, vi sono dei buchi temporali, ad esempio tra le posizioni 11 e 12, ciò è dovuto all'algoritmo di controllo che nel caso di posizione invariata o di calcolo di una posizione non raggiungibile nel tempo trascorso, scarta il valore ed evita di scriverlo nella tabella.

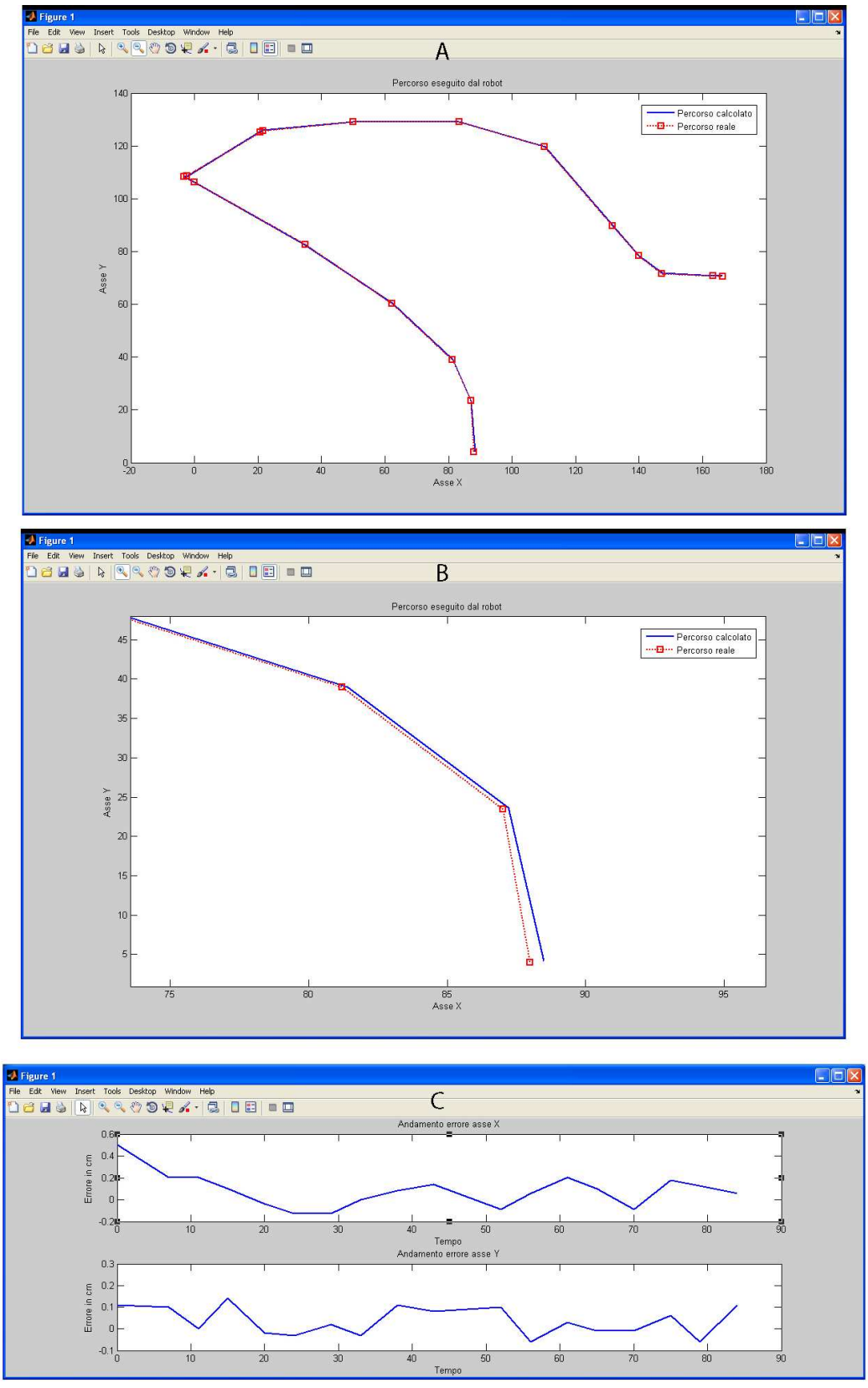


Figura 4.6: Analisi dell'errore di individuazione esempio 2

Tramite lo stesso accorgimento usato nell'esempio precedente è risultato possibile effettuare una stima dell'errore di individuazione del robot e sono stati sovrapposti i due percorsi, quello calcolato dal software e quello misurato sul piano di lavoro (Fig. 4.6 A), come si nota anche in questo i risultati del calcolo sono stati ottimi anche se resta comunque presente una piccola percentuale di errore (Fig. 4.6 B). L'andamento di tale errore è stato tracciato separatamente rispetto ai due assi (Fig. 4.6 C) e risulta essere inferiore ai 0.6 cm.

4.3 Esempio 3: Confronto Incrociato

Questo ultimo esempio è stato realizzato con l'ausilio di due webcam contemporaneamente montate sulla struttura e collegate a due diversi computer. Dopo aver sincronizzato gli orologi dei due pc è stato possibile avviare le fasi di calibrazione e monitaraggio del robot.

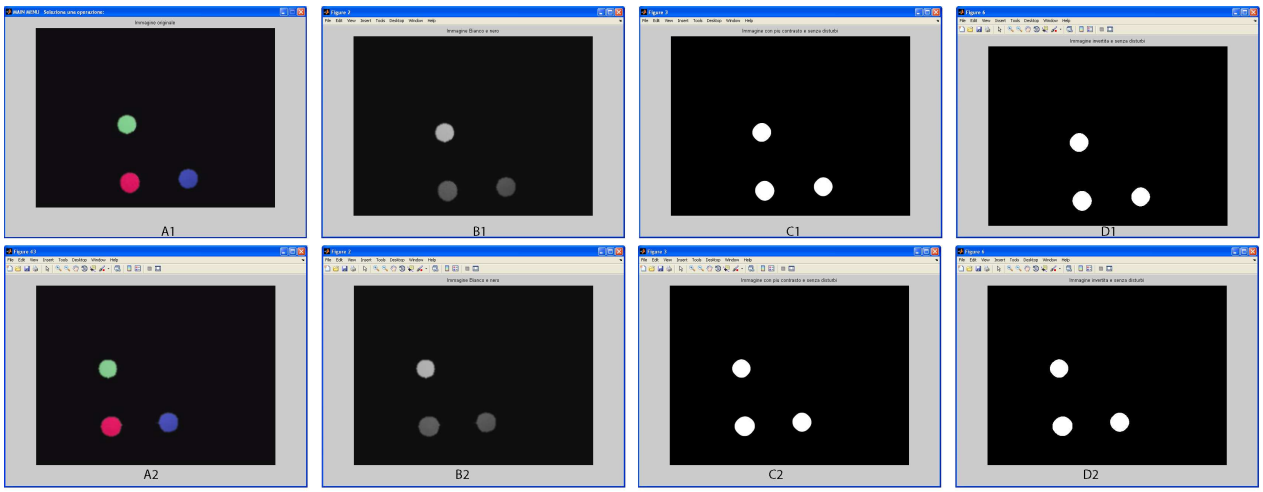


Figura 4.7: Procedura di calibrazione assi esempio 3

Tralasciando anche in questo caso la fase di filtraggio della calibrazione assi[2] (Fig. 4.7) andiamo ad analizzare i risultati calcolati dal software.

[2]Nella prima riga della figura sono riportate le immagini catturate dal computer 1 mentre nella seconda quelle catturate dal computer 2.

A1

Figure 2

File Edit View Insert Tools Desktop Window Help

	Rosso	Verde	Blue	Orientamento/Scarto Diagonale	Stima Altezza/Errore medio ogni Pixel
X	252	244	409	-2.9370	202.7021
Y	414	258	403	0.9399	0.1945

A2

Figure 2

File Edit View Insert Tools Desktop Window Help

	Rosso	Verde	Blue	Orientamento/Scarto Diagonale	Stima Altezza/Errore medio ogni Pixel
X	201	192	354	-3.3250	206.4859
Y	377	222	366	0.6909	0.1902

B1

Figure 3

File Edit View Insert Tools Desktop Window Help

	X	Y	ORIENTAMENTO	ORE	MINUTI	SECONDI	FLAG	VELOCITA	Cm X	Cm Y	X ASSI	Y ASSI	Secondi dall Inizio
1	191.1133	160.4597	0	14	46	25.0350	1	0	0	0	-47.8160	256.3269	0
2	191.1133	160.4597	0	14	46	25.0350	1	NaN	-21.0422	112.8006	-47.8160	256.3269	0
3	197.5577	164.0846	-32.2947	14	46	29.3310	1	1.7211	-18.2917	111.0622	-41.5659	252.3766	4.2960
4	225.7516	179.3899	-31.4326	14	46	31.4140	1	15.4010	-6.2459	103.7000	-14.1931	235.6468	6.3790
5	252.3345	195.0980	-33.5162	14	46	33.5570	1	14.4083	5.0827	96.1971	11.5500	218.5974	8.5220
6	277.3063	210.0005	-33.7646	14	46	35.7200	1	13.4445	15.7215	89.0846	35.7254	202.4349	10.6850
7	324.8332	239	-34.3272	14	46	37.9330	1	25.1585	35.9551	75.2680	81.7041	171.0384	12.8980
8	336.1108	241.8194	-16.9732	14	46	40.1270	1	5.2984	40.8479	73.7747	92.8224	167.6448	15.0920
9	348.1940	242.6249	-6.7510	14	46	42.2600	1	5.6774	46.1401	73.1482	104.8484	166.2213	17.2250
10	360.2771	240.6111	6.5254	14	46	44.4530	1	5.5859	51.4959	73.7608	117.0188	167.6134	19.4180
11	384.4433	230.1390	20.4917	14	46	46.5760	1	12.4059	62.3528	77.8182	141.6899	176.8334	21.5410
12	396.5265	216.8476	44.7894	14	46	48.6890	1	8.5011	67.9629	83.3872	154.4381	189.4883	23.6540
13	413.4428	195.5008	48.6679	14	46	50.8720	1	12.4768	75.8787	92.3875	172.4261	209.9404	25.8370
14	427.1370	177.7788	49.3688	14	46	52.9350	1	10.8562	82.2967	99.8672	187.0103	226.9374	27.9000
15	431.1647	171.7373	53.3730	14	46	55.1480	1	3.2811	84.2031	102.4316	191.3423	232.7646	30.1130
16	437.6091	164.0846	46.9621	14	46	57.2310	1	4.8030	87.2078	105.6496	198.1702	240.0770	32.1960
17	448.8867	159.2514	20.2616	14	46	59.4740	1	5.4702	92.2732	107.5194	209.6806	244.3261	34.4390
18	457.7476	160.4597	-10.7021	14	47	1.5870	1	4.2324	96.1402	106.7886	218.4681	242.6654	36.5520
19	490.7748	195.5008	-49.6316	14	47	5.9040	1	11.1542	109.8651	90.6438	249.6565	205.9781	40.8690
20	514.9411	225.7086	-54.2772	14	47	10.1000	1	9.2195	119.8048	76.8230	272.2432	174.5718	45.0650
21	536.6907	254.3053	-55.6817	14	47	14.4960	1	8.1729	128.7187	63.7647	292.4991	144.8982	49.4610
22	539.9129	276.0549	-84.5100	14	47	18.6720	1	5.2651	129.6444	54.1333	294.6026	123.0121	53.6370
23	538.3018	286.1242	-102.0272	14	47	22.8480	1	2.4419	128.7093	49.7443	292.4777	113.0386	57.8130
24	539.9129	286.1242	-2.9370	14	47	48.7550	1	0.0622	129.4173	49.7080	294.0867	112.9560	83.7200
25	538.3018	286.1242	177.0630	14	48	1.7640	1	0.1238	128.7093	49.7443	292.4777	113.0386	96.7290

B2

Figure 2

File Edit View Insert Tools Desktop Window Help

	X	Y	ORIENTAMENTO	ORE	MINUTI	SECONDI	FLAG	VELOCITA	Cm X	Cm Y	X ASSI	Y ASSI	Secondi dall Inizio
1	145.0869	130.0842	0	14	46	24.8900	1	0	0	0	-41.4980	249.7431	0
2	145.0869	130.0842	0	14	46	24.8900	1	NaN	-18.5537	111.6600	-41.4980	249.7431	0
3	150.7554	132.5136	-26.5236	14	46	29.0150	1	1.4951	-16.0866	110.4287	-35.9800	246.9890	4.1250
4	184.7663	150.3288	-30.9710	14	46	31.5930	1	14.8931	-1.3680	101.5950	-3.0596	227.2312	6.7030
5	221.2065	171.3831	-33.3434	14	46	34.9840	1	12.4109	14.3511	91.2525	32.0981	204.0988	10.0940
6	263.3152	200.5353	-38.0201	14	46	37.9680	1	17.1633	32.3902	77.1486	72.4451	172.5534	13.0780
7	290.8478	213.4918	-28.5261	14	46	40.5310	1	11.8724	44.3433	70.6515	99.1799	158.0218	15.6410
8	305.4239	214.3016	-6.5048	14	46	43.4840	1	4.9436	50.8283	69.9121	113.6845	156.3680	18.5940
9	334.5761	206.2038	12.1991	14	46	46.5460	1	9.8811	64.0503	72.7706	143.2573	162.7614	21.6560
10	354.8207	187.9837	38.6622	14	46	49.5780	1	8.9829	73.5589	80.3781	164.5246	179.7766	24.6880
11	379.9239	156.4022	48.1948	14	46	52.2960	1	14.8429	85.5826	93.8234	191.4173	209.8490	27.4060
12	389.6413	144.2554	48.0152	14	46	55	1	5.7527	90.2349	98.9931	201.8228	221.4117	30.1100
13	400.9783	135.7527	33.5449	14	46	57.4530	1	5.7771	95.5156	102.4943	213.6338	229.2426	32.5630
14	412.3152	132.5136	12.6204	14	46	59.8590	1	4.9005	100.6598	103.6461	225.1396	231.8187	34.9690
15	415.5544	133.7282	-23.8810	14	47	2.6560	1	1.2368	102.0741	103.0199	228.3028	230.4182	37.7660
16	435.7989	150.7337	-43.3552	14	47	6	1	7.9064	110.6692	94.9046	247.5270	212.2672	41.1100
17	464.9511	195.6766	-60.3556	14	47	8.9530	1	18.1408	122.5158	74.0885	274.0234	165.7091	44.0630
18	469.0000	199.3206	-45.3122	14	47	12.1250	1	1.7173	124.2285	72.3570	277.8542	161.8364	47.2350
19	483.5761	219.1603	-57.0205	14	47	14.9530	1	8.7053	130.2200	63.1237	291.2551	141.1847	50.0630
20	488.4348	226.4484	-59.6349	14	47	17.6400	1	3.2598	132.1997	59.7447	295.6829	133.6272	52.7500
21	490.0544	256.8152	-90.2721	14	47	20.8280	1	9.5389	132.1352	46.1485	295.5384	103.2175	55.9380
22	0	0	0	0	0	0	0	0	0	0	0	0	0

Figura 4.8: Risultati della calibrazione e del tracking esempio 3

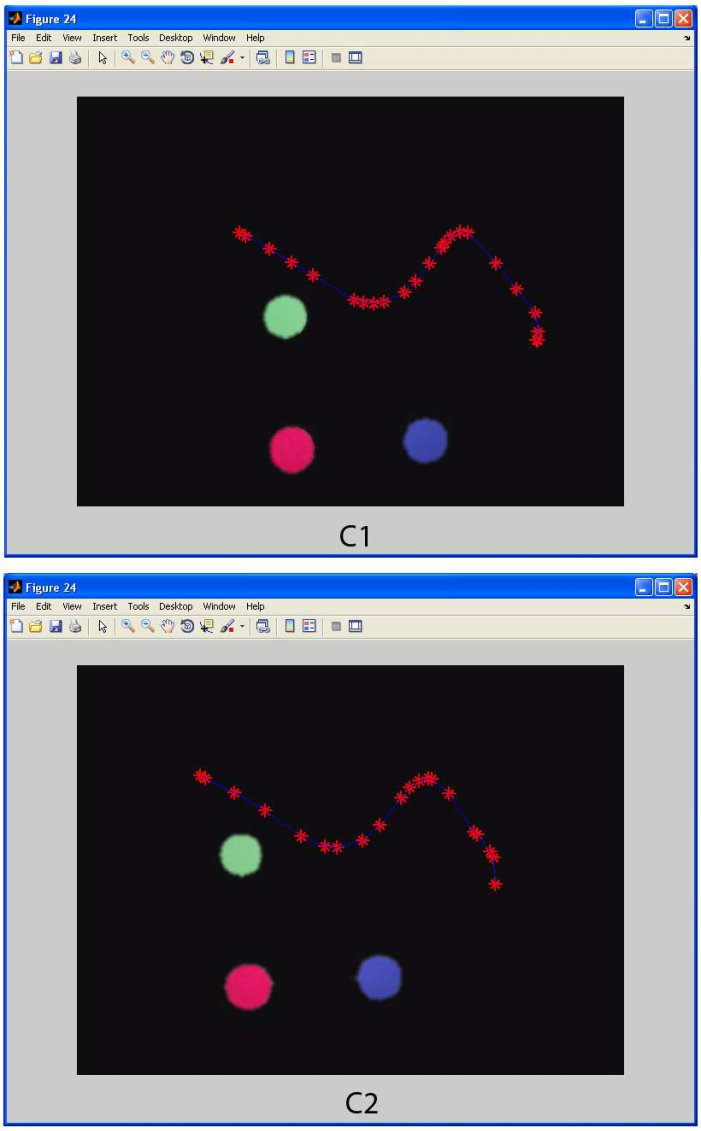


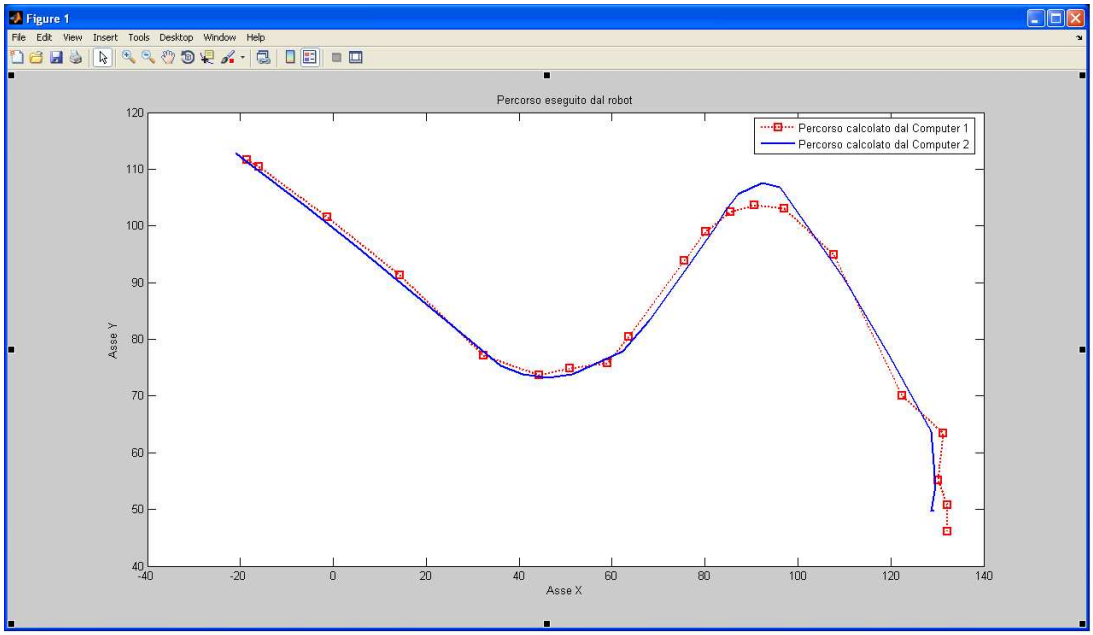
Figura 4.9: Confronto tra i due percorsi calcolati

Vediamo nelle figure 4.8 A1 e A2, i risultati delle calibrazioni effettuate sui due computer a confronto, come è possibile notare i valori calcolati sono leggermente diversi, ciò è dovuto in primo luogo al fatto che le due webcam sono collocate in due posizioni diverse[3], in secondo luogo al fatto che la seconda webcam non sia perfettamente perpendicolare al piano di lavoro. Infatti essendo la struttura realizzata nata per ospitare una sola webcam la seconda è stata fissata con il materiale reperibile in

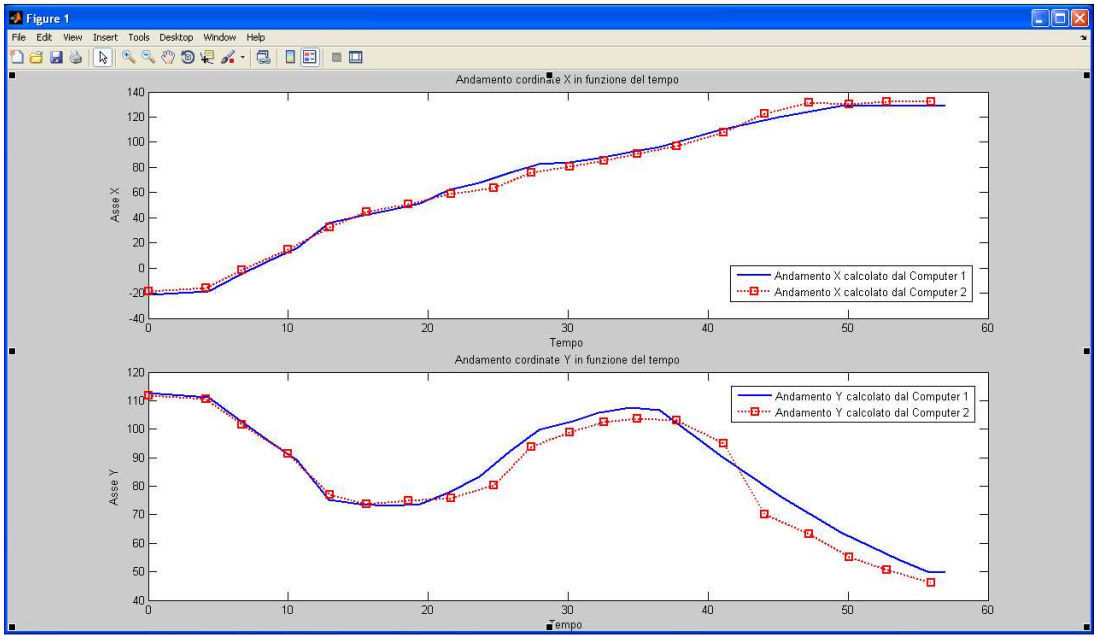
[3]Tale affermazione riguarda in particolar modo il fatto che sia le posizioni dei riferimenti, sia l'angolo di orientamento e sia il valore dell'altezza stimata, siano diversi.

laboratorio, dunque la sua posizione non è perfetta, questo fatto spiega il perchè dell'elevato valore dello scarto percentuale tra le diagonali di una delle due webcam pari a 0.9399, un valore quasi limite che comporterà come si vedrà in seguito un'errore nella trasformazione tra coordinate fisse e quelle mobili e dunque un'errore complessivo nell' individuazione del robot.

Nelle figure 4.8 B1 e B2 troviamo le tabelle relative ai dati del tracking calcolati da due PC, come è possibile notare il computer 1 avendo caratteristiche tecniche superiori al secondo nello stesso intervallo di tempo è riuscito ad elaborare 25 fotogrammi anziché 22. Come si può osservare nelle figure 4.9 C1 e C2, i due percorsi calcolati sembrano essere identici ma mettendo a confronto i due percorsi sulla stessa immagine si notano alcune differenze.



A



B

Figura 4.10: Confronto tra i percorsi calcolati dai due computer

Come si nota nella figura 4.10 A, i due percorsi si discostano in particolar modo nella fase finale del percorso, tali differenze nelle traiettorie sono dovuti come già accennato al cattivo posizionamento

della webcam che ha portato ad errori nel calcolo dei valori nella matrice di rototraslazione "M" e dunque al calcolo errato delle posizione del robot in uno dei due casi. In particolar modo l'errore risulta significativo (Fig. 4.10 B) lungo l'asse Y dove lo scarto tra le due misure calcolate raggiunge i 5-6 cm. Questo comunque resta sempre un buon valore, infatti, essendo l'area di lavoro ripresa di circa 360x300 cm, l'errore percentuale tra le posizioni stimate dai due PC resta inferiore all'1.5%.

Capitolo 5

Conclusioni e sviluppi futuri

Il lavoro di questa tesi ha portato alcuni contributi interessanti, per questo è possibile individuare alcune linee di sviluppo ulteriori verso cui far confluire il software realizzato, infatti come già accennato nell'introduzione i contesti applicativi in cui è possibile utilizzare le tecniche discusse in questa tesi sono molteplici. Fondamentalmente l'applicazione è stata concepita come uno strumento che permetta di misurare in modo accurato la posizione di un'oggetto nel piano di lavoro, con lo scopo di ottenere un "ground true", ovvero un riscontro che risulta essere di fondamentale importanza nelle fasi di test di algoritmi di localizzazione. Inoltre questa tesi mira alla realizzazione di uno dei mattoni fondamentali per la costruzione di sistemi robotici più avanzati.

Il programma come già visto, è in grado di riconoscere un piano di lavoro con le coordinate mobili indicate dall'utente, riconoscere colori, individuare e inseguire un oggetto che si muove sul piano di lavoro, dunque trova impiego in una qualsiasi applicazione che richieda il tracciamento di oggetti ed il calcolo delle coordinate in tempo reale, anche in ambienti esterni e ricchi di rumore.

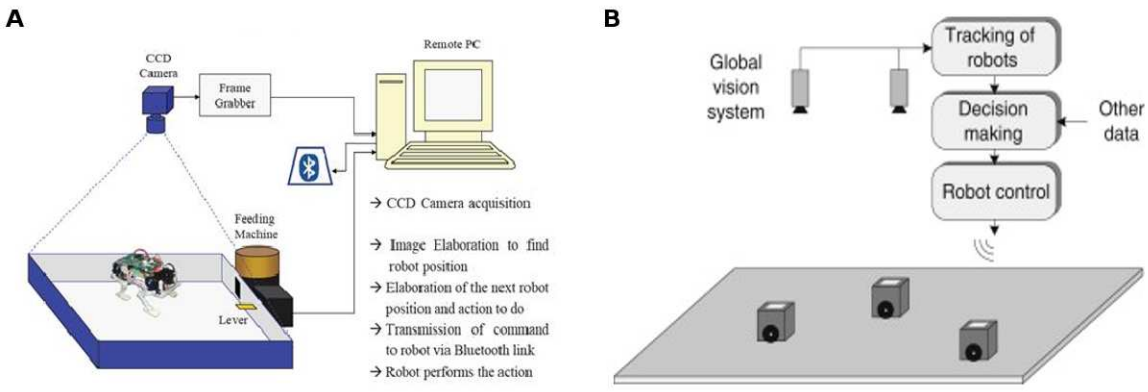


Figura 5.1: Esempi di sistemi chiusi per il controllo di robot mobili

Il prossimo passo da compiere, a mio parere, è quello di integrare tale software con quello del robot, in modo tale da poter restituire i valori di posizione calcolati direttamente al software del robot stesso, così facendo sarà possibile pianificare percorsi ed evitare ostacoli (Fig. 5.1 A). Inoltre tale meccanismo potrà essere esteso anche a sistemi più complessi dotati di piú webcam in grado di monitorare spazi di lavoro più estesi (Fig. 5.1 B) oppure apportando piccole modifiche nel codice il software può essere in grado di analizzare il movimento di più oggetti, come più robot o un robot ed altri ostacoli mobili riconoscibile sempre tramite diversi colori.

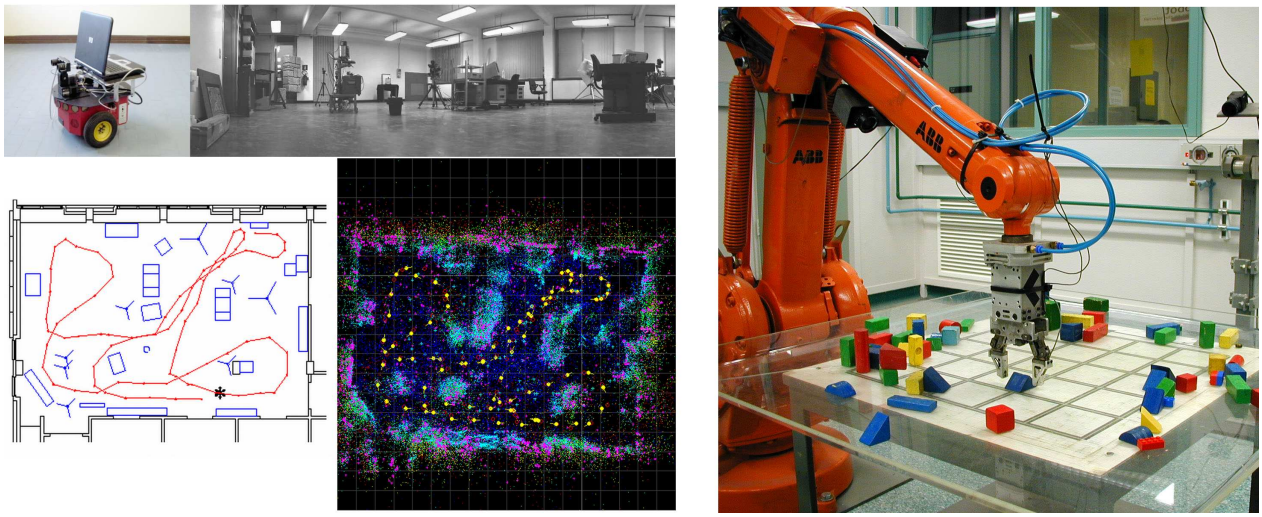


Figura 5.2: Possibili applicazioni future

Un ulteriore sviluppo potrebbe essere quello di estendere tale software alla visione sterescopica con l'utilizzo di due webcam simultaneamente, in tale modo si potrebbe ricostruire un mappa tridimensionale dell'ambiente circostante e dunque anche in tale caso si potrebbero riconoscere ed evitare ostacoli mobili e fissi ed individuare il percorso ottimo tra due punti della mappa (Fig. 5.2), oppure nel caso di robot manipolatori sarebbe possibile restituire le coordinate dell'oggetto da prendere direttamente al manipolatore (Fig. 5.2).

Figura 5.3: Possibile sistema di video sorveglianza

Inoltre il software potrebbe essere modificato al fine di realizzare un programma di sorveglianza per il tracciamento di persone o automezzi (Fig. 5.3). Infine la struttura portante realizzata potrà essere utilizzata per qualsiasi operazione che richieda l'uso di una webcam fissa per il monitoraggio.

Capitolo 6

Appendice A
Correzione distorsione radiale

6.1 Introduzione

La distorsione di un'immagine deriva da piccole deficienze ottiche, dovute ad imperfezioni nelle lenti ed al loro non corretto allineamento, gli effetti di ciò sono comunemente detti aberrazioni le quali causano una degradazione dell'immagine finale(Fig. 6.1). Al contrario degli altri tipi di aberrazioni la distorsione non degrada la qualità dell'immagine ma ne influenza fortemente la geometria.

Possiamo individuare fondamentalmente due tipi di distorsione: una radiale che varia allontantanandosi dal centro dell'immagine, l'altra tangenziale che invece varia in modo perpendicolare alla prima. Quantitativamenente la deformazione tangenziale è trascurabile rispetto a quella radiale e pertanto viene di norma ignorata durante la fase di correzione dell'immagine. La quasi totalità della

Figura 6.1: Distorsioni

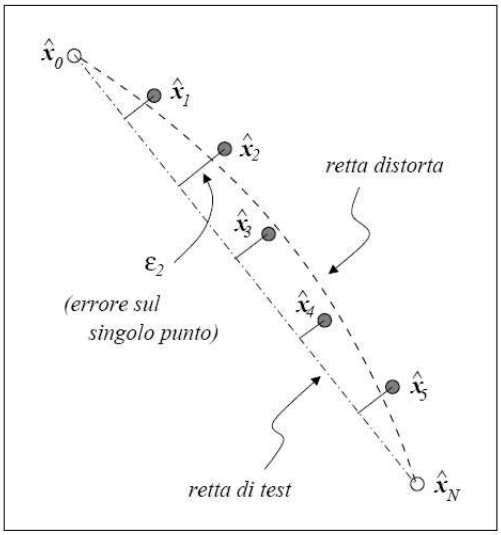


Figura 6.2: Effetto distorsione

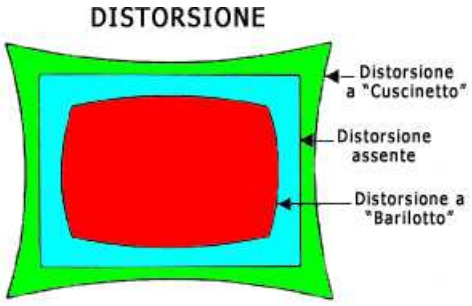


Figura 6.3: Tipi di distorsione

deformazione è di natura radiale, questo spiega perché ai bordi dell'immagine l'effetto è più evidente.

L'effetto visibile della distorsione radiale(Fig. 6.2) è quello di curvare quelle che nella realtà sono rette o meglio di spostare dei generici punti lungo una direzione radiale rispetto alla loro posizione effettiva.

Se lo spostamento avviene in direzione centrifuga (Fig. 6.3) la distorsione è detta a "botte", se al contrario è centripeda viene detta a "cuscino". Esistono opportune funzioni in grado di descrivere e correggere questi tipi di distorsione ed internazionalmente viene definito uno standard per la struttura matematica di tale funzione:

$$Distorsione = A_1 * r(r^2 - R_0^2) + A_2 * r(r^4 - R_0^4)$$

aquistando una fotocamera metrica viene fornito un certificato che riporta il valore di tali variabili dove "r" è la distanza focale e le altre 3 variabili descrivono la distorsione secondo tale standard. Al contrario invece lavorando con fotocamere amatoriali, come nel nostro caso, è necessario stimare tale distorsione con opportuni algoritmi come vedremo nei prossimi paragrafi.

6.2 Modello di Pinhole

Nella maggior parte dei casi la precisione di un sistema visivo dipende fortemente dall'accuratezza con cui è stata eseguita la calibrazione della "camera", dunque questo si rivela un passo molto importante nella realizzazione di un qualsiasi sistema ottico.

Calibrare una fotocamera significa estrarne la posizione del piano di terra nel riferimento scelto ed una serie di parametri necessari alla successiva elaborazione delle immagini e come si vedrà in seguito un metodo per effettuare tale calibrazione è quello di inquadrare una scacchiera di dimensioni note collocata sul piano terra. Comunemente tali parametri sono divisi in due categorie: parametri estrinseci [1] ed intrinseci, ma solo questi ultimi sono quelli che a noi interessano in quanto dipendono strettamente dall'ottica del dispositivo in uso.

Tali parameri includono:

- lunghezza focale (f_0)
- fattore di scala (s_u)
- centro dell'immagine ($C_c = [u_0, v_0]$)
- fattore di distorsione radiale (k)

Pirma di procedere con la fase di calibrazione vera e propria sembra opportuno richiamare il modello della camera oscura (anche detto di Pinhole), il quale a dispetto della sua semplicità riesce a modellare

[1] Tali parametri sono necessari alla trasormazione da coordinate dell'immagine a coordinate reali e sono stati affrontati nel Capitolo 2.

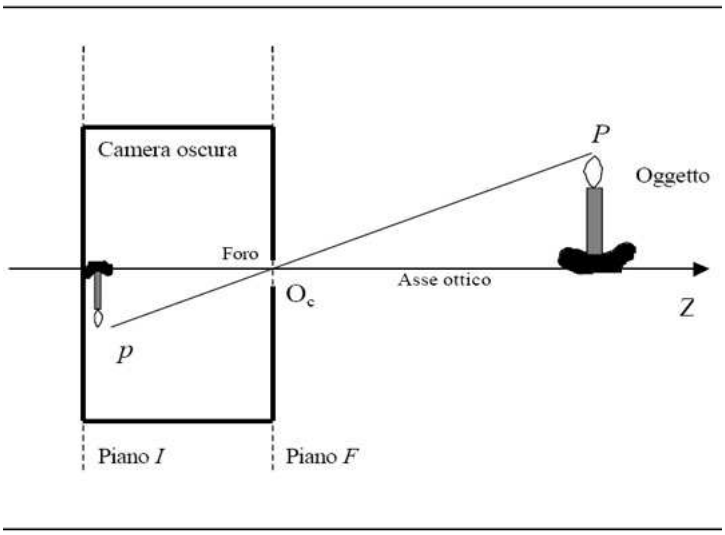


Figura 6.4: Il modello della camera oscura

abbastanza bene un generico sistema video CCD. Si consideri una scatola chiusa in cui è stato praticato un piccolo foro sul piano F (Figura 6.4). I raggi luminosi che partono da un oggetto esterno entrano nel foro e formano sul piano I l'immagine invertita di tale oggetto.

I piani F e I sono detti rispettivamente "Piano focale" e "Piano immagine"(Figura 6.5) e la distanza tra loro è detta distanza focale f_0. Il punto in cui è stato praticato il foro si indica come "centro ottico", mentre la retta normale al "piano focale" passante per il foro è detta "asse ottico", infine la proiezione invertita dell'oggetto sul "piano immagine" è detta "proiezione percettiva" dell'oggetto. Si supponga adesso di scegliere un sistema di riferimento di assi cartesiani X Y Z di centro O_c come in figura 6.5 in cui l'asse X è stato scelto ortogonale al piano individuato dall'asse ottico e da Y. Se P è un punto dell'oggetto, la sua posizione nel sistema di riferimento scelto sarà espressa dal vettore $X_p = [X_p\ Y_p\ Z_p]^T$. La proiezione percettiva di P sul piano immagine individua il punto ξ_p la cui posizione è data dal vettore $\overline{\xi_p} = [x_p\ y_p\ z_p]^T$.

Per passare dalle coordinate spaziali di $\overline{X_p}$ a quelle immagine di $\overline{\xi_p}$ è necessaria la seguente matrice

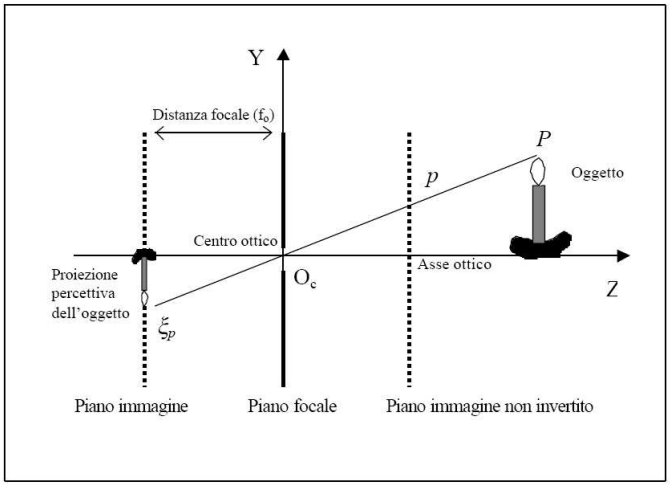


Figura 6.5: Schematizzazione del modello della camera oscura

di rototraslazione.

$$\begin{bmatrix} x_p \\ y_p \\ z_p \end{bmatrix} = \begin{bmatrix} m_{11} & m_{12} & m_{13} \\ m_{21} & m_{22} & m_{23} \\ m_{31} & m_{32} & m_{33} \end{bmatrix} \begin{bmatrix} X_p \\ Y_p \\ Z_p \end{bmatrix} + \begin{bmatrix} x_0 \\ y_0 \\ z_0 \end{bmatrix}$$

dove:

$m_{12} = sen\omega sen\phi cos\kappa - cos\omega sen\kappa$

$m_{13} = cos\omega sen\phi cos\kappa + sen\omega sen\kappa$

$m_{21} = cos\phi cos\kappa$

$m_{22} = sen\omega sen\phi sen\kappa + cos\omega cos\kappa$

$m_{23} = cos\omega sen\phi sen\kappa - sen\omega cos\kappa$

$m_{31} = -sen\phi$

$m_{32} = sen\omega cos\phi$

$m_{33} = cos\omega cos\phi$

Lo spostamento complessivo è rappresentato usando gli angoli di Eulero $\omega,\ \phi,\ \kappa$ che definiscono una sequenza di rotazioni elementari rispettivamente intorno ai tre assi X, Y, e Z.

Poiché vale il principio di omotetia[2] tra i due triangoli di diagonali $\overline{O_cP}$ e $\overline{O_c\xi_p}$ possiamo scrivere:

[2]L'omotetia è un particolare tipo di similitudine tra triangoli.

$\frac{X_p}{Z_p}=\frac{x_p}{z_p}$ e $\frac{Y_p}{Z_p}=\frac{y_p}{z_p}$

ora dato che z_p è pari alla lunghezza focale f_0, il vettore ξ_p può essere espresso nel seguente modo:

$$\xi_p=\begin{bmatrix} -f_0\frac{X_p}{Z_p} \\ -f_0\frac{Y_p}{Z_p} \\ -f_0 \end{bmatrix}$$

dove $z_p = -f_0$ trattandosi della proiezione percettiva e dunque invertita dell'oggetto.

Nella pratica può risultare tuttavia più comodo riferirsi non alla proiezione percettiva dell'oggetto sul piano immagine (che fornisce una visione ribaltata dell'oggetto stesso), ma bensì alla sua proiezione non invertita descritta dal vettore $\overline{x}_p$

$$\overline{x}_p=\begin{bmatrix} f_0\frac{X_p}{Z_p} \\ f_0\frac{Y_p}{Z_p} \\ f_0 \end{bmatrix}$$

È importante però tenere presente il fatto che nel piano dell'immagine le coordinate saranno bidimensionali e dunque usando sempre il modello di Pinhole le proiezioni del punto $\overline{X_p}$ nel piano immagine possono essere espresse come:

$$\begin{bmatrix} u_p \\ v_p \end{bmatrix}=\frac{f_0}{Z_p}\begin{bmatrix} x_p \\ y_p \end{bmatrix}.$$

L'unità di misura nell'immagine è il pixel dunque risulta necessario il coefficiente "D"[3] per passare dalle coordinate metriche in pixel,come segue:

$$\begin{bmatrix} u_p^{'} \\ v_p \end{bmatrix}=\begin{bmatrix} D*s_u u_p \\ D*v_p \end{bmatrix}+\begin{bmatrix} u_0 \\ v_0 \end{bmatrix}$$

La distorsione radiale può essere approssimata usando la seguente espressione[4]:

[3]Tale coefficiente è stato opportunamente calcolato nel Capitolo 2 durante la fase di "Calibrazione assi".
[4]L'apice (r) stà ad indicare solamente che si tratta della distorsione radiale.

$$\begin{bmatrix} \delta_{u_p^{(r)}} \\ \delta_{v_p^{(r)}} \end{bmatrix} = \begin{bmatrix} u_p(k_1 r_p^2 + k_2 r_p^4 + ...) \\ v_p(k_1 r_p^2 + k_2 r_p^4 + ...) \end{bmatrix}$$

dove k_1 e k_2 sono i coefficienti della distorsione radiale e $r_p = \sqrt{u_p^2 + v_p^2}$.

Il centro di curvatura delle lenti non risulta sempre perfettamente collineare, questo introduce un'altro tipo di distorsione molto comune, il "decentramento dell'immagine" che può essere scomposto in due componenti una radiale ed una tangenziale, di cui però la componente radiale risulta essere trascurabile, al contrario quella tangenziale è descritta dalla seguente equazione:

$$\begin{bmatrix} \delta_{u_p^{(t)}} \\ \delta_{v_p^{(t)}} \end{bmatrix} = \begin{bmatrix} 2p_1 u_p v_p + p_2(r_p^2 + 2u_p^2) \\ p_1(r_p^2 + 2v_p^2) + 2p_2 u_p v_p \end{bmatrix}$$

Un modello appropriato in grado di consentire una calibrazione accurata può essere ottenuto combinando il modello di Pinhole con le componenti della distorsione radiale e tangenziale precedentemente illustrate ottenendo il seguente modello:

$$\begin{bmatrix} u_p \\ v_p \end{bmatrix} = \begin{bmatrix} D * s_u(u_p + \delta u_p^{(r)} + \delta u_p^{(t)} \\ D * (v_p + \delta v_p^{(r)} + \delta v_p^{(t)}) \end{bmatrix} + \begin{bmatrix} u_0 \\ v_0 \end{bmatrix}$$

nella quale troviamo proprio tutti i parametri intrinseci della camera che saranno le nostre incognite da calcolare.

6.3 Stima dei parametri della webcam

In questo paragrafo verrà trattato sommariamente, per ragioni di spazio, un metodo per la stima dei parametri intrinseci della camera tramite DLT. La trasformazione lineare diretta (DLT) fu orginariamente sviluppata da Abdel-Aziz e Karara [1].

Il metodo dell DLT è basato sul modello di Pinhole e non tiene conto delle componenti non lineari della distorsione radiale e tangenziale. La trasformazione dalle coordinate dell'oggetto reale a quelle

immagine può essere risolta usando la matrice A e può essere scritta la seguente equazione:

$$\begin{bmatrix} u_p w_p \\ v_p w_p \\ w_p \end{bmatrix} = \begin{bmatrix} a_{11} & a_{12} & a_{13} & a_{14} \\ a_{21} & a_{22} & a_{23} & a_{24} \\ a_{31} & a_{32} & a_{33} & a_{34} \end{bmatrix} \begin{bmatrix} X_p \\ Y_p \\ Z_p \\ 1 \end{bmatrix}$$

dove la matrice 3 X 4 è la matrice A i cui parametri possono essere calcolati nel seguente modo

$A = \lambda V^{-1} B^{-1} F M T$

$$\text{V}= \begin{bmatrix} 1 & 0 & -u_0 \\ 0 & 1 & -v_0 \\ 0 & 0 & 1 \end{bmatrix} \quad \text{B}= \begin{bmatrix} 1-b_1 & b_2 & 0 \\ b_2 & 1-b_1 & 0 \\ 0 & 0 & 1 \end{bmatrix} \quad \text{F}= \begin{bmatrix} f_0 & 0 & 0 \\ 0 & f_0 & 0 \\ 0 & 0 & 1 \end{bmatrix}$$

dove λ risulta essere un fattore di scala complessivo, le matrici M e T sono le matrici di rotazione e traslazione per passare dalle coordinate reali a quelle immagine [5], b_1 e b_2 sono coefficienti lineari della distorsione, f_0 è la lunghezza focale.

Una procedura dettagliata per la stima di tali parametri è consultabile nel testo di Melen [2].

[5] vedi Appendice A, Paragrafo 2

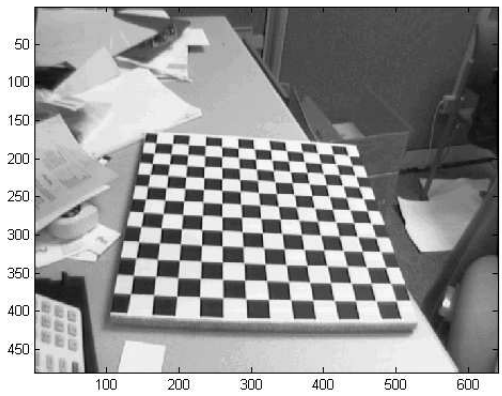


Figura 6.6: Scacchiera per la calibrazione

6.4 Calibrazione obiettivo

Come già anticipato nell'introduzione, lo scopo della calibrazione della telecamera è quello di stimare i parametri intrinseci (lunghezza focale, centro dell'immagine e fattore di distorsione radiale) al fine di poter eliminare totalmente, o comunque in gran parte, l'effetto della distorsione radiale.

Tale operazione può essere svolta posizionando una scacchiera di forma e dimensione nota sul piano di terra (Fig. 6.6). A partire dall'immagine catturata dalla telecamera si confrontano le proiezioni degli spigoli o dei vertici della scacchiera sul piano immagine con quanto si conosce a priori della scacchiera stessa (ad esempio le dimensioni dei quadretti ed il loro numero). Si prenda come riferimento la Figura 6.7, nella quale è stato scelto, oltre al solito sistema di riferimento di assi cartesiani $(0_c,\ X, Y, Z)$ con 0_c coincidente con il centro ottico, anche il sistema di riferimento $(0_d,\ X_d, Y_d, Z_d)$ solidale con il piano di terra, in cui il piano individuato dagli assi X_d e Y_d coincide con il piano di terra.

Si consideri un punto P_k sulla scacchiera e si indichi con $\overline{X_d}^k$ il vettore le cui coordinate individuano la posizione di P_k nel riferimento $(0_d,\ X_d, Y_d, Z_d)$. Analogamente si indichi con $\overline{X_c}^k$ il vettore le cui coordinate individuano la posizione di P_k nel riferimento $(0_c,\ X, Y, Z)$. Infine si indichi con $\overline{x_c}^k$ il vettore immagine individuante il punto p_k proiezione di P_k nel piano immagine.

L'unica cosa di cui si è a conoscenza sono le coordinate di P_k nel riferimento solidale con la scacchiera

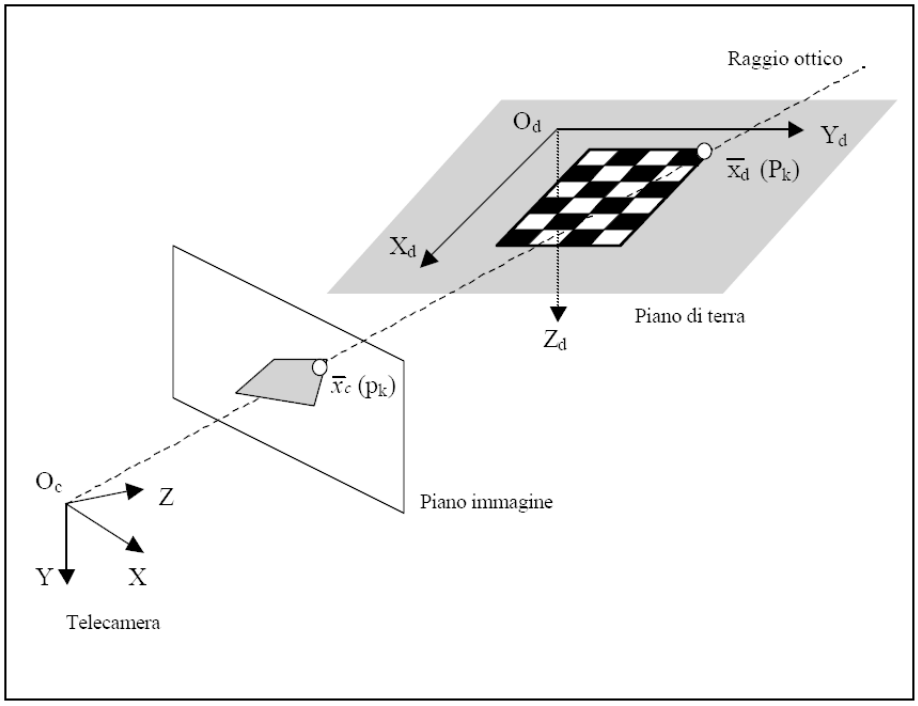


Figura 6.7: Schematizzazione delle grandezze in gioco nell'operazione di calibrazione

(dunque il vettore $\overline{X_d}^k$ è noto). In realtà esiste un'unica trasformazione rigida che permette di passare dal riferimento $(0_d,\ X_d, Y_d, Z_d)$ a quello $(0_c,\ X, Y, Z)$ e vicecersa, proprio quello definito nel paragrafo precedente tramite le matrici M e T e la relazione:

$\overline{X_c}^k = M * \overline{X_d}^k + T$

per approfondimenti vedi anche nota bibliografica [3] di Heikkila.

6.5 Guida alla prima calibrazione

Il software che è stato utilizzato per la calibrazione ‘e un toolbox di Matlab completamente gratuito reperibile nel sito "http://www.vision.caltech.edu/bouguetj/".

le operazioni preliminari da fare sono le seguenti:

- scarica dalla sezione download del sito il file .zip contenente tutto il necessario
- decomprimi l'archivio in una cartella chiamata ad esempio "Tool-calibration"

Figura 6.8: Selezione metodo di lavoro

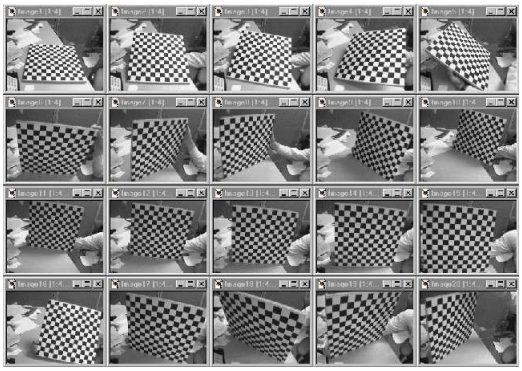


Figura 6.9: Esempio Immagini

- inserisci tale cartella nella directory "toolbox" di Matlab
- avvia Matlab e esegui "file/setPath.." , esegui il comando "Add folder" ed aggiungi la cartella "Tool-calibraton" precedentemente creata
- salva le impostazioni e riavvia Matlab.

Ora il TOOLBOX è pronto all'uso e può essere eseguito direttamente digitando dalla shell di Matlab il seguente comando "calib(underscore)gui". Apparirà una finestra (Fig. 6.8) dalla quale è possible scegliere la modalità di lavoro desiderata, se si hanno a disposizione almeno 512 MB di RAM la più opportuna e veloce risulta sicuramente la modalità "Standard". Dopo aver selezionato una delle modalita sarà possibile iniziare il lavoro di calibrazione vero e proprio.

Nell'archivio .zip precedentemente scaricato troverete un file "pattern.eps" stampatelo su un cartoncino e questa sarà la nostra scacchiera. È importante che tale scacchiera sia perfettamente piana, con le caselle ben contrastate e perfettamente identiche. Dobbiamo ora riprendere almeno 20 immagini della scacchiera orientate in maniera differente l'una dall'altra(Fig. 6.9), stando attenti che:

Figura 6.10: Interfaccia grafica

- non sia mai perfettamente perpendicolare all'asse della camera
- in ogni scatto ci siano almeno 4 caselle per lato e possibilmente in numero diverso sui due lati
- tutti i vertici siano presenti nell'immagine
- la scacchiera riempia il più possibile il campo di ripresa.

Dalla schermata in Figura 6.10 selezionare "Read Images" e scrivere nella shell di Matlab il nome delle immagini senza numero e senza estensione, ad esempio:
se gli scatti sono stati chiamati Immagine2, Immagine3...ect, dobbiamo scrivere solo "Immagine", fatto ciò il programma cercherà tutte le immagini così chiamate [6].
Ora selezionate l'estensione della vostra immagine come richiesto nella shell, una volta caricate tutte le immagini in memoria tornate all'interfaccia grafica (Fig. 6.10) e selezionate il comando "Extract Grid Corners", premere "Invio" dalla shell per esaminare tutte le immagini. Ora verranno richiesti alcuni parametri riguardanti il layout delle finestre, basta lasciare quelli di default premendo tre volte il tasto "Enter" ed infine una quarta volta per fare in modo che il software riconosca automaticamente il numero di vertici nell'immagine.

Il passo successivo è quello di andare ad individuare, con quattro click del mouse, i vertici estremi della scacchiera (Fig. 6.11) una volta fatto ciò (Fig. 6.12), verrà richiesta la dimensione delle caselle della scacchiera in mm[7], a questo punto è sufficiente misurare tale dimensione sul foglio stampato e riportarla nella shell, ed il numero di caselle sull'asse X e Y,

[6] Se durante questa fase compare il messaggio "OUT OF MEMORY" non avete abbastanza RAM installata dunque dovete scegliere la seconda modalità di lavoro "Memory efficient".
[7] Questa parte della procedura deve essere effettuata solo per la prima immagine, infatti a partire dalla successive il software proporrà automaticamente i vertici della scacchiera stimati, nel caso in cui non fossero tutti esatti è comunque possibile apportare delle correzioni.

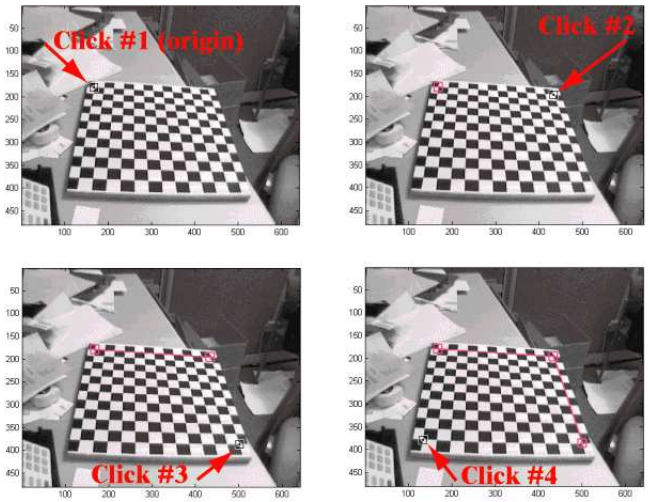


Figura 6.11: Esempio di individuazione vertici

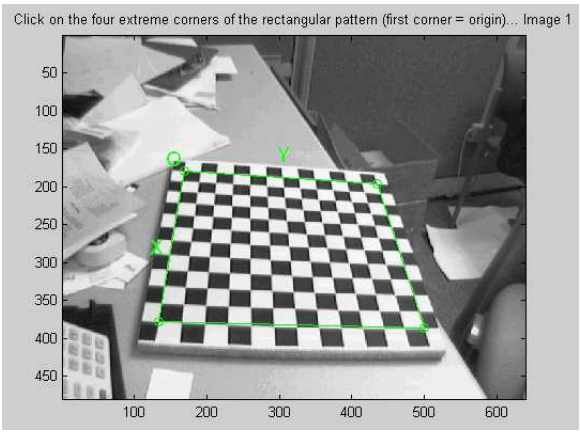


Figura 6.12: Vertici individuati

fatto tutto questo il programma riconoscerà automaticamente i vertici delle figure(Fig. 6.13 in quanto proprio da ciò dipenderà l'accuratezza delle stime e dunque del nostro sistema complessivo.

A questo punto sarà sufficiente ripetere tale procedura per tutte le immagini successive, prestando molta attenzione all'individuazione dei vertici.

Il passo successivo da effettuare è quello di selezionare il comando "Calibration" dal menù grafico (Fig. 6.10), attraverso il quale verranno analizzati tutti i dati precedentemente acquisiti estraendo tutti i parametri entrinseci ed intrinseci, come descritto nel paragrafo "Stima dei parametri" di questa appendice. Dopodiché, sempre nel Menù grafico del toolbox sarà possibile analizzare i dati

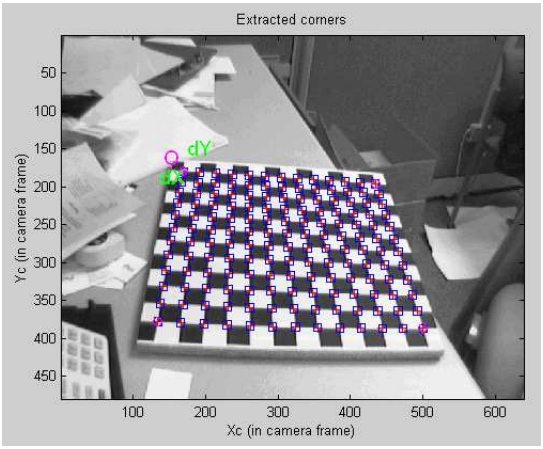


Figura 6.13: Individuazione automatica dei vertici

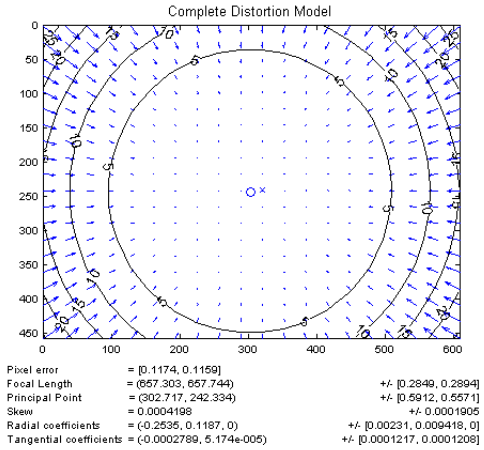


Figura 6.14: Modellazione della distorsione radiale

rilevati. Selezionando "Show calib results"(Fig. 6.14) verrà mostrata una rappresentazione grafica che mostra l'entità della distorsione radiale nell'immagine e verranno elencati i valori dei parametri entrinseci della camera. Selezionando al contrario il comando "Show Extrinsic"(Fig. 6.15) verrà mostrata una stima della posizione di ripresa durante lo scatto delle foto usate per la calibrazione, ovvero per ciascuna foto scattata verrà mostrata la posizione e l'orientamento stimato della webcam rispetto al piano terra. Il comando "Analyse error" mostra la distribuzione dell'errore nella stima dei vertici all'interno dell'immagine(Fig. 6.16) inoltre, facendo un click sul pixel desiderato, verrà mostrato il valore esatto dell'errore (Fig. 6.17). Questo strumento risulta essere di grande utilità nel

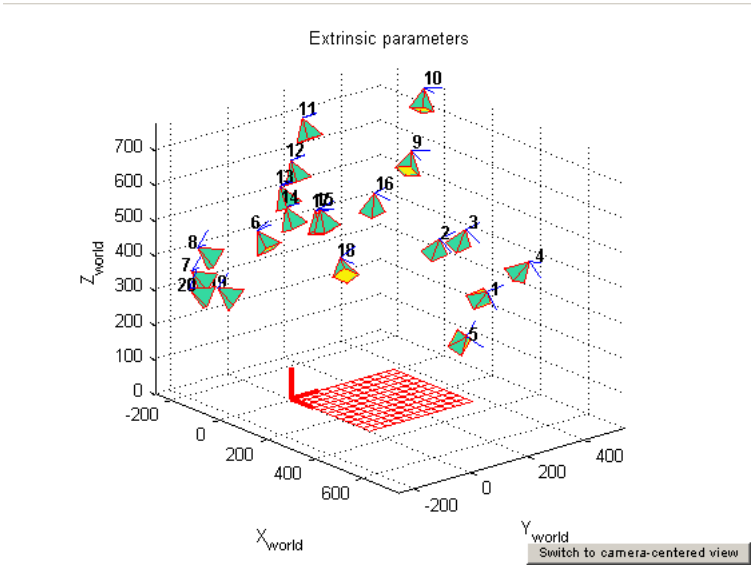


Figura 6.15: Stima parametri estrinseci

caso la stima del vertice si discosti di molto dalla sua posizione reale, in questo caso sarà possibile ricalcolarlo in modo corretto(Fig. 6.18). Tramite il comando "Undistort image" è possibile eliminare l'effetto della distorsione su una o piu immagini (Fig. 6.19) , proprio la funzione che più spesso verrà utilizzata nel programma per l'inseguimento del robot. Inoltre tramite i comandi "Save" e "Export data" è possibile creare con il primo comando un file .mat nel quale vengono salvati tutti i parametri precedentemente calcolati, ciò permette di poterlo caricare in un secondo momento, tramite il comando "Load" e dunque evita di dover ripetere ogni volta l'operazione di calibrazione. Il secondo comando invece permette di esportare i parametri della calibrazione in due diversi formati comunemente accetati da numerosi programmi di ritocco fotografico e di telemetria.

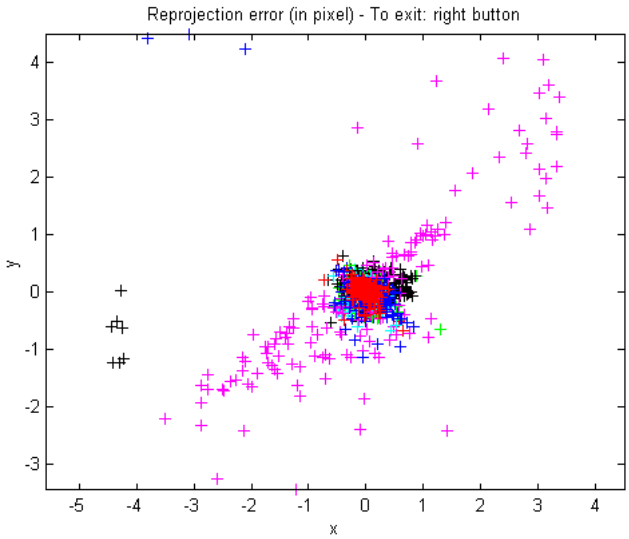


Figura 6.16: Distribuzione dell'errore di stima nella posizione dei vertici

```
Selected image: 18
Selected point index: 145
Pattern coordinates (in units of (dX,dY)): (X,Y)=(0,0)
Image coordinates (in pixel): (45.63,23.27)
Pixel error = (0.58234,0.39466)
Window size: (wintx,winty) = (5,5)
```

Figura 6.17: Dati dell'errore di stima del pixel selezionato

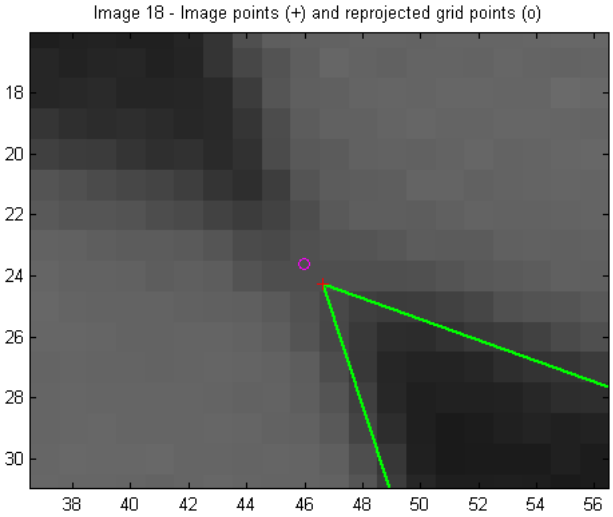


Figura 6.18: Esempio di errore nella stima della posizione del vertice

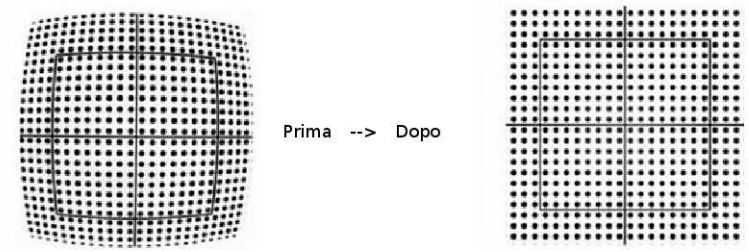


Figura 6.19: Correzione della distorsione radiale

Capitolo 7

Appendice B
Guida pratica all'utilizzo del sistema

7.1 Introduzione

Tale seconda appendice vuole essere una guida all'utilizzo del software realizzato. Verrà prima descritta la fase di installazione e configurazione del programma usato per la cattura dei fotogrammi, poi l'uso vero e proprio del programma sotto MatLab.

7.2 Configurazione ed utilizzo di CaptureMax

CaptureMax 2.55 è il software esterno usato per catturare i fotogrammi dalla webcam, una volta installato tale programma è possibile passare alla configurazione delle impostazioni principali.

Effettuando un click sull'icona "*options*" (Fig. 7.1) A, si aprirà un menu contenente tutte le impostazioni del programma.

- Selezioniamo "*Capture*" (Fig. 7.1 B) modificando il valore di "Image Capture" a 0 minuti 1 secondo, in tale modo il programma andrà a catturare un fotogramma al secondo, dopodiché impostiamo la directory in cui andare a salvare le immagini catturate[1]. La cartella in cui

[1]Per ragioni di praticità si è scelto di sovrascrivere l'immagine ad ogni cattura, dunque l'ultima immagine catturata si chiamerà sempre "*last.jpg*", tale scelta ha comportato l'inserimento nel programma di alcune righe di codice per il controllo e la sincronizzazione di lettura e scrittura su file, infatti poteva accadere che l'immagine veniva aperta da MatLab mentre CaptureMax la stava ancora finendo di scrivere, dunque l'immagine risultava incompleta ed il programma terminava in modo non corretto.

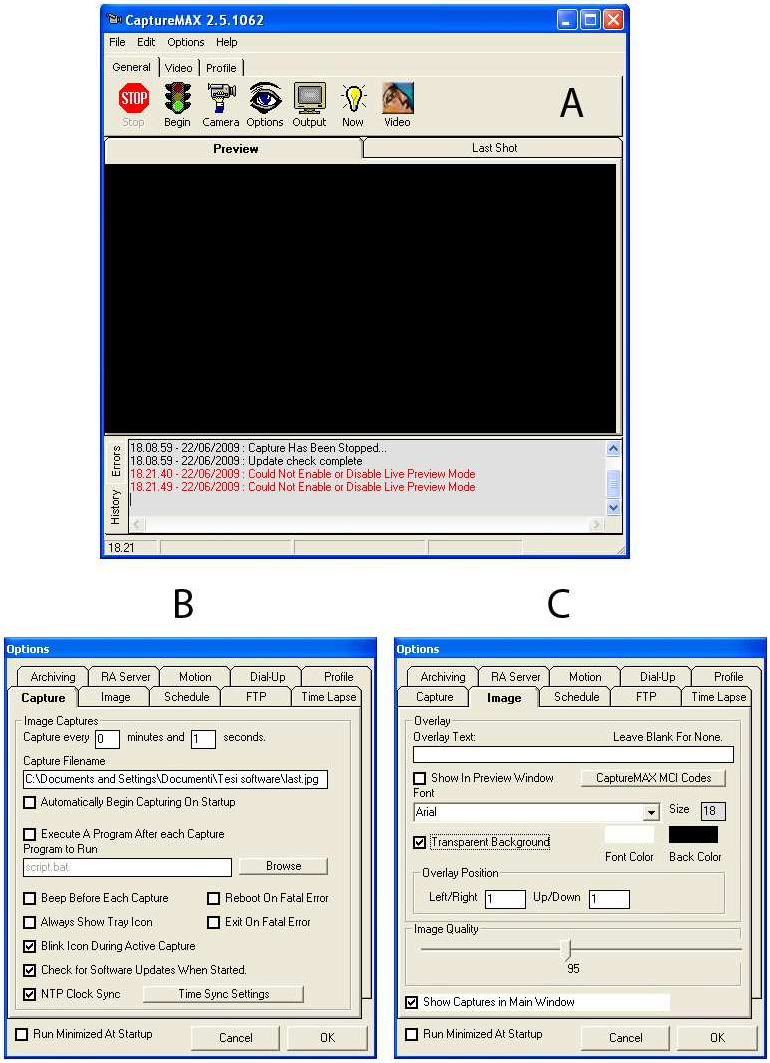


Figura 7.1: Configurazione delle impostazioni di CaptureMax

dover andare a salvare l'immagine dovrà essere la stessa in cui abbiamo i file MatLab con il codice sorgente del software di calibrazione e inseguimento e l'immagine dovrà essere chiamata "*last.jpg*".

- Selezionare "*Image*" (Fig. 7.1 C) e lasciare vuota l' "*Overlay Text*", in tale modo si evita che sull'immagine vengano stampati data ed ora di cattura.

- Selezionare "Ok" per uscire dal menù salvando le modifiche apportate.

Ora è necessario impostare alcuni valori realativi al formato e alla webcam. Apriamo l'opzione "*output*" (Fig. 7.2 A) ed impostiamo il formato dell'immagine a RGB e la dimensione a 640x480. Selezionando ora l'opzione "*camera*" (Fig. 7.2 B) si accederà alle opzioni della videocamera gestite dai driver Philips. Impostiamo il valore di FPS al valore desiderato[2], l'importante è che tale valore sia superiore a 15, infatti al di sotto di tale valore alcune impostazioni non sono attive. L'opzione "*controllo completamente automatico*" è bene che sia disabilitata, in quanto nel caso avvengano variazioni di illuminazione nell'ambiente di lavoro il programma andrebbe a modificare i valori di luminosità e contrasto dell'immagine e dunque il software non sarebbe più in grado di sfondare l'immagine.

Infine prima di iniziare la cattura vera e propria selezioniamo dal menù la voce "$funzioni$" (Fig. 7.2 C), spuntiamo l'opzione "$modifica\ sfondo$", selezioniamo "$sfoglia$" e carichiamo in memoria l'immagine di sfondo nera di dimensioni 640x480 precedentemente creata[3], dopodiché è possibile selezionare "$Acquisisci\ foto$" facendo ben attenzione al fatto che nell'area di lavoro non debbano esserci né il robot né tanto meno il calibratore. È possibile inoltre una volta catturata l'immagine, regolare l'intensita del filtro, generalmente è bene impostarlo intorno al 50-75%, poi premere "*ok*" per uscire e salvare le impostazioni.

Una volta effettuate tali operazioni premere "*start*" dall'interfaccia principale di CaptureMax per iniziare a catturare i fotogrammi.

7.3 Manuale di Calibrazione e Tracking

Una volta effettuata la procedura di configurazione ed aver lanciato la cattura dei fotogrammi tramite CaptureMax il sistema è attivo e pronto ad iniziare le procedure di calibrazione assi e monitaraggio. Il programma realizzato all'interno MatLab risulta essere molto semplice ed intuitivo. Per lanciare l'interfaccia grafica basta selezionare il file "$interfaccia.m$" e lanciare l'interprete MatLab con il

[2] Attenzione, un valore troppo elevato, come 60, potrebbe comportare il rallentamento del sistema nel caso il computer non abbia requisiti sufficienti.

[3] Tale immagine è già presente nella cartella del codice sorgente.

tasto $F5$. Si aprirà una finestra grafica con un menù (Fig. 7.3) caratterizzata da 8 tasti, la quale evita la scrittura di fastidiosi comandi da shell. Il primo tasto "*Calibrazione Webcam*" avvia la procedura di calibrazione degli assi, dunque dopo aver inserito, all'interno dell'area di lavoro, il calibratore nella posizione desiderata, è possibile lanciare tale comando, che usando l'algoritmo descritto nel Capitolo 2, individua gli assi di lavoro. Una volta terminata in modo corretto[4] tale procedura è possibile visualizzare i dati calcolati nell'operazone di calibrazione avviando il secondo comando del menù grafico "*Visualizza Dati Calibrazione*". I valori calcolati sarano mostrati in un apposita tabella (Fig. 7.4), in cui il valore che mostra la percentuale di errore tra la diagonale reale e quella ideale risulta essere molto importante per la buona riuscita delle misurazioni, tale valore deve essere di norma inferiore all'1%. Il terzo tasto "*Tracking del Robot*" avvia la procedura di individuazione e tracciamento della posizione del robot all'interno dello spazio di lavoro, subito dopo aver premuto tale tasto bisognerà inserire all'interno della shell di MatLab per quanti minuti si desidera tracciare il robot e premere invio per iniziare il monitoraggio. Al termine di tale operazione sarà possibile visualizzare il percorso effettuato dal robot rispetto ai riferimenti del calibratore attraverso il comando "*Analisi Traiettoria*", tale percorso sarà tanto più accurato, quanto più frequenti saranno le misurazioni eseguite. Invece, lanciando il comando "*Visualizza Dati Tracking*" verrà visualizzata una finestra (Fig. 7.5) contenente tutti i valori calcolati durante l'intero periodo di monitoraggio del robot. Il pulsante "*Esporta Dati Tracking*" esporta i dati calcolati in un file .rtf esterno (Fig. 7.6), in tale modo è possibile leggerli senza alcun problema anche da un qualsiasi altro linguaggio di programmazione quale C, C++, Java.

Infine il comando"*Reset Manuale*" resetta manualmente tutte le variabili di sistema del programma, così facendo si ha la certezza di cancellare tutti i dati precedentemente calcolati. Per terminare la finestra grafica è possibile alternativamente premere il tasto "*Exit*" oppure il comando chiudi in alto a destra.

[4]Nel caso in cui non si presenti alcun messaggio di errore, la procedura è avvenuta in modo corretto, altrimenti bisognerà ripeterla.

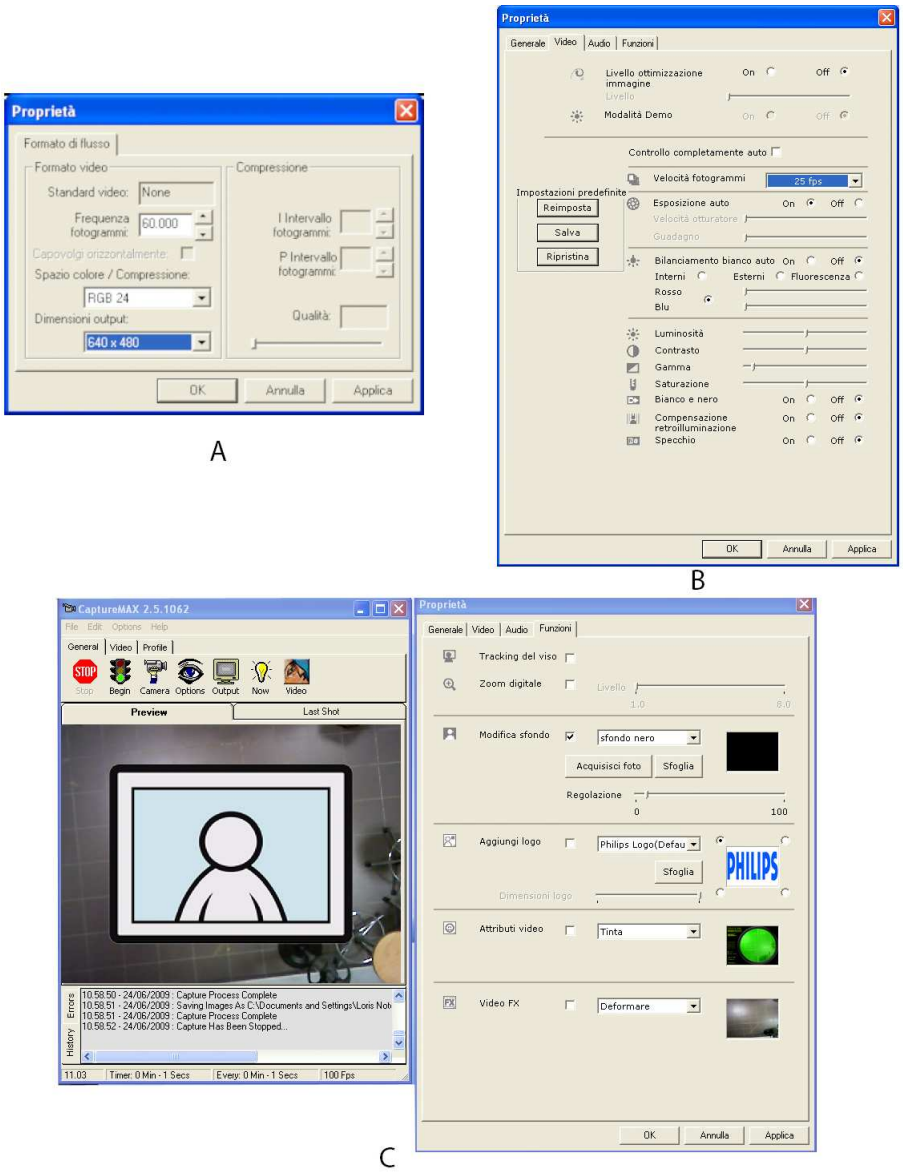


Figura 7.2: Impostazioni pre-cattura

Figura 7.3: Interfaccia utente

Coordinate X Rosso	Coordinate X Verde	Cordinate X Blue	Orientamento Assi rispetto al bordo dell'immagine	Stima dell'altezza della Webcam da terra in cm
Coordinate Y Rosso	Coordinate Y Verde	Coordinate Y Blue	Scarto percentuale tra la misura della diagonale ideale e quella reale	Errore medio in pixel dovuto alla distorsione prospettica

Figura 7.4: Tabella calibrazione

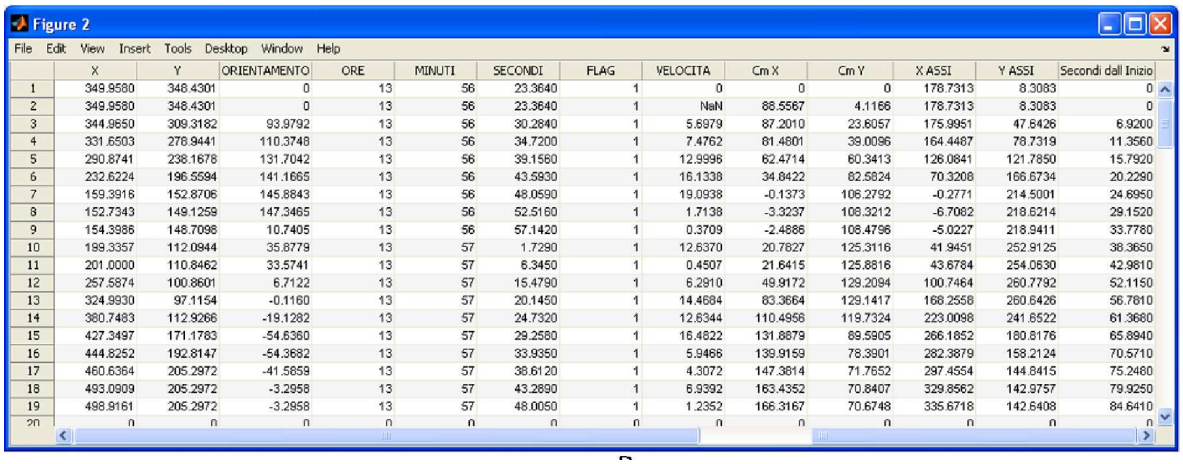
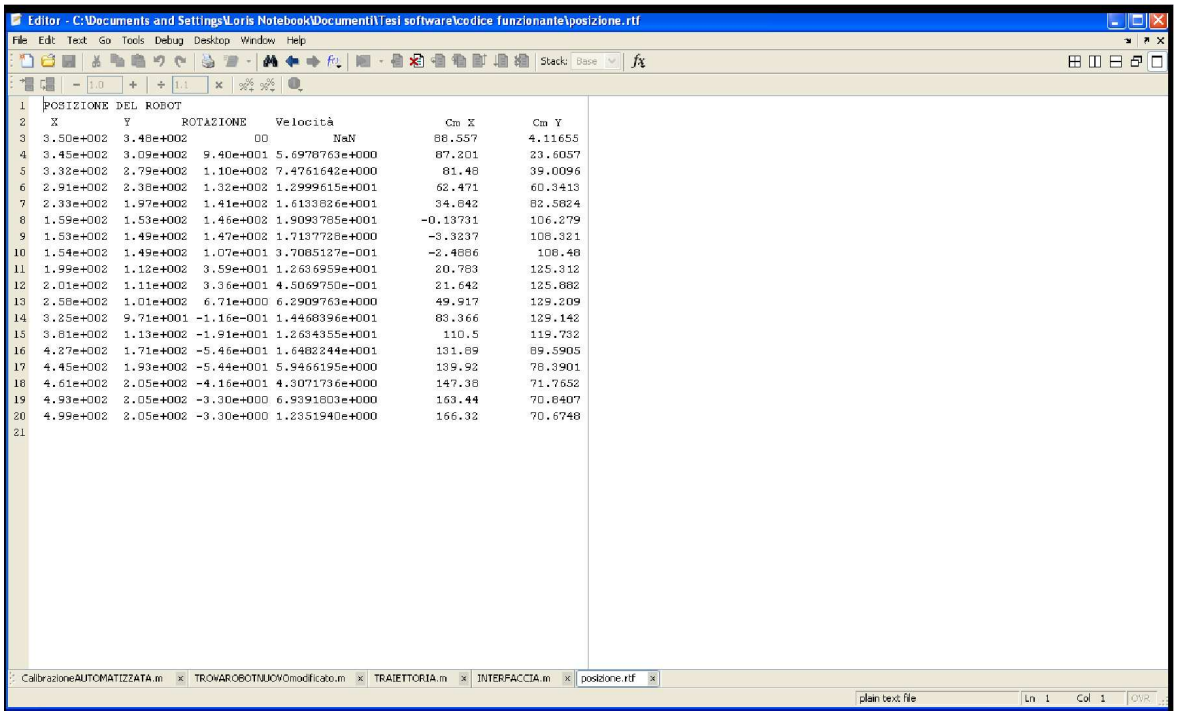


Figure 2

File Edit View Insert Tools Desktop Window Help

	X	Y	ORIENTAMENTO	ORE	MINUTI	SECONDI	FLAG	VELOCITA	Cm X	Cm Y	X ASSI	Y ASSI	Secondi dall Inizio
1	349.9580	348.4301	0	13	56	23.3640	1	0	0	0	178.7313	8.3083	0
2	349.9580	348.4301	0	13	56	23.3640	1	NaN	88.5567	4.1166	178.7313	8.3083	0
3	344.9650	309.3182	93.9792	13	56	30.2840	1	5.6979	87.2010	23.6057	175.9951	47.6426	6.9200
4	331.6503	278.9441	110.3748	13	56	34.7200	1	7.4762	81.4801	39.0086	164.4487	78.7319	11.3560
5	290.8741	238.1678	131.7042	13	56	39.1560	1	12.9998	62.4714	60.3413	126.0841	121.7850	15.7920
6	232.6224	196.5594	141.1665	13	56	43.5930	1	16.1338	34.8422	82.5824	70.3208	166.6734	20.2290
7	159.3916	152.8706	145.8843	13	56	48.0590	1	19.0938	-0.1373	106.2792	-0.2771	214.5001	24.6950
8	152.7343	149.1259	147.3465	13	56	52.5160	1	1.7138	-3.3237	108.3212	-6.7082	218.6214	29.1520
9	154.3986	148.7098	10.7405	13	56	57.1420	1	0.3709	-2.4886	108.4796	-5.0227	218.9411	33.7780
10	199.3357	112.0944	35.8779	13	57	1.7290	1	12.6370	20.7827	125.3116	41.9451	252.9125	38.3650
11	201.0000	110.8462	33.5741	13	57	6.3450	1	0.4507	21.6415	125.8816	43.6784	254.0630	42.9810
12	257.5874	100.8601	6.7122	13	57	15.4790	1	6.2910	49.9172	129.2094	100.7464	260.7792	52.1150
13	324.9930	97.1154	-0.1160	13	57	20.1450	1	14.4684	83.3664	129.1417	168.2558	260.6426	56.7810
14	380.7483	112.9266	-19.1282	13	57	24.7320	1	12.8344	110.4958	119.7324	223.0098	241.6522	61.3680
15	427.3497	171.1783	-54.6360	13	57	29.2580	1	16.4822	131.8679	89.5905	266.1852	180.8176	65.8940
16	444.8252	192.8147	-54.3682	13	57	33.9350	1	5.9466	139.9159	78.3901	282.3879	158.2124	70.5710
17	460.6364	205.2972	-41.5659	13	57	38.6120	1	4.3072	147.3814	71.7652	297.4554	144.8415	75.2480
18	493.0909	205.2972	-3.2958	13	57	43.2890	1	6.9392	163.4352	70.8407	329.8562	142.9757	79.9250
19	498.9161	205.2972	-3.2958	13	57	48.0050	1	1.2352	166.3167	70.6748	335.6718	142.6408	84.6410
20	0	0	0	0	0	0	0	0	0	0	0	0	0

Figura 7.5: Tabella dati Tracking



	X	Y	ROTAZIONE	Velocità	Cm X	Cm Y
1	POSIZIONE DEL ROBOT					
2	X	Y	ROTAZIONE	Velocità	Cm X	Cm Y
3	3.50e+002	3.48e+002	00	NaN	88.557	4.11655
4	3.45e+002	3.09e+002	9.40e+001	5.6978763e+000	87.201	23.6057
5	3.32e+002	2.79e+002	1.10e+002	7.4761642e+000	81.48	39.0096
6	2.91e+002	2.38e+002	1.32e+002	1.2999615e+001	62.471	60.3413
7	2.33e+002	1.97e+002	1.41e+002	1.6133826e+001	34.842	82.5824
8	1.59e+002	1.53e+002	1.46e+002	1.9093785e+001	-0.13731	106.279
9	1.53e+002	1.49e+002	1.47e+002	1.7137728e+000	-3.3237	108.321
10	1.54e+002	1.49e+002	1.07e+001	3.7085127e-001	-2.4886	108.48
11	1.99e+002	1.12e+002	3.59e+001	1.2636959e+001	20.783	125.312
12	2.01e+002	1.11e+002	3.36e+001	4.5069750e-001	21.642	125.882
13	2.58e+002	1.01e+002	6.71e+000	6.2909763e+000	49.917	129.209
14	3.25e+002	9.71e+001	-1.16e-001	1.4468396e+001	83.366	129.142
15	3.81e+002	1.13e+002	-1.91e+001	1.2634355e+001	110.5	119.732
16	4.27e+002	1.71e+002	-5.46e+001	1.6482244e+001	131.89	89.5905
17	4.45e+002	1.93e+002	-5.44e+001	5.9466195e+000	139.92	78.3901
18	4.61e+002	2.05e+002	-4.16e+001	4.3071736e+000	147.38	71.7652
19	4.93e+002	2.05e+002	-3.30e+000	6.9391803e+000	163.44	70.8407
20	4.99e+002	2.05e+002	-3.30e+000	1.2351940e+000	166.32	70.6748
21						

Figura 7.6: Dati esportati nel file di testo

Bibliografia

[1] Abdel-Aziz,Y.I. Karara *"Direct linear trasformation into onject space coordinates in close-range photogrammetry"*, Urbana, Illios-1998.

[2] Melen T., *"Geometrical modelling and calibration of video camera"*, Optical Engineenring-1993.

[3] Heikkila,J. Silven, *"Calibration procedure for short focal lenght off-the-shelf CCD cameras"*, International Conference on Pattern Recognition-1996.

[4] A. Von Koenismarck, *"Cinema 4d Guida Avanzata"*, Pearson Education-2003.

[5] Rafael C. Gonzalez,Rchard E. Woods,Stefen L. Eddins *"Digital Image Processing using MatLab"*, Pearson, Prentice Hall, 2004.

[6] User's Guide , *"Video and Image Processing Blockset 2"*, The Math Work, March 2007 release.

[7] P.Foggia,A.Limongiello,M.Vento, *"Conference on Automation, Robotics and Autonomous Systems"*, Cairo, Egypt, 19-21 December, 2005.

[8] Berthold K. P. Horn, *"Robot Vision"*, Department of Electrical Engineering and Computer Science, MIT Rapporto interno, 2006.

Siti visitati:

www.wikipedia.com

www.swarm-robotics.com

www.adinf.unisa.it