UNIVERSITÀ POLITECNICA DELLE MARCHE
FACOLTÀ DI INGEGNERIA

Corso di Laurea Magistrale in Ingegneria Meccanica

Applicazione di robotica collaborativa ed intelligenza artificiale al gioco degli scacchi

Application of collaborative robotics and artificial intelligence to the game of chess

Relatore:
Luca Carbonari

Tesi di Laurea di:
Simone Salucci

Correlatore:
Massimo Callegari

A.A. 2021/2022

UNIVERSITÀ POLITECNICA DELLE MARCHE
FACOLTÀ DI INGEGNERIA
CORSO DI LAUREA MAGISTRALE IN INGEGNERIA MECCANICA
Via Brecce Bianche – 60131 Ancona (AN), Italy

INDICE

INTRODUZIONE 4

CAPITOLO 1: LA ROBOTICA COLLABORATIVA 6

CAPITOLO 2: LA VISIONE ARTIFICIALE 10

CAPITOLO 3: ARCHITETTURA GENERALE DEL DIMOSTRATORE 15

CAPITOLO 4: IL ROBOT COLLABORATIVO YUMI 17
- 4.1: CARATTERISTICHE FISICHE 17
- 4.2 MECCANICA E PRESTAZIONI 19
- 4.3 ROBOTSTUDIO 21

CAPITOLO 5: PROGETTAZIONE DEL SISTEMA 22
- 5.1 LA SCACCHIERA 22
- 5.2 IL SISTEMA DI VISIONE 26
- 5.3 L'INTELLIGENZA ARTIFICIALE DI GIOCO 38
- 5.4 LA PROGRAMMAZIONE DEL ROBOT 40
- 5.5 LA PRESA DEI PEZZI 46

CONCLUSIONI 48

RIFERIMENTI 49

INTRODUZIONE

Il lavoro descritto in questa tesi è stato svolto come tirocinio curricolare presso il Dipartimento di Ingegneria Industriale e Scienze Matematiche dell'Università Politecnica delle Marche.

L'obiettivo è stato quello di implementare una stazione di gioco basata su tecnologie come la robotica collaborativa, l' analisi di immagini per la visione artificiale e la programmazione.

In particolare è stata sviluppata una applicazione che permette lo svolgimento di una partita di scacchi tra un giocatore umano e il robot collaborativo YuMi di ABB.

Le tecnologie utilizzate sono alcune tra quelle che ho approfondito durante il corso di laurea magistrale, questa tesi quindi rispecchia pienamente gli argomenti trattati durante il percorso di studi e li mette in pratica.

Saranno descritte tutte le componenti del gioco e i vari problemi affrontati nello sviluppo fino ad arrivare all'applicazione definitiva.

Per raggiungere questo risultato sono stati utilizzati diversi protocolli e linguaggi nella programmazione dei componenti software necessari:

- Il sistema di visione necessario per visualizzare le mosse effettuate e l'intelligenza di gioco sono scritti in linguaggio Python tramite l'editor Visual Studio Code;
- Il collegamento con il robot avviene tramite il protocollo di comunicazione TCP/IP;
- I movimenti del robot sono programmati sull'ambiente simulativo Robotstudio tramite il linguaggio Rapid.

Per la parte fisica del dimostratore invece è stato necessario disegnare e stampare in 3D alcuni elementi come un supporto per la webcam e delle dita sostitutive per le pinze robot.

- Sono stati utilizzati i software Solidedge e NX.

Il lavoro è suddiviso in cinque capitoli e presenta inizialmente una introduzione generale delle tecnologie coinvolte nel progetto:

- il primo capitolo descrive lo sviluppo della robotica collaborativa evidenziando le caratteristiche dei cobot e le loro differenze rispetto alle macchine automatiche tradizionali;
- nel secondo capitolo viene affrontato il tema della visione artificiale, sono presentate le problematiche relative all'illuminazione, la rappresentazione digitale delle immagini e la loro analisi;

Successivamente sono trattati i problemi pratici affrontati nello sviluppo del programma:

- nel terzo capitolo viene spiegato il funzionamento generale del software che per permette lo svolgimento del gioco;
- il quarto capitolo descrive tutte le caratteristiche e le potenzialità del robot ABB YuMi, utilizzato come dimostratore in questa tesi;
- infine il quinto capitolo tratta nello specifico le singole componenti del sistema, soffermandosi anche sulle problematiche incontrate e le soluzioni alternative necessarie per superarle.

Le attività pratiche svolte durante il tirocinio sono state svolte presso il laboratorio MIR (Mechatronics and Industrial Robots) [1] dell' Università Politecnica delle Marche, ad Ancona.

CAPITOLO 1: LA ROBOTICA COLLABORATIVA

I primi robot sono stati sviluppati per il lavoro in fabbrica, dove sono impiegati per eseguire operazioni altamente ripetitive, per lavorare in ambienti ostili o per manipolare carichi eccessivi per un operatore umano. Il loro impiego è da sempre stato finalizzato al settore industriale, soprattutto nel settore automobilistico.

A causa della pericolosità di queste macchine in movimento, l'accesso nella loro area di lavoro deve necessariamente essere interdetto durante l'esecuzione dei compiti.

I robot tradizionali sono quindi in grado di sostituire completamente il lavoro umano e sono studiati proprio per questo motivo. Nelle linee produttive automatizzate, le celle di lavoro sono costruite attorno a uno o più robot e sono presenti delle barriere fisiche di accesso che impediscono agli operatori di trovarsi in condizioni di pericolo. Le protezioni sono molto ingombranti, richiedono un notevole spazio a disposizione e hanno anche costi elevati.

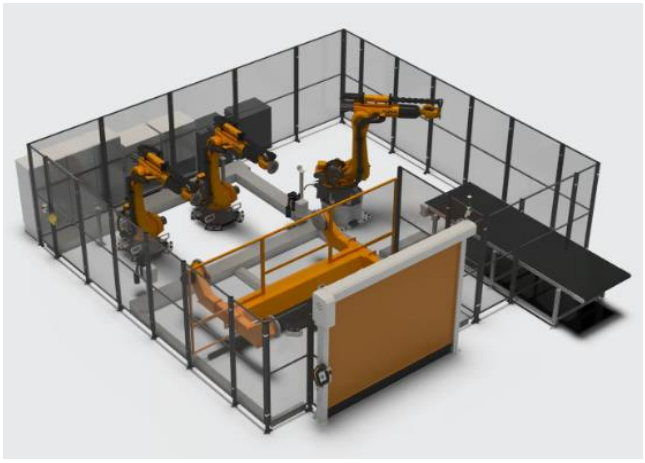

Figura 1.1: i robot industriali lavorano all'interno di uno spazio di lavoro sicuro

Le prestazioni dei robot sono notevoli, infatti garantiscono alta ripetitività e precisione nei compiti con un ridotto tempo ciclo, caratteristiche ideali per i processi produttivi automatizzati.

È possibile trovare macchine automatiche anche in ambienti tradizionali, la robotica "di oggi" si è evoluta ed è presente anche fuori dai reparti di produzione. Basta pensare ai diffusissimi robot automatici per le pulizie domestiche.

Allo stesso tempo anche la robotica tradizionale si è sviluppata tramite l'integrazione di nuovi sensori ma soprattutto grazie all'utilizzo di strumenti di simulazione, programmazione off-line e la possibilità di lavorare in spazi condivisi con grande flessibilità nello svolgimento dei compiti.

A partire dal 2010 si inizia a parlare del paradigma di industria 4.0: il termine indica una tendenza all'automazione industriale che integra nuove tecnologie per migliorare le condizioni di lavoro e aumentare la produttività [2]. Industria 4.0 trova la sua espressione nella smart factory.

Viene focalizzata l'attenzione sulla cooperazione tra uomo e macchina, ma questa volta con lo scopo di migliorare il lavoro umano e non sostituirlo. La nuova tecnologia è a servizio dell'uomo cioè è in grado di collaborare con esso.

Figura 1.2:

uno dei pilastri fondamentali su cui si basa Industria 4.0 è la robotica collaborativa [2].

Questa è necessaria per far fronte alle principali richieste del mercato attuale: brevi tempi di consegna e personalizzazione di massa. Le macchine garantiscono forza e precisione, l'umano pensa e agisce in modo dinamico: unendoli si ottiene un sistema produttivo altamente flessibile

Già descritti a partire dal 1996 da J. Edward Colgate e Micheal Pashkin [3], ma di recente diffusione sul mercato, i robot collaborativi, o cobot, sono macchine di nuova generazione, leggere e flessibili che lavorano in sicurezza insieme all'uomo nelle smart factories, senza barriere o gabbie protettive di divisione.

Permettono un miglioramento delle condizioni di lavoro e un aumento della produttività [4]: il robot facilita le operazioni di montaggio scomode, garantisce il movimento di utensili pesanti o

pericolosi e soprattutto la collaborazione porta al superamento della dicotomia tra lavorazioni manuali e automatizzate.

Per poter lavorare senza protezioni fisiche, i cobot sono limitati in velocità e forza, possono movimentare carichi relativamente bassi con precisioni non altissime. Sono quindi dotati di sistemi di sicurezza aggiuntivi [5] rispetto ai robot tradizionali che li rendono più costosi.

La sicurezza dell' operatore è garantita tramite protezioni software realizzate con sensori, rivestimenti in 'pelle' capacitiva, sistemi di visione e di prossimità, ma anche da accorgimenti sulla struttura fisica: le macchine non hanno cavi o componenti esposti e sono rivestite con materiali morbidi al fine di evitare incidenti.

I cobot sono semplici da installare e, grazie all'assenza di gabbie protettive, consentono un sfruttamento più efficiente dello spazio di lavoro.

La programmazione è semplice e non richiede personale altamente qualificato.

Infine l'utilizzo di questi sistemi, anche se costosi, aumenta la produttività dell'impianto.

Per il prossimo decennio si stima infatti una enorme crescita nel mercato di queste macchine [6].

Figura 1.3: i robot collaborativi operano a stretto contatto con l'uomo

Come per tutti i sistemi industriali, le caratteristiche dei cobot sono descritte dalle normative: la Direttiva Macchine definisce i requisiti essenziali di sicurezza e salute pubblica ai quali devono rispondere prima della loro immissione sul mercato [7]. Inoltre la specifica tecnica ISO/TS 15066 del 2016: "Robots and robotic devices: collaborative robots" [8] tratta esclusivamente i requisiti di sicurezza aggiuntivi per robot collaborativi e definisce:

- l'arresto monitorato (SMS): interruzione dei movimenti del robot se un operatore entra nell'area collaborativa durante lo svolgimento di un compito autonomo. Questo tipo di

arresto non avviene in emergenza quindi non è richiesto un riavvio del robot: il movimento riparte in autonomia quando l'operatore si allontana;

- guida manuale (HG): l'operatore può azionare il meccanismo muovendolo a mano;
- monitoraggio della velocità e della separazione (SSM): robot e operatore possono muoversi contemporaneamente nello spazio di lavoro collaborativo. Il rischio viene ridotto garantendo sempre una distanza sufficiente tra i due che, se viene superata, causa l'arresto della macchina. Quando i movimenti sono più rapidi, la distanza di sicurezza aumenta.
- Limitazione della potenza e della forza (PFL): il cobot non può mai superare dei valori limite calcolati tramite una valutazione dei rischi. Deve inoltre essere in grado di dissipare le forze in caso di impatto.

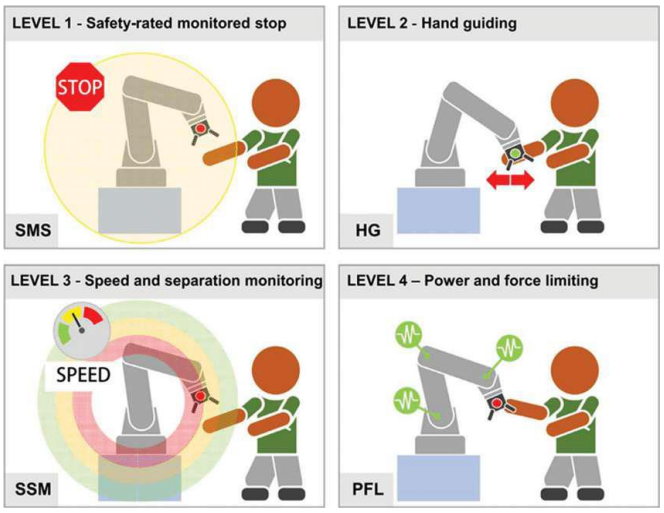

Figura 1.4: classi di requisiti di sicurezza definiti dalla normativa per i robot collaborativi

Tra i robot collaborativi più venduti [7] troviamo il Fanuc CRX, UR5, e ABB Yumi, che è stato utilizzato in questa tesi. Le sue caratteristiche saranno descritte in seguito.

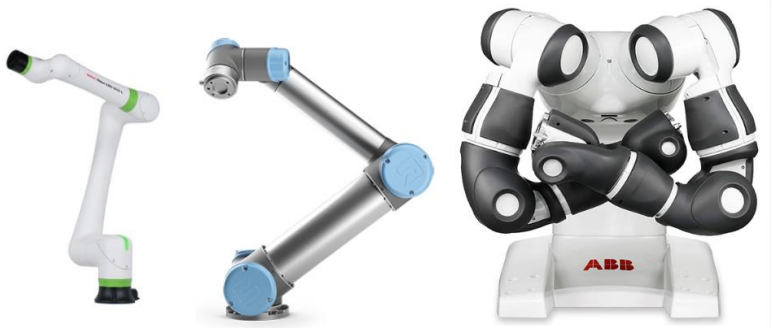

Figura 1.5: Esempi di robot collaborativi

CAPITOLO 2: LA VISIONE ARTIFICIALE

La visione artificiale o computer vision è un campo dell'intelligenza artificiale che consente ad un sistema di elaborare immagini o video e da questi estrarre informazioni, riconoscere oggetti e intraprendere azioni a partire dai segnali ricavati [9].

L'obiettivo è quello di imitare la nostra vista ma, a differenza di un essere umano, un sistema elettronico non dispone di tutta l'esperienza maturata nel corso della vita nel riconoscimento di immagini. Uno stesso oggetto infatti può essere visualizzato da diverse angolazioni, può essere in movimento o fermo e il suo colore può essere molto variabile. Questi parametri sono per noi semplici da generalizzare grazie alla nostra esperienza.

La computer vision cerca di replicare questa capacità utilizzando telecamere, dati e algoritmi.

Il sistema deve essere addestrato in un tempo molto breve quindi sono necessari tantissimi dati che il software deve 'studiare', cioè una serie di immagini da analizzare nelle quali l'oggetto si trova in configurazioni molto varie tra loro.

Attraverso le caratteristiche che rileva in questi esempi il sistema si allena e migliora nel riconoscimento, risultato possibile tramite deep learning e reti neurali convoluzionali [10].

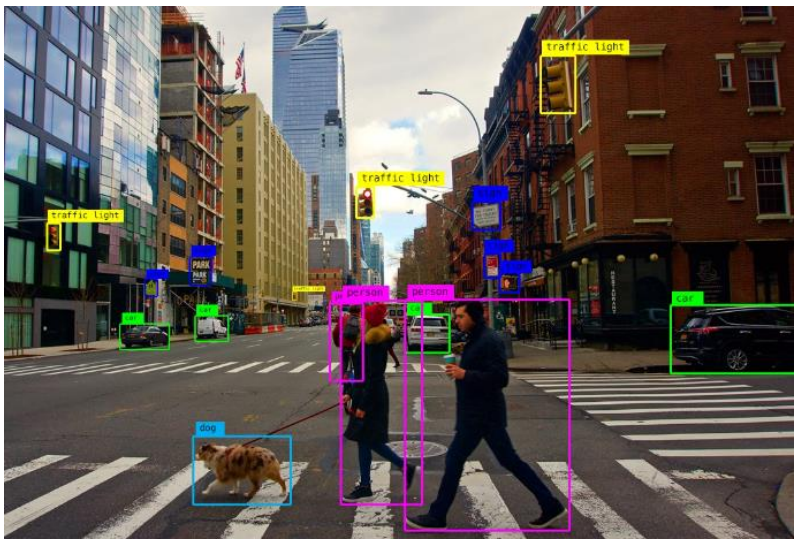

Figura 2.1: Classificazione di oggetti.

tramite intelligenza artificiale e machine learning è possibile insegnare a una macchina a riconoscere e catalogare oggetti. Spesso si utilizza un dataset di training iniziale per portare il sistema al libello di apprendimento simile a quello umano. In questo caso la tecnica viene detta 'supervisionata'.

Figura 2.2: La visione artificiale è presente in molte applicazioni concrete: nella produzione industriale viene utilizzata per verificare i pezzi in lavorazione, riconoscere i difetti o verificare il loro orientamento;
nei veicoli a guida autonoma permette di riconoscere auto, ostacoli e segnali stradali nello spazio in cui si muovono.

È possibile programmare diverse tipologie di compiti:

1) Object detection: verificare la presenza di un marker o di un oggetto all'interno dei dati analizzati;
2) classificazione di immagini: ricercare oggetti o forme note all'interno dell'immagine e classificarle nei loro gruppi di appartenenza;
3) object tracking: seguire la traiettoria di un oggetto dopo averlo rilevato;
4) content-based image retrieval: cercare e recuperare immagini da grandi archivi di dati basandosi sul contenuto delle immagini stesse o da una loro regione di interesse (ROI).

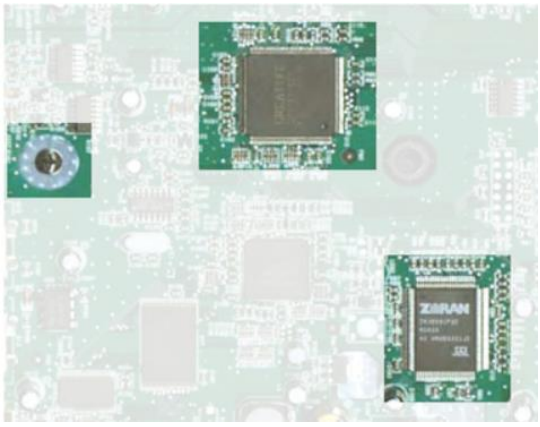

Figura 2.3: esempio di analisi industriale tramite immagini.
Una ROI si ottiene estraendo dalla foto una o più sottomatrici che contengono le informazioni desiderate.
Sono escluse le parti poco significative dei dati.

Le componenti fondamentali di un sistema di visione sono [11]:

1) la luce: l'illuminazione della scena da osservare è il parametro fondamentale per il funzionamento del sistema. Un ottimo software di analisi di immagini non riesce ad ottenere risultati soddisfacenti se la luce utilizzata non è di buona qualità.

   I parametri fisici della luce dipendono dal tipo di sorgente utilizzata, alcuni esempi sono le lampade alogene, i led o i laser. Un altro parametro fondamentale è la direzione dell'illuminazione: diverse geometrie mettono in evidenza dettagli particolari dell'oggetto.

   I possibili effetti dovuti ad una errata illuminazione possono essere [12]:

   - riflessi
   - ombreggiamento
   - basso contrasto
   - blooming
   - basso rapporto segnale/rumore

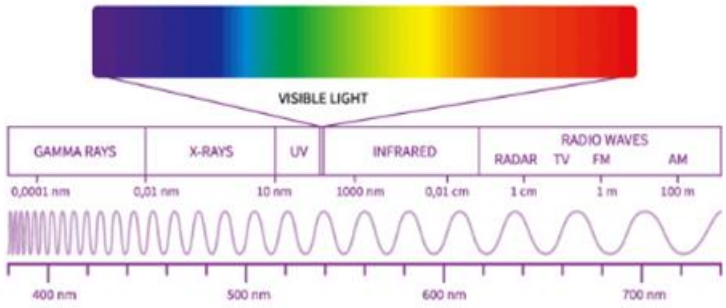

Figura 2.4: spettro elettromagnetico della luce.

Le diverse sorgenti presenti sul mercato hanno proprietà diverse in termini di coerenza, spettro cromatico e temporale, polarizzazione. Si parla di illuminazione anche in sistemi di misura tramite immagini che sfruttano luce non visibile come l'infrarosso, l'UV e i raggi X.

2) l mezzo di propagazione: eventuali interfacce tra mezzi di propagazione differenti causano una variazione dell'indice di rifrazione. La traiettoria della luce presenta un punto angoloso a causa delle diverse velocità di propagazione. Il fenomeno varia in base alla lunghezza d'onda della radiazione. Inoltre l'intensità dell'illuminazione può essere alterata.

3) il target da osservare: deve presentare una superficie cooperativa, cioè con caratteristiche tali da riuscire ad ottenere una buona osservazione di essa in relazione alla sorgente

luminosa disponibile. In base alle necessità, una superficie cooperativa può essere di tipo diffusivo, assorbente, retroriflettente o totalmente riflettente.

4) l'obiettivo: al suo interno si trovano una serie di lenti che mettono a fuoco la luce in un punto ben preciso e contrastano vari tipi di aberrazioni e distorsioni.
   L'obiettivo gestisce la quantità di luce disponibile e determina anche la risoluzione spaziale dell'immagine.

Figura 2.5: F Number. Il rapporto tra lunghezza focale e diametro della lente è una misura della luce che può passare ed arrivare al sensore, può essere modificata tramite l'apertura o chiusura del diaframma.

5) il sensore: la luce catturata dall'obiettivo viene messa a fuoco su una matrice di sensori quando l'otturatore viene aperto.
   I sensori trasformano l'intensità luminosa in un segnale elettrico, poi rappresentato da una certo numero di bit. I singoli elementi fotosensibili sono i pixel.
   Modificando i parametri del sensore si hanno altri gradi di libertà per la gestione della luce.

Figura 2.6: Tempo di esposizione. Modifica l'intervallo tempo di apertura dell'otturatore: È necessario regolarlo in modo diverso a seconda della quantità di luce presente e del movimento della scena per evitare di ottenere immagini mosse.

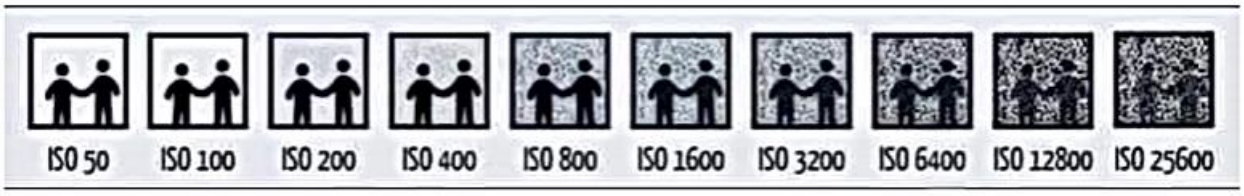

Figura 2.7: Sensibilità ISO. Il sensore può essere più o meno ricettivo alla luce modificando la sua sensibilità: con un valore elevato l'immagine sarà chiara anche in mancanza di luce ma diventa anche più sgranata.

6) l'algoritmo di elaborazione: l'immagine digitale ottenuta è una distribuzione bidimensionale di intensità luminosa [11]. Viene trattata dal pc come una matrice multidimensionale ed è possibile agire sui valori al suo interno come se fosse una normale variabile (un array).

   I filtri esaltano le varie tipologie di frequenze spaziali dell'immagine. Si definisce attorno ad ogni pixel un kernel, cioè una sottomatrice di dimensione scelta. L'intensità del pixel viene variata in base ad una certa combinazione dei valori che si trovano in quelli adiacenti. Si fa 'scorrere' il kernel su tutta la dimensione della matrice di partenza per ottenere l'immagine filtrata. I filtri spaziali esaltano o smussano i bordi degli oggetti oppure esaltano dei dettagli.

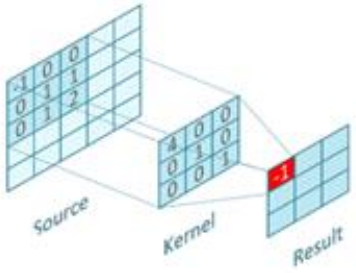

Figura 2.8: variando i pesi moltiplicativi per i valori del kernel sono possibili diverse operazioni di filtraggio

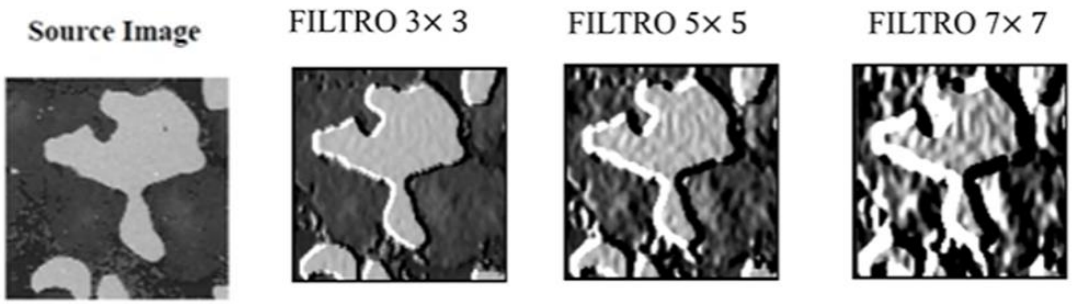

Figura 2.9: Esempio di applicazione del filtro gradiente. L'intensità varia con la dimensione del kernel

CAPITOLO 3: ARCHITETTURA GENERALE DEL DIMOSTRATORE

Come esempio di utilizzo delle tecnologie appena trattate, in questa tesi di laurea è stato richiesto di implementare un dimostratore di robotica collaborativa in grado di permettere lo svolgimento di una partita a scacchi tra un utente umano e il robot collaborativo YuMi.
In particolare si tratta di predisporre un set-up sperimentale che comprende sia le strutture fisiche che software necessarie.

Figura 3.1

Utilizzando le tecnologie descritte è possibile effettuare una partita a scacchi tra un giocatore umano e un robot.

Per raggiungere questo risultato è necessario posizionare le classiche pedine in un supporto completamente raggiungibile dai bracci del robot e comunicare ad esso le mosse effettuate dal giocatore umano, la macchina deve elaborare una risposta ed effettuarla fisicamente sulla scacchiera.
Le componenti fisiche necessarie per il gioco sono solamente una scacchiera con i relativi pezzi e il robot; il software contiene il codice che gestisce la scacchiera elettronica, l'intelligenza di gioco e la movimentazione del robot.

Per comprendere il lavoro è comodo pensare di suddividerlo in piccoli sottoproblemi che sono stati affrontati uno dopo l'altro durante le ore di tirocinio:

1) nella fase iniziale del lavoro è stata progettata una scacchiera elettronica scegliendo tra alcune possibili tecnologie e facendo attenzione soprattutto alle sue dimensioni. Il

parametro fondamentale da tenere in considerazione è il campo di lavoro del robot e la sua estensione è stata valutata attraverso la simulazione.

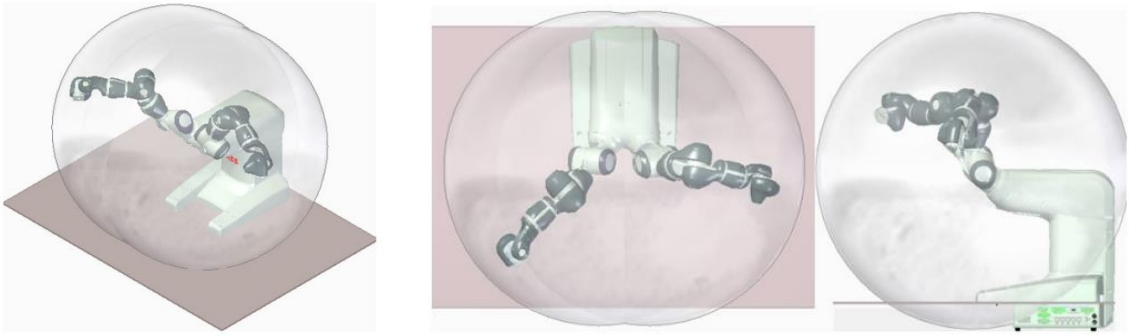

Figura 3.2: rappresentazione grafica dello spazio di lavoro di YuMi [13]

2) L'installazione della scacchiera deve avvenire su di un supporto adeguato per permettere l'interazione del giocatore umano e del robot su di essa;
3) A partire dai dati ottenuti dalla scacchiera elettronica, si deve risale alle mosse effettuate dal giocatore umano. In questa applicazione esso giocherà solo con il colore bianco.
4) La mossa effettuata deve essere salvata in una adeguata struttura dati ed utilizzata come input in una intelligenza artificiale di gioco che sia in grado di generare una mossa di risposta per il giocatore nero;
5) Infine il robot deve muoversi e giocare fisicamente la sua mossa come giocatore nero, la movimentazione deve essere programmata a partire dalla mossa di risposta calcolata dal software. È necessario anche studiare un metodo per effettuare la presa dei pezzi.

CAPITOLO 4: IL ROBOT COLLABORATIVO YUMI

Figura 4.1: ABB IRB 14000 - YUMI

YuMi è un robot articolato presentato alla fiera di Hannover del 2015 dalla casa costruttrice ABB. Definito dalla stessa azienda come il primo vero robot collaborativo al mondo, infatti Il suo nome è l'acronimo di "You and Me".

Si distingue per la presenza di due bracci con un totale di 14 gradi di libertà che garantiscono un aspetto molto simile ad un umano oltre che un ampio raggio d'azione.

YuMi si inserisce perfettamente nel nuovo contesto dell'automazione: il suo scopo non è quello di sostituire l'uomo ma di aiutarlo in operazioni di precisione su piccoli componenti.

4.1: CARATTERISTICHE FISICHE

Osservando YuMi [14] si notano tutte le caratteristiche principali di un robot collaborativo:

I suoi bracci hanno una struttura interna in magnesio, che garantisce la giusta leggerezza e sono ricoperte da un involucro in plastica con una sottile imbottitura morbida per attutire al meglio i possibili urti nello spazio di lavoro. Il rivestimento nasconde tutti i motori, i cavi e i circuiti dell'aria compressa che non sono per niente visibili dall'esterno.

L' assenza di strutture esterne sporgenti è un elemento di sicurezza fondamentale sia per l'operatore che condivide con YuMi lo spazio di lavoro, ma anche per il robot stesso che sarà meno soggetto a danneggiamenti o rotture accidentali.

Figura 4.2

Dettagli di un braccio di YuMi scoperto e, a destra, completamente assemblato

Inoltre il controllore del robot, che ha la funzione di muovere e controllare il sistema, risulta integrato nello stesso case esterno. L'assenza di una ulteriore struttura aggiuntiva in cui alloggiarlo garantisce un notevole risparmio di spazio e una forte semplificazione del layout in una eventuale cella di lavoro.

Figura 4.3: Collegamenti interni del controller

l'integrazione tra le due apparecchiature risparmia eventuali collegamenti esterni, possibili cause di urti o interferenze con le macchine in movimento o con gli operatori.

La struttura complessiva è compatta e semplice da trasportare grazie al suo peso di soli 38kg.

La base di appoggio misura 339mm*497mm e consente un montaggio in posizione esclusivamente orizzontale, su un tavolo o una qualunque superficie piana.

Il robot a disposizione si trova posizionato sopra un carrello mobile in alluminio appositamente progettato in una precedente tesi di laurea [13], mostrato in figura 4.4

Figura 4.4: Modello di YuMi collegato al carrello

4.2 MECCANICA E PRESTAZIONI

Dal punto di vista meccanico, YuMi è un robot ridondante poiché ogni braccio presenta 7 assi indipendenti. Un qualunque compito da effettuare nello spazio coinvolge al massimo 6 gradi di libertà, quindi il problema cinematico di posizione risulta sempre indeterminato e la soluzione può essere scelta tra un numero infinito di alternative.

È quasi sempre possibile scegliere la configurazione più comoda per i bracci a seconda del compito da eseguire. Grazie alla ridondanza YuMi esegue operazioni ad elevata destrezza in uno spazio condiviso con l'operatore umano: può raggiungere una certa posa con diverse configurazioni del braccio, evitando singolarità cinematiche quindi evitando di sollecitare i motori eccessivamente, oppure mantenendo fissa la posa del terminale per essere pronto tempestivamente al successivo compito.

I bracci del robot si muovono attorno agli assi descritti nella tabella. Si può notare che nella numerazione dei giunti, il settimo non fa parte del dispositivo terminale ma è inserito al centro della catena cinematica, tra il secondo e il terzo.

ASSE	TIPO DI MOVIMENTO	RANGE DI MOVIMENTO	VELOCITA' MAX
1	Rotazione	-168.5°/+ 168.5°	180°/s
2	Piegatura	-143.5°/43.5°	180°/s
7	Rotazione	-168.5°/+ 168.5°	180°/s
3	Piegatura	-123.5°/+80°	180°/s
4	Rotazione	-290°/+290°	400°/s
5	Piegatura	-88°/+138°	400°/s
6	Rotazione	-229°/+229°	400°/s

Le sue prestazioni sono notevolmente inferiori a quelle di un robot industriale ma proprio queste caratteristiche lo rendono adatto ad operare a stretto contatto con l'operatore umano [4].

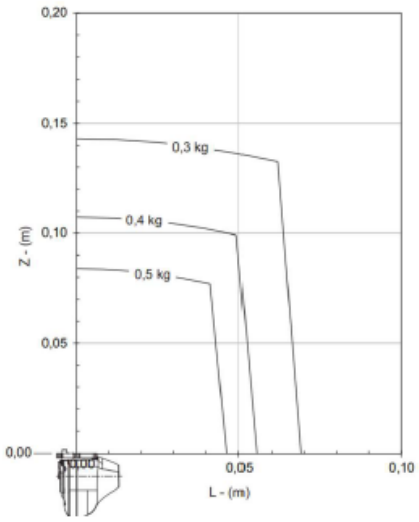

Figura 4.5: Diagramma di carico [14]

Il terminale di YuMi raggiunge una velocità massima di 1,5m/s con una ripetibilità di 0.02mm e il peso che sopporta è al massimo di 0.5kg per ogni braccio.

Ogni braccio di YuMi viene equipaggiato con una pinza sulla quale possono essere presenti una serie di equipaggiamenti diversi.

Figura 4.6: tipologie diverse di pinze ABB in commercio

In base al tipo di gripper sono disponibili ventose, sensori o sistemi di visione integrati per visualizzare lo spazio di lavoro della pinza oltre alle dita che, nella versione di base, consentono di sopportare circa 10000 prese con una forza massima di 20N [15] e sono controllabili in posizione e in forza.

Tramite i disegni a disposizione sul manuale è comunque possibile progettare dei gripper su misura per il compito da svolgere se quelle standard non vanno bene.

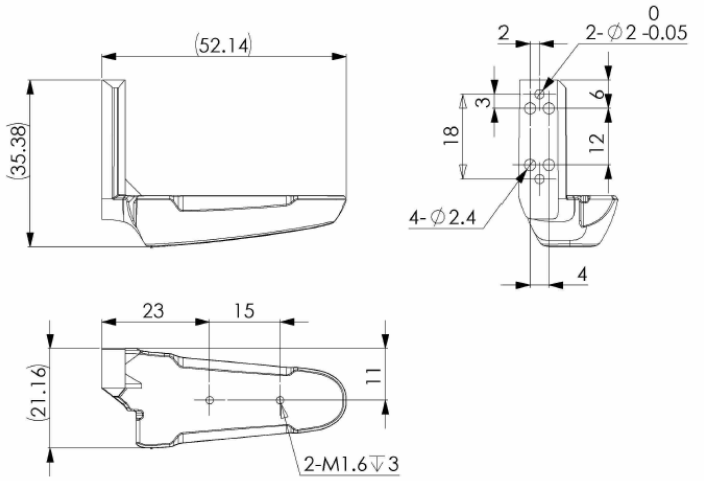

Figura 4.7: Tavole delle dita di YuMi [15]

Tramite i disegni a disposizione sul manuale è possibile progettare pinze su misura per il compito da svolgere.

È possibile che la forma o la resistenza meccanica delle dita standard non siano adeguate

4.3 ROBOTSTUDIO

Il software Robotstudio [16] consente di effettuare una grande quantità di operazioni sui robot simulando il loro comportamento reale. All'interno è disponibile un editor per la programmazione in codice Rapid, un linguaggio ad alto livello dedicato alla movimentazione dei robot di ABB. Il linguaggio consente la programmazione in diverse modalità:

- in modalità online il PC si connette ad un controller fisico ed è possibile programmare i task tramite teach pendant e anche tramite l'editor.

  La programmazione online garantisce una ottima accuratezza delle pose, che sono apprese dal robot con una guida manuale;
- in modalità offline un controller virtuale analogo a quello fisico permette una accurata simulazione dei movimenti del robot. Consente di lavorare sui codici senza avere fisicamente a disposizione il robot. Se la macchina è impegnata in una linea produttiva non è necessario fermarla per scrivere il nuovo programma.

  Inoltre l'editor di testo permette di individuare facilmente gli errori logici nella programmazione tramite un debugger.

CAPITOLO 5: PROGETTAZIONE DEL SISTEMA

In questo capitolo viene descritta dettagliatamente la progettazione del sistema di gioco: la scacchiera, il sistema di visione, l'algoritmo di gioco e la programmazione del robot.

5.1 LA SCACCHIERA

Una prima idea è stata quella di utilizzare una scacchiera elettronica a disposizione del DIISM che riconosce automaticamente le mosse effettuate tramite una tavola sensorizzata e pedine magnetiche e comunicandole al PC con un collegamento USB.

La scacchiera elettronica in dotazione ha un lato di 50cm ed è stato necessario verificare le sue dimensioni in confronto allo spazio di lavoro di YuMi. Infatti per una corretta movimentazione, è necessario che il robot sia in grado di raggiungere ogni punto sulla scacchiera senza portare i giunti a fine corsa. Tramite il software Robotstudio e la simulazione di uno spostamento manuale è stato constatato che risultava troppo grande ed è quindi stata scartata. È stato necessario quindi progettare una nuova scacchiera e una tecnologia per rilevare gli spostamenti dei pezzi.

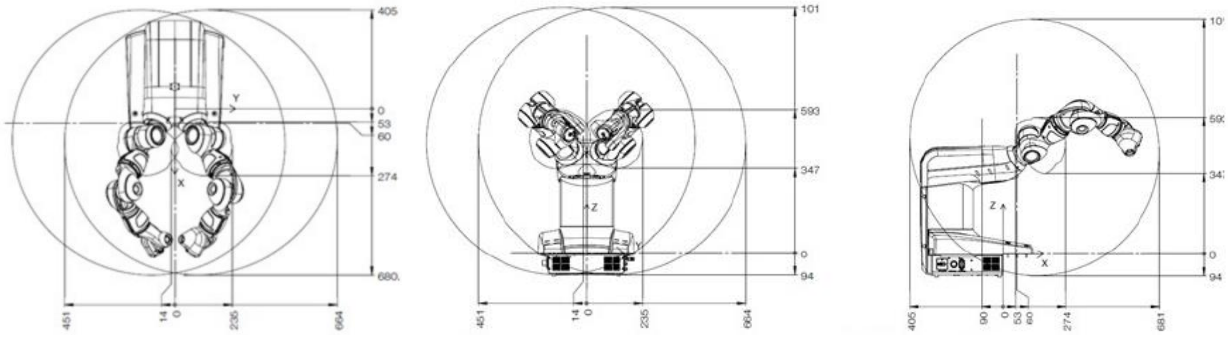

Figura 5.1: Spazio di lavoro di YuMi

Una dimensione della scacchiera compatibile con i bracci del robot è di 36 cm, ed è stata valutata sempre tramite l'ambiente di simulazione.

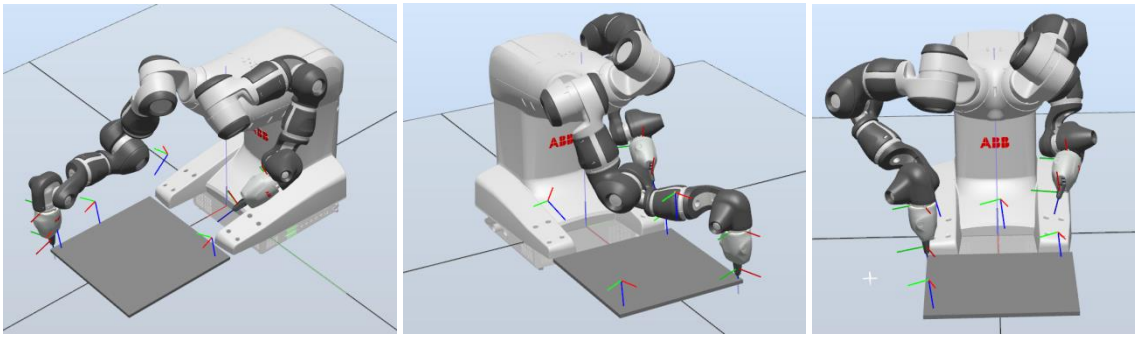

Figura 5.2: Verifica dello spazio di lavoro.

Tramite il posizionamento manuale del robot nei punti estremi della superficie di gioco si verifica che il braccio sia in grado di raggiungerli perfettamente.

Non potendo più disporre della scacchiera elettronica, per il rilevamento delle mosse durante la partita è stato deciso di implementare un sistema di visione artificiale.

Le pedine degli scacchi hanno diversi colori e forme molto varie, l'idea più semplice per il loro riconoscimento si realizza tramite una lastra trasparente in plexiglass su cui disegnare la scacchiera: una webcam posta al di sotto della lastra può così vedere in trasparenza l'impronta delle pedine che è per tutte di forma circolare.

In questo modo è molto semplice riconscere la loro presenza in uno scacchetto.

Figura 5.3: i pezzi degli scacchi presentano forme molto variegate

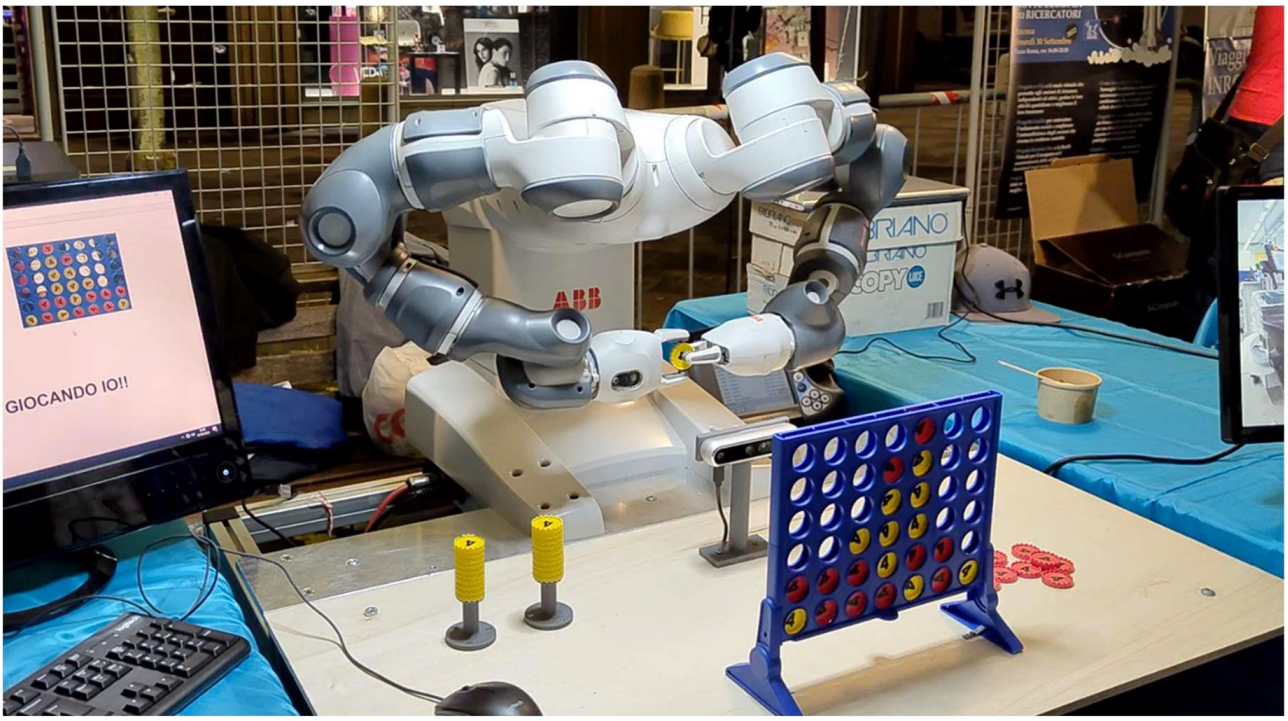

Figura 5.4: In un precedente progetto sviluppato su YuMi [17], lo spazio di gioco è stato realizzato tramite una tavola di dimensioni 1000mm x 500mm collegata al carrello.

Tramite il software di modellazione Siemens NX è stato eseguito un primo modello 3D di una tavola da inserire sul carrello in cui è collegato YuMi per sostituire quella già presente, pensata per il gioco Forza 4, oppure per permettere di scegliere tra entrambi i giochi integrandoli sulla stessa tavola.

Il modello disegnato è stato pensato per questo ultimo caso: le sue caratteristiche principali sono state scelte principalmente per la nuova applicazione sugli scacchi ma le misure e le posizioni delle varie componenti progettate lasciano spazio ai materiali presenti nel precedente lavoro.

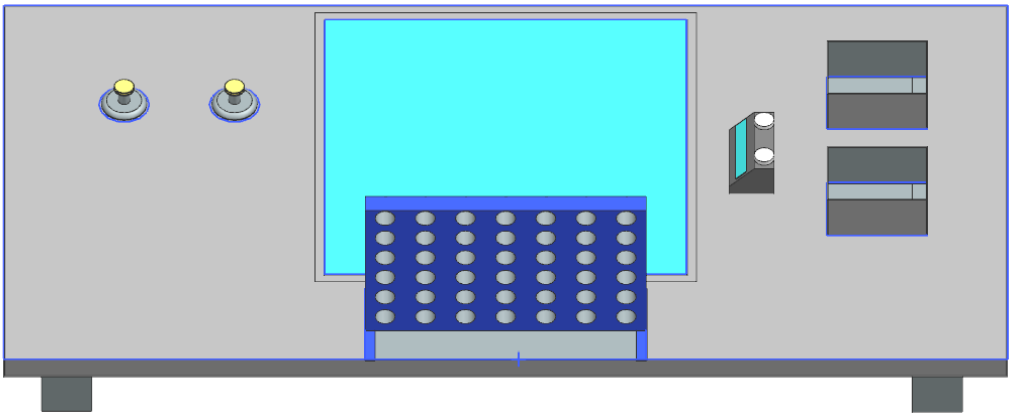

Figura 5.5: Modello CAD, vista dall'alto della tavola

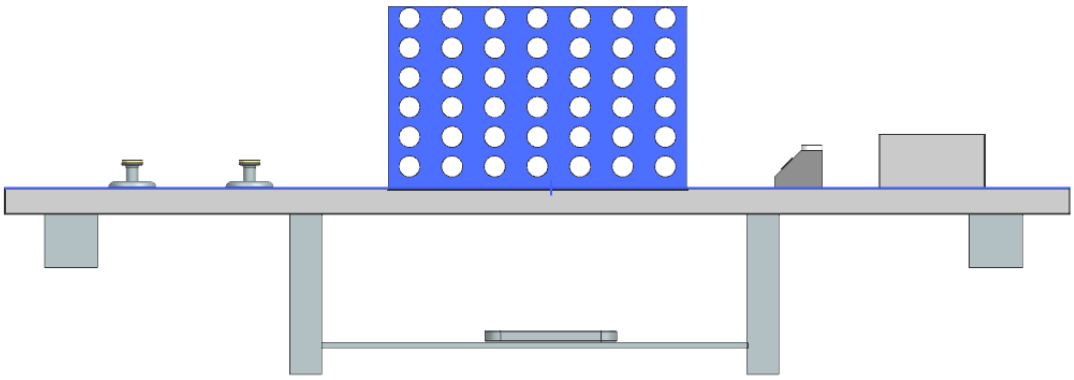

Figura 5.6: Modello CAD: vista frontale della tavola

Di seguito sono elencate le caratteristiche della tavola:

- Le sue dimensioni complessive sul piano sono di 1000mm*500mm;
- La lastra in plexiglass trasparente è quadrata di lato 380mm: oltre alla superficie utile di gioco rimangono 10mm di margine su ogni lato per garantire l'appoggio di essa sulla tavola principale e il suo ancoraggio;
- Ai lati della scacchiera sono presenti un eventuale cronometro e dei contenitori per disporre i pezzi mangiati durante il gioco;
- Un apposito sostegno inferiore sostiene la webcam sotto il piano di gioco: è costituito da due aste verticali e una piccola tavoletta su cui fissare o avvitare la telecamera;
  resta per ora da determinare la distanza ottimale alla quale posizionarla per ottenere una inquadratura completa della superficie di gioco.
- Le due travi disposte lungo l'asse longitudinale consentono l'assemblaggio della tavola al carrello che ospita YuMi.

Il disegno mostra il progetto per una possibile soluzione definitiva, da implementare alla fine del progetto dopo le parti più impegnative di programmazione.
Una prima struttura di prova è stata invece realizzata con sostegni in alluminio collegati con bulloni e una lastra di plexiglass di dimensioni 50cmx50cm.
La lastra è stata forata ai vertici per consentire un collegamento filettato di quest'ultima con le sbarre di alluminio, solo su due estremi.
Da queste sbarre di supporto scendono anche le aste verticali per il sostegno della webcam. Anche la tavoletta inferiore per la webcam è stata provvisoriamente realizzata con una di queste barre.

È possibile variare la distanza a cui posizionarla in un intervallo piuttosto ampio, partendo dalla superficie fino ad una profondità di circa 50 cm. Il posizionamento esatto sarà discusso dopo la scelta della webcam cosi come il suo collegamento alla struttura.

Nelle ultime settimane di lavoro, il trasferimento del robot YuMi presso l' I-Labs di Jesi ha impedito di sviluppare la tavola per lo spazio di lavoro definitivo e, come vedremo in seguito, anche la programmazione fisica delle movimentazioni.

Il supporto in alluminio quindi è stato utilizzato in forma definitiva per tutto lo svolgimento della tesi.

5.2 IL SISTEMA DI VISIONE

Il sistema di visione programmato per questa applicazione si occupa di acquisire immagini della scacchiera al fine di risalire agli spostamenti delle pedine, cioè alle mosse fatte dai giocatori durante la partita.

Viene inizialmente introdotta la notazione algebrica: è il metodo utilizzato dalla federazione internazionale degli scacchi (FIDE) per registrare e descrivere le partite.

Una scacchiera è divisa in quadretti disposti in 8 righe e 8 colonne e la loro numerazione viene effettuata utilizzando come riferimento la vista del giocatore bianco.

Ogni casa è identificata da:

- Una lettera da 'a' ad 'h' che indica la colonna. La più a sinistra è quella indicata da 'a' rispetto al giocatore bianco;
- Un numero da 1 ad 8 indica la traversa, ossia la riga.

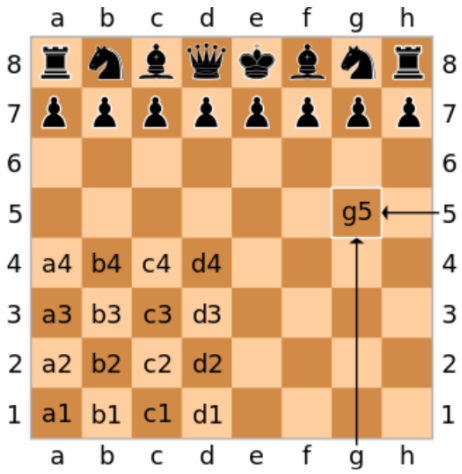

Figura 5.7: ogni casa è descritta tramite una sequenza di due caratteri che determinano univocamente la sua posizione nella scacchiera.

Durante il gioco, per indicare le mosse effettuate è sufficiente unire le coordinate della casa di partenza della pedina spostata con quelle della casa di arrivo tramite la notazione vista in precedenza.

È possibile specificare anche il tipo di pezzo spostato ma non è necessario:

all'inizio di ogni partita la disposizione è sempre la stessa quindi è sufficiente indicare le coordinate delle caselle coinvolte.

Inoltre esistono anche mosse come l'arrocco o la presa en passant, le loro sigle non sono discusse poiché queste mosse speciali non sono state implementate.

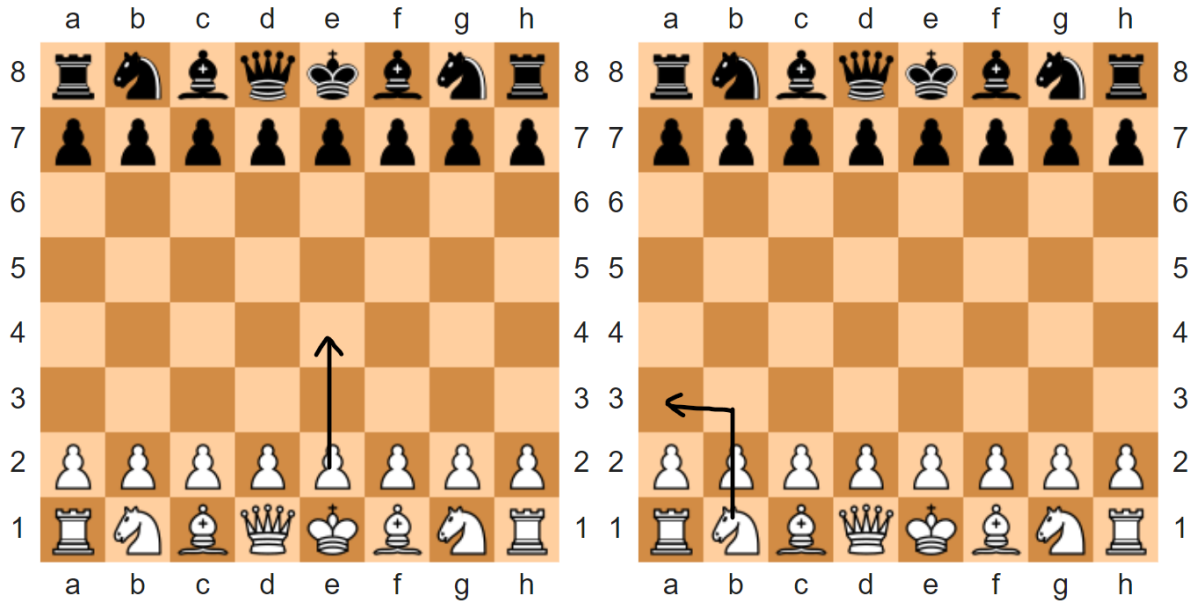

Figura 5.8: esempi di scritture in notazione algebrica.
A sinistra il movimento di un pedone descritto dalla sequenza 'e2e4';
a destra un cavallo effettua il movimento 'b1a3'.

Il sistema di visione consente di acquisire immagini durante il gioco e ricavare la mossa effettuata dall'utente umano scritta in notazione algebrica.

La webcam utilizzata è una Intel Realsense Depth camera D455 [18]:

Figura 5.9: Webcam Intel D455

Il dispositivo è piccolo e con un ampio campo visivo, quindi è perfetta per le applicazioni tipiche della robotica e della visione artificiale.

Nell'applicazione sviluppata viene accesa solamente la telecamera RGB per inquadrare dal basso il piano del gioco e controllare le mosse effettuate da entrambi i giocatori ma la webcam permette anche la visione stereoscopica tramite ulteriori sensori e un proiettore IR.

La telecamera RGB ha le seguenti caratteristiche [18][19]:

	INTEL D455 CAMERA
SENSORE RGB	Global shutter
RISOLUZIONE E FRAME RATE	1280 x 800 a 30fps
FOV SENSORE RGB (HxV)	90°x 65°
RANGE IDEALE	0,6m/6m
DIMENSIONI (L x H x D)	124mm x 26mm x 29mm
INTERFACCIA	USB typeC 3.1 gen1
MONTAGGIO	Due fori M4 sul retro

Dal campo di vista della webcam (FOV), considerando la più stringente condizione sulla direzione verticale, si ottiene che la distanza ottimale alla quale posizionare il dispositivo è di circa 30/35 cm sotto la scacchiera. In questo modo è possibile visualizzare esattamente tutta la superficie di gioco, Con il software Solidedge è stato disegnato e stampato in 3D un supporto per collegare la webcam alla sbarra di supporto sfruttando i suoi fori M4 posizionati nella parte posteriore.

Il pezzo si appoggia sulla sbarra ed è tenuto al centro tramite le alette verticali, inoltre presenta i fori per il collegamento con la struttura e quelli per avvitare la webcam.

Figura 5.10: modello CAD del supporto e realizzazione fisica con webcam collegata

Il codice che gestisce il sistema è scritto in Python tramite l'editor Visual Studio Code, uno dei linguaggi più diffusi al mondo [20] e che combina la semplicità di scrittura alle performance.

Il software deve ottenere dalle immagini le posizioni occupate e quelle libere sulla scacchiera, salvando questa informazione in una apposita struttura dati ad ogni turno.

Il riconoscimento di una mossa durante il gioco avviene analizzando le differenze tra due immagini successive scattate prima e dopo lo spostamento.

È importante evidenziare che:

- la configurazione iniziale del gioco è sempre la stessa, non è quindi necessario riconoscere il tipo di pezzo presente in ogni scacco;
- le pedine sono di colore bianco e nero e hanno forme complesse, quindi oltre al riconoscimento delle caselle occupate si potrebbe pensare di distinguere pure il tipo di pezzo e il colore.

Il primo codice sviluppato non discrimina alcuna caratteristica delle pedine, ma si limita a distinguere le case libere da quelle occupate.

Anche se questo metodo si è rivelato insufficiente per la rilevazione delle le mosse del gioco, sarà ora descritto poiché fornisce un importante risultato: dimostra quali sono le informazioni minime necessarie per il funzionamento del sistema di visione.

Il plexiglass è stato coperto con un foglio di carta bianca molto sottile, con le dimensioni della scacchiera, in modo da eliminare i riflessi di luce dovuti alla superficie trasparente e allo stesso tempo permettere di osservare in trasparenza la pianta circolare delle pedine.

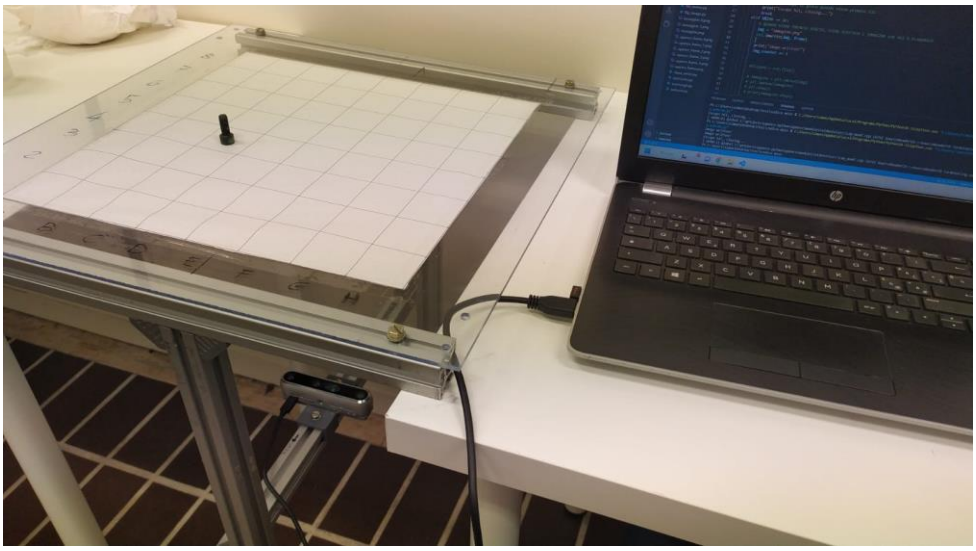

Figura 5.11: disposizione della scacchiera con copertura in carta: la base dei pezzi risulta retroilluminata dalle lampade del laboratorio. Si notano i dettagli della struttura e il collegamento della webcam su di essa, in basso.

A partire dall'immagine scattata si seleziona una ROI per scartare i bordi esterni alla scacchiera poi si effettua una analisi geometrica per distinguere le posizioni occupate.
La trasformata di Hough può essere utilizzata per ricavare i centri delle circonferenze rilevate.
Le loro coordinate in pixel sono poi ricondotte alla notazione algebrica.

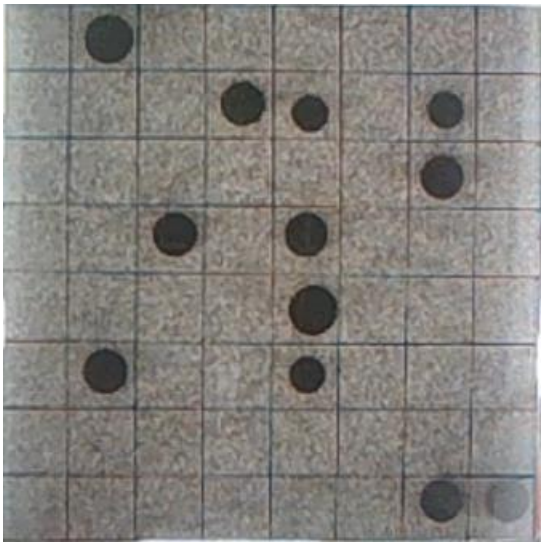

Figura 5.12:
Esempio di una ROI estratta durante la programmazione. Tramite il rivestimento in carta non sono presenti riflessi e le impronte dei pezzi sono facilmente riconoscibili.

Le posizioni sono salvate in un array in cui sono possibili i valori 1 (casa occupata) e 0 (casa libera).

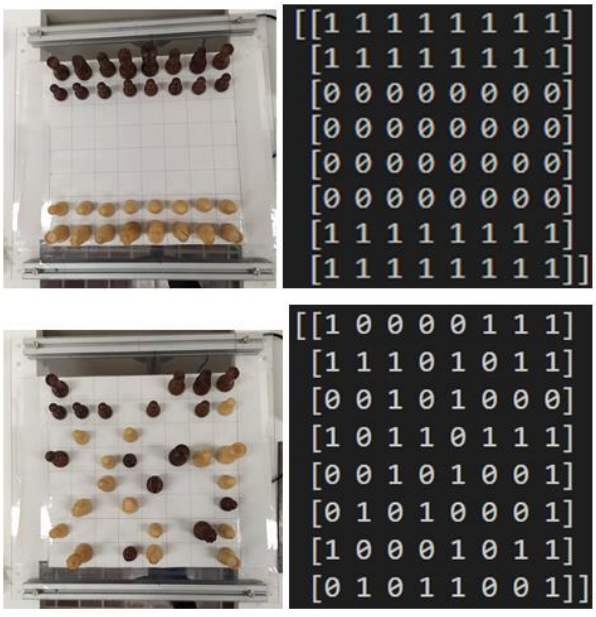

Figura 5.13: test effettuati tramite soglia binaria.
Le posizioni sono salvate in un array utilizzando la libreria Numpy

Scattando una foto dopo ogni mossa e mantenendo in memoria la configurazione precedente, è possibile effettuare una sottrazione tra le due matrici ottenute per evidenziare lo spostamento effettuato.

```
[[ 0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0]
 [ 0  0  1  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0]
 [ 0  0 -1  0  0  0  0  0]
 [ 0  0  0  0  0  0  0  0]]
```

Figura 5.14: matrice ottenuta dalla differenza.

Tutte le posizioni occupate prima e dopo lo spostamento sono state cancellate; si evidenziano i punti di partenza e di fine del movimento. La mossa effettuata in questo caso è 'c2c4'

La modalità descritta non consente di gestire le situazioni che si presentano quando avviene una mangiata delle pedine: se un giocatore ha la possibilità di mangiare più di una pedina avversaria utilizzando lo stesso pezzo, è impossibile riconoscere lo spostamento effettuato senza aver discriminato il colore dei pezzi in gioco.

Quando più mosse legali sono disponibili, il calcolo della differenza tra le matrici delle configurazioni genera una indeterminazione.

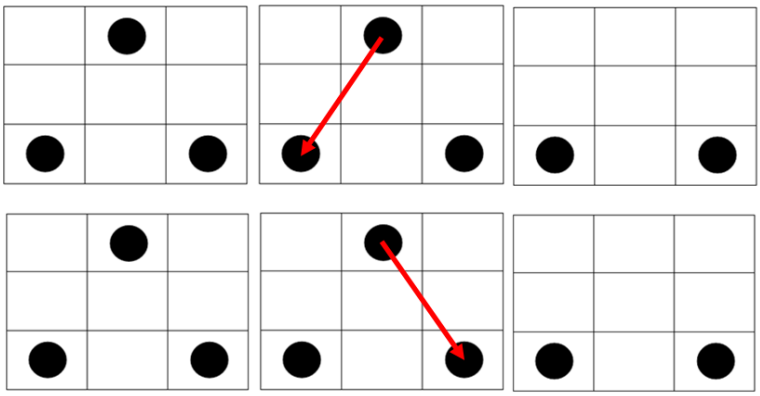

Figura 5.15: Indeterminazione nelle mosse: sono raffigurate le sottomatrici di una ipotetica configurazione di gioco.

A sinistra è mostrata la posizione iniziale: Il pezzo superiore è nero, gli altri due sono bianchi ma hanno la stessa rappresentazione poiché il metodo finora utilizzato non discrimina le pedine dei due giocatori.

Se per il nero è possibile mangiare entrambi i pezzi bianchi, con mosse differenti, si ottiene la stessa configurazione finale ed è impossibile distinguere le due mosse tramite il metodo sviluppato.

La differenza tra le due matrici è identica.

Per questo la detecion delle case occupate non è da sola sufficiente: bisogna conoscere anche il colore delle pedine.

La soluzione definitiva adottata prevede l'utilizzo di marker specifici da posizionare sotto ogni singola pedina per permettere il riconoscimento del loro colore.

Con essi la ricerca delle posizioni occupate è immediata: se un marker viene individuato, allora significa che in quel punto è presente un pezzo degli scacchi.

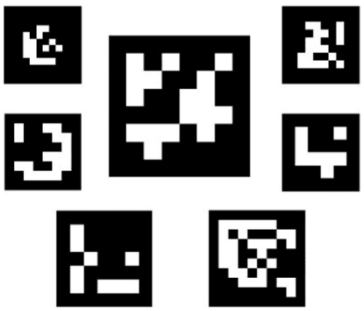

Figura 5.16: Aruco marker.

Sviluppati dall' Artificial Vision Application group (AVA) dell' università di Cordoba [21], appartengono alla categoria dei fiducial marker, target che vanno posizionati nel campo di vista di un sistema di imaging artificiale come punti di riferimento o di misura.

Sono stati utilizzati gli ArUco marker [21] , molto diffusi in applicazioni di computer vision come la navigazione autonoma dei robot o la realtà aumentata.

Hanno una forma quadrata con un bordo nero esterno e una matrice binaria interna che determina il loro numero identificativo o ID e permettono di effettuare una stima della loro posizione e orientamento nel campo di misura.

Sono raccolti in dizionari specifici in base alla loro dimensione, distinti in base al numero di bit che li compongono. Un dizionario è composto da un certo numero di marker univoci.

Esistono per il linguaggio Python una serie di funzioni dedicate da utilizzare insieme alla nota libreria per l'analisi di immagini OpenCV.

La creazione dei marker avviene grazie ad apposite funzioni contenute nella libreria 'aruco', oppure tramite un sito web.

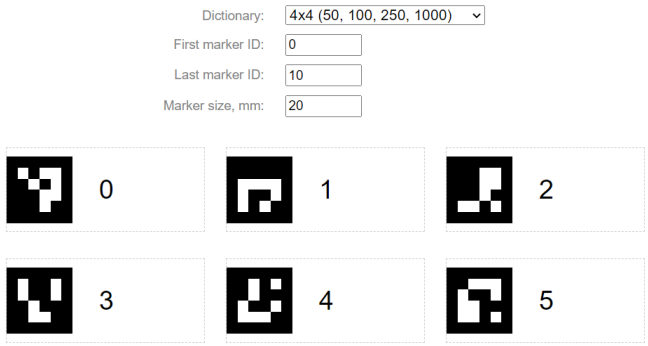

Figura 5.17: Generazione dei marker.

Nel sito è sufficiente specificare i seguenti parametri:

il dizionario scelto, il numero di id univoci necessari, cioè la dimensione del dizionario e la dimensione in mm dei marker.

Viene generato un file da stampare che contiene i marker desiderati.

Successivamente si effettua il vero e proprio processo di detection:

prendendo in input una immagine che contiene dei marker, le fuzioni di libreria forniscono come risultato [21]:

- il numero di marker rilevati;
- un elenco con tutti gli ID univoci presenti;
- Le coordinate in pixel dei quattro angoli per ogni marker.

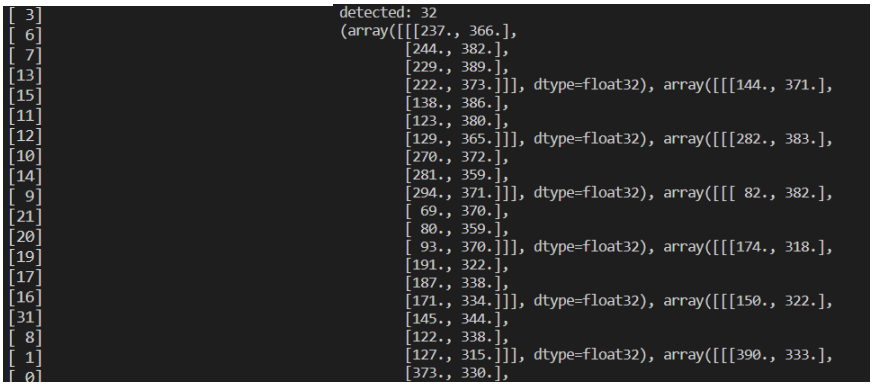

Figura 5.18: Risultati della detection: a sinistra un elenco con gli ID rilevati, a destra la lista con le informazioni sui vertici

Il processo di ricerca avviene in due fasi [21]:

1) dopo l'applicazione di opportuni filtri che esaltano i contorni degli oggetti si cercano nell'immagine delle forme geometriche simili ad un quadrato che possono essere candidate ad essere dei marker. Le forme che non rispondono ai requisiti sono scartate.
2) dopo aver scelto tutti i candidati, viene analizzata la loro codifica interna.
   Una soglia permette di distinguere le parti bianche dalle nere e contarle. Se i marker effettivamente rilevati appartengono al dizionario scelto, allora le loro caratteristiche sono trasferite in output.
3) Ulteriori funzioni della libreria permettono di ottenere l'orientamento dei marker. Questa funzionalità non è stata utilizzata nella tesi.

Nella nuova configurazione le pedine sono posizionate sopra alla scacchiera trasparente ma senza interporre il foglio sottile ed ognuna appoggia sopra ad un marker univoco.

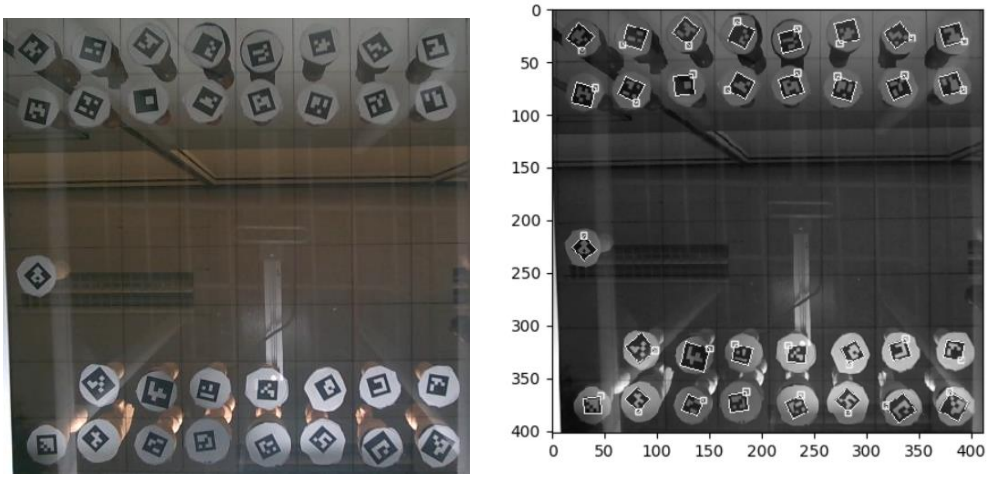

Figura 5.19: visuale inquadrata dal sistema.
Con la rimozione dello strato di carta sono presenti nel campo di vista i riflessi della zona sottostante alla scacchiera e la luminosità dell'immagine è molto più influenzata dai parametri dell'illuminazione.
A destra: i marker rilevati vengono evidenziati.

Il software può commettere errori nella detection a causa dell'illuminazione, ma questi consistono sempre in un numero di marker rilevati in difetto rispetto a quelli presenti.
Il problema è causato da una illuminazione non uniforme di ogni singolo marker: non sono rilevati se sono troppo o poco esposti.

In fotografia lo scatto in modalità High Dynamic Range o HDR [11] permette di ottenere una immagine con un intervallo tra le aree scure e quelle chiare molto ampio attraverso scatti multipli ad esposizione variabile. Una elaborazione permette di ricostruire una nuova immagine da quelle di partenza in cui i dettagli sovraesposti o sottoesposti sono stati corretti.

Figura 5.20 : esempio di elaborazione HDR

Per mitigare il problema della errata esposizione dei marker si utilizza una tecnica simile: ad ogni mossa si effettua uno scatto multiplo con esposizione variabile. La regolazione avviene attraverso la libreria OpenCV [22], che aggiorna automaticamente i parametri della webcam.

Non è necessario ricostruire l'immagine correttamente esposta: ad ogni scatto le posizioni visualizzate sono salvate e aggiunte a quelle trovate con il livello precedente di esposizione.

Le prove sono state effettuate in diverse condizioni di illuminazione:

- in laboratorio con i neon accesi: questa illuminazione ha fornito i risultati peggiori a causa delle caratteristiche delle lampade: il loro spettro non costante nel tempo e i risultati sono più o meno accurati in base all'istante di scatto della foto.
- in laboratorio con la luce spenta ma utilizzando una luce led con geometrie differenti: i migliori risultati sono stati ottenuti con una luce parallela al piano di gioco, con assenza di luce diretta o riflessa nel campo visivo della telecamera.
- in casa con la luce naturale proveniente dalle finestre.

Le varie prove hanno mostrato l'importanza di avere una illuminazione strutturata specifica per l'applicazione che si sta sviluppando nel campo della visione artificiale. Come spiegato nel secondo capitolo, l'illuminazione è il parametro fondamentale per un sistema di visione: è stato possibile verificare che con una luce di cattiva qualità il software non fornisce i risultati aspettati anche se il codice è perfettamente funzionante.

Tra i risultati della detection bisogna memorizzare:

- la posizione in pixel di uno qualunque dei 4 vertici dei marker, infatti non è importante l'orientamento e comunque sono tutti compresi nella casa di appartenenza;
- Gli ID: basta distinguere il loro valore: i bianchi hanno valore tra 0 e 15, i neri hanno quelli da 15 a 31.

La posizione in pixel dei vertici trovati si converte nel sistema di riferimento della scacchiera e le posizioni occupate sono memorizzate ad ogni turno in un array bidimensionale che contiene:

- 0 nelle caselle in cui non è presente alcun pezzo;
- 2 nelle caselle con i pezzi bianchi;
- 3 nelle caselle con i pezzi neri.

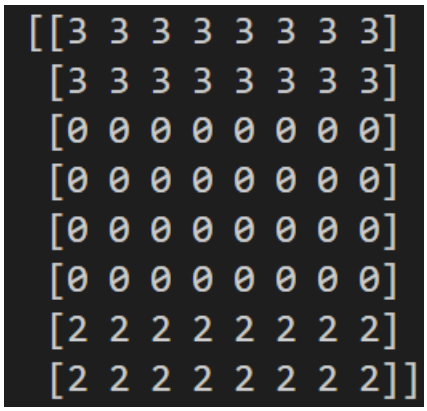

Figura 5.21: matrice delle posizioni nella configurazione iniziale del gioco

La pressione del tasto 'spazio' permette il cambio del turno e il ricalcolo dell' array aggiornato, sia da parte del giocatore umano, sia da parte del robot. I valori precedenti sono tenuti in memoria.
Per poter ottenere la mossa effettuata in notazione algebrica, ad ogni turno viene calcolata la differenza tra le due configurazioni: nella matrice risultante scompaiono tutti i valori relativi alle pedine che non sono state mosse mentre sono evidenziate le posizioni in cui è iniziato e finito li spostamento. Solamente due posizioni saranno occupate da numeri diversi da 0 e il loro valore è diverso in base al tipo di mossa effettuata. Per conoscere le case coinvolte basta cercare gli indici della matrice in cui sono presenti questi valori.

Questo metodo esclude la possibilità di spostamenti multipli come quelli che si verificano nel caso di mosse speciali, queste quindi non sono riconosciute dal sistema.

Figure 5.22, 5.23, 5.24, 5.25 : si distinguono 4 casi diversi per individuare le mosse effettuate:
la differenza avviene tra la configurazione attuale e quella presente prima della mossa. Sono utilizzati gli array numpy [23]

1) Muove il bianco e non mangia:

$$\begin{pmatrix}3&3&3&3&3&3&3&3\\3&3&3&3&3&3&3&3\\0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0\\0&0&0&2&0&0&0&0\\0&0&0&0&0&0&0&0\\2&2&2&0&2&2&2&2\\2&2&2&2&2&2&2&2\end{pmatrix} - \begin{pmatrix}3&3&3&3&3&3&3&3\\3&3&3&3&3&3&3&3\\0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0\\2&2&2&2&2&2&2&2\\2&2&2&2&2&2&2&2\end{pmatrix} = \begin{pmatrix}0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0\\0&0&0&2&0&0&0&0\\0&0&0&0&0&0&0&0\\0&0&0&-2&0&0&0&0\\0&0&0&0&0&0&0&0\end{pmatrix}$$

2) Muove il nero e non mangia. Il caso è analogo al primo ma si otterranno i valori -3 e 3 rispettivamente nelle case di partenza e di arrivo:

$$\begin{pmatrix}3&3&3&3&3&3&3&3\\3&3&3&0&3&3&3&3\\0&0&0&0&0&0&0&0\\0&0&0&3&0&0&0&0\\0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0\\2&2&2&2&2&2&2&2\\2&2&2&2&2&2&2&2\end{pmatrix} - \begin{pmatrix}3&3&3&3&3&3&3&3\\3&3&3&3&3&3&3&3\\0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0\\2&2&2&2&2&2&2&2\\2&2&2&2&2&2&2&2\end{pmatrix} = \begin{pmatrix}0&0&0&0&0&0&0&0\\0&0&0&-3&0&0&0&0\\0&0&0&0&0&0&0&0\\0&0&0&3&0&0&0&0\\0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0\end{pmatrix}$$

3) Muove il bianco e mangia: in questo caso i valori ottenuti sono diversi, lo spostamento va dalla casa con il '-2' a quella che contiene '-1'

$$\begin{pmatrix}3&3&3&3&3&3&3&3\\3&3&3&0&3&3&3&3\\0&0&0&0&0&0&0&0\\0&0&0&2&0&0&0&0\\0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0\\2&2&2&2&0&2&2&2\\2&2&2&2&2&2&2&2\end{pmatrix} - \begin{pmatrix}3&3&3&3&3&3&3&3\\3&3&3&0&3&3&3&3\\0&0&0&0&0&0&0&0\\0&0&0&3&0&0&0&0\\0&0&0&0&2&0&0&0\\0&0&0&0&0&0&0&0\\2&2&2&2&0&2&2&2\\2&2&2&2&2&2&2&2\end{pmatrix} = \begin{pmatrix}0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0\\0&0&0&-1&0&0&0&0\\0&0&0&0&-2&0&0&0\\0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0\end{pmatrix}$$

4) Muove il nero e mangia: il movimento è dal '-3' al '1'

$$\begin{pmatrix}3&3&3&3&3&3&3&3\\3&3&3&0&3&3&3&3\\0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0\\0&0&0&0&3&0&0&0\\0&0&0&0&0&0&0&0\\2&2&2&2&0&2&2&2\\2&2&2&2&2&2&2&2\end{pmatrix} - \begin{pmatrix}3&3&3&3&3&3&3&3\\3&3&3&0&3&3&3&3\\0&0&0&0&0&0&0&0\\0&0&0&3&0&0&0&0\\0&0&0&0&2&0&0&0\\0&0&0&0&0&0&0&0\\2&2&2&2&0&2&2&2\\2&2&2&2&2&2&2&2\end{pmatrix} = \begin{pmatrix}0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0\\0&0&0&-3&0&0&0&0\\0&0&0&0&1&0&0&0\\0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0\\0&0&0&0&0&0&0&0\end{pmatrix}$$

Un indice tiene in memoria il numero di pezzi che devono essere presenti sulla tavola alla mossa successiva e viene diminuito nel caso di una mangiata. Durante il seguente turno il sistema deve rilevare questo numero di pezzi per non effettuare errori.

Le posizioni delle case coinvolte nello spostamento vanno ora scritte in notazione algebrica. Come visto in precedenza essa è composta da lettere e numeri.
Gli indici che individuano le posizioni dei valori nella matrice sono compresi in un range numerico con valori da 0 a 7, è necessario effettuare due trasformazioni su queste sequenze:

- La coordinata che indica le colonne deve essere scritta con una lettera variabile tra 'a' e 'h', la trasformazione avviene tramite un dizionario che associa ad ogni numero la lettera corrispondente;
- La coordinata che indica le traverse deve variare invece nell'intervallo 1-8.

Dopo questa elaborazione la mossa è correttamente scritta in notazione algebrica e viene salvata come stringa. Il processo viene effettuato anche per il giocatore nero con il fine di aggiornare la scacchiera e mantenerla in memoria ed eventualmente per modificare il numero di pezzi in gioco.

5.3 L'INTELLIGENZA ARTIFICIALE DI GIOCO

Dopo aver ottenuto lo spostamento l'informazione sulla mossa è codificata in una stringa di 4 caratteri come spiegato in precedenza.
Il turno del giocatore nero, cioè YuMi, è controllato da una intelligenza artificiale che è in grado di calcolare la mossa da compiere.
Questo programma deve ricevere in ingresso la mossa effettuata dal giocatore umano e deve fornire come uscita la mossa di risposta che effettuerà il robot.
In rete è possibile trovare tantissimi siti e applicazioni che permettono di effettuare una partita a scacchi contro un computer, questi programmi sono quindi ben noti e disponibili gratuitamente.
Questa parte del gioco è stata semplicemente scaricata in codice Python a causa della sua complessità e vista la grande disponibilità di algoritmi online.
Il codice utilizzato contiene l'algoritmo 'Sunfish' [24]: scritto in linguaggio Python da Thomas Dybdahl Ahle, sviluppatore software presso il Basic Algorithms Research Group (BARC) dell'università di Copenhagen, è un codice open source abbastanaza semplice poiché è stato sviluppato per ragioni didattiche.
Si basa su un algoritmo ricorsivo Minimax: sceglie la mossa successiva da effettuare associando un valore ad ogni configurazione del gioco.
È facile capire il suo funzionamento quando la partita è vicina alla fine. Siccome è possibile vincere, perdere o pareggiare, allora:

- se il giocatore A può vincere con una sola mossa, la mossa migliore è quella vincente;

- se il giocatore B sa che una data mossa porterà A alla vittoria nel suo prossimo turno, mentre un'altra lo porterà a pareggiare, la migliore mossa del giocatore B è quella che lo porterà alla patta.

Ad ogni giocata l'algoritmo cerca la mossa migliore da effettuare simulando lo svolgimento probabile che porta alla fine del gioco e risalendo verso la configurazione attuale.

Si assume che ogni giocatore cerchi di massimizzare le sue probabilità di vincere minimizzando quelle dell' avversario.

L'algoritmo contiene una funzione di valutazione delle posizioni che indica quanto è desiderabile ottenere una certa configurazione di gioco in base al suo punteggio associato. Il programma quindi con la sua mossa cerca di minimizzare il valore della migliore posizione ottenibile dall'avversario.

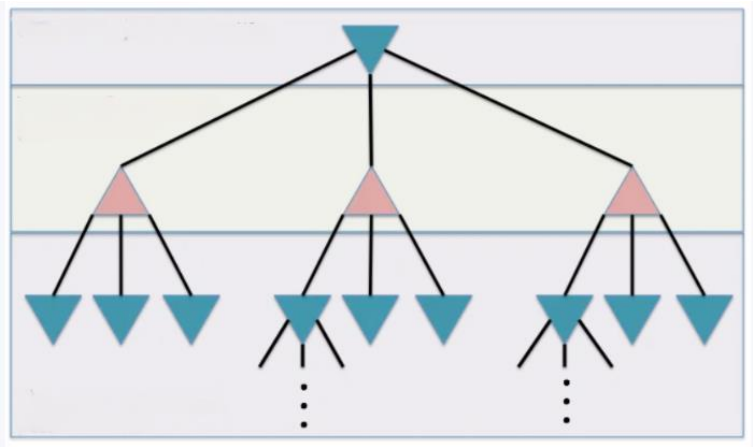

Figura 5.26 : il gioco può essere immaginato come un albero in cui la radice è la configurazione attuale dopo la mossa dell'umano. Al secondo livello sono contenute le possibili mosse di risposta massimizzanti e per ognuna di esse l'algoritmo valuta le configurazioni future dopo il successivo turno dell'umano. La mossa effettuata sarà quella che minimizza il punteggio raggiunto dall'utente umano.

Negli scacchi questa valutazione è molto complessa poiché non è possibile risalire a tutte le mosse disponibili da una certa configurazione quindi è calcolata una stima della possibilità di vittoria raggiungibili tramite ogni mossa.

Tramite appositi algoritmi di 'potatura' possono essere migliorate le prestazioni: si riduce il numero di mosse accettabili a partire dall' 'albero' che contiene tutte quelle a disposizione.

In generale l'algoritmo minimax garantisce la soluzione ottima solo nei giochi a somma zero cioè quelli in cui il guadagno o la perdita di uno dei giocatori è perfettamente bilanciato rispetto alla perdita o guadagno dell'altro giocatore.

Il codice ‘Sunfish’ consente quindi di rispondere alle mosse del giocatore umano che si troverà a giocare con un livello di difficoltà non altissimo.
Il programma genera una sequenza di 4 caratteri (una stringa) in notazione algebrica per le mosse dei i pezzi neri ed è necessaria per movimentare il robot.
L’interfaccia di gioco sul terminale mostra ad ogni turno la mossa effettuata in notazione algebrica e la configurazione attuale del gioco.

Figura 5.27: Esempio di sequenza interattiva mostrata in un turno di gioco
Il programma stampa la configurazione del gioco dopo aver rilevato la mossa del giocatore bianco, poi risponde con la mossa calcolata dall’algoritmo

5.4 LA PROGRAMMAZIONE DEL ROBOT

La programmazione è stata effettuata in linguaggio Rapid tramite il software Robotstudio che permette di simulare tutte le funzionalità dei robot di ABB.
È un linguaggio di programmazione ad alto livello costituito da una sequenza di istruzioni che descrivono il comportamento della macchina.
Una serie di argomenti aggiuntivi definiscono i parametri di ogni istruzione.
È possibile memorizzare informazioni in strutture dati di tre tipi [25]:

- costanti: il loro valore è statico e non può essere modificato;
- variabili: è possibile assegnare un nuovo valore ad essi durante l’esecuzione del programma;

- persistenti: il loro valore si mantiene tra tutti i robot comandati dallo stesso controller.

All'interno della stazione di lavoro sono stati inseriti il robot e tutti i componenti della stazione di lavoro.

Tramite la simulazione del movimento lineare del TCP, è già stato valutato lo spazio di lavoro all'inizio dell'attività.

L'avvio del controller virtuale permette di effettuare una programmazione offline, senza avere il robot fisico a disposizione. Il vantaggio principale di questo metodo è la possibilità di scrivere il nuovo codice senza interrompere la macchina, se questa sta lavorando in una linea produttiva.

Valutando i tempi a disposizione dopo il trasferimento del robot YuMi presso l'I-Labs di Jesi [26], è stato scelto di approfondire lo sviluppo del sistema di visione effettuando esclusivamente in simulazione la programmazione dei compiti.

Alternativamente il codice Rapid si può scrivere in modalità online attraverso il teach pendant.

La programmazione online permette invece di effettuare una guida manuale sui target da raggiungere nei movimenti.

Durante una partita a scacchi è necessario poter spostare le pedine in qualunque punto della scacchiera e nel caso di una mangiata bisogna eliminare dal gioco una pedina del giocatore avversario, quindi:

1. si possono dividere equamente i compiti tra i due bracci di YuMi: ognuno si occupa di ciò che succede nella metà della scacchiera più vicina ad esso.

   I due singoli bracci devono essere in grado di gestire i movimenti in tutta la superficie che controllano, sia le mosse normali che quelle con una mangiata.

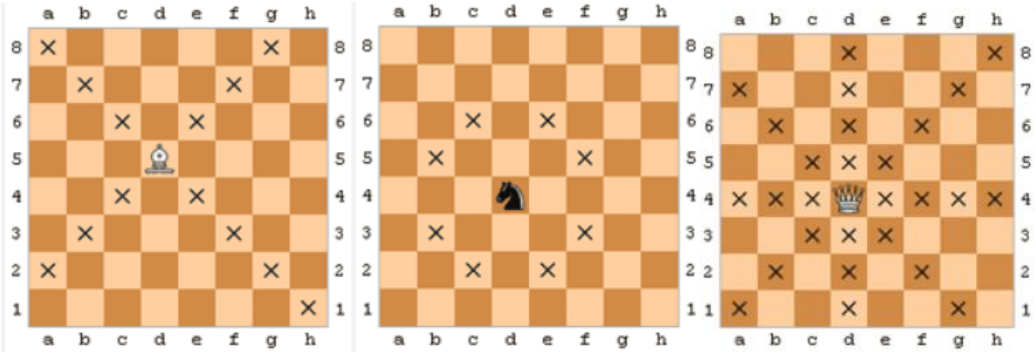

Figura 5.28: esempi dei range di movimento disponibili ad alcuni pezzi.

I movimenti consentiti sono molto variabili tra le diverse pedine, alcune di esse non hanno né una direzione né una distanza prestabilita di movimento e possono coprire grandi distanze in una sola mossa.

Una intera divisione dei compiti sarebbe necessaria se i due bracci singolarmente non riuscissero a coprire l'intera superficie della tavola e nel caso di lunghi spostamenti orizzontali o diagonali, bisogna effettuare un passaggio del pezzo da un braccio all'altro durante la mossa.

2 Ognuno dei bracci di YuMi copre benissimo tutta la tavola di gioco quindi la prima scelta è stata scartata a causa di numerose complicazioni alla programmazione non necessarie.
È stato programmato il braccio destro per le movimentazioni normali, mentre il sinistro si muove solo per mangiare le pedine bianche, quando necessario.
In caso di una mangiata si muoverà prima il braccio sinistro, per liberare la casella, e poi il destro che effettua lo spostamento del pezzo nero.

All'interno del software, YuMi muove il sistema di riferimento del TCP nello spazio tridimensionale rispetto ad un sistema di riferimento universale così orientato:

- asse x: asse longitudinale, in rosso;
- asse y: asse trasversale, in verde;
- asse z: asse verticale, in blu.

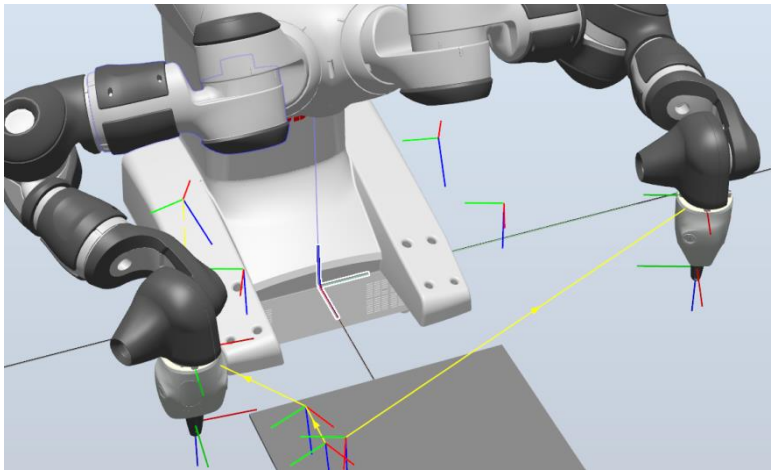

Figura 5.29: Sistemi di riferimento.
al centro della base di YuMi è evidenziato il sistema di riferimento universale. Le altre terne visualizzate nello spazio di lavoro sono le configurazioni assunte dai TCP dei due bracci nella movimentazione attualmente inserita.

L'asse longitudinale è invertito rispetto al sistema di riferimento di gioco utilizzato finora, dal punto di vista del giocatore bianco.
Per muovere il robot in uno qualunque degli scacchi della tavola di gioco, non è consigliabile apprendere tutte le 64 pose ma sono salvate solo quelle relative alle case estreme per una

colonna e per una traversa, tutte le case intermedie sono raggiunte tramite interpolazione delle loro coordinate per migliorare la precisione nel raggiungimento dei target.
Le case da raggiungere sono aggiornate ad ogni mossa partendo dalla posa memorizzata in base alle coordinate ricevute dall'intelligenza di gioco.
Siccome l'algoritmo Sunfish fornisce le coordinate in notazione algebrica, queste sono trasformate in sequenze di caratteri puramente numerici attraverso un secondo dizionario prima di essere inviate al robot e il sistema di riferimento è ruotato per essere concorde a quello universale di YuMi.

La comunicazione tra YuMi e il codice di gioco in Python avviene tramite il protocollo TCP/IP.
Esso consente ai dispositivi connessi ad internet di comunicare tra loro attraverso una rete. Il protocollo suddivide i dati da trasmettere in pacchetti e li riassembla nel messaggio completo una volta raggiunta la destinazione. L'invio di dati in piccoli gruppi garantisce una maggiore accuratezza rispetto all'invio contemporaneo di tutte le informazioni. TCP/IP oggi è uno standard per tutte le comunicazione internet.
Robotstudio permette di simulare la connessione al robot senza averlo disponibile fisicamente, il controller virtuale assolve questa funzionalità in assenza di quello fisico.
Per connettere i due dispositivi si utilizza una tecnologia websocket tramite le librerie standard disponibili in Python che permettono una comunicazione full-duplex [27].
Una architettura di rete server-client locale (LAN) permette ai due dispositivi di comunicare e scambiarsi dati. È sufficiente la trasmissione di pochi caratteri per garantire la movimentazione del robot.

Figura 5.30: Connessione LAN virtuale.
Il controller virtuale (Robotstudio) assume il ruolo di server, mentre il PC è il client (Visual Studio Code).

Siccome il PC ed il controller virtuale sono sullo stesso supporto fisico, si effettua una connessione sull' indirizzo 127.0.0.1 e viene visualizzato un messaggio di conferma nel terminale se l'operazione va a buon fine, prima dell'avvio del gioco.

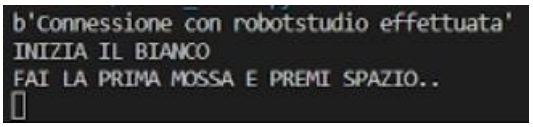

Figura 5.31: Inizio del gioco

Per effettuare le mosse, il robot riceve alcuni caratteri dalla rete salvati in una stringa: queste stringhe sono modificate prima di essere spedite aggiungendo un quinto carattere.
Quando una mossa non prevede alcuna mangiata, ai caratteri della notazione algebrica viene aggiunto il carattere '0', altrimenti viene aggiunta una 'x' che attiva, se necessario, il braccio sinistro del robot.

I controller virtuali di ABB gestiscono la funzionalità Multimove: essa consente ad un unico controller di gestire diverse unità meccaniche coordinate tra loro.
Si possono comandare fino a 36 assi contemporaneamente ed eseguire fino a sette task di movimento suddivisi tra quattro manipolatori [28].
Le modalità di lavoro sono:

- movimenti indipendenti;
- movimenti semicoordinati;
- movimenti coordinati sincronizzati.

Il robot YuMi presenta due manipolatori gestiti da un solo controllore, quindi è necessario utilizzare una modalità Multimove.
Per l'applicazione sviluppata è sufficiente effettuare dei movimenti semicoordinati:
I task di movimento sono scritti in due script separati per ogni singolo manipolatore ed esiste solamente una piccola dipendenza tra i due bracci: nel caso di mangiata di una pedina bianca, si inizia a muovere il braccio sinistro di YuMi mentre il destro deve momentaneamente aspettare per evitare conflitti nello spazio di lavoro.
L'interazione viene gestita tramite una variabile persistente unita ad una istruzione di attesa.
Quando l'intelligenza di gioco produce la mossa da effettuare, la stringa viene modificata aggiungendo il carattere di controllo come visto in precedenza.
La stringa ottenuta viene memorizzata in una variabile persistente, disponibile ad entrambi i manipolatori. Quando l'ultimo carattere assume il valore 'x':

- il braccio destro rimane in attesa per un certo numero di secondi;

- il braccio sinistro si attiva e effettua la rimozione della pedina bianca, collocata nel punto indicato dalla seconda coordinata nella notazione algebrica;
- dopodiché il braccio destro si riattiva e sposta la sua pedina nera nella posizione liberata.

Il robot dopo ogni mossa porta i suoi bracci in una posizione di attesa ad una certa altezza dal piano di gioco. Questa distanza è necessaria per evitare di colpire le pedine in gioco con le dita del manipolatore, e permette di effettuare con precisione la presa dopo l'allineamento verticale.

Alla ricezione della mossa compie i seguenti movimenti in sequenza, con il braccio destro:

- mantenendo la stessa quota, si porta sopra la casa con il pezzo da muovere;
- si abbassa, prende il pezzo e si risolleva;
- si sposta verso la casa di arrivo e lascia il pezzo
- torna alla posizione di attesa dopo aver premuto il pulsante per il cambio turno.

Se nella mossa viene mangiato un pezzo bianco, la sequenza viene ritardata del tempo necessario alla rimozione, tramite il braccio sinistro.

Tutti i task sono effettuati in modalità MoveL: è un'istruzione che sposta il robot linearmente dalla sua posizione a quella specificata.

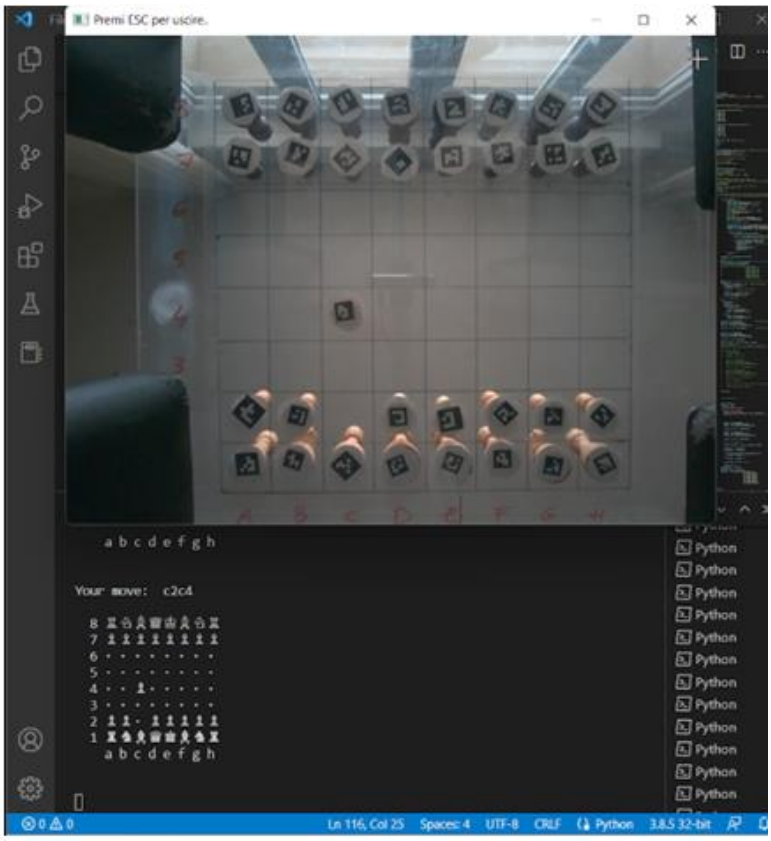

Figura 5.23: Configurazione visualizzata durante il gioco

in alto si trova la finestra grafica con la visuale catturata dalla webcam, in basso la mossa effettuata viene riconosciuta e registrata nel gioco dopo aver aggiornato graficamente la situazione del gioco.

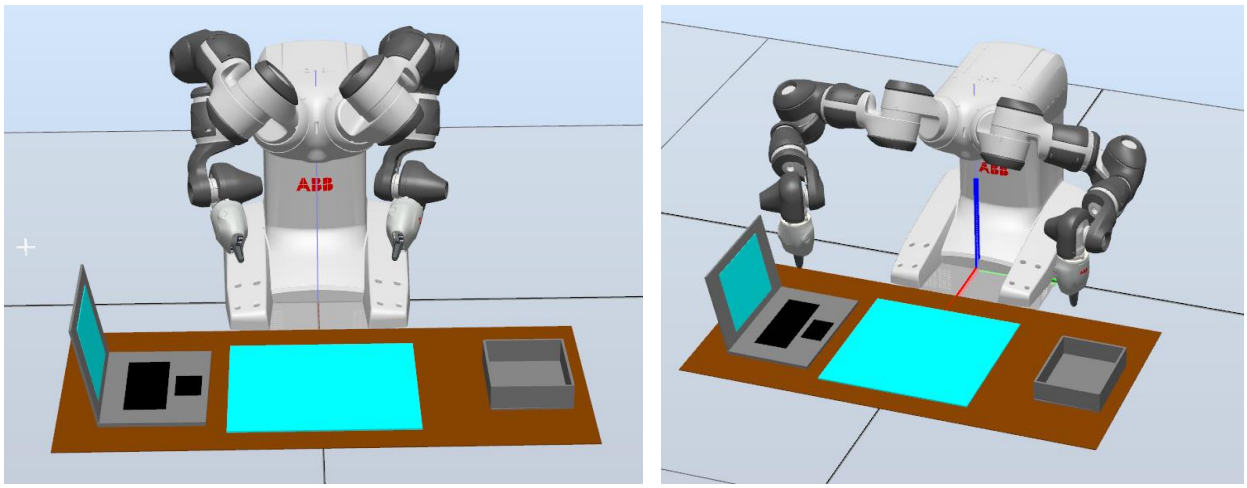

Figura 5.24: Disposizione dello spazio di lavoro

A sinistra: posizione di partenza prima dell'avvio del gioco

A destra: YuMi in attesa di effettuare la mossa successiva

Nell'ambiente virtuale di Robotstudio è stato riprodotto un esempio della tavola di gioco definitiva. Oltre alla scacchiera trasparente sono presenti:

- a sinistra il PC, posizionato sotto al braccio di YuMi in posizione di attesa.
  Alla fine di ogni turno sia il giocatore che YuMi devono premere lo spazio, come visto in precedenza;
- un contenitore raccoglie le pedine mangiate rilasciate dal braccio sinistro del robot

5.5 LA PRESA DEI PEZZI

Un ultima parte fondamentale per il gioco è la movimentazione delle pedine tramite il robot.

Oltre a dover effettuare i corretti spostamenti, YuMi deve riuscire ad afferrare con elevata precisione e ripetibilità tutti i pezzi degli scacchi.

Come già visto per il loro riconoscimento tramite immagini, i pezzi sono molto differenti tra loro nella forma e nell'altezza e in particolare il cavallo presenta anche un andamento non assialsimmetrico.

È possibile afferrare i pezzi nella loro parte inferiore circolare ma le dita standard in dotazione con il gripper di YuMi non sono adeguate per il compito.

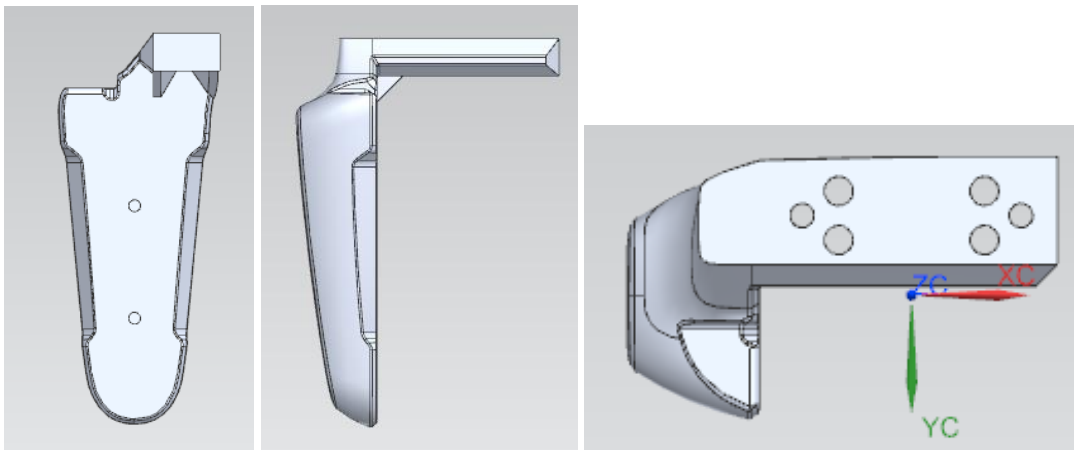

Figura 5.25: Modello CAD delle dita in dotazione

Utilizzando i disegni disponibili sul manuale (riportati in figura 4.7) è possibile progettare delle dita sostitutive adatte al compito da svolgere.

Le caratteristiche da considerare sono le seguenti:

- La presa dei pezzi deve avvenire in modo stabile senza urtare o spostare le altre pedine posizionate nel gioco. È consigliabile quindi afferrare i pezzi con la pinza posta sulla diagonale dello scacco, che ha una misura di circa 6cm.
- L'altezza dei pezzi è variabile tra 5cm per i pedoni fino ad un massimo di 9,5cm per le regine;
- La pinza in configurazione chiusa deve riuscire ad afferrare saldamente la base circolare dei pezzi.

È stato sviluppato un primo modello di base tramite il software Siemens NX: le sue caratteristiche rispondono ai requisiti descritti. Le caratteristiche strutturali sono da valutare dopo la stampa.

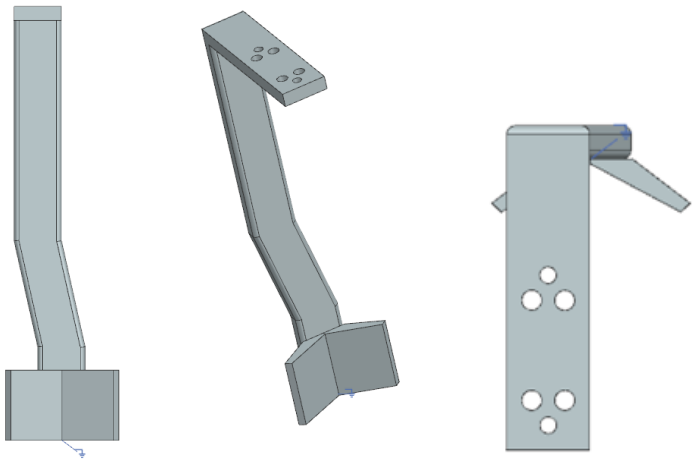

Figura 5.26: Modello CAD per le dita sostitutive

CONCLUSIONI

In questa tesi di laurea magistrale è stata sviluppata una applicazione interattiva che permette lo svolgimento di una partita di scacchi contro il robot YuMi.

Il progetto è un esempio dei risultati raggiungibili grazie alle nuove tecnologie sviluppate nel contesto di industria 4.0, e dimostra che il loro utilizzo si estende anche fuori dall'ambiente industriale.

- Grazie alla visione artificiale è stato possibile trasferire le posizioni e le mosse del gioco su adeguate strutture dati: i principali problemi riscontrati sono stati quelli relativi all'illuminazione. Con numerose prove sono stati messi in pratica i concetti di computer vision, programmazione e analisi di immagini studiati nei corsi teorici durante il percorso di studi ed è stata confermata la necessità di una illuminazione strutturata per il compito da eseguire. Per il corretto funzionamento del sistema infatti è necessario evitare forti luci dirette sul tavolo di gioco oppure condizioni di scarsa illuminazione.

- Il trasferimento del robot ha impedito la programmazione dei compiti sulla macchina fisica, ma il metodo utilizzato ha messo in evidenza i vantaggi e le possibilità offerte dalla programmazione offline. È stato possibile programmare i task in assenza del robot e, siccome il codice sviluppato in Rapid è lo stesso che viene trasferito al controller reale, collegando il vero robot al sistema le movimentazioni sono già disponibili;

- Vista la complessità del gioco degli scacchi, sono possibili alcune modifiche ed estensioni al lavoro già svolto:
  tramite il riconoscimento univoco di tutti i pezzi in gioco, sarebbe possibile gestire mosse speciali come l'arrocco oppure utilizzare queste informazioni ad una per ottenere una interfaccia grafica più ampia e dettagliata.
  Utilizzando più algoritmi di intelligenza artificiale si potrebbe far selezionare all' utente il livello di difficoltà del gioco.
  Infine, la pressione dello spazio può essere utilizzata anche per inserire il tempo all'interno della partita funzionando come un classico cronometro.

RIFERIMENTI

[1] https://www.diism.univpm.it/attività-ricerca/meccatronica

[2] Mazzuto G., Dispense del corso “Smart factories”, UNIVPM, 2021

[3] Colgate, J.E.; Edward, J.; Peshkin, M.A.; Wannasuphoprasit, W. “Cobots: Robots for Collaboration with Human Operators.” In Proceedings of the 1996 ASME International Mechanical Engineering Congress and Exposition, Atlanta, GA, USA, 17–22 November 1996;

[4] Faccio, M., Bottin, M. & Rosati, G. “Collaborative and traditional robotic assembly: a model comparison.” Int J Adv Manuf Technol 102, 1355–1372 (2019).

[5] Gaskill, S.; Andato, S. “Safety issues in modern applications of robots.” Reliability Engineering & System Safety 1996, 53, 301–307.

[6] “Research and Markets”, Guinnes Centre, Taylors Lane, Ireland

[7] Callegari M., Dispense del corso “Robotica industriale”, UNIVPM, 2021

[8] https://www.iso.org/obp

[9] https://www.ibm.com/it-it/topics/computer-vision

[10] Bhatt, Dulari, et al. "CNN variants for computer vision: History, architecture, application, challenges and future scope." Electronics 10.20 (2021): 2470.

[11] Castellini P., Dispense del corso “Sistemi di misura e visione”, UNIVPM, 2021

[12] https://www.adobe.com/it/creativecloud/photography/discover/exposure-in-photography.html

[13] Chiodi M., “*Progettazione di un carrello mobile per il robot ABB YuMi*”, tesi di laurea triennale in ingegneria meccanica, UNIVPM, A.A. 2018/2019

[14] ABB, “Operating Manual – IRB 14000”

[15] ABB, “Product manual – IRB 14000 gripper”

[16] https://new.abb.com/products/robotics/robotstudio

[17] Emiliani F., “*Sviluppo di un’applicazione dimostrativa delle potenzialità di robot collaborativi, intelligenza artificiale e sistemi di visione*”, tesi di laurea magistrale in ingegneria meccanica, UNIVPM, A.A. 2021/2022

[18] https://www.intelrealsense.com

[19] Intel, “*Intel RealSense Camera 400 Series Product Family Datasheet*”

[20] https://www.tiobe.com/tiobe-index/python/

[21] https://pyimagesearch.com/2020/12/21/detecting-aruco-markers-with-opencv-and-python/

[22] https://opencv.org/

[23] https://numpy.org/

[24] https://github.com/thomasahle/sunfish

[25] ABB, “Technical reference manual - RAPID”

[26] https://www.i-labs.it/it

[27] Alessandrini M., Dispense del corso “Elementi di elettronica per l’ingegneria meccatronica”, UNIVPM, 2020

[28] ABB, “Application manual – MultiMove”