DOTTORATO DI RICERCA
in
SCIENZE COMPUTAZIONALI E INFORMATICHE
Ciclo XIX

Consorzio tra Università di Catania, Università di Napoli Federico II,
Seconda Università di Napoli, Università di Palermo, Università di Salerno

SEDE AMMINISTRATIVA: UNIVERSITÀ DI NAPOLI FEDERICO II

Paolo Coraggio

Interazione Uomo-Robot Finalizzata alla Cooperazione

TESI DI DOTTORATO DI RICERCA

IL COORDINATORE
Prof. Luigi M. Ricciardi

A Michele,
mio figlio

Indice

INTRODUZIONE

Nel corso dell'ultimo decennio la ricerca robotica ha registrato una notevole espansione dovuta a vari fattori tra cui: la disponibilità di nuovi congegni di rilevazione (elementi necessari dell'apparato sensoristico dei robot) sempre più affidabili, precisi ed economici; lo sviluppo di capacità di calcolo sempre più potenti e veloci; un'attenzione crescente da parte di mass-media e opinione pubblica. Questo ultimo fatto, sebbene non si tratti di un fenomeno nuovo nell'ambito delle scienze in generale e della robotica in particolare, può essere esaminato sotto diversi aspetti. La robotica ha da sempre affascinato l'immaginario collettivo e, negli ultimi anni, nell'opinione pubblica si sta rafforzando l'idea che le prestazioni delle applicazioni reali stiano, in qualche modo, raggiungendo quelle aspettative che fino a ieri erano delle chimere. Il grande pubblico sta considerando sempre di più come possibile (e in alcuni casi auspicabile) l'introduzione di robot nella vita quotidiana degli esseri umani: come partner in ambito lavorativo, come ausilio nella riabilitazione e assistenza ai disabili, o, addirittura, come "compagni di giochi" nel tempo libero. Infine, altro effetto collaterale, l'attenzione del pubblico verso i robot ha fatto convergere in questo campo l'interesse di ricercatori di settori scientifici diversi.

Il lavoro di questa tesi ha come obiettivo lo studio dell'interazione uomo-robot finalizzata alla cooperazione: in particolare, ci si è limitati al campo dell'interazione fisica tra un robot e un utente umano in un ambito lavorativo industriale. Dal punto di vista storico, il settore industriale è stato uno dei primi nel quale sono stati introdotti sistemi automatici che sostituissero gli esseri uomini nei compiti più usuranti, ripetitivi e/o pericolosi; ciononostante, l'autonomia del robot, intesa come la sua capacità di prendere decisioni ed eseguire azioni senza l'intervento diretto di un essere umano, in questo settore è sempre stata pressoché inesistente. Le considerazioni circa la sicurezza degli utenti umani hanno portato a una netta divisione dello spazio operativo delle macchine da quello degli utilizzatori dei sistema.

Lo studio delle caratteristiche relative all'autonomia è stato, invece, portato avanti in altri ambiti di ricerca, come ad esempio l'informatica e l'intelligenza artificiale, con diversi campi applicativi come, ad esempio, la *robotica di servizio*. La differenza sostanziale è che, in questo ultimo campo, il possibile contatto fisico tra robot ed essere umano ha di solito conseguenze non

disastrose. La ricerca sulla robotica autonoma, iniziata verso la fine degli anni '70, ha prodotto risultati apprezzabili tanto dal punto di vista teorico, con l'introduzione di metodologie per il governo dei sistemi robotici, che da quello applicativo, con una serie di robot che hanno raggiunto un mercato non solo limitato al campo della ricerca scientifica.

Lo sviluppo tecnologico degli ultimi anni permette di continuare ad approfondire temi di ricerca in robotica autonoma che servano a migliorare l'interazione tra i robot e gli esseri umani, cercando nuovi approcci alla progettazione di questi sistemi. Inoltre, oggi si assiste al tentativo di introdurre in ambito industriale concetti già consolidati in questo settore, per rendere, finalmente, possibile un contatto tra robot e utente umano.

L'interazione uomo-robot è, così, un settore di ricerca robotica nuovo, con molte possibilità di applicazioni. Il contatto tra uomo è macchina, però, deve essere "sicuro" (deve cioè salvaguardare l'incolumità dell'utente), e "fattivo" (ossia realmente utile allo scopo prefissato). L'ambito industriale si presta in maniera particolarmente adatta a ricerche sull'interazione sicura, in quanto l'uso di strutture robotiche particolarmente pesanti, e dunque pericolose, ne accentua le caratteristiche.

Il contenuto di questa tesi è stato organizzato in 4 capitoli e 2 appendici

Nel capitolo 1 si fornisce una rassegna delle architetture per progettare e implementare i sistemi per il governo di robot (visti come agenti autonomi) che siano in grado di svolgere compiti senza l'intervento diretto degli esseri umani che li hanno costruiti. Tale panoramica include le architetture robotiche che nel corso degli anni sono state proposte in letteratura, e i metodi di progettazione più usati.

Nel capitolo 2 si introduce il problema dell'interazione uomo-robot (o *Human-Robot Interaction – HRI*). Una cooperazione tra robot e utente umano può infatti, in alcune fasi dello svolgimento di un determinato task comune, prevedere un loro contatto fisico: in questi casi si parla specificamente dell'interazione "fisica" uomo-robot. Si discute, quindi, il problema di come un agente autonomo artificiale (il robot) possa entrare in contatto con un essere umano in tutta sicurezza. Per fare questo, risulterà indispensabile progettare il sistema di governo del robot partendo innanzitutto dalle caratteristiche dell'utente con il quale il robot entra in contatto. Sarà quindi cruciale un'appropriata modellazione, nell'ordine, dell'utente, dell'interazione e infine dell'ambiente dove essi cooperano.

Il capitolo 3 è dedicato alla descrizione di un compito visuomotorio generale in ambito industriale. Si individuano, quindi, i modelli minimi più appropriati per un'interazione sicura: sarà descritto il modo in cui è possibile modellare, in maniera efficace, i task e saranno descritti gli algoritmi individuati per la modellazione dell'utente (soprattutto gli algoritmi di visione). Se i sensori esterni al robot sono deputati per la modellazione dell'utente e dell'ambiente, quelli interni permetteranno una modellazione del robot tale da permettere l'interazione sicura. Si descriveranno, infine, i principi del progetto PAVA (Percezione Attesa & Visione Attiva) nel quale un compito di *HRI* fisica è stato eseguito da manipolatore industriale reale.

Infine, nel capitolo 4, attraverso la spiegazione dell'algoritmo per il governo del robot, è descritta l'effettiva integrazione visuomotoria. È, inoltre, introdotto il set-up sperimentale utilizzato e le procedure seguite negli esperimenti effettuati.

CAPITOLO 1

Le architetture robotiche

Il vecchio Rossum, grande filosofo, [...] cercò di imitare
con una sintesi chimica la sostanza viva detta protoplasma
finché un bel giorno scoprì una sostanza il cui comportamento
era del tutto uguale a quello della sostanza viva
sebbene presentasse una differente composizione chimica,
era l'anno 1932
Karel Capec – R.U.R, 1920.

1.1. Introduzione

È il 1920 quando lo scrittore ceco Karel Capek utilizza, per la prima volta, il termine "robot" all'interno della sua opera drammatica R.U.R. – Rossum's Universal Robots. Questo termine, suggeritagli dal fratello Josef (scrittore e pittore cubista che in un suo racconto del 1917, *Opilec* – l'ubriacone – aveva introdotto i personaggi di R.U.R. chiamandoli, però, "automa"), deriva dal ceco *robota* che significa "lavoro forzato" e fu utilizzata per indicare degli esseri nati per liberare l'umanità dalla schiavitù della fatica fisica. Sebbene nell'opera originale un robot era un essere organico artificiale, da allora tale termine ha indicato un artefatto fisico. Non è difficile riconoscere nelle pagine di Capek, che visse a Praga negli anni del suo lavoro di giornalista e scrittore, l'influsso di un'altra figura leggendaria: il Golem, il gigante di argilla buono e obbediente nato per difendere il popolo ebraico dai suoi persecutori. La caratteristica comune ai robot di Capek e al Golem è l'uso che fanno della loro forza fisica al servizio degli esseri umani.

Di poco precedenti o coevi all'opera di Capek sono comunque i tentativi di studiare macchine che fossero dotate di capacità di autoregolazione e autocontrollo paragonabili a quelle

proprie degli organismi viventi [Cordeschi, 1998]. Al di là della pura forza fisica, dunque, un robot ha cominciato ad assumere, o meglio si è cercato che un robot assumesse, caratteristiche decisionali più evolute. Secondo l'Associazione dell'Industria della Robotica (RIA), "un robot è un manipolatore riprogrammabile, multi-funzionale, progettato per trasportare materiale, parti, attrezzi, o apparecchiature specializzate attraverso movimenti variabili programmati e per compiere una varietà di compiti" [Jablonsky e Posey, 1985].

In prima istanza, la robotica può essere intesa come il collegamento intelligente di percezione e azione, e un robot come una macchina capace di estrarre informazioni dal suo ambiente e usare la conoscenza sul suo mondo per muoversi in sicurezza in una maniera significativa e mirata. Come progettare tale macchina è il primo, fondamentale passo per la sua realizzazione.

I questo capitolo si darà una definizione di cosa sia un'architettura robotica, di quali problemi, il suo studio, si pone e come risolverli. Segue (paragrafi 1.2.1 e 1.2.2) una breve panoramica delle prime architetture proposte in letteratura, discutendo le metodologie proposte e i limiti che esse avevano. Sono poi discusse le architetture cosiddette *behavior-based*. Tali architetture presentano spunti e soluzioni interessanti per la realizzazione di un sistema di governo del robot che potranno essere utilizzate, in seguito, nella realizzazione di un sistema di interazione uomo-robot.

1.2. Le architetture robotiche

Un robot può essere visto come un artefatto fisico dotato di sensori, per percepire alcune caratteristiche dell'ambiente nel quale è immerso, e di attuatori per muoversi in esso e modificarlo. Come progettare le connessioni tra sensori e attuatori, come gestire un sistema di controllo affinché un tale sistema complesso pianifichi ed esegua le azioni più appropriate, è la base dello studio delle cosiddette "architetture robotiche".

Una definizione di architettura robotica la si può mutuare da quella che Stone [Stone, 1980] aveva proposto per l'architettura di un calcolatore:

"*Computer architecture is the discipline devoted to the design of highly specific and individual computers from a collection of common building blocks*"

che, in questo ambito, potrebbe essere tradotta come: "un'architettura robotica è ciò che è inerente alla progettazione di robot individuali e altamente specifici partendo da un insieme di componenti costituenti software comuni". Secondo Hayes-Roth [Hayes-Roth, 1995], con il termine "architettura" ci si riferisce alla "progettazione astratta di una classe di agenti; più

specificatamente dell'insieme di componenti strutturali nei quali avvengono la percezione, il ragionamento e le azioni; delle funzionalità specifiche di ogni componente e della loro interconnessione topologica". Mataric [Mataric, 1992] dà una definizione alternativa: "un'architettura fornisce i principi di base per organizzare un sistema di controllo. Oltre a definire la sua struttura, un'architettura deve indicare i limiti entro i quali risolvere i suoi problemi".

Nel corso degli anni sono state proposte diverse tecniche e approcci per progettare e costruire il sistema di governo di un robot. In prima istanza, tali tecniche si possono suddividere, storicamente e metodologicamente, in due categorie: deliberative e reattive. Le prime richiedono essenzialmente una conoscenza (relativamente) completa del mondo per poi utilizzarla per predire gli effetti delle azioni del robot; le seconde si basano su un accoppiamento (quasi) diretto tra informazione ricevuta dai sensori e azione corrispondente degli attuatori.

1.2.1. Il paradigma deliberativo/gerarchico

Le architetture che si basano sul paradigma deliberativo/gerarchico nascono storicamente dai risultati ottenuti nell'ambito delle ricerche in Intelligenza Artificiale (AI) e hanno come peculiarità principale l'uso di metodi di ragionamento deliberativo. Tali sistemi hanno spesso caratteristiche comuni tra le quali:

- sono gerarchiche nella struttura con una suddivisione abbastanza netta e identificabile delle funzionalità (simile a certi controlli gerarchici delle aziende);
- la comunicazione e il controllo avviene in maniera predeterminata e prevedibile, salendo e scendendo nella gerarchia;
- livelli più alti nella gerarchia forniscono sub goal per livelli subordinati inferiori;
- lo scopo della pianificazione, sia dal punto di vista spaziale che temporale, cambia scendendo di livello nella gerarchia. I tempi di elaborazione divengono minori e le considerazioni spaziali divengono vieppiù locali nei livelli inferiori;
- si basano fondamentalmente su una rappresentazione simbolica del modello del mondo.

Un robot deliberativo in primo luogo, e in maniera sequenziale, percepisce con i sensori l'ambiente in cui è immerso e ne costruisce un modello. Dopo, senza tener conto di altri dati sensoriali che nel frattempo potrebbero giungergli ed essere elaborati, pianifica tutte le direttive che gli attuatori devono eseguire per raggiungere il goal e comincia ad agire seguendo la prima direttiva. A conclusione del primo ciclo *sense-plan-act*, il robot raccoglie nuove informazioni dai sensori che includono anche le conseguenze della sua azione, ripianifica e agisce di nuovo (a volte anche senza far "fisicamente" nulla). Le informazioni fornite dai sensori vengono immagazzinate in una struttura dati (il *modello del mondo*) per poi essere elaborate. Nel modello del mondo del paradigma gerarchico sono contenuti: a) una rappresentazione a priori del mondo in cui il robot è immerso; b) le informazioni sensoriali; c) ogni altra ulteriore informazione di tipo "cognitivo" (ad esempio l'obiettivo di alto livello da raggiungere). Un motore inferenziale (di solito un dimostratore automatico di teoremi) rappresenta il sistema di controllo del robot: sulla base del modello del mondo, si inferisce la migliore azione da eseguire. Basandosi sul paradigma *sense-plan-act*, il controllo di un robot deliberativo richiede forti ipotesi di fondo sul modello del mondo; innanzitutto che la conoscenza sulla quale inferire sia consistente, sempre disponibile e certa. Se l'informazione a disposizione è inaccurata o è cambiata dall'istante in cui è stata ottenuta, il risultato del processo deliberativo può essere erroneo. In un mondo dinamico, dove gli oggetti si muovono in maniera arbitraria, è potenzialmente pericoloso basarsi su informazioni passate che potrebbero non essere più valide. I modelli che rappresentano il mondo sono costitutivi sia da una conoscenza a priori dell'ambiente, che dai dati sensoriali: tali modelli sono perciò il supporto per il ragionamento deliberativo del robot. Il pianificatore è l'elemento centrale per la costruzione del robot "deliberativo". Un piano è un qualsiasi processo gerarchico che controlla l'ordine nel quale una sequenza di operazioni viene eseguita [Miller et al., 1960]. Tra i pianificatori classici più conosciuti si può citare STRIPS [Fikes e Nilsson, 1971], un dimostratore automatico di teoremi che sfruttava la logica del primo ordine per sviluppare un pianificatore per la navigazione. A STRIPS fece seguito ABSTRIPS [Sacerdoti, 1974], il suo miglioramento, che utilizzava una gerarchia di spazi di astrazione per migliorare l'efficienza di STRIPS, raffinando i dettagli di un piano quando questi diventavano più significativi. HACKER [Sussman, 1975] era un sistema che eseguiva una ricerca in una libreria di procedure per proporre, alla fine, un piano. NOAH [Sacerdoti, 1975] era un pianificatore gerarchico la cui peculiarità era l'uso di una decomposizione del problema al fine di risolvere dei sottoproblemi, riordinandoli se necessario. Alcuni tra i robot costruiti all'epoca facevano uso di tali dimostratori o, comunque, dipendevano da una rappresentazione simbolica dell'ambiente circostante. Ad esempio, Shakey [Nilsson, 1969], costruito allo Stanford Research Institute, si basava su STRIPS. In questo caso il pianificatore

usava l'informazione contenuta in un modello simbolico del mondo per determinare quali azioni eseguire per raggiungere il proprio scopo in un determinato istante. HILARE [Giralt et al., 1984] adoperava un sistema di pianificazione che utilizzava uno spazio di rappresentazioni a più livelli: i modelli geometrici rappresentavano le distanze, mentre un modello relazionale esprimeva le connessioni tra stanze e corridoi. Stanford Cart [Moravec, 1977] era una piattaforma Robotica che utilizzava un sistema di visione stereo per la navigazione. Gli ostacoli riconosciuti dalla videocamera erano modellati da sfere che li racchiudevano ed erano immagazzinati in una mappa del mondo interna al sistema di controllo del robot. Stanford Cart utilizzava, poi, un algoritmo per la ricerca dei percorsi minimi.

Il problema maggiore della architetture deliberative è il cosiddetto *frame problem*. Il sistema di governo di un robot deliberativo deve essere in grado di modellare tutte, o quasi, le entità presenti nell'ambiente; dunque il numero di fatti (o assiomi) che il sistema inferenziale del robot deve considerare e manipolare diventa rapidamente intrattabile dal punto di vista computazionale per applicazioni reali. A ciò vanno aggiunti il cosiddetto problema della traduzione (ossia tradurre il mondo reale in una descrizione accurata e adeguata affinché possa essere utile e maneggevole) e il problema rappresentazione/ragionamento (cioè come rappresentare simbolicamente l'informazione sui processi che intercorrono tra entità complesse reali rendendo il robot capace di "ragionare" tramite questa informazione, così da fornire risultati utili in tempo reale [Wooldridge, 1999]).

1.2.2. Il paradigma reattivo

I problemi legati al paradigma dell'AI simbolica portarono i ricercatori a investigare ipotesi alternative che prevedevano di: scartare le rappresentazioni simboliche e i processi di generazione di decisioni basati sulla loro manipolazione sintattica; affermare l'idea che il comportamento intelligente e razionale sia strettamente legato all'ambiente nel quale l'agente si trova a operare – il comportamento intelligente non è avulso dal contesto ma è il risultato dell'interazione tra l'agente e il suo ambiente; ipotizzare che il comportamento intelligente si manifesti attraverso l'interazione di vari componenti basilari più semplici. Una delle voci più critiche nei riguardi dell'AI simbolica (e capostipite delle architetture reattive) è certamente Rodney Brooks. Nei suoi articoli [Brooks, 1991a] [Brooks, 1991b] Brooks propone tre tesi chiave:

- il comportamento intelligente può aver luogo senza che vi sia una rappresentazione esplicita di modelli del tipo proposto dall'AI simbolica;
- il comportamento intelligente può essere generato senza un ragionamento astratto esplicito del tipo proposto dall'AI simbolica;
- l'intelligenza è una proprietà *emergente* di alcuni sistemi complessi.

Brooks, inoltre, identifica altre due problematiche che guidano il suo lavoro e cioè la collocazione (*situatedeness*) e la personificazione (*embodiement*): 1) l'intelligenza "reale" è posizionata (collocata) nel mondo, non in sistemi slegati dal contesto come i dimostratori automatici di teoremi o i sistemi esperti; 2) il comportamento intelligente si impone come il risultato dell'interazione tra l'agente e il suo ambiente. In più, l'intelligenza è nello "sguardo dello spettatore", ossia non è più una proprietà isolata o innata.

Volendo riassumere, in una frase, le architetture reattive, queste rappresentano una tecnica che associa in maniera diretta la percezione e all'azione (tipicamente nel governo dei motori) per produrre delle risposte in tempo utile in ambienti dinamici e non strutturati. Le architetture reattive, come quelle a sussunzione che vedremo in seguito, usano una decomposizione orientata al compito da perseguire: si hanno una serie di moduli paralleli, eseguiti contemporaneamente, che perseguono task specifici. Come notato da vari ricercatori, tra i quali Brooks, i sistemi deliberativi possono essere caratterizzati attraverso una decomposizione in una struttura orizzontale (figura 1): il task viene suddiviso nell'organizzazione *sense-plan-act* e l'elaborazione procede da modulo a modulo seguendo un flusso orizzontale.

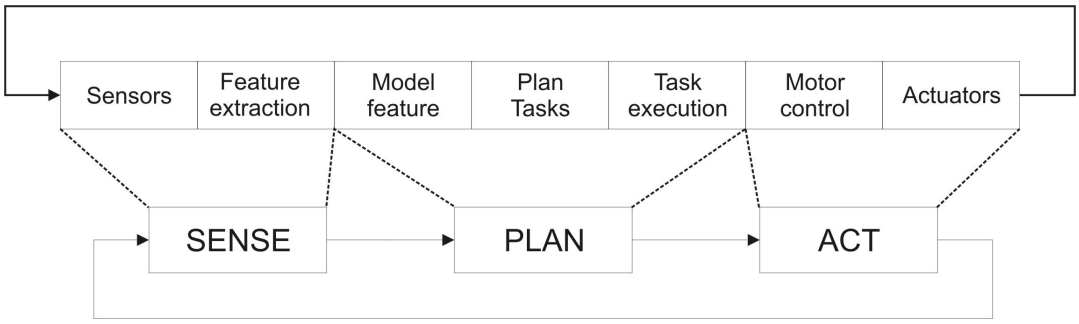


Fig. 1 – decomposizione dello schema classico deliberativo

Partendo da suggestioni etologiche, le architetture reattive si fondano sull'assioma che l'intelligenza (o il comportamento intelligente) è stratificata secondo una decomposizione verticale (figura 2). Il diagramma che visualizza le architetture reattive mette in risalto un

accoppiamento tra sensori e attuatori che, rispetto alle architetture deliberative, risulta già più funzionale dal punto di vista dei tempi di elaborazione. I vari moduli che caratterizzano il governo del robot devono servire come base per le azioni del robot mentre il suo comportamento complessivo è quello che emerge.

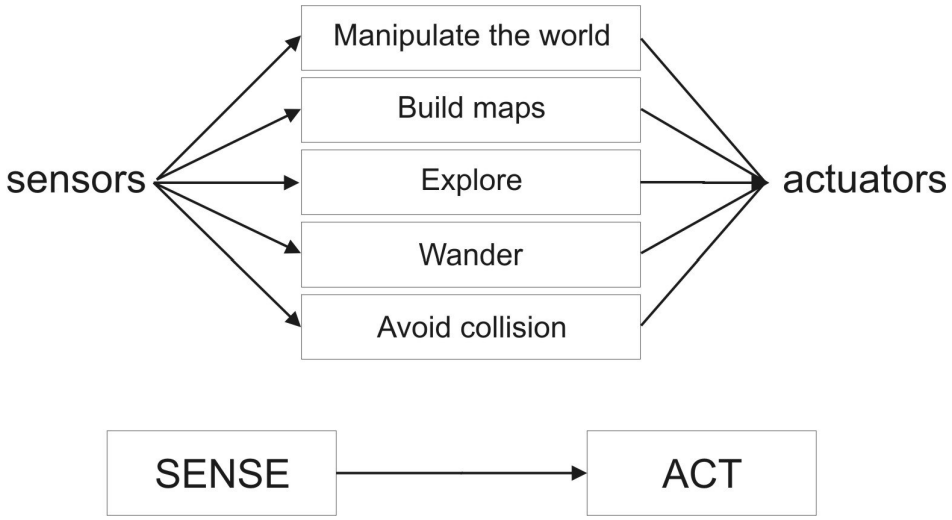


figura 2 – decomposizione verticale nelle architetture reattive

1.3. I Robot Behavior Based

1.3.1. Introduzione al termine *behavior*

Come detto in precedenza (cfr., paragrafo 1.1), la robotica può essere vista come la disciplina che si occupa del collegamento "intelligente" tra percezione e azione. Cosa sia l'intelligenza, come nasca e come si esprima è ancora una questione aperta ma, essendo un robot un agente che opera in un ambiente, ha senso parlare di un suo "comportamento intelligente". Non sorprende, dunque, che la comunità robotica si sia rivolta agli studi etologici per avere suggerimenti su come progettare e costruire macchine, artefatti che mostrino un comportamento cosiddetto "intelligente". Una definizione operativa di "intelligenza" – adatta a questo campo – potrebbe essere: una caratteristica di un sistema (biologico o meno) dotato di capacità di migliorare le sue possibilità di sopravvivenza nel mondo reale e di competere (o cooperare) con altri agenti a questo scopo.

Il termine *behavior* (comportamento), che in robotica inizia a circolare all'inizio degli anni '80, viene mutuato dagli studi psicologici precedenti a questi anni. Il comportamentismo (o *behaviourismo* o *psicologia comportamentale*) appare nella scena degli studi in psicologia nei primi anni

del XX secolo sviluppato da J. B. Watson [Watson, 1925] agli inizi del Novecento, basandosi sulla considerazione che il comportamento esplicito è l'unica unità di analisi scientificamente studiabile della psicologia. In robotica, invece, il termine "behavior" ha più spiegazioni intuitive che formali. La sua importanza deriva dall'assunto che se in natura si osserva un comportamento intelligente, questo può essere replicato in un robot[1]. Dunque i sistemi robotici possono essere studiati considerando il comportamento biologico dei sistemi animali. Infatti lo studio del comportamento animale può fornire modelli che un ricercatore può rendere operativi su un sistema robotico, partendo dalla capacità più semplice di un organismo vivente: percepire e agire nell'ambiente in maniera significativa e intenzionale.

Se molti ricercatori continuano a ispirarsi a considerazioni biologiche, altri le ignorano a vantaggio di soluzioni prettamente ingegneristiche. Le ragioni sono principalmente due. La prima è che l'hardware che sottende un sistema biologico e quello, invece, che forma il sistema decisionale di un robot sono profondamente diversi. I sistemi biologici sono il risultato di un bagaglio evoluzionistico che non è necessario (né, in taluni casi, possibile) nella costruzione di un robot. La seconda è che la conoscenza del funzionamento del sottostrato biologico è spesso ancora inadeguata per supportare la validità dei risultati raggiunti in un campo riportandoli nell'altro. Ciononostante, i ricercatori "comportamentali" affermano che per la robotica si possono raggiungere risultati importanti solo attraverso lo studio delle neuroscienze, la psicologia e l'etologia.

1.3.2. Il comportamento di un robot

Il comportamento degli esseri viventi serve spesso d'ispirazione per la progettazione dei robot. I sistemi così costruiti prendono il nome di "behavior based".

Forse il modo più semplice di definire un comportamento elementare di un robot è quello di adottare un concetto sostenuto nel campo della psicologia comportamentalista: un behavior è una reazione a uno stimolo. Questa visione pragmatica ci rende capaci di esprimere come un robot dovrebbe interagire con il suo ambiente. Aderendo a questa visione, però, ci si restringe al solo campo dei sistemi puramente reattivi.

[1] Sebbene sia un argomento di ricerca interessantissimo, non ci addentreremo in questioni su cosa significhi effettivamente replicare un comportamento intelligente piuttosto che "imitarlo" o "dotare un sistema di comportamento intelligente"

1.3.2.1. I sistemi reattivi

Nei sistemi reattivi i comportamenti servono come fondamenta dalle quali costruire le azioni del robot. Un behavior, in questi sistemi, consiste tipicamente di una semplice coppia sensomotoria dove l'attività sensoriale fornisce l'informazione necessaria per soddisfare l'applicabilità di una risposta particolare riflessiva a basso livello. Per generare le risposte motorie, viene quindi evitata qualsiasi rappresentazione astratta, esplicita della conoscenza. Un tale approccio si rileva particolarmente fruttuoso in ambienti altamente dinamici, imprevedibili e potenzialmente ostili per chi li abita. La costruzione di modelli astratti del mondo risulta un processo particolarmente dispendioso e troppo sensibile agli errori riducendo, così, la correttezza dell'azione potenziale di un robot nella maggior parte degli ambienti dove esso opera. I sistemi reattivi sono intrinsecamente modulari dal punto di vista della progettazione software. Ciò permette a chi li costruisce di espandere le capacità di un robot aggiungendo nuovi behavior senza dover, ogni volta, riprogettare il sistema.

Riassumendo, le linee guida di quasi tutte le architetture che seguono il paradigma reattivo sono:

1. i robot sono agenti situati che operano in una nicchia ecologica;
2. i behavior rappresentato i blocchi fondamentali per la generazione delle azioni del robot e il behavior complessivo è quello che emerge;
3. i behavior sono specifici e hanno una percezione ed elaborazione locale;
4. modelli di behavior animali sono spesso citati come una base per questi sistemi o per un particolare behavior.

1.3.3. Come esprimere i behavior

Esistono diversi metodi per esprimere formalmente i behavior di un robot. Tra i più interessanti proposti in letteratura ci sono i diagrammi stimolo-risposta (SR diagrams), la notazione funzionale, i diagrammi ad automi a stati finiti (FSA – Final State Automata).

I diagrammi SR sono il metodo più intuitivo ma meno formale per esprimere i behavior di un robot. Qualsiasi behavior può essere rappresentato come una risposta, calcolata da uno specifico behavior, generata da un dato stimolo. Quando si hanno più behavior differenti le loro uscite vengono raccolte da un modulo deputato alla coordinazione che produce, in ogni istante, la risposta appropriata, in termini di comandi motori, agli stimoli recepiti dai sensori. Se si usa una

notazione funzionale si possono utilizzare i metodi matematici per descrivere le relazioni tra ingresso e uscita del behavior:

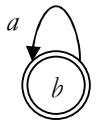
$$b(s) = r$$

che significa che il behavior *b* ha una risposta *r* allo stimolo *s*.

In un sistema puramente reattivo il tempo non è un argomento di *b*, dato che la risposta del behavior è istantanea e indipendente dalla storia temporale del sistema. Se si hanno più behavior che producono risposte differenti ci sarà una funzione di coordinazione che determinerà cosa fare con le uscite dei vari behavior. Attraverso la notazione funzionale risulta abbastanza semplice tradurre tale rappresentazione in un programma per il calcolatore, utilizzando, spesso, il LISP come linguaggio di programmazione.

1.3.3.1. Automi a Stati Finiti (FSA)

L'introduzione degli automi a Stati Finiti si rivela particolarmente utili quando si devono descrivere aggregati e sequenze di behavior [Arbib et al., 1981]. Questa metodologia rende esplicito il behavior attivo in un dato istante di tempo e la transizione tra behavior. Si rivelano poco utili, invece, quando si tratta di codificare un singolo, semplice behavior.



q	*input*	$\delta(q,input)$
b	*a*	*b*

figura 3 – FSA per un behavior "primitivo"

In fig. 3 il cerchio con *b* rappresenta lo stato dove il behavior *b* è attivo e accetta ogni stimolo. Il simbolo *a*, in questo caso, denota tutti gli input. Un automa a stati finiti M può essere definito da una quadrupla (Q, δ, q_0, F) dove Q rappresenta l'insieme di tutti gli stati (behavior) permessi; δ è la funzione di transizione che mette in relazione l'input e lo stato corrente con un altro stato del sistema, volendo anche con lo stato stesso; q_0 denota la configurazione del

behavior iniziale; F rappresenta l'insieme degli stati accettori, un sottoinsieme di Q, che indica il completamento del task sensomotorio. δ può essere rappresentata tramite una tabella dove gli archi rappresentano le transizioni di stato nell'FSA che vengono invocate dagli stimoli percettivi in ingresso.

I vantaggi degli FSA si apprezzano principalmente quando si deve specificare un sistema di controllo dei behavior particolarmente complesso dove interi insiemi di behavior primitivi si attivano e disattivano, mentre il robot soddisfa un obiettivo di alto livello [Arkin e MacKenzie, 1994]. Anche quando si vuole modellare il comportamento in sequenza di vari behavior, tale metodologia si rivela particolarmente fruttuosa [Gat e Dorais, 1994]. Gli FSA forniscono un meccanismo valido per esprimere le varie relazioni tra i diversi insiemi di behavior e hanno ricevuto un ampio consenso nella comunità robotica per esprimere i sistemi di controllo. Ad esempio, in MELDOG [Tachi e Komoriya, 1985] si usa una mappa di automi per catturare le azioni che dovrebbero essere eseguite in varie zone dell'ambiente. Gli FSA sono stati utilizzati per controllare robot con sistemi di navigazione basati sulla visione [Fok e Kabuka, 1991], [Tsai e Chen, 1986]. Gli automi a stati finiti sono stati utilizzati anche da Brooks [Brooks, 1986] per le sue architetture a sussunzione.

1.3.3.2. Codifica dei behavior

Per codificare la risposta di un behavior a uno stimolo percepito, si deve creare una correlazione funzionale tra stimolo (sensore) e controllo motorio (attuatore). Per costruire una tale correlazione bisogna definire, innanzitutto, le caratteristiche delle risposte motorie. In prima istanza, queste ultime possono essere divise in due componenti ortogonali: la *forza* e l'*orientazione*. La forza si riferisce al modulo della risposta motoria a un dato stimolo percettivo. Essa si può estrinsecare in termini di velocità o di forze esercitate dagli attuatori (si veda il caso di un manipolatore). La forza delle risposte può, in alcuni casi, essere indipendente dalla forza dello stimolo ma modulata da fattori o comportamenti esogeni quali le intenzioni del robot (ossia suoi possibili goal interni) o l'assuefazione allo stimolo (quanto spesso quel determinato stimolo è stato attivato). L'orientazione denota la direzione dell'azione, conseguenza dello stimolo (ad esempio allontanarsi da uno stimolo o avvicinarsi verso un attrattore). La realizzazione di questa

componente direzionale richiede comunque conoscenze sulla cinematica[2] del robot e potrebbe essere indipendente dalla forza dello stimolo.

La codifica di un behavior può essere, inoltre, discreta o continua. Nel caso di codifica discreta la funzione che mappa gli stimoli percettivi in risposte motorie consiste in un insieme finito di coppie <situazione, azione>. La risposta a un determinato stimolo può essere anche molto semplice (come un comando di stop), o più complessa (come l'elicitazione di una sequenza di azioni). Un'altra strategia che usa la codifica discreta è quella che fa uso di sistemi basati su regole. Qui la correlazione tra input e output è rappresentata da regole della forma if-then:

IF antecedente **THEN** conseguente

dove l'antecedente consiste in una lista di precondizioni che devono essere soddisfatte affinché la regola sia applicabile, mentre il conseguente è costituito dalle risposte motorie associate.

Se più di una regola è applicabile a una determinata situazione, bisognerà progettare una strategia di risoluzione di conflitti che, di solito, sceglie solo una delle regole applicabili, usando una funzione di valutazione appropriata. In letteratura, alcuni sistemi behavior based codificano i behavior con regole fuzzy [Saffiotti et al., 1993], [Goodridge e Luo, 1994]. Esempi di sistemi inferenziali che trattano i sistemi rule based sono Gapps [Kaelbling e Rosenschein, 1991], e il metodo teleoreattivo sviluppato da Nilsson [Nilsson,1994]. Per le architetture a sussunzione è stato proposto un Behavior Language [Brooks, 1990] che usava un approccio rule-based con una sintassi simile al LISP.

La codifica continua permette al robot di avere uno spazio infinito di potenziali reazioni al suo ambiente. Invece di avere un insieme numerabile di risposte che discretizzano il modo in cui il robot può muoversi (avanti, indietro, a sinistra, a destra, accelerando, rallentando…), una funzione matematica trasforma l'input sensoriale in una reazione del behavior. Uno dei metodi più conosciuti per implementare una risposta continua si basa su una tecnica detta metodo dei campi di potenziale. Khatib [Khatib, 1985] e Krog [Krog, 1984] svilupparono tale metodologia per la generazione di traiettorie regolari (*smooth*) sia per piattaforme robotiche mobili che per manipolatori. Tale metodo genera un campo che rappresenta uno spazio di navigazione che si basa su una funzione potenziale arbitraria. I punti dello spazio che rappresentano posizioni goal sono trattati come attrattori (generatori di campi attrattivi) mentre gli ostacoli sono visti come repulsori. Si costruiscono campi separati a seconda della funzione potenziale per rappresentare la

[2] Per *cinematica* di un robot si intende lo studio degli aspetti relativi alla sua posizione, velocità e accelerazione. Questa include, in particolare, tutte le proprietà fisiche del robot. La *dinamica* di un robot estende la sua cinematica e include lo studio delle forze prodotte dal suo moto.

relazione tra il robot e ogni oggetto presente nel campo d'azione del sensore. Questi campi vengono poi combinati usando, di solito, un principio di sovrapposizione degli effetti per ottenere un unico campo globale [Latombe, 1991]. Come rilevato in [Koren e Borenstein, 1991], i campi di potenziale presentano alcune limitazioni. Innanzitutto possono presentare dei minimi locali (posizioni dello spazio nelle quali il robot può rimanere "intrappolato") o far avere al robot dei comportamenti oscillatori ciclici. Per evitare questi problemi sono state proposte alcune soluzioni, tra le quali gli harmonic potential fields [Kim e Khosla, 1992] o i campi di potenziale time-varying [Tianmiao e Bo, 1992]. Arkin [Arkin, 1987] descrisse la possibilità di introdurre un rumore casuale per uscire dai minimi locali.

1.3.3.3. Assemblare i behavior

Una volta richiamati i metodi per descrivere i behavior individuali, illustriamo alcuni metodi per costruire sistemi che consistano di più behavior. Prima di discutere le tecniche proposte in letteratura, è bene esaminare brevemente una controversa questione che riguarda la nozione di "behavior emergente" (*emergent behavior*). La manifestazione di un dato comportamento viene spesso invocata in termini quasi mistici quando si parla di capacità dei sistemi behavior based. La possibilità che un behavior si manifesti ("emerga") implica una capacità olistica dove la somma è considerevolmente più grande delle parti che la formano. Spesso accade che ciò che avviene in un sistema behavior based si riveli una sorpresa per chi progetta il sistema stesso; ma la fonte di tale sorpresa è da ricercare in un'analisi insufficiente dei behavior che costituiscono i blocchi fondamentali, e della loro coordinazione, o in qualcos'altro? La nozione di "emergenza" come fenomeno mistico deve essere abbandonata, ma il concetto in termini precisamente definiti può risultare ancora utile. Numerosi ricercatori hanno discusso il termine *emergenza* sotto vari aspetti: secondo Moravec [Moravec, 1988] la manifestazione di un behavior è l'apparizione di una nuova proprietà del sistema visto nel suo insieme. Steels [Steels, 1990] afferma che "una funzionalità globale emerge dall'interazione parallela di behavior locali". Per Brooks [Brooks, 1991a] "l'intelligenza emerge dall'interazione dei componenti del sistema" (dove le funzionalità del sistema, ossia la capacità di pianificazione, la percezione, la mobilità etc., sono il risultato delle componenti che generano il behavior). Per McFarland e Bosser [McFarland e Bosser, 1993] "la funzionalità emergente si estrinseca attraverso l'interazione tra componenti che non sono state progettate per quella specifica funzione particolare". L'elemento comune tra le varie correnti di pensiero è che l'emergenza è una proprietà di una collezione di componenti che interagiscono (in

questo caso i behavior). Ma questo non risponde a un'altra domanda: perché una collezione coordinata dei behavior individuali, ben definiti dal punto di vista funzionale, dovrebbe dunque produrre dei risultati nuovi o non predetti? La risposta a questo quesito potrebbe essere che, per quanto si possa progettare a priori un robot con tutti i suoi behavior, l'ambiente nel quale è immerso ha troppe variabili che non possono essere predette e/o modellate in precedenza. A questo si può aggiungere un altro fattore rilevante: un robot è dotato di sensori: oggetti per il rilevamento di misure fisiche e quindi, intrinsecamente, affette da errori non sempre prevedibili e modellabili. Più l'ambiente diventa complesso, maggiore è il numero di sensori e attuatori, anche essi affetti da errore, utilizzati, più diventa difficile prevedere e controllare il comportamento di un robot.

Le funzioni di coordinazione sono algoritmi che non posseggono alcuna "proprietà magica" e da cui non ci si aspetta alcuna sorpresa. In alcuni casi esse sono particolarmente semplici e dirette come, ad esempio, scegliere il behavior posto al primo posto di una determinata graduatoria o quello più dominante; in altri possono risultare più complesse, considerando la fusione o l'attivazione di più behavior. Al di là di questo, sono generalmente funzioni deterministiche e certamente calcolabili. Perchè allora l'emergenza, intesa come caratteristica di un sistema behavior based, si manifesta quando queste operano in un ambiente reale? Perché è difficile predire *esattamente* un behavior? La risposta a questa domanda è contenuta non tanto nel robot quanto nella sua relazione con l'ambiente. Per la maggior parte delle situazioni nelle quasi si applica il paradigma behavior based è il mondo, l'ambiente che non offre alla possibilità di avere una sua modellazione analitica. Il mondo reale è incerto e dinamico. Inoltre, se non sopratutto, il processo percettivo è per sua natura "corrotto": non esistono modelli dei sensori validi per mondi aperti. Se fosse possibile costruire un modello del mondo che catturi tutte le sue proprietà, allora il concetto di emergenza (manifestazione) non avrebbe senso: si potrebbero fare predizioni accurate. Riassumendo, si può concludere che le proprietà "emergenti" dei sistemi behavior based sono una conseguenza della complessità del mondo nel quale l'agente robotico risiede, con in più la difficoltà di percepirlo in maniera precisa.

Si possono individuare due metodologie principali per modellare la funzione di coordinazione: i metodi competitivi e quelli cooperativi. Entrambi propongono diverse strategie di realizzazione.

1.3.3.3.1. Metodi competitivi

Un conflitto nel sistema decisionale del robot può avere luogo quando due o più behavior sono attivi e ognuno fornisce indipendentemente la sua risposta. I metodi competitivi propongono un modo di coordinare le risposte e quindi risolvere i conflitti. La coordinazione può essere spesso vista come una rete cosiddetta "winner-take-all" nella quale la singola risposta del behavior "vincente" è l'unica presa in considerazione dal robot quando questi esegue un'azione. Questo tipo di strategia competitiva può essere attuata in modi diversi.

La funzione di coordinazione ad "arbitrato" è una tecnica che prevede, in uscita, una singola risposta comportamentale. Tale funzione può assumere la forma di una struttura con priorità fissa nella quale esiste una stretta gerarchia tra i behavior, tipicamente attraverso l'uso di meccanismi detti di soppressione e inibizione utilizzati a cascata. Questa è, ad esempio, la caratteristica delle architetture a sussunzione [Brooks, 1986]. Anche i metodi di action-selection [Maes, 1990] utilizzano una procedura definita a priori dell'output di un singolo behavior, ma questa avviene in maniera meno autocratica. Qui i behavior competono in maniera attiva l'uno contro l'altro attraverso l'uso di livelli di attivazione guidati sia dagli obiettivi dell'agente, sia dalle informazioni sensoriali che provengono dall'ambiente. Non si stabilisce nessuna stretta gerarchia a priori: piuttosto i behavior competono per il controllo in ogni istante di tempo. La decisione su quale behavior prevalga avviene a *run time* quando si sceglie quello "più attivo", senza che esista una gerarchia preesistente. Un altro metodo competitivo prevede un criterio di votazione per i behavior, eseguendo l'azione che ha ricevuto più voti [Rosenblat e Payton, 1989]. Questa tecnica è stata usata in un'architettura distribuita per un robot con compiti di navigazione [Rosenblatt, 1995]. In questo caso, invece di avere ogni behavior codificato come un insieme di risposte basate su un insieme di regole, ogni behavior assegna un numero di voti attraverso un insieme predefinito di risposte motorie discrete. Per un compito di navigazione di un veicolo terrestre autonomo, l'insieme di risposte comportamentali per azioni che riguardano il comando dello sterzo è costituito da {[hard-left, soft-left, straight-ahead, soft-right, hard-right]}. Ogni behavior attivo {goal-seeking, obstacle-avoidance, road-following, cross-country, teleoperation, map-based navigation} ha un dato numero di voti (g_i) e una distribuzione costruita dall'utente per allocare questi voti. Si ha una funzione di scelta che, attraverso una strategia del tipo "winner-take-all", sceglie l'azione che ha ricevuto più voti. In questo modo, si ottiene un livello di cooperazione dei behavior, anche se il metodo è ancora arbitrario nella scelta della singola risposta.

1.3.3.3.2. Metodi cooperativi

I metodi cooperativi forniscono un'alternativa ai metodi competitivi. In questo caso si prevede una fusione dei comportamenti, che può avvenire attraverso modalità differenti, e che fornisce al sistema di governo del robot la capacità di usare in maniera concorrente l'output di più di un behavior nello stesso istante.

La questione centrale nel combinare gli output dei behavior è trovare una rappresentazione che lo permetta. Il metodo dei campi di potenziale fornisce uno dei formalismi più adatti. Come detto in precedenza, il metodo più diretto è quello di operare una somma vettoriale. Se inoltre si introduce un parametro f_i che misura il guadagno[3] di un determinato behavior, lo si può utilizzare come moltiplicatore del modulo del vettore relativo a quel behavior.

In altri casi [Payton et al., 1992] la combinazione dei vari behavior viene effettuata evitando l'uso di singoli valori di risposta discreti. Le risposte di ogni behavior vincolano invece le variabili di controllo in questo modo:

- *Zona*: definizione di limiti inferiore e superiore per una variabile di controllo;
- *Spike*: la designazione di un singolo valore che risulta da un arbitrariato standard basato su priorità;
- *Clamp*: definizione di limiti inferiore e superiore per una variabile di controllo.

In definitiva, il sistema fornisce i vincoli nei quali deve operare il sistema di controllo per soddisfare le necessità dei behavior.

Un altro metodo [Saffiotti et al, 1995] fa uso di metodi formali, assegnando a ogni behavior una funzione che può essere combinata utilizzando operatori logici specializzati (chiamati *t-norms*), definendo così una sorta di logica a più valori. La singola azione da espletare viene scelta da un insieme di azioni create mescolando le azioni primitive pesate.

[3] Il guadagno è, in questo caso, un peso (assegnato a-priori o determinato dinamicamente) associato a un behavior che ne esprime l'importanza nelle diverse fasi del funzionamento del robot

1.3.4. Le Architetture Behavior Based

Nei paragrafi precedenti si è visto come esprimere, codificare e coordinare i behavior. Per progettare e costruire dei sistemi robotici behavior based, bisogna imporre dei vincoli ai metodi specifici visti in precedenza. Le architetture robotiche sono, in definitiva, sistemi e specifici software che forniscono linguaggi e strumenti per la costruzione di sistemi behavior based.

Le architetture behavior based condividono alcune particolarità comuni:

- enfasi sull'importanza di accoppiare le capacità percettive con quelle attuative;
- mancanza di una rappresentazione simbolica della conoscenza;
- decomposizione in unità contestuali significative (behavior o coppie situazione-azione).

Sebbene tali architetture condividano apparentemente una tale filosofia comune, esistono profonde differenza tra loro che includono:

- granularità della decomposizione dei behavior;
- la base della specificazione dei behavior (etologica, sperimentale, etc.);
- il metodo di codifica della risposta (discreto o continuo);
- il metodo di coordinazione utilizzato (competitivo o cooperativo);
- i metodi di programmazione; il supporto di linguaggi disponibili; la riusabilità del codice.

Dato che nel corso degli anni sono state proposte varie architetture robotiche, ci si può chiedere in cosa differiscono e perché un'architettura può essere migliore di un'altra. Le architetture sono costruite a partire da componenti che possono essere di natura diversa. I modi in cui connettere tali componenti può facilitare, in date circostanze, alcuni tipi di progettazione robotica rispetto ad altri. Tutte le architetture behavior based sono essenzialmente linguaggi software o framework che specificano e controllano il robot. Se possono variare i livelli di astrazione, la potenza rispetto alla "calcolabilità" del problema rimane immutata. Ciò significa che ogni architettura proposta finora ha trovato una sua nicchia o dominio di applicazione specifico.

I sistemi behavior based rappresentano una valida soluzione quando l'ambiente reale nel quale è immerso il robot non può essere caratterizzato o modellato in maniera accurata. Se, ipoteticamente, si riuscisse ad avere un ambiente (e, soprattutto, dei sensori) modellabile senza errori, statico, i sistemi puramente behavior based non sarebbero necessariamente la soluzione migliore rispetto, ad esempio, a soluzioni di tipo gerarchico/deliberativo. Nonostante gli sforzi, le incertezze di misura e di interpretazione non possono essere rimosse. Inoltre, più l'ambiente cambia quando il robot agisce più il valore di un qualsiasi piano generato a priori diminuisce e la rappresentazione della conoscenza immagazzinata in un determinato istante di tempo diviene poco significativa.

Descriviamo due esempi di architetture behavior based descrivendo brevemente quelle a sussunzione mentre si analizzerà più dettagliatamente i sistemi basati sul cosiddetto schema theory.

1.3.4.1. Architetture a Sussunzione

Nate come risposta al paradigma gerarchico e alla costruzione di robot come Shakey che seguivano quel paradigma, le architetture a sussunzione si basano prevalentemente sui seguenti principi enunciati da Brooks:

- un comportamento complesso può non essere necessariamente il prodotto di un sistema di controllo complesso;
- l'intelligenza è "nell'occhio di chi osserva"
- il mondo è il miglior modello di se stesso;
- la semplicità è una virtù;
- i robot devono essere economici;
- la robustezza rispetto alla presenza di rumore o di sensori con prestazioni che degradano è un obiettivo del progettista;
- la pianificazione è solo un modo per evitare di immaginare cosa fare dopo;
- è importante avere tutta la potenza di calcolo su dispositivi *on board* al robot;
- i sistemi dovrebbero essere costruiti in maniera incrementale;

- nessuna rappresentazione, nessuna calibrazione, nessuna potenza di calcolo particolarmente elevata, nessuna comunicazione con una larghezza di banda eccessiva.

I behavior in un'architettura a sussunzione sono rappresentati come moduli separati che elaborano obiettivi individuali in maniera concorrente e asincrona. A un livello più basso, ogni behavior è rappresentato utilizzando un modello di macchina a stati finita incrementale (AFSM – augmented finite state machine). L'AFSM incapsula una funzione di trasformazione particolare β_i mentre i segnali provenienti come stimoli o quelli che rappresentano la risposta del behavior possono essere "soppressi" o "inibiti" da altri behavior attivi. Ogni AFSM esegue un'azione ed è responsabile della sua propria percezione del mondo; non esistono una memoria globale, dei bus, o dei clock. Con una tale progettazione, ogni modulo può essere mappato in un proprio processore. Infine non c'è un modello centrale del mondo o una rappresentazione globale dei sensori.

La coordinazione in questa architettura si esplica attraverso due meccanismi principali: la soppressione e l'inibizione: l'inibizione viene utilizzata per impedire che un segnale trasmesso da un AFSM raggiunga gli attuatori; la soppressione evita che il segnale venga trasmesso e lo sostituisce con un altro. Attraverso tali meccanismi si dota il sistema di un arbitrariato implicit basato su priorità nei quali i livelli più in alto nella gerarchia esplicano compiti di maggiore competenza.

1.3.4.2. Schema-theory

L'uso di schemi come modello filosofico per la spiegazione del comportamento lo si può far risalire a Kant (diciottesimo secolo). Gli schemi, in questo caso, erano definiti come un mezzo attraverso il quale la capacità di comprensione è capace di categorizzare le percezioni sensoriali nel processo della realizzazione della conoscenza o dell'esperienza. Una teoria di schemi neuro psicologici emerse nei primi anni del ventesimo secolo. La loro prima applicazione si deve agli sforzi di Head e Holmes [Head e Holmes 1911] per spiegare il meccanismo di controllo della postura negli esseri umani. Lo schema theory ha influenzato anche la psicologia fornendo uno strumento per costruire un astrazione che legasse il "cervello" con la "mente" (ossia la capacità di elaborazione con il suo substrato fisico). I lavori di Bartlett [Bartlett, 1932] e Piaget [Piaget, 1971] usavano lo schema theory come meccanismo per esprimere modelli della memoria e dell'apprendimento. Neisser [Neisser ,1976] nei suoi lavori presentava un modello cognitivo di

interazione tra i comportamenti motori nella forma di schemi interdipendenti con la percezione in un contesto di ciclo percettivo. Norman e Shallice [Norman e Shallice, 1986] usavano gli schemi come strumento per differenziare due classi di comportamento, volontario e automatico, e proposero un modello cognitivo che faceva uso di un meccanismo di scheduling antagonistico per la cooperazione e competizione dei behavior. Arbib [Arbib, 1981] fu in primo a considerare applicazioni dello schema theory ai sistemi robotici. L'estensione di questi principi sono stati anche applicati a sistemi di computer vision [Riseman e Hanson, 1987].

Esistono molte definizioni di schema, spesso fortemente influenzate dalle loro aree di applicazione (computazionale, neuroscientifiche, psicologiche, psicologiche). Alcuni esempi sono:

- una sequenza di azioni così come una sequenza per le azioni – a pattern of/for action [Neisser, 1976];
- un controllo adattivo che fa uso di una procedura di identificazione per aggiornare la sua rappresentazione dell'oggetto che deve essere controllato;
- un'unità funzional che riceve informazioni speciali, che anticipa un contenuto percettivo possibile, lo confronta con l'informazione percepita [Koy-Oberthur, 1989];
- un'unità percettiva che corrisponde a un'unità mentale [Piaget, 1971].

La definizione operativa, secondo Arkin, è la seguente: uno schema è un'unità di base di comportamento a partire dal quale si possono costruire delle azioni complesse; essa è composta della conoscenza di come agire o percepire così come dei processi computazionali che danno luogo a questa entità. Lo schema theory fornisce un metodo per codificare il comportamento di un robot a una granularità più grezza delle reti neurali mentre conserva gli aspetti di un controllo concorrente cooperativo/competitivo comune ai modelli neuroscientifici.

Lo schema theory, come definito in [Arbib, 1995], fornisce un linguaggio, ad alto livello di astrazione, per la teoria del cervello, la psicologia cognitiva, e l'Intelligenza Artificiale Distribuita (DAI). Questi è un'astrazione ad alto livello attraverso la quale il comportamento può essere espresso in maniera modulare. In questa cornice, un behavior è uno schema che è composto da un schema motorio ed uno schema percettivo.

Lo schema percettivo (*perceptual schema*) rappresenta i processi per il riconoscimento di un dato dominio di interazione, con alcuni parametri che rappresentano alcune proprietà quali le dimensioni, la posizione e il moto. Un assemblaggio degli schemi percettivi fornisce una stima dello stato ambientale con una rappresentazione di obiettivi e bisogni. Nuovi input sensoriali, così come i processi interni, aggiornano l'assemblaggio degli schemi. Lo stato interno è aggiornato anche dalla conoscenza dello stato di esecuzione dei piani correnti costruiti dallo

schema motorio (*motor schema*) molto simili ai sistemi di controllo. I motor schema possono essere combinati con i perceptual schema per formare programmi di controllo coordinato che controllano il passaggio di pattern in ingresso e uscita di co-attivazione, con meccanismi per il passaggio di parametri di controllo dagli schemi percettivi a quelli motori.

Il livello di attività di un'istanza di uno schema percettivo rappresenta un "livello di confidenza" che l'oggetto rappresentato dallo schema sia presente; mentre per lo schema motorio può segnalare il suo "grado di prontezza" per controllare una qualche sequenza di azioni. Le istanza dello schema possono diventare attivate in risposta ad alcuni pattern di input da stimoli sensoriali (nel qual caso si dice che sono *data-driven*) o in risposta ad altre istanze di schema che sono già attive (*hypotesis driver*). Il livello di attività di uno schema può essere uno dei molti parametri che lo caratterizzano (uno schema per una "palla" può includere parametri per le sue dimensioni, colore, e velocità). È così importante distinguere tra "livello di attività" come parametro particolare di uno schema dall' "attività neuronale" che varierà a seconda delle diverse implementazioni neurali di uno schema.

Lo schema theory fornisce le capacità per specificare e progettare sistemi behavior-based [Arbib, 1992]. Il behavior, infatti, può essere visto come uno schema composto, a sua volta, da uno schema motorio e uno percettivo (figura. 4).

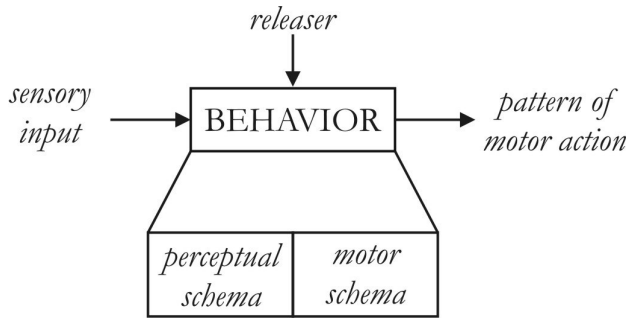


figura 4 – rappresentazione di un behavior attraverso lo schema theory

Il *releaser*[4] agisce come un segnale di controllo per attivare un behaviour. Se un behaviour non è attivato dal suo releaser, esso non risponde ai segnali di input e quindi non produce output motorio.

[4] Il termine *releaser* è ereditato dagli studi etologici [Tinberg, 1953],[Lorenz, 1981]. L'aspetto fondamentale di questi studi fu che un comportamento animale è innescato da una qualche percezione che l'animale riceve alla quale fanno seguito uno o più behaviour in cascata.

L'idea di schema ha una sua controparte informatica nel concetto di classe nella programmazione ad oggetti (OOP). Se uno schema è una modalità generica per fare un'attività, come andare in bicicletta, una classe "schema" in un qualsiasi linguaggio a oggetti può contenere sia dati (conoscenze, modelli, releaser) sia metodi (gli algoritmi che permettono di percepire e agire) per realizzarla. Dopo che uno schema è stato rappresentato come una classe, i parametri (ad esempio tipo di bicicletta, altezza del sellino, posizione del manubrio) possono essere forniti all'oggetto al momento dell'instanziazione (quando un oggetto è creato dalla classe). La capacità tipica di astrazione tipica del paradigma a oggetti si riflette, in questo caso, nel fatto che uno schema è, per sua natura, una generalizzazione di un'attività: la sua istanziazione è la concretizzazione particolare effettiva di tale attività.

Riassumendo, si possono così considerare i concetti di schema percettivo e motorio in termini derivanti dall'etologia e dalla psicologia cognitiva in questo modo:

- un behavior riceve input sensoriali e produce azioni motorie come output;
- un behavior può essere rappresentato come uno schema che, essenzialmente, è un costrutto di programmazione a oggetti (OOP);
- un behavior è attivato da releaser;
- la trasformazione di input proveniente dai sensori in output costituiti da azioni motorie può essere divisa in due sottoprocessi che coinvolgono uno schema percettivo e uno schema motorio

1.4. Compendio al capitolo 1

Le architetture robotiche discusse in questo capitolo hanno come obiettivo principale la costruzione di un robot visto, essenzialmente, come connessione "intelligente" tra i suoi sensori e i suoi attuatori. Il governo del robot (la connessione "intelligente") viene costruito utilizzando il robot stesso come suo centro di interesse primario: la progettazione si sviluppa, partendo da una descrizione dell'ambiente della struttura del robot e del suo comportamento desiderato, con il robot come sistema di riferimento principale. Una tale visione permette di costruire, all'aumentare dei comportamenti implementati per il robot, artefatti dotati di un grado di autonomia sempre crescente. Quando si vuole costruire un sistema robotico in grado di interagire con un essere umano, però, bisogna considerare altri fattori che possono influenzare il *modus operandi* della costruzione di un robot. In questo caso è partner di cooperazione (l'essere umano)

che diventa il protagonista dell'interazione: se si vuole costruire un robot che entri in contatto con gli esseri umani e che possa cooperare con questi, si dovrà cambiare il centro di interesse passando da una progettazione robot-centrica a una uomo-centrica (o *human-centric*). Come si vedrà nel prossimo capitolo, esigenze quali la modellazione dell'essere umano, la capacità, per un robot, di interpretazione di comandi e intenzioni possono diventare primarie. A ciò, si aggiunge la necessità imprescindibile che l'interazione avvenga innanzitutto in maniera sicura salvaguardando l'incolumità dell'essere umano.

Alcune delle soluzioni proposte in questo capitolo si rivelano ancora utili, se non necessarie, per la costruzione del sistema robotico. In particolare, si vedrà che una schematizzazione attraverso un automa a stati finiti per esprimere la sequenza di funzionamento di un robot può risultare ancora utile (specie in ambienti con un grado di autonomia non elevato). Inoltre, una codifica delle risposte del robot attraverso campi di potenziale può risultare decisiva per ottenere risposte in *real-time* che evitino situazioni potenzialmente catastrofiche.

CAPITOLO 2

L'interazione uomo-robot

2.1. Introduzione

La robotica è un settore di ricerca in continua evoluzione che, negli ultimi anni, ha avuto un notevole sviluppo. Tra i fattori che hanno favorito tale sviluppo ci sono la riduzione dei costi di dispositivi di calcolo sempre più potenti, la produzione di strumenti per la rilevazione delle caratteristiche ambientali e di attuatori sempre più robusti e precisi. Allo stato attuale, la maggior parte dei robot operano in ambienti industriali dove eseguono principalmente compiti quali l'assemblaggio o il trasporto di materiali, ma non appare lontano il momento in cui i robot potranno fornire direttamente dei servizi a utenti umani condividendo lo stesso spazio operativo.

Tutti questi progressi hanno reso possibile lo sviluppo di una nuova generazione di robot di servizio progettati per *assistere* utenti umani direttamente nel loro posto di lavoro, nel tempo libero e a casa. Dovendo uomo e robot condividere lo stesso ambiente, è nata da pochi anni una nuova disciplina all'interno del vasto campo della robotica: lo studio dell'interazione uomo-robot (*Human-Robot Interaction – HRI*).

In questo capitolo, si illustreranno (paragrafo 2.2) alcuni sistemi robotici che attualmente operano i diversi ambiti della società; di come il problema dell'interazione uomo robot sia recentemente nato in ambito scientifico e delle prime considerazioni che sono sorte in tale ambito (paragrafo 2.3). Restringendoci, infine, al settore dell'interazione fisica uomo-robot, ossia quando si prevede un contatto fisico tra un sistema robotico e un suo utente, verrà proposto uno schema generale dell'interazione fisica "sicura", ossia che abbia l'incolumità dell'essere umano come suo primo requisito essenziale (paragrafo 2.4).

2.2. I sistemi robotici attuali

Nel rapporto più recente delle Nazioni Unite (U.N.) sulla robotica [U.N. and I.F.R.R., 2002], questa è raggruppata in tre categorie principali: quella industriale, quella orientata a servizi professionali e quella di servizio personale. La robotica industriale è quella che ha avuto per prima un rilevante successo commerciale, diffondendosi maggiormente rispetto alle altre, più giovani, categorie. Un robot industriale presenta tre elementi essenziali:

1. modifica il suo ambiente fisico – ad esempio raccogliendo oggetti da un posto preciso e ricollocandoli in un altro posto;
2. è (totalmente) controllato da un elaboratore esterno;
3. opera in un ambiente industriale fortemente strutturato – ad esempio su dei nastri trasportatori.

Il confine tra robot industriale e dispositivi automatici industriali non-robotici può apparire alquanto labile; il termine robot è, in questi casi, usato per riferirsi a sistemi con elementi di attuazione multipli organizzati in catene (ad es. i bracci robotici). Applicazioni classiche per la robotica industriale comprendono compiti di saldatura, assemblaggio, impacchettamento, trasporto, manipolazione di materiale potenzialmente pericoloso per gli esseri umani e così via.

Fino alla fine del 2002, la maggior parte dei robot industriali era installata nell'industria automobilistica dove il rapporto tra lavoratori umani e robot è approssimativamente di dieci a uno (sempre secondo il rapporto U.N. e I.F.R.R. del 2002). Nel 2001, sempre le Nazioni Unite hanno stimato in 780.600 le unità di approvvigionamento di robot industriali, un numero che è aumentato del 25% fino al 2005. Inoltre, il costo medio dei robot industriali e diminuito dell'88.8% tra il 1990 e il 2001 mentre negli Stati Uniti, nello stesso periodo, il costo del lavoro del personale aumentava del 50.8%. Questi due andamenti opposti continuano ad aprire nuove opportunità per dispositivi robotici che siano in grado, se non di sostituire la manodopera umana, almeno di aiutare gli utenti nei lavori più pesanti e onerosi. Inoltre, in passato l'introduzione dei robot in ambienti industriali rientrava esclusivamente nell'ottica della sostituzione della manodopera "umana" e dunque l'interazione uomo-robot era studiata quasi esclusivamente come la progettazione e implementazione di interfacce uomo-macchina per programmare e/o riprogrammare i robot. La condivisione dello spazio di lavoro da parte di uomini e robot in un ambiente industriale apre, a questo punto, nuovi scenari e nuovi campi di ricerca.

La robotica orientata a servizi professionali costituisce un settore di sviluppo tecnologico e metodologico più recente. Sebbene la ricerca in questo campo sia per lo più ai primordi, essa sta crescendo più rapidamente della robotica industriale. Come per la robotica industriale, un robot per servizi professionali agisce, modificando e navigando, nel proprio spazio fisico. Il suo compito principale è quello di assiste le persone nel raggiungimento di obiettivi professionali, molto al di fuori di un ambiente fortemente strutturato quale quello industriale. Alcuni di questi robot operano in ambienti inaccessibili per le persone, come i robot che svuotano i residui nucleari [Blackmon et al, 1999], [Brady et al., 1998] o determinano la posizione di mine abbandonate [Thrun et al, 2003]. Altri assistono negli ospedali, come l'Helpmate robot [King e Weiman, 1990] che trasporta cibo e medicinali negli ospedali, o i sistemi robotici chirurgici usati per assistere i medici nelle loro procedure. I manipolatori robotici sono di solito anche usati nei laboratori chimici e biologici, dove trattano e manipolano sostanze (ad esempio il sangue) con una velocità e precisione impossibile per utenti umani; studi recenti hanno indagato la possibilità di introdurre aghi nelle vene attraverso manipolatori robotici [Zivanovic e Davies, 2000]. Secondo il rapporto delle Nazioni Unite citato in precedenza, il totale delle operazioni in cui sono stati utilizzati dei robot per servizi professionali nel 2001 è stato di 12400: nel 2004 tali operazioni sono praticamente raddoppiate. In questo campo, le interazioni dirette con gli esseri umani hanno superato quelle della robotica industriale da quando i robot di servizio condividono gli stessi spazi fisici dei loro utenti.

I robot orientati a servizi personalizzati possiedono il più alto tasso di crescita[5]. Tali robot assistono o intrattengono le persone in faccende domestiche o attività di ricreazione. Alcuni esempi includono gli aspirapolvere, taglia erbe, robot che assistono persone anziane o disabili, sedie a rotelle automatizzate e giocattoli. I robot per servizi personalizzati stanno cominciando a diventare una realtà visibile al grande pubblico: la sola vendita di giocattoli robotici è destinata ad aumentare in maniera significativa nei prossimi anni. Molti di questi robot interagiscono con le persone che, in genere, non possiedono standard o requisiti speciali per interagire con un robot. Trovare mezzi efficaci di interazione risulta una questione ancora più cruciale in questo nuovo segmento di mercato della tecnologia robotica che nell'industria robotica o la robotica orientata a servizi professionali.

Il cambiamento di ottica tra la robotica industriale e quella dei servizi personali e l'aumento di robot che operano in rapporto stretto con la gente ha aumentato la quantità di ricerche e sfide nel campo della loro progettazione. Alcune di queste sfide riguardano, ad esempio, il costo e la sicurezza. Dalla prospettiva dell'HRI, la più importante caratteristica di

[5] Secondo stime ottimistiche il numero di tali robot crescerà da 176.500 nel 2001 a 2.021.000 nel 2005.

questi nuovi obiettivi è che i robot di servizio condividono gli spazi fisici con le persone. In alcune applicazioni, queste persone possono essere professionisti che possono essere addestrati per usare i robot; in altre, possono essere bambini, persone anziane o disabili la cui capacità di adattarsi alla tecnologia robotica può essere limitata. La progettazione di un'interfaccia, mentre dipende dallo specifico target dell'applicazione, richiederà considerazioni sostanziali sull'utente finale del sistema robotico. In questo risiede una delle sfide più interessanti che, oggi, il settore della ricerca robotica si trova ad affrontare.

2.3. Interazione uomo-robot

I robot sono sempre più presenti nella vita di ogni giorno in ambienti dove convivono con esseri umani e in molti aspetti della nostra società, dagli usi militari a quelli medici, per il tempo libero e negli uffici, usati affinché camminino, nuotino, volino o siano coprotagonisti di missioni spaziali.

Come evidenziato in [Burke et al., 2004], uno dei principali deterrenti per l'integrazione dei robot in sistemi integrati uomo-robot è l'attuale punto di vista predominante nella progettazione dei robot: l'approccio robot-centrico, dove lo schema predominante parte dal punto di vista del robot. Sebbene questa prospettiva si riveli particolarmente appropriata e soddisfacente nello sviluppo delle piattaforme hardware e software orientate ai robot, non si prendono tuttavia in considerazione concetti propri del lavoro di squadra e quindi, considerandoli non primari, la loro integrazione risulta difficile da realizzare. Nell'interazione uomo-robot, bisogna tener conto di come queste due componenti possano formare dei team sinergici[6] e dunque considerare nuovi approcci nella progettazione dei sistemi uomo-robot: cinematiche più flessibili per i manipolatori, una miglior comprensione della divisione di responsabilità tra le capacità "intellettive" degli uomini e quelle di "elaborazione" dei robot; algoritmi di apprendimento, sensori migliori ma, soprattutto, algoritmi migliori per la fusione delle informazioni provenienti da tali sensori. Ovviamente un tale tipo di lavoro può essere affrontato solo in un'ottica interdisciplinare che comprenda i contributi di gruppi di ricerca di varie aree ed estrazioni. Le aree di ricerca maggiormente coinvolte sono, in ordine sparso, la robotica, lo studio dell'interazione uomo-macchina, le scienze cognitive, la psicologia e le scienze sociali. Il contributo di tutti questi gruppi di lavoro dovrebbe portare alla progettazione di team cooperativi e collaborativi formati da uomini e robot dove i vari membri di questa "squadra" eseguono

[6] L'interazione è una modalità che va al di la della semplice condivisione di uno spazio comune. Ciò che fa la differenza è la presenza di un obiettivo condiviso tra uomo e robot.

compiti specifici a seconda delle loro varie abilità. Così come nel lavoro di gruppo usuale, le responsabilità e i ruoli nei team misti uomo-robot dovrebbero essere assegnati in maniera dinamica. I membri del team dovrebbero essere in grado di riconoscere le situazioni che cambiano e adattarsi per assicurare che la missione globale abbia successo. La sfida della ricerca interdisciplinare sull'Human-Robot-Interaction è, essenzialmente, capire come combinare, in maniera effettiva ed efficace, le teorie attuali e quelle nuove (così come nuovi modelli) che possono essere sviluppate per progettare team eterogenei uomo-robot. Una lista di direzioni di ricerca finora proposte includono:

- studi sull'intervento umano con differenti livelli di autonomia;
- studi cognitivi sulle limitazioni delle capacità intellettive degli esseri umani in compiti tipici di interazione tra uomo e robot, come, ad esempio, limitazioni sul ragionamento spaziale, velocità di reazione, consistenza, effetti degli sforzi (della fatica);
- modalità di interazione, sia in input che in output, che si estrinsecano attraverso strumenti che oggigiorno sono di uso comune – tastiere, mouse, monitor – e che possono essere usati in vari ambienti fisici;
- livelli di astrazione appropriati per comandi efficaci ma intuitivi e controllo di robot;
- sviluppo dei ruoli per robot ed esseri umani all'interno di team, basandosi su studi sulla divisione di compiti tra esseri umani, scambio di ruoli, e comportamenti in cui non ci devono essere interferenze;
- adattabilità degli esseri umani, robot, e team misti a seconda della natura dinamica delle situazioni;
- interfacce con gli utenti scalabili per permettere ad utenti umani di lavorare in maniera efficiente con team multi-robot;
- progettazione di tool per lo sviluppo di interfacce uomo-robot;
- architetture robotiche e modelli del mondo che supportino l'evoluzione dei robot;
- metodologie di valutazioni e metriche per la stima dei progressi nei team uomo-robot.

Sebbene questa lista non comprenda tutti i possibili aspetti dell'interazione uomo-robot, essa può essere comunque usata come punto di partenza. Da questa prima, sommaria tassonomia

delle direzioni di ricerca, infatti, si possono già distinguere aspetti che coinvolgono una interazione "fisica" da altri più propriamente "cognitivi".

2.3.1. Il concetto di "autonomia" nell'Interazione Uomo-Robot

In generale con il termine "autonomia" si connotano quei sistemi (software o hardware) che agiscono indipendentemente dall'intervento diretto di esseri umani [Weiss, 1999]. Tale definizione è puramente operativa e così generica da potersi adattare a varie situazioni. Un robot autonomo, ad esempio, può essere quello che, una volta progettato e implementato, esegue il suo compito senza ricevere più istruzioni su come scegliere la migliore azione da eseguire. Quando, poi, il sistema deve interagire in maniera significativa con utenti umani, l'autonomia diventa un concetto più ricco che merita alcune considerazioni.

L'autonomia è intimamente legata alla capacità dei robot di compiere modifiche nel proprio spazio operativo. Robot diversi mostrano differenti gradi di autonomia il quale è spesso misurato mettendo in relazione quanto un ambiente può essere cambiato sotto l'effetto delle azioni del robot rispetto al tempo medio tra due fallimenti delle azioni del robot[7] stesso [Thrun, 2004]. L'interazione dei robot con gli esseri umani non può essere studiata senza considerare il grado di autonomia di un robot, poiché esso è un fattore determinante che riguarda le funzioni che un robot può eseguire, e il livello al quale l'interazione ha luogo.

I tre tipi di robotica, mensionati nel paragrafo precedente, sono caratterizzati da diversi livelli di autonomia, i quali sono collegati alla complessità dell'ambiente nel quale i robot operano. Non sorprende il fatto che i robot industriali operino con un livello di autonomia minimo. Nelle installazioni industriali, l'ambiente è solitamente altamente strutturato per rendere i robot capaci di eseguire i loro compiti in maniera alquanto automatica. Per esempio, i robot impegnati in compiti di *keep-and-place* (ossia raccogliere e poggiare un determinato oggetto) hanno informazioni estremamente precise sulle proprietà fisiche degli oggetti da manipolare (dimensioni, peso, etc.), e sulle posizioni relative agli oggetti da prendere e i luoghi dove poggiarli. I veicoli da trasporto senza conducente nelle installazioni industriali seguono spesso percorsi stabiliti da fili guida o indicati da segni speciali apposti sul pavimento. Come suggeriscono questi esempi, un'attenta organizzazione dello spazio di certo minimizza la quantità di autonomia richiesta – un ingrediente chiave del successo commerciale dell'industria robotica.

[7] Interessante il fatto che Thrun metta in relazione il concetto di autonomia con la possibilità, per il robot, di avere fallimenti

Questo quadro risulta abbastanza differente nei robot di servizo. Mentre le modifiche dell'ambiente rappresentano ancora una caratteristica saliente – ad esempio si può utilizzare un sistema satellitare basato sul GPS per determinare la posizione di un robot in compiti di outdoor mentre questi modifica il suo ambiente e monitorare, così, le sue capacità attuative – la complessità degli ambienti nei quali operano tali robot prevede un grado di autonomia maggiore rispetto a quelli operanti nell'industria. L'importanza dell'autonomia nella robotica di servizio diventa ovvia in applicazioni quali un compito di guida nei musei come descritto in [Thrun et al., 2000]. Se il robot si trovasse ad operare in un museo vuoto, questi sarebbe capace di seguire ciecamente la stessa traiettoria all'infinito – così come i robot industriali tendono a eseguire ripetutamente la stessa, esatta sequenza di azioni – dato che non ci sarebbe nulla, nell'ambiente, che possa modificare il comportamento motorio del robot una volta che questi deve eseguire un determinato compito (ossia una determinata traiettoria tra le varie stanze del museo). Il comportamento imprevedibile dei visitatori forza il robot a compiere delle deviazioni. La possibilità di compiere traiettorie non programmate a priori dota questi robot di capacità superiori alla maggior parte dei loro omologhi impegnati in applicazioni industriali.

Lo studio delle tecnologie che rendono capace un robot di essere autonomo è stato un nucleo focale della ricerca robotica negli anni passati. Una branca di tale ricerca riguarda l'acquisizione di modelli dell'ambiente. Se, ad esempio, una mappa 2-D è solo una proiezione del vero spazio tridimensionale ciononostante questa informazione, unita a un sistema di pianificazione, è sufficiente per far navigare un robot in assenza di modifiche ambientali. Altri studi si sono incentrati sulla possibilità di rilevare e modellare le persone. In genere, i robot che operano a stretto contatto con la gente devono essere dotati di un alto grado di autonomia, in parte a causa della sicurezza e in parte perché la gente è meno prevedibile della maggior parte degli oggetti inanimati. E' pratica comune dotare i robot di servizio di sensori capaci di rilevare e tracciare la posizione delle persone [Schulz et al., 2001]. Alcuni ricercatori sono andati oltre la tecnologia dei vari dispositivi per costruire robot che imparano il comportamento usuale della gente per rispondere prontamente ai loro spostamenti [Bennewitz et al., 2003].

2.4. L'interazione fisica uomo-robot

L'interazione fisica, specie in ambito industriale, si è basata prevalentemente sul principio dell'isolamento della controparte meccanica da quella umana (quindi a tutti gli effetti evitando l'interazione) [ANSI, 1999]. Le azioni che il robot può compiere devono essere effettuate in un'area prescritta di sicurezza: quando si rileva la presenza di un intruso in tale area, il robot ha,

come unico comportamento predefinito, un comando di emergenza che lo ferma[8]. Tale vincolo risulta, ovviamente, inappropriato se si vuole che robot e utente umano non siano isolati tra loro ma che condividano parte dello stesso spazio operativo. Si pensi solo a quante volte il sistema dovrebbe fermarsi e partire da zero ogni volta che un utente entra nello spazio di lavoro del robot. La sicurezza intrinseca del progetto e l'affidabilità dei sistemi di controllo di questi robot diventano dei nuovi criteri per la valutazione delle prestazioni. In passato, la qualità di un robot si misurava in termini di precisione e ripetibilità delle operazioni di posizionamento. Adesso, bisogna individuare parametri quantitativi che traducano in standard di produzione i criteri di sicurezza. I robot attuali sono ancora molto pericolosi per l'interazione, e non esistono criteri di sicurezza standardizzati ai quali uniformarsi, né è maturo lo studio di interfacce in linguaggio naturale che permettano, ad esempio, di arrestare un robot in maniera intuitiva in caso di emergenza. Le due parole-chiave sono quindi "sicurezza" e "affidabilità".

Un robot è in grado di esercitare delle forze, da utilizzare per compiere un lavoro pesante. Se è necessaria la generazione di forze per sopperire a limiti fisici umani, la sicurezza è messa a rischio dalle masse in gioco. La robotica vista come "connessione intelligente tra percezione ed azione" implica una certa autonomia dei sistemi il che rende impossibile prevedere ogni movimento dei bracci meccanici: se si vuole dotare i robot di autonomia, non è possibile pre-programmare tutte le azioni da operare in un contesto che sia, al contrario di una linea di produzione industriale, strutturato in modo non noto e imprevedibile. Pertanto, l'autonomia necessaria a un robot domestico o comunque in grado di fornire servizi, sottintende un'adeguata valutazione della sicurezza dei sistemi impiegati. Fino ad oggi è sempre stata operata una dicotomia tra "cervello" e "corpo" dei robot, affidando lo studio del primo a neuroscienziati, a esperti in scienze cognitive e a informatici, e lo studio della struttura meccanica e dei controllori agli ingegneri elettronici, meccanici e ai cibernetici. Ora, nelle applicazioni reali della robotica, si nota come la prospettiva fisica sia sempre prioritaria, e che le architetture per il governo di un robot non possano essere indipendenti dalla sua "struttura fisica".

Attualmente, la regolamentazione per la sicurezza nell'utilizzo di robot assume alcune ipotesi in riferimento all'ambiente di lavoro che sono valide per tutti i contesti produttivi odierni, ma completamente fuori luogo per applicazioni di stretta interazione tra uomo e robot. Il principale riferimento internazionale è il già citato standard ANSI/RIA R15.06-1999 (American National Standard for Industrial Robots and Robot Systems - Safety Requirements). Questo documento presenta i provvedimenti da adottare per la sicurezza personale in ambienti industriali dove siano adoperati robot manipolatori. Esiste uno standard complementare (ANSI/UL 1740) a

[8] Lo standard Europeo EN-775 contiene, praticamente, gli stessi vincoli.

cui uniformarsi perché il progetto hardware sia a norma dello R15.06. Tuttavia, quello che colpisce è ancora la mancanza di indicazioni per il caso in cui robot e operatori debbano *condividere lo spazio di lavoro*, cioè per ambienti non strutturati e non protetti dalle gabbie prescritte. Sulla base di osservazioni provenienti da enti normatori in tutto il mondo, è in corso una revisione dello standard ISO 10218, che è l'equivalente internazionale dello R15.06, affinché si considerino anche aspetti relativi ai circuiti di controllo e alla presenza di robot mobili su ruote. I più importanti produttori europei di robot (ABB, Kuka Roboter, Reis Roboter) hanno incluso moduli software per rallentare le operazioni dei robot manipolatori industriali all'approssimarsi di operatori nello spazio di lavoro, e ciò può essere sufficiente per la sicurezza in un contesto industriale. In ogni caso, criteri che permettano di definire la sicurezza dell'interazione con i robot sono strettamente collegati ad una quantificazione delle possibili conseguenze di impatti con i robot in un particolare contesto. La sicurezza e l'affidabilità, quindi, devono essere messe in relazione con i singoli componenti del progetto di un robot, dai meccanismi ai motori, dai sensori ai sistemi di controllo, studiando in che modo malfunzionamenti ed errori possano tradursi in movimenti imprevisti e collisioni.

In alcuni casi appare evidente che la dimensione "fisica" possa diventare anche più importante degli aspetti cognitivi (soprattutto nel caso di comportamenti autonomi del robot), perché movimenti inattesi delle persone possono trasformarsi in tragici impatti. In ogni caso, aspetti "cognitivi" sono fondamentali per dotare il robot di interfacce e sistemi di fusione sensoriale che li rendano più "consapevoli" e adatti all'interazione con persone. Un possibile approccio all'interazione tra uomo e robot prevede che vengano enfatizzate le caratteristiche "migliori" di uomini e robot: sia quindi l'uomo a fornire il controllo del compito e l'esperienza, sia il robot a rendere disponibile la forza bruta e la resistenza alla fatica. Se, però, analizziamo un compito tipico per un robot di manipolazione che debba operare in ambienti domestici, capiamo subito che il robot deve essere capace anche di pianificare, muoversi, esplorare, controllare la presenza di persone, schivare alcuni oggetti e afferrarne altri, sempre rispettando i vincoli di sicurezza.

Le architetture di controllo devono prevedere la possibilità di gestire errori dei vari componenti al fine di rendere gli eventi non catastrofici, mentre i sistemi sensoriali devono fornire un'immagine fedele della posizione, della direzione ed eventualmente dell'espressione e interpretazione di comandi vocali delle persone presenti nell'ambiente di lavoro; infine, motori e sistemi di attuazione del movimento delle "mani" devono permettere di non danneggiare un utente e di assecondarne movimenti ed intenzioni. Più in dettaglio, per ottenere condizioni di sicurezza accettabili a partire dal progetto meccanico, l'eliminazione di spigoli vivi e la

sostituzione dei bracci in acciaio con strutture più leggere è il primo passo per ridurre la possibilità di danneggiare un utente. Materiali leggeri ma rigidi, con protezioni passive sull'intera struttura di un braccio di manipolazione robotico permettono di ridurre le conseguenze di impatti inattesi durante il funzionamento. Dal punto di vista dell'attuazione del moto, poi, i motori devono essere collocati in una posizione prossima al suolo, per ridurre l'inerzia dei bracci, con la conseguenza, però, di complicare il progetto a causa dell'introduzione di cavi, tendini e cinghie per la trasmissione del moto. Diverse tecniche sono state proposte anche per cambiare la rigidezza del robot in caso di contatto, attraverso l'impiego di elementi elastici. Infine, è chiaro che solo sensori affidabili possono contribuire a migliorare la sicurezza nell'interazione: il sistema deve sempre sapere quale sia la sua posizione nello spazio, e dove si trovino gli utenti (in particolare, le loro teste e le loro mani) con i quali dovrà interagire. Infine, gli algoritmi di controllo sono fondamentali per un'adeguata elaborazione dei dati e per una regolazione opportuna del moto.

2.5. Uno schema generale per la interazione fisica sicura *uomo-robot*

Come accennato in precedenza, la progettazione di un sistema di interazione fisica uomo-robot può risultare differente dallo schema classico dove il sistema di governo del robot è costruito a partire dalle sue esigenze particolari [Arkin, 1998], [Murphy, 2000]. Già Kidd agli inizi degli anni '90 [Kidd, 1992] afferma che in un sistema robotico che coopera fisicamente con gli esseri umani, condividendo lo stesso spazio fisico, bisogna sempre tener conto delle capacità degli esseri umani. Kidd affermava che chi progetta un tale sistema dovrebbe usare la tecnologia dei robot per aiutare e arricchire le capacità degli utenti piuttosto che sostituirli. Inoltre, fa notare che la ricerca robotica, fino ad allora, tendeva a focalizzarsi su questioni che erano governate innanzitutto da esigenze legislative legate esclusivamente alla sicurezza. Ciò comportava che nella fase di progettazione di un sistema robotico si privilegiava il punto di vista del robot – robot-centred design – tenendo in secondo piano le caratteristiche e le esigenze dell'utente umano – human-centred design. In una progettazione cosiddetta "human-centred" l'interazione uomo-robot è sviluppata superando, o tentando di superare, le questioni puramente tecnologiche e concentrandosi su problemi di più alto livello quali l'allocazione di task tra persona e robot, la sicurezza, la struttura dei gruppi di lavoro. Tali questioni devono essere prese in considerazione

negli stadi iniziali della progettazione: se sono considerate solo in quelli finali, le problematiche diventano secondarie e possono risultare di impatto poco rilevante sugli aspetti progettuali.

Diventa, dunque, fondamentale in questo nuovo approccio (a) come rilevare le potenziali situazioni (b) come affrontarle e controllarle sempre cercando di soddisfare il goal di cooperazione principale.

In figura 1 è illustrato un possibile schema generale per la interazione e cooperazione fisica uomo/robot.

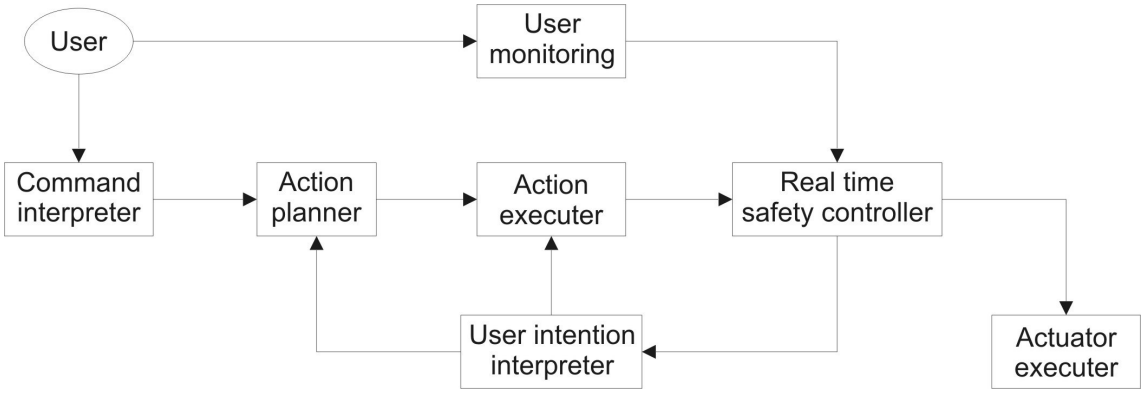


figura 1 – schema di HRI fisica sicura

In uno schema tradizionale di progettazione robotico (ad esempio il paradigma behavior based alla Arkin), il sistema di governo è costruito a partire da quelli che sono i suoi obiettivi particolari e l'interazione con gli utenti è governata considerando un puro feedback sensoriale che, in qualche modo, modella l'utente a partire dalle esigenze del robot. In questo caso, invece, il sistema decisionale del robot è costruito a partire dal modello di utente con il quale deve interagire e dal goal cooperativo principale. La modellazione dell'utente, intesa come le caratteristiche sensoriali che il robot deve essere in grado di elaborare per poter interagire con gli esseri umani, agevola l'interazione e permette che questa possa avvenire in maniera sicura. Il nuovo punto di vista, human-centred, parte dall'esigenza di interagire con gli esseri umani: come interpretarne i comandi e, in una certa misura, come interpretarne le intenzioni

Nello schema proposto, il *command interpreter* è un modulo deputato alla traduzione dei comandi impartiti dall'utente umano (comandi che possono essere impartiti in linguaggio naturale, con un sistema codificato di gesti o, più semplicemente, attraverso una tastiera) in un insieme di azioni da eseguire e/o posizioni target da raggiungere. Tale modulo riveste

un'importanza fondamentale per l'interazione "cognitiva" tra uomo e robot e può essere studiato e progettato sotto diversi punti di vista. Le interfacce che si basano su comandi vocali, ad esempio, rappresentano un settore di ricerca di frontiera [Cappelli and Giovannetti, 2004] sebbene non sia considerato, attualmente, il mezzo ideale per ogni situazione.

Il modulo di pianificazione (*Action planner*) calcola le possibili traiettorie sicure per il robot a partire da uno stato iniziale a quello finale fornito dal *command interpreter*. In generale, può essere considerato come un modulo di pianificazione globale che genera la sequenza "migliore" da essere eseguita se il sistema non rivela situazioni di pericolo o l'utente non impartisce nuove direttive.

Il modulo *action executer* ha il compito di eseguire la sequenza di azioni pianificata suddividendola in maniera opportuna a seconda delle circostanti ambientali attuali. Se, ad esempio, durante l'esecuzione del task l'utente è in prossimità del robot ma comunque a una certa distanza di sicurezza, l'*action executer* può comandare di rallentare la velocità del robot lasciando inalterata la traiettoria da seguire.

Quando si sviluppa un sistema per la cooperazione uomo/robot risulta cruciale il modo in cui il robot è in grado di percepire gli esseri umani presenti nella scena. La scelta di come rappresentare gli utenti del sistema e, poi, di come utilizzarla al meglio (*User monitoring*) è fondamentale per il sistema globale. Tale rappresentazione può dipendere principalmente da due fattori: l'ambiente nel quale si esplica l'interazione e gli obiettivi che si devono raggiungere cooperativamente[9]. Se l'obiettivo principale dell'interazione richiede al robot la capacità di evitare un essere umano o di avvicinarsi ad alcune sue parti, allora non risulta appropriato considerare l'utente come un mero ostacolo (come, ad esempio, in [Traver et al. 2000]). Se da un lato alcune zone del corpo devono (quasi) sempre esser considerate come ostacoli (ad esempio la testa), dall'altro bisogna lasciare la possibilità che, in diversi momenti della sua esecuzione, il robot consideri altre zone del corpo dell'utente (ad esempio le mani) alternativamente come ostacoli o come "zone di contatto". Ciò può essere raggiunto se si dota il sistema della capacità di individuare e, soprattutto, discriminare tra diverse parti del corpo di un essere umano. In un ambiente di lavorazione industriale, particolare importanza rivestono le mani e il volto. Durante l'interazione, inoltre, l'utente deve essere monitorato con una certa frequenza, frequenza che potrebbe variare con il grado di pericolosità dell'interazione attuale stessa. A ciò si aggiunge il fatto che l'informazione sulla posizione dell'utente può essere usata per modificare la velocità del robot lungo la traiettoria modificata.

[9] La scelta dei sensori con i quali rilevare gli esseri umani può essere conseguenza di questi due fattori.

Il compito primario di un sistema per l'interazione sicura uomo/robot, nel quale quest'ultimo rappresenta la componente "più pericolosa", è quello di salvaguardare l'incolumità dell'utente umano. E' dunque necessaria l'implementazione di un modulo che rilevi i pericoli imminenti e che, in maniera reattiva e in tempi stretti, abbia la possibilità di evitare le collisioni più pericolose. Il modulo *real time safety controller* è responsabile di tale controllo reattivo ai cambiamenti improvvisi che possono avvenire nell'ambiente e che non sono anticipati durante la fase di pianificazione delle traiettorie. In definitiva, tale modulo ha il compito di confrontare i dati provenienti dai sensori che monitorano l'attività dell'utente con quelli provenienti dall' *action executer*: se la traiettoria pianificata è incompatibile con le esigenze di sicurezza, si attiva un meccanismo di controllo per prevenire collisioni pericolose e, in un secondo momento, si cerca di ripianificare un nuovo percorso per i soddisfacimento del task. Da notare che l'uscita di tale modulo è comunque un'azione del robot, sia che esso debba continuare con la traiettoria pianificata sia che il sistema si debba, in qualche modo, riconfigurare (un comando agli attuatori).

Una volta individuate le caratteristiche salienti che un sistema di interazione uomo-robot deve possedere, i dettagli implementativi dei vari moduli dipenderanno dall'ambiente nel quale avviene l'interazione, dalla sensoristica a disposizione, degli obiettivi che devono essere portati a termine.

2.6. Compendio al Capitolo 2

L'interazione uomo-robot è un settore di ricerca giovane ma già riconosciuto, nell'ambito scientifico, come fertile e promettente. Se la costruzione di robot autonomi ha portato a risultati importanti e consolidati, quando lo scopo principale di un robot è l'interazione con un essere umano si impongono ulteriori considerazioni. Se si vuole un sistema robotico che cooperi in maniera significativa e possa interagire fisicamente, bisogna mettere al centro della tua progettazione l'essere umano. Ciò significa che diventa predominate la possibilità di poterlo modellare affinché, innanzitutto, la cooperazione avvenga in termini sicuri e, inoltre, il compito di cooperazione possa essere effettivamente svolto.

Lo schema proposto in questo capitolo (paragrafo 2.5) pone l'utente al centro dell'interazione individuando i blocchi, all'interno della progettazione del sistema robotico complessivo, dove questi possa essere modellato e il meccanismo che ne garantisce un'interazione sicura.

CAPITOLO 3

Un compito visuomotorio di interazione uomo-robot

3.1. Introduzione

Come detto nei capitoli precedenti, quello industriale è stato uno dei primi settori che ha favorito lo sviluppo della robotica, ancora oggi alla continua ricerca di nuove strategie e soluzioni per sistemi automatici. A differenza del passato, è oggi possibile (e auspicabile) introdurre dei sistemi robotici che siano più flessibili rispetto a quelli classici e, soprattutto, siano in grado di prendere alcune decisioni in maniera autonoma. Ovviamente il primo grado di flessibilità e autonomia di un robot industriale in un ambiente coabitato da esseri umani è quello di assicurarsi di non poter provocare danni agli utenti del sistema – e in seconda battuta, a se stesso. Se un robot ha un qualunque obiettivo (sia esso di assemblaggio, di trasporto) e deve svolgerlo in presenza di esseri umani, è giocoforza che sia in grado di rilevare la presenza degli utenti e di evitare collisioni potenzialmente pericolose[10]. La progettazione di un siffatto sistema robotico industriale prevede innanzitutto una modellazione degli agenti che cooperano nell'ambiente comune (l'utente umano e il robot) e una modellazione dell'interazione che parta da considerazioni di sicurezza se l'interazione prevede un contatto fisico. Oltre a una descrizione dell'ambiente attraverso le sue caratteristiche, è qui possibile, se non a volte necessario, dotarlo di proprietà che favoriscano l'interazione complessiva uomo-robot. Si noti come, in un tale approccio, si privilegi, durante la progettazione del sistema, partire da considerazioni sull'utente e la sua interazione piuttosto che da una modellazione del robot che risulta, in questo caso, funzionale all'interazione stessa.

[10] In seguito si prenderà in considerazione e si distinguerà quando l'interazione tra uomo e robot dovrà essere esclusivamente elusiva (ossia il robot dovrà evitare il contatto con gli esseri umani) e quando, invece, il robot dovrà "inseguire" un utente come nel caso in cui dovrà porgergli un oggetto.

3.2. Un Task Cooperativo

Si consideri un compito (*task*) generale, in ambiente industriale, di cooperazione uomo-robot che preveda la loro interazione fisica. Affinché tale compito sia significativo, si dovrà prevedere la possibilità, da parte del robot, di ricevere un ordine per poi eseguire in maniera autonoma le procedure necessarie per il suo espletamento. Durante il suo funzionamento, il robot dovrà navigare all'interno del suo ambiente sotto la condizione che questi possa essere coabitato da esseri umani. Infine, per l'espletamento del compito, ci dovrà essere una fase che prevede il contatto tra robot e utente o, comunque, la possibilità che la struttura del robot lo approcci l'essere umano.

Ad esempio, si può pensare al seguente task cooperativo: il robot ricevere, come ordine da un utente umano, la ricerca su un tavolo di lavoro di un determinato utensile richiesto dall'utente stesso e porgerglielo in una determinata posizione dello spazio. L'esecuzione del task globale dovrà, ovviamente, essere subordinata alla sicurezza dell'interazione uomo-robot: nei suoi vari spostamenti, il robot dovrà aver cura di non urtare l'utente o altri esseri umani che possono apparire nella scena.

Per la progettazione di un tale sistema si possono prevedere almeno i seguenti quattro passi:

1. modellazione appropriata dell'utente e del robot, dove per appropriata si intende che ne permetta la risoluzione del task cooperativo salvaguardando l'incolumità dell'utente;
2. individuazione della sensoristica necessaria alla modellazione del passo 1 e delle caratteristiche minime che l'ambiente di convivenza deve possedere;
3. individuazione di blocchi di comportamento che il robot deve esibire nelle varie fasi di funzionamento;
4. progettazione e implementazione dei moduli software/hardware.

Si noti, innanzitutto, che per il punto 2. alcune caratteristiche ambientali possono portare a scelte della sensoristica necessaria e, viceversa, l'utilizzo di particolari sensori porta a strutturare l'ambiente in un determinato modo. Inoltre, il punto 4 discende direttamente dai blocchi di comportamento individuati per la progettazione del sistema.

3.3. Modellazione uomo/robot

Per soddisfare i suoi obiettivi, il manipolatore si muove all'interno dell'ambiente coabitato da esseri umani e può, in alcuni casi, venire in contatto con essi. Affinché si evitino possibili collisioni pericolose, la navigazione del sistema robotico deve essere progettata e implementata con il vincolo principale della sicurezza dell'utente umano. La traiettorie che il manipolatore effettuerà per raggiungere i vari punti del suo spazio operativo dovranno tenere conto delle possibili interazioni con l'utente stesso e quindi non possono essere programmate a priori. È necessario, quindi, un meccanismo di navigazione sicura che tiene conto di tutte le problematiche discusse in precedenza. Per fare ciò si tiene conto dello schema di HRI sicura introdotto nel secondo capitolo, paragrafo 2.5.

Innanzitutto si deve poter modellare l'utente in maniera efficace. Le caratteristiche salienti da considerare per un essere umano in un ambiente industriale sono essenzialmente il volto e le mani. Il volto dell'utente è la zona più delicata da considerare e la traiettoria del manipolatore non dovrà mai intersecarlo nel suo percorso. Il sistema percettivo del robot, dunque, dovrà essere in grado di riconoscere la presenza di un volto nella scena coabitata dal robot e dall'uomo. L'algoritmo di visione che effettuerà tale riconoscimento dovrà godere essenzialmente delle seguenti proprietà: basso carico computazionale e quindi un tempo di campionamento del monitoraggio del volto il più basso possibile; robustezza del rilevamento del volto: il sistema dovrà essere sempre in grado di riconoscere la presenza dell'utente evitando soprattutto i falsi negativi; generalità rispetto ai possibili volti presenti nella scena, capacità di riconoscere un volto a prescindere, quanto più possibile, dalle sue caratteristiche di colore e luminosità.

Per quanto riguarda il riconoscimento delle mani, si è supposto che l'utente debba indossare dei guanti per poter interagire con il robot, sfruttando anche la possibilità di poter strutturare l'ambiente. Una tale ipotesi risulta ragionevole dato che nella maggior parte degli ambienti industriali l'uso dei guanti da lavoro è obbligatorio. Si ha così un modo di introdurre nell'ambiente una caratteristica distintiva che può permettere al sistema complessivo di migliorare le sue prestazioni. Innanzitutto, la presenza di guanti di un determinato colore permette di avere un modo di discriminare che tipo di utente è presente nella scena (colori differenti possono portare a distinguere tra utenti con compiti diversi e dunque con modalità di interazione possibili diverse); inoltre il carico computazionale del sistema di visione può essere alleviato in maniera sensibile dato che possono essere implementati degli algoritmi ad hoc particolarmente efficienti. Tale efficienza si esplica inoltre in un'alta capacità discriminatoria (capacità di distinguere e classificare con un'alta affidabilità porzioni dell'immagine con colori diversi) e un basso tempo di

elaborazione. L'introduzione di un segno distintivo di riferimento (*landmark*) porta, di contro, la necessità di ridurre la presenza nell'ambiente di elementi che possano, in qualche modo, trarre in inganno il sistema percettivo generando falsi positivi che ne pregiudichino l'affidabilità. La scelta dei colori come landmark può ridurre questo tipo di inconveniente dato l'ambiente di lavoro in esame: si può immaginare che in un'officina meccanica gli utenti debbano indossare tute da lavoro con determinati colori e che non sia particolarmente difficile connotare gli attrezzi presenti nella scena con colori determinati a priori.

Una volta modellato in maniera appropriata l'essere umano coinvolto nel compito di interazione e cooperazione, è necessaria una modellazione della struttura robotica quanto più funzionale possibile. Una conoscenza troppo dettagliata della struttura del manipolatore può risultare intrattabile, con troppi parametri da considerare e dunque elaborare, se si vuole che la reazione del manipolatore alla presenza dell'essere umano avvenga in tempo utile. Per fare ciò, si può schematizzare la struttura del braccio robotico con uno scheletro formato da segmenti che congiungono i suoi giunti associando a tali segmenti dei volumi di sicurezza che li ricoprono (anche l'essere umano sarà "ricoperto" da un suo volume di sicurezza). Si vedrà in seguito che una tale scelta permette da un lato di semplificare le procedure di calcolo per quanto riguarda il rilevamento delle distanze minime tra tutti i punti del manipolatore e l'utente umano e la susseguente reazione in caso di possibile collisione, dall'altro di tener conto degli errori di misura propri di un sistema robotico (errori per quanto riguarda il sistema propriocettivo sulla posizione del manipolatore e quelli riguardanti la posizione dell'essere umano per quanto riguarda il sistema esterocettivo).

3.4. Sensoristica utilizzata

In un compito di interazione cooperativa uomo robot questi deve essere in grado, essenzialmente, di poter rilevare la presenza di soggetti all'interno dell'area di lavoro e anche un poco all'esterno di essa. Inoltre, ha bisogno di poter effettuare la ricerca di un oggetto da afferrare posto su una postazione di lavoro predefinita. La presenza di due telecamere, dunque, è un requisito minimo per il compimento del task globale. Si deve prevedere, inoltre, la possibilità di disporre almeno di un sensore di forza posto sull'organo terminale del braccio robotico che permetta di rilevare una forza esercitata in punta sul manipolatore che può essere interpretata

come la raccolta, da parte dell'utente, dell'oggetto afferrato[11]. Riassumendo, la sensoristica minima prevista per il compito in esame è la seguente:

- una telecamera deputata al rilevamento di esseri umani presenti nella scena $\boldsymbol{p}_{hum}=(x_{hum}, y_{hum}, z_{hum})$ nei termini definiti dalla modellazione dell'utente;
- una telecamera (che può essere solidale all'*end effector* del braccio robotico o fissa al di sopra del tavolo di lavoro) che individui le coordinate degli attrezzi $\boldsymbol{p}_{obj}=(x_{obj},y_{obj})$ che servono all'utente;
- un sensore di forza in punta all'end-effector $\boldsymbol{f}_{usr}=(f_x,f_y,f_z)$.

Di seguito, la prima videocamera sarà indicata con CAM1, la seconda con CAM2, e il sensore di forza con FORCE. Da notare che la posizione degli oggetti è espressa attraverso coordinare bidimensionali dato che si può supporre che l'altezza del tavolo da lavoro sia nota; la collocazione dell'essere umano è rappresentata, invece, dalla sua posizione x,y nel piano immagine della telecamera e da una terza coordinata, z, che ne rappresenta la distanza da essa. In seguito, quando si tratterà di integrare il sistema di visione con il sistema di controllo del manipolatore, sarà necessario una trasformazione di coordinate dal sistema solidale alla telecamera a quello definito "sistema mondo" comune alla videocamera e al manipolatore.

3.5. Struttura dell'ambiente

Un ambiente industriale è di solito costruito *ad hoc* per poter soddisfare degli standard di sicurezza obbligatori per le aziende. Se si vuole che il sistema robotico interagisca fisicamente con gli utenti, la modellazione dell'ambiente non può essere fatta considerando solo le caratteristiche del robot (la sua struttura, i suoi sensori, i suoi attuatori e il suo sistema di controllo) ma, piuttosto, deve partire da considerazioni che permettano e favoriscano l'interazione. Ad esempio, un manipolatore robotico, è caratterizzato da uno spazio operativo limitato: avendo una base fissa e lunghezza dei bracci limitata, non può raggiungere tutte le posizioni dello spazio[12]. L'area di interazione possibile tra robot e utente viene, così, a delimitarsi; inoltre la conoscenza a priori

[11] In realtà l'oggetto potrà essere rilasciato non solo in base al valore registrato dal sensore in forza ma anche dal fatto che il robot si trova in una "posizione di interazione" (vedere in seguito).

[12] Le direttive in campo industriale prevedono, ad esempio, che se un braccio industriale come un COMAU è in movimento, non sia possibile la presenza di alcun essere umano all'interno di una gabbia di sicurezza che delimita lo spazio operativo del manipolatore

di tutte le possibili posizioni del manipolatore può facilitare e guidare la progettazione del sistema complessivo. Oltre allo spazio operativo (*workspace*) del manipolatore, si possono individuare zone o aree di lavoro e alcuni punti "fissi" all'interno dell'ambiente stesso. Possibili aree di lavoro individuabili sono:

- Volume 1: comprende il tavolo di lavoro e la posizione di riposo del robot;
- Volume 2: comprende la "zona di interazione" e la posizione di riposo del robot;
- Volume 3: è la zona al di fuori del *workspace* del braccio meccanico. Non è raggiungibile dall'*end-effector* – area di non interazione.

L'individuazione di questi volumi permette, in generale, di rendere più flessibile il comportamento complessivo del robot: a seconda della presenza dei vari soggetti (robot, oggetti da prendere, esseri umani) all'interno dei tre volumi, il sistema potrà essere in grado di effettuare differenti valutazioni sul suo stato ed esibire quindi un comportamento flessibile. Oltre queste aree di lavoro, è possibile individuare dei punti, all'interno dell'ambiente, salienti per le operazioni che il robot dovrà effettuare. Questi potranno essere una:

- posizione di riposo del robot $\boldsymbol{p}_{rest}=\{x_0,y_0,z_0\}$;
- posizione sul tavolo di lavoro $\boldsymbol{p}_{table}=\{x_{table},y_{table},z_{table}\}$ dove l'*end effector* si andrà a collocare per iniziare l'esplorazione del tavolo di lavoro;
- posizione di interazione $\boldsymbol{p}_{int}=\{x_{int},y_{int},z_{int}\}$ dove l'utente è autorizzato a prendere gli oggetti che il robot gli deve porgere.

Indicheremo, nel prosieguo del capitolo, con $\boldsymbol{p}_a=\{x_a,y_a,z_a\}$ la posizione generica al tempo *t* dell'*end effector*.

3.6. Architettura complessiva del sistema

L'attività del sistema robotico come successione di stati per il soddisfacimento del task può essere rappresentata con un automa a stati finiti (cfr capitolo 1 paragrafo 1.3.3.1). Tale rappresentazione (figura 1) risulta utile in quanto permette di evidenziare le varie fasi di esecuzione del sistema e permette, così, di costruire dei moduli indipendenti che possono essere progettati, implementati e testati in maniera autonoma gli uni dagli altri. Per ogni modulo bisogna poi descrivere gli input che riceve dai sensori o da altri moduli ad esso collegati e gli output relativi all'elaborazione propria del modulo stesso. Inoltre, alcune variabili (o parametri) del sistema aiuteranno a descrivere lo stato del sistema e discrimineranno, in alcuni casi, il passaggio da uno stato a un altro. Come vedremo, inoltre, alcuni blocchi rappresenteranno l'interazione fisica tra uomo e robot e potranno essere progettati e implementati secondo lo schema descritto in precedenza (cfr. capitolo 2 paragrafo 2.5).

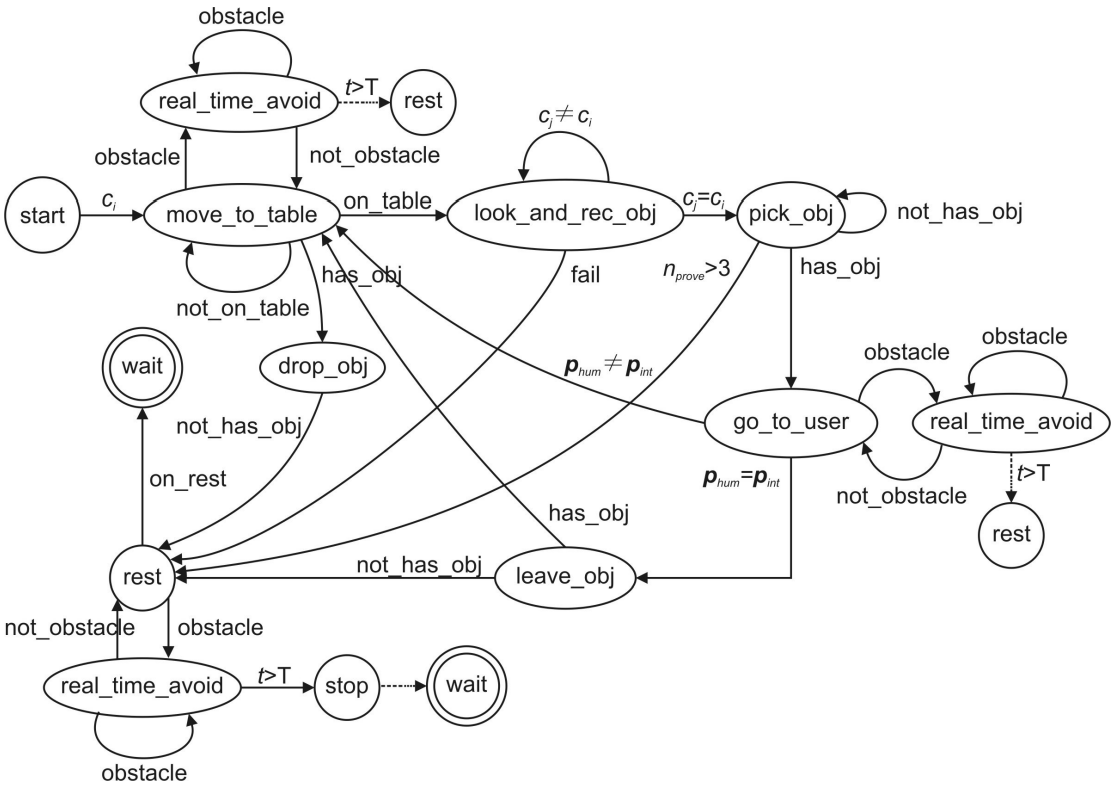


figura 1 – rappresentazione FSA

L'insieme degli stati dell'automa è Q = {*start*, *move_to_table*, *real_time_avoid*, *look_and_recognize_object*, *pick_object*, *go_to_user*, *leave_object*, *drop_object*, *rest*, *stop*, *wait*}, lo stato iniziale è *start*, quello finale (per singolo loop) è *wait*. Inoltre, si può introdurre una variabile di sistema has_object che rappresenta un parametro interno del sistema che indica se il robot ha un oggetto all'interno del *gripper* o meno.

La descrizione dei singoli stati è la seguente:

start: input: si attiva quando riceve il comando prendi_oggetto{c_i} e la variabile has_object == NULL. L'output dello stato è l'attivazione di *move_to_table*, al quale viene passato come parametro l'oggetto da cercare c_i.

move_to_table: in questo stato il manipolatore si sposta dalla sua posizione di riposo fino a quella al di sopra del tavolo da lavoro dove sono collocati gli attrezzi. Finché il manipolatore non raggiunge tale posizione, il sistema rimane in tale stato: la condizione di uscita di *move_to_table* è $\mathbf{p}_a(t) == \mathbf{p}_{tab}$ ossia il braccio ha raggiunto il punto da dove iniziare la ricerca dell'oggetto da prendere o, se tale punto risulta occupato, una posizione all'interno di un intervallo di confidenza. Se in tale stato has_object == FALSE (in figura la variabile non è rappresentata per mancanza di spazio), il sistema passa il controllo a *look_and_recognize_object*; se *has_object*==TRUE, il braccio si colloca nella posizione $\mathbf{p}_{ci}$ (vedi *pick_object*) e passa nello stato *drop_object*. *move_to_table* deve generare, attraverso lo stato real_time_avoid, le traiettorie sicure per raggiungere la sua posizione target. Se la telecamera fissa rileva la presenza di un essere umano che interseca la traiettora del robot, questi passa nello stato real_time_avoid.

real_time_avoid: in questo stato si gestisce la rappresentazione dell'utente che meglio si adatta alle caratteristiche del sistema globale. Nel caso specifico di un progetto di interazione con un compito di presa e consegna di oggetti, il monitoraggio dell'utente prevede il tracciamento del volto e delle mani di quest'ultimo. Tali posizioni dovranno essere processate in maniera opportuna per far si che il manipolatore, durante l'esecuzione delle sue traiettorie possibili, non abbia un contatto improprio con l'essere umano[13]. L'attivazione dello stato *real_time_avoid*[14] avviene sempre se la traiettoria programmata del manipolatore interseca quella del volto dell'utente mentre questo non avviene se si considerano le sue mani. Quando il manipolatore si muove verso il tavolo (*move_to_table*) o verso la posizione di riposo (*rest*), le mani devono essere salvaguardate e dunque il comportamento del manipolatore sarà quello di evitarle, se, invece, l'utente deve prendere l'oggetto dal manipolatore, le sue mani non rappresenteranno un ostacolo

[13] come si vedrà in seguito, nel caso pratico presentato nel seguente lavoro di tesi, l'idea per generare delle traiettorie sicure è quella di considerare dei "*volumi di sicurezza*" centrati sulle posizioni rilevate da CAM2

[14] Può essere considerato come l'interfaccia tra sensore e robot o un "sensore virtuale".

(ossia non si genereranno traiettorie atte ad allontanarsi da esse) bensì un attrattore da avvicinare, comunque, con cautela (si rallenta la velocità del manipolatore in fase di approccio). Per evitare situazioni di stallo, in cui il manipolatore tenta in maniera indefinita di raggiungere una determinata posizione, si può limitare nel tempo (con delle variabili T_1 e T_2) l'effetto di questo comportamento: se, dopo un certo intervallo di tempo in cui il manipolatore cerca di raggiungere la sua posizione target evitando collisioni, il sistema dovrà cambiare la sua posizione obiettivo o tornando nella sua posizione di riposo o fermandosi definitivamente.

look_and_recognize_object: tale stato si attiva quando move_to_table ha successo e gli manda in ingresso c_i, la variabile che contiene l'informazione sull'oggetto da riconoscere. look_and_recognize_object cerca di riconoscere l'oggetto richiesto e individuare le coordinate del suo centro di massa[15] per poi attivare lo stato pick_object. Qualora il sistema non fosse in grado di riconoscere l'oggetto (fail) si genera un segnale di *warning* all'utente e il controllo passa nello stato *rest*.

pick_object: una volta individuate le coordinate del centro di massa dell'oggetto (o comunque del punto di presa), in pick_object si genera ed esegue la sequenza di azioni per afferrare l'oggetto. Se l'azione ha successo, la variabile has_object è impostata a TRUE e vengono memorizzate le coordinate $\mathbf{p}_{obj} = \{x_{obj}, y_{obj}\}$ dove si trovava l'oggetto e si attiva lo stato *go_to_user*; altrimenti, dopo, ad esempio, tre tentativi fatti per diverse angolazioni del gripper, si ha o un segnale che avverte l'utente di tale fallimento e/o il riposizionamento nella posizione di riposo, passando allo stato rest.

drop_object: tale stato si attiva quando bisogna ricollocare l'oggetto nella posizione sul tavolo da dove è stato preso in un primo momento. L'attivazione di drop_object sottende la condizione che il manipolatore, dopo aver eseguito l'azione di presa dell'oggetto, non sia riuscito per qualche motivo a consegnarlo all'utente (ad esempio quest'ultimo è uscito dalla scena). Si esce da questo stato quando si apre il *gripper* e la variabile has_object viene impostata a FALSE. Quando drop_object ha successo, si passa allo stato rest, ossia il manipolatore si andrà a ricollocare nella sua posizione iniziale di riposo.

go_to_user: ha l'obiettivo di spostare l'end effector del manipolatore dalla posizione che occupava dopo aver preso l'oggetto fino alla zona di interazione con l'utente. Per fare ciò, bisogna controllare che nel punto di interazione $\mathbf{p}_{int}$ ci sia un utente umano (monitorato da CAM2), altrimenti calcolare il nuovo punto di arrivo come quello all'interno di una sfera predefinita (l'intorno sferico del punto dove si presume che comunque l'interazione possa aver luogo) e più prossimo a $\mathbf{p}_{int}$. Si generano quindi i comandi motori opportuni e la condizione di

[15] Si potrà avere, ad esempio, che il centro di massa è noto (noto l'oggetto) oppure lo si dovrà calcolare una volta riconosciuto l'oggetto

uscita è $\mathbf{p}_a = \mathbf{p}_{int}$ e $\mathbf{p}_{hum} = \mathbf{p}_{int}$. Se $\mathbf{p}_{hum} \neq \mathbf{p}_{int}$ (l'utente non è presente nel volume di interazione) e il controllo passa nello stato *move_to_table* per posare, poi, l'oggetto.

leave_object: se la forza in punta dovuta all'interazione con l'utente è maggiore di una certa soglia $\mathbf{f}_{thres}$, il gripper si apre, l'azione ha successo, la variabile has_object è impostata a FALSE e il sistema passa nello stato rest. Altrimenti il controllo viene passato allo stato move_to_table.

rest: è lo stato in cui si effettua il ricollocamento dell'end effector nella sua posizione di riposo salvaguardando la sicurezza degli esseri umani presenti nella scena. Ciò accade se: (a) l'azione leave_object ha avuto successo, (b) l'utente è fuori dal raggio di azione del manipolatore e il braccio ha posato l'oggetto c_i sul tavolo. L'output di tale modulo sono i comandi motori per raggiungere la posizione $\mathbf{p}_{rest}$. Una vola che $\mathbf{p}_a == \mathbf{p}_{rest}$, il sistema passa nello stato wait in attesa di ricevere ulteriori ordini. Da notare che se da rest si passa a real_time_avoid (ossia si incontra un ostacolo) dopo un periodo di tempo T che il manipolatore ha provato a raggiungere la posizione di riposo, il sistema si blocca (stato stop) in attesa di ulteriori ordini. Una tale procedura è resa necessaria affinché il sistema non entri in una situazione di stallo.

In figura 2 è rappresentata la tabella delle funzioni di transizione tra stati.

δ	Q	*input*	$\delta(q,input)$
	Start	c_i	move_to_table
	move_to_table	not_on_table && not_obstacle	move_to_table
	move_to_table	on_table && not_has_object	look_and_rec_obj
	move_to_table	obstacle	real_time_avoid
	move_to_table	on_table && has_object	drop_object
	real_time_avoid	obstacle	real_time_avoid
	real_time_avoid	obstacle && t>T	rest
	real_time_avoid	obstacle && try_rest && t>T	stop
	real_time_avoid	not_obstacle	move_to_table
	look_and_rec_obj	$c_j \neq c_i$	look_and_rec_obj
	look_and_rec_obj	$c_j = c_i$	pick_obj
	look_and_rec_obj	*fail*	rest
	pick_obj	not_has_obj & $n_{prove} \leq 3$	pick_obj
	pick_obj	not_has_obj & $n_{prove} > 3$	rest
	pick_obj	has_obj	go_to_user
	go_to_user	$\mathbf{p}_{hum} \neq \mathbf{p}_{int}$ && not_obstalce	move_to_table

	go_to_user	$\mathbf{p}_{hum}=\mathbf{p}_{int}$ && $\mathbf{p}_{a}=\mathbf{p}_{int}$ && not_obstacle	leave_object
	go_to_user	$\mathbf{p}_{hum}=\mathbf{p}_{int}$ && Obstacle	real_time_avoid
	leave_object	has_object	move_to_table
	leave_object	not_has_obj	rest
	drop_object	not_has_obj	rest
	Rest	Obstacle	real_time_avoid
	Rest	not_on_rest && not_obstacle	rest
	Rest	on_rest	wait
	Stop	All	wait

figura 2 – rappresentazione della funzione di transizione

Il comportamento "a regime" del sistema può essere, inoltre, descritto come segue: il robot riceve il comando, da un utente, di ricercare un utensile posto su un tavolo di lavoro. Il primo compito del robot è quello, dunque, di posizionarsi in una determinata posizione al di sopra di esso per cercarlo. Una volta trovato, lo afferra con il *gripper* posto all'estremità dell'end effector e lo porta in una zona dell'ambiente dove si presume che l'utente sia pronto per riceverlo (o meglio, è previsto che lui sia presente per l'interazione). Se il sensore di forza si trova nella zona di spazio consentita per l'interazione e rileva un valore al di sopra di una certa soglia rilascia l'oggetto. In ogni fase in cui il manipolatore si sposta all'interno dell'ambiente, si deve attivare lo stato *real_time_avoid* che permette la navigazione "sicura" del braccio. Riassumendo, l'*interazione* uomo/robot si esplica essenzialmente: nella richiesta dell'utensile; quando il manipolatore si deve collocare nei vari punti dell'ambiente ed evitare gli utenti presenti nella scena, e, infine, quando l'utente raccoglie l'utensile esercitando una forza in punta dell'end effector. Si noti che una rappresentazione unica delle mani dell'utente può essere usata dal sistema in maniera differente nelle diverse fasi dell'interazione: in un caso rappresentano "ostacoli" in un altro punti di attrazione. La *collaborazione* tra il robot e l'uomo, inoltre, è subordinata al riconoscimento, da parte del sistema, dell'oggetto da porgere all'utente.

3.7. Il Progetto PAVA

Il compito cooperativo precedentemente esposto è stato realizzato nell'ambito del progetto PAVA (Percezione Attesa & Visione Attiva), finanziato dalla Regione Campania e svolto in collaborazione con il dipartimento di Informatica e Sistemistica della Facoltà di Ingegneria della "Federico II" e con il dipartimento di Ingegneria dell'Informazione e Ingegneria Elettrica dell'Università di Salerno. L'obiettivo di tale progetto è stato lo sviluppo e l'applicazione di teorie dell'HRI (*Human-Robot Interaction*) in ambito industriale allo scopo di ottenere un sistema che interagisca fisicamente con gli utenti umani in maniera flessibile e sicura (salvaguardando l'incolumità prima dell'utente e poi del sistema robotico mentre si cerca di soddisfare gli obiettivi di collaborazione).

Per descrivere meglio i moduli software implementati, riconsideriamo gli schemi per l'interazione uomo-robot e l'architettura del sistema complessivo descritti nei paragrafi precedenti (crf cap. II paragrafo 2.4 e cap. III paragrafo 6). In figura 3 è riproposta l'architettura del sistema con evidenziati i blocchi che rappresentano le peculiarità del sistema.

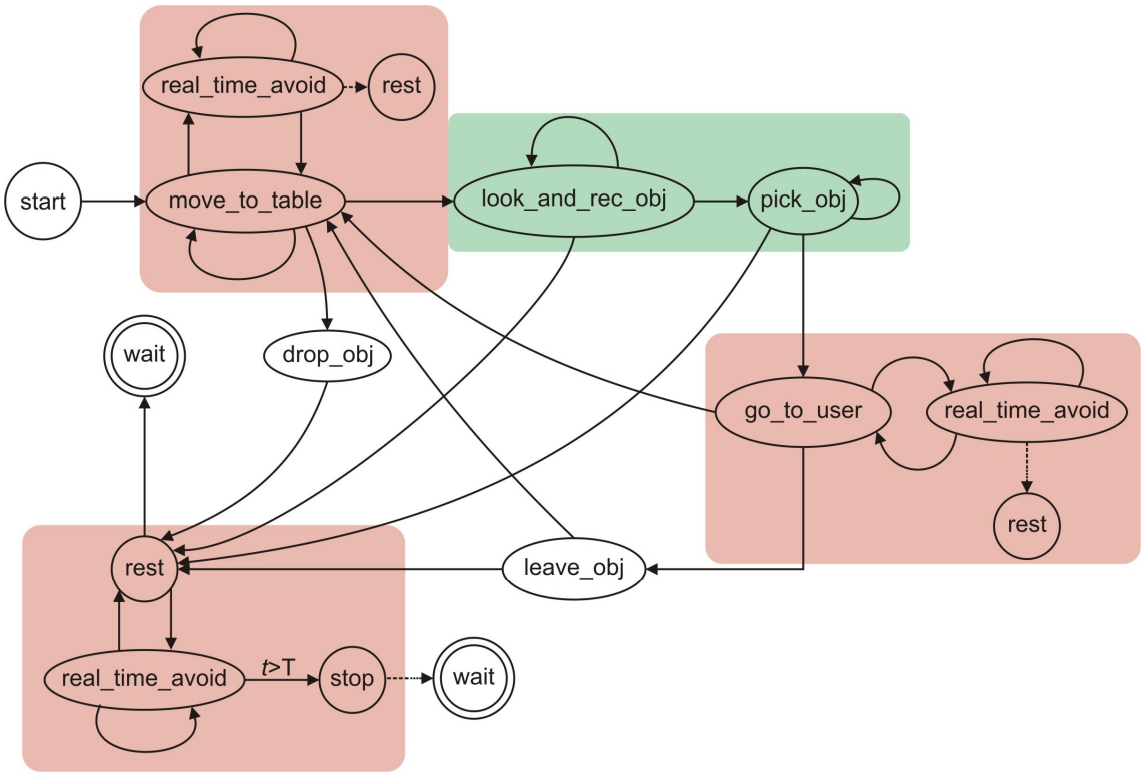


figura 3 – architettura del sistema rivista come gruppi software

Nel progetto PAVA i sensori che forniscono le informazioni più salienti sono le telecamere: quella fissa (la CAM2 vista in precedenza), monitorando gli esseri umani presenti del sistema, governa in qualche modo l'interazione fisica tra robot e utente, quella montata sull'organo terminale del manipolatore (CAM1) consente al sistema di espletare il suo compito di manipolazione. In figura 3 in verde è evidenziata la parte riguardante la parte di riconoscimento e presa dell'oggetto mentre in rosso è evidenziato il blocco software che garantisce la navigazione sicura del manipolatore (blocco progettato, come vedremo, secondo lo schema di interazione sicura presentato nel precedente capitolo). I blocchi in figura 3 al di fuori delle zone colorate sono di implementazione intuitiva e, di conseguenza, non si ritiene necessaria una loro descrizione dettagliata.

3.7.1. Visual Based Manipulation

Una delle funzionalità di base che deve possedere un robot di servizio è la capacità di manipolare oggetti in un ambiente semi-strutturato [Christensen et al, 2003]. Uno dei task più comuni è quello del "*fetch – and – carry*". Un compito di "presa" dove si usa la visione quale sensore esterocettivo può essere suddiviso, in generale, in diversi subtask:

i. **Rilevazione/ Detection**: data una immagine della scena, il sistema di visione deve essere in grado di fornire una risposta (binaria) al quesito se l'oggetto desiderato è presente o meno nella scena stessa. Ciò implica che nel campo visivo della videocamera possano esserci oggetti simili a quello da ricercare o che l'oggetto stesso non sia presente.

ii. **Segmentazione/Segmentation**: se l'oggetto è stato rilevato, bisogna ricercare e ritagliare la porzione di immagine dove è effettivamente presente l'oggetto. Questo ulteriore passo serve anche ad affinare l'ipotesi che l'oggetto rilevato sia effettivamente quello richiesto.

iii. **Riconoscimento/Recognition**: l'oggetto è stato riconosciuto e si forniscono misure sul grado di riconoscimento. A seconda dell'implementazione, questa parte può anche fornire una posizione parziale/totale dell'oggetto [Tarabanis et al, 1995].

iv. **Allineamento/Alignment**: dopo che l'oggetto è stato localizzato, il manipolatore deve arrivare in prossimità dell'oggetto (in generale viene fatto con un controllo pilotato dal sistema visivo). Molti sistemi robotici ignorano i primi tre passi e si concentrano solo sull'aspetto dell'allineamento. Gli approcci differiscono nel numero di videocamere usate (stand-alone vs eye-in-hand) e per lo spazio di controllo usato (basato sull'immagine, basato sulla posizione, usando un sistema 2D 1/2 di visual servoing). Per effettuare l'allineamento attraverso un loop-chiuso, il sistema deve essere in grado di "inseguire" le caratteristiche dell'oggetto durante i movimenti del braccio e, se necessario, aggiornare la loro posizione (specie se gli oggetti nel frattempo si sono mossi o l'allineamento e la presa vengono effettuate quando il robot si muove).

v. **Presa dell'oggetto/Grasping**: dopo aver effettuato l'allineamento, inizia la sequenza di grasping. Per oggetti semplici (scatole, cilindri) si può pensare a modalità di apprendimento che vengono usate a seconda della posizione corrente dell'oggetto. Per oggetti, invece, che possono risultare più difficili da afferrare, si dovrà considerare la fusione dei dati provenienti dal sensore visivo con quelli dei sensori di forza (o di contatto) per permettere delle prese più flessibili.

vi. **Manipolazione/Manipulation**: in molti casi, si richiede al sistema robotico di porgere l'oggetto in un modo particolare, ad esempio per incastrarlo in un altro. Ciò può richiedere che l'oggetto debba essere manipolato prima di essere rilasciato dall'end-effector.

Tutti questi punti sono stati studiati in maniera diffusa in diversi settori di ricerca: visione artificiale (*computer vision*), controllo, ottimizzazione, integrazione sensoriale etc. etc. Ciononostante, la loro interazione e integrazione è un'area ancora abbastanza nuova.

A dispetto della loro complessità computazionale, l'uso di modelli CAD è uno degli approcci più popolari per risolvere la maggior parte di questi punti: stima della posizione e tracking, [Lowe, 1991], [Koller et al., 1993], allineamento, [Merchand et al., 2000], [Wunsch e Hirzinger, 1997], e grasping, [Miller e Allen, 1999], [Bicchi e Kumar, 2000].

Nel progetto PAVA, per i punti i), ii), e iii), si è usato un approccio differente.

3.7.1.1. Riconoscimento degli oggetti tramite la Trasformata SIFT

La tecnica SIFT (Scale Invariant Feature Transform) [Lowe, 2004] è un metodo che nasce per estrarre caratteristiche distintive invarianti dalle immagini e che permette di eseguire dei confronti tra viste differenti di un oggetto o di una scena. L'idea di base della trasformata SIFT è quella di generare una rappresentazione sparsa dell'immagine in uno spazio euclideo di caratteristiche che siano altamente distintive, nel senso che vedremo in seguito, ma anche invariante per trasformazioni applicabili all'immagine. Tali vettori di caratteristiche vengono poi usati per individuare delle corrispondenze tra punti distintivi appartenenti alla scena oppure a uno specifico oggetto raffigurato da differenti punti di vista. Per "altamente distintive", si intende che una singola caratteristica può essere confrontata, con un'alta probabilità, con un ampio database di caratteristiche formato da molte altre immagini. Ciò implica che è possibile individuare correttamente la presenza di un singolo vettore con alta probabilità anche all'interno di un database molto ampio di vettori estratti da altre immagini.

Tra i motivi che rendono la trasformata SIFT interessante per l'applicazione di fetch and carry, ci sono:

- il carico computazionale relativamente basso;
- la peculiarità che i vettori di caratteristiche risultano invarianti per traslazioni e rotazioni 2D nel piano dell'immagine, e per la ridefinizione del fattore di scala tra le immagini.

Tali invarianze sono il risultato del metodo mediante il quale i vettori di caratteristiche sono estratti dall'immagine. Inoltre, i risultati sperimentali hanno dimostrato che si ottengono buoni risultati anche quando l'oggetto è sottoposto a distorsioni affini e, soprattutto, aggiunta di rumore nell'immagine e variazione di luminosità nell'ambiente (per i dettagli sulla trasformata SIFT si rimanda all'appendice A).

L'utilizzo della trasformata SIFT fornisce le informazioni necessarie per i punti i, ii e iii descritti nel paragrafo precedente. Tale metodo fornisce anche l'angolo, rispetto al piano dell'immagine e dunque al piano di lavoro dove sono posizionati gli oggetti da prendere, e l'orientazione degli oggetti stessi. In questo modo, l'allineamento che dovrà eseguire il manipolatore per effettuare il movimento di presa con il gripper risulta altamente facilitato. La robustezza e affidabilità rispetto al rumore presente nella scena e ai cambi di illuminazione ambientale sono altre caratteristiche che lo rendono adatto a un ambiente industriale quando la

telecamera preposta al riconoscimento dell'oggetto è solidale all'organo terminale del manipolatore.

3.7.2. Navigazione sicura del manipolatore

La modellazione dell'utente umano e del manipolatore influenza l'implementazione di un modulo di real time safety controller efficiente. Gli algoritmi di visione che rivelano la presenza di un volto nella scena e di cosiddetti blob[16] di determinati colori oltre a dare una risposta binaria sulla presenza o meno nella scena di queste feature, devono anche essere in grado di fornire informazioni sulla loro posizione all'interno dell'ambiente istante per istante. L'algoritmo di visione per il rilevamento del volto deve essere anche in grado di individuare un rettangolo (*box*) all'interno del quale è racchiuso il volto, così come l'algoritmo di color detection deve essere in grado di calcolare il centro del *cluster* di pixel del colore ricercato. A partire dalla rilevazione di tali punti (il centro del volto e delle mani) si creano anche in questo caso dei "volumi di sicurezza" che li avvolgono e si controlla, istante per istante, che non si intersechino con i volumi della struttura dell'apparato robotico, durante l'esecuzione delle sue traiettorie. Quando ciò dovesse accadere, si attiva il meccanismo reattivo che modifica la traiettoria del manipolatore generando i comandi motori opportuni.

3.7.2.1. Algoritmi per il rilevamento del volto e delle mani

In letteratura sono stati presentati molti algoritmi e soluzioni per il riconoscimento di un volto in una scena attraverso l'uso di una videocamera. La soluzione da adottare in questo caso deve rispondere a due requisiti fondamentali: rapidità di esecuzione e minimizzazione dei falsi negativi. La necessità di un algoritmo rapido per il riconoscimento della posizione della testa nasce dal fatto, ovvio, che il manipolatore deve essere in grado di modificare in tempo utile la sua traiettoria se il volto è rilevato a una distanza inferiore a quella di sicurezza. D'altro canto, la rapidità di riconoscimento non può andare a discapito della robustezza del sistema: bisogna essere sicuri di riconoscere sempre ogni volto presente nella scena limitando, per quanto possibile, i casi in cui il sistema di visione non riconosca volti laddove non si presentino. Per fare ciò, si è adottata la soluzione proposta da Viola e Jones in [Viola e Jones, 2004]. Rispetto ad altre soluzioni proposte in letteratura (ad esempio [Schneiderman e Kanade, 2000],[Roth et al., 2000]),

[16] Con il termine blob si intendono insiemi di pixel dell'immagine connessi tra di loro. Il blob è, così, una porzione connessa dell'immagine formata da pixel dello stesso colore.

tale *framework* è capace di elaborare immagini estratte da un flusso video in tempi estremamente rapidi raggiungendo percentuali di riconoscimenti molto alte e garantendo, al contempo, la velocità di elaborazione richiesta.

Il successo di tale applicazione per la face detection si fonda su tre punti chiave. Il primo è l'introduzione di una nuova rappresentazione delle immagini chiamata *integral image* che permette una valutazione delle *feature* dell'immagine stessa molto rapida. Il secondo è l'adozione di un classificatore semplice ed efficiente che è costruito scegliendo un numero piccolo di feature più importanti da un'ampia libreria di *feature* potenziali utilizzando AdaBoost[17] [Freund e Schapire, 1995]. Per garantire dei tempi di elaborazione bassi, il processo di apprendimento deve escludere la maggior parte delle *feature* disponibili e focalizzarsi su un insieme piccolo di caratteristiche più importanti. La selezione delle *feature* viene effettuata, quindi, attraverso l'algoritmo di apprendimento AdaBoost usato per un processo di selezione delle caratteristiche. Il terzo aspetto rilevante è l'utilizzo di un metodo per combinare in successione vari classificatori, vieppiù complessi, in cascata. In questo modo è possibile aumentare in maniera decisiva la velocità di rilevazione focalizzando l'attenzione su regioni dell'immagine che risultano più "promettenti". L'idea dietro gli approcci che utilizzano i "fuochi di attenzione" è quella di determinare in maniera rapida in quale parte dell'immagine sia più "promettente" trovare un volto ([Tsotsos et al, 1995],[Itti et al, 1998],[Fleuret e German, 2001]). A queste regioni viene riservata un'elaborazione più complessa (per una descrizione più dettagliata, si rimanda all'appendice A).

Per quanto riguarda il riconoscimento di blob di colore rosso e blu, è stato utilizzato l'algoritmo descritto in [Boccignone et al., 2004] che fa riferimento, sviluppandolo, al tradizionale algoritmo EM (Expectation and Maximization) [Dempster et al., 1977]. L'algoritmo EM è uno tra i metodi più usati per la segmentazione delle immagini, in questa sua variante si rileva particolarmente efficace quando si segmentano le immagini e si classificano in base al colore della regione classificata, ottenendo, così, un algoritmo particolarmente affidabile.

[17] AdaBoost è una tecnica di classificazione che si basa sulla combinazione di diverse realizzazioni di uno stesso classificatore di base. Ad ogni punto viene assegnato un peso ed iterativamente vengono costruiti dei modelli dipendenti dai pesi. Ad ogni iterazione i pesi sono modificati. In particolare, quelli relativi ai campioni maggiormente sbagliati vengono incrementati e quelli relativi ai campioni classificati correttamente vengono decrementati. L'algoritmo è così forzato a "concentrarsi" sui campioni più difficili. Il classificatore finale è ottenuto con una votazione pesata dei modelli realizzati. L'algoritmo AdaBoost è uno dei metodi di *boosting* più utilizzati e di maggior successo, ma, mentre preserva la sua generale e pratica applicabilità, fornendo ottime prestazioni su dati non rumorosi, lavori teorici e applicativi mostrano che può incorrere nel problema del sovrapprendimento (*overfitting*) quando applicato a dati rumorosi.

3.7.2.2. Il modulo di visione complessivo

Il sistema di visione complessivo ha dunque le seguenti caratteristiche:

a. è in grado di riconoscere un volto e identificare la sua posizione all'interno di una terna di assi solidale al centro della videocamera;

b. classifica blob di colore rosso e blu individuandone la posizione all'interno del piano della videocamera. In questo caso non ci si è preoccupati di individuare la loro distanza dall'origine degli assi della videocamera in quanto, in prima approssimazione, la possiamo dedurre dalla posizione del volto.

In figura 4 è mostrata un'istantanea del sistema globale. Nella finestra "`Eye`" si visualizza il flusso (*streaming*) video della videocamera, la finestra "Color and face Detection" evidenzia il volto e i blob rosso e blu individuati, in quella "Blob Detection for Red" e "Blob Detection for Blue" sono presenti esclusivamente i blob rispettivamente rosso e blu discriminati dal sistema. Nella console sono stampati a video i risultati dell'elaborazione del modulo di vision. Nell'area evidenziata in verde ci sono i risultati relativi alla *frame* di riferimento. Più precisamente si ha:

- Current Pan position e Current Tilt position indicano le posizioni correnti di pan e tilt della videocamera;
- Red Blob Coordinates e Blue Blob Coordinates sono le coordinate nel piano dell'immagine rispettivamente del centro del blob rosso e del centro di quello blu. La coordinata (0,0) è fissata al centro del piano della telecamera (figura 5);
- Con x-> e y-> si indicano le coordinate del centro del volto riconosciuto nel sistema di riferimento assoluto della videocamera (gli assi in rosso di figura 2);
- Con Camera Coordinates si riportano le coordinate della videocamera rispetto alle nuove posizioni di pan e tilt;
- Infine Face Coordinates riporta le coordinate del centro del volto rispetto al sistema di riferimento centrato nell'origine dell'immagine. La coordinata z si riferisce alla distanza del volto dal centro della telecamera.

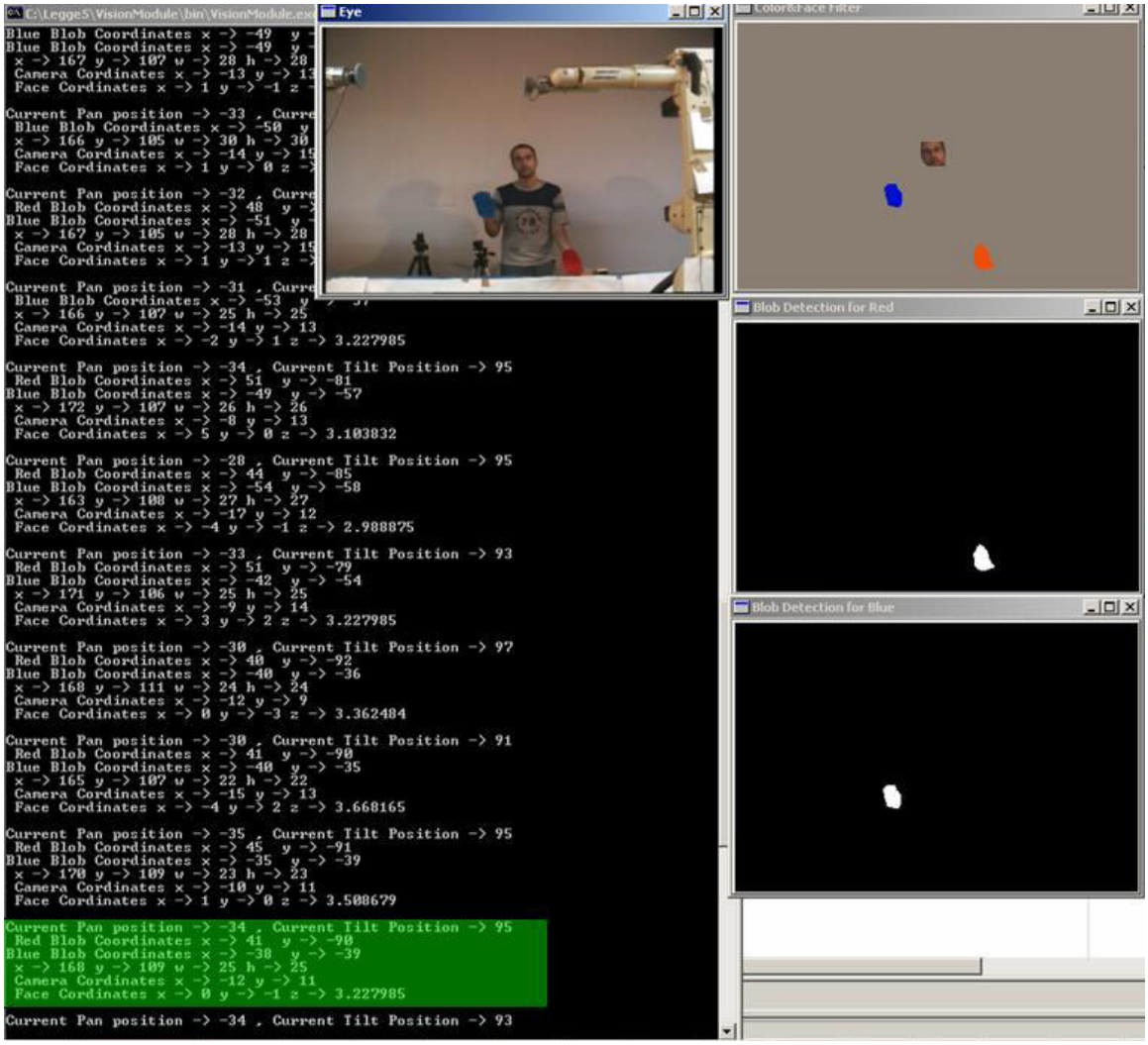


Figura 4 – snapshot del sistema globale

In figura 5 è mostrato un esempio di fotogramma (*frame*) elaborato dal sistema con evidenziati i due sistemi di riferimento utilizzati per l'individuazione delle posizioni del volto (per le mani si utilizzano gli stessi riferimenti). In figura 6, invece, sono riportate le varie analisi visive. Si noti che la possibilità di avere come interfaccia del sistema una schermata che riassuma in maniera visiva più intuitiva le varie elaborazione possa essere d'aiuto per una prima analisi delle prestazioni del sistema di visione. In figura 7, inoltre, sono riportati i risultati dell'elaborazione relativi alla frame di figura 5.

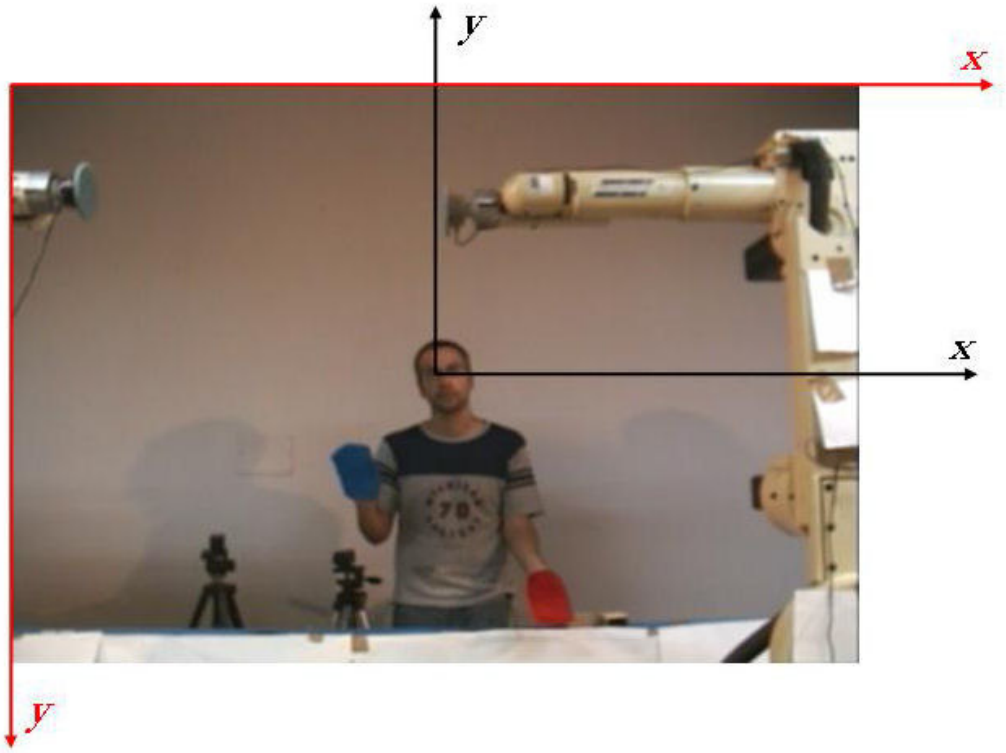


Figura 5 – sistema di riferimento della videocamera

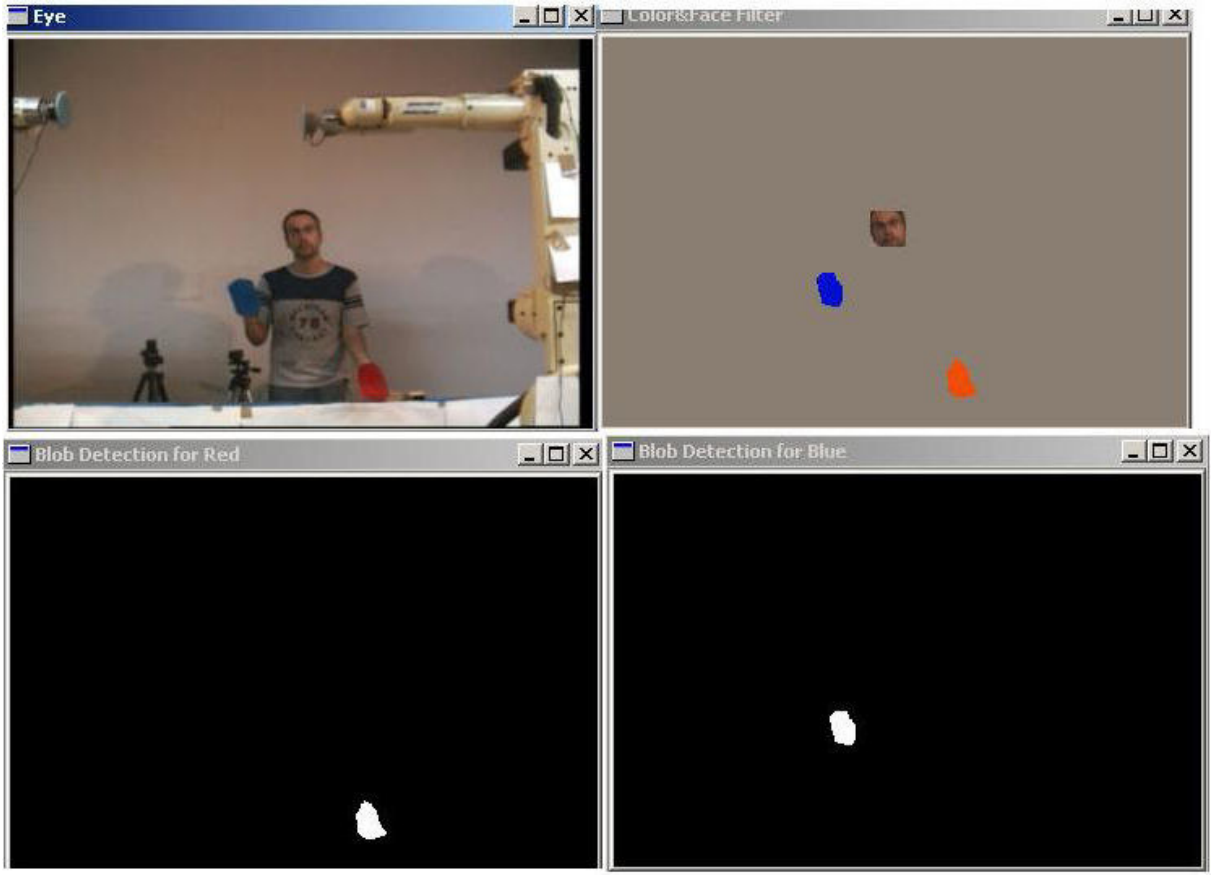


Figura 6 – finestre riassuntive

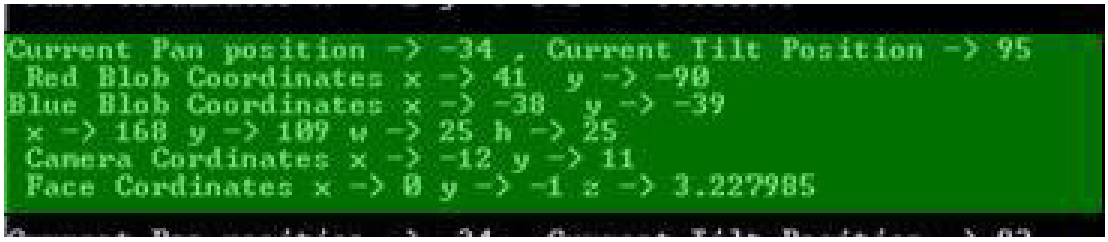


Figura 7 – visualizzazione delle coordinate

In figura 8 viene mostrato come, cambiando utente, il sistema sia in grado comunque di riconoscere il suo volto. Figura 9 e 10 riassumono, infine, il risultato delle varie elaborazioni.

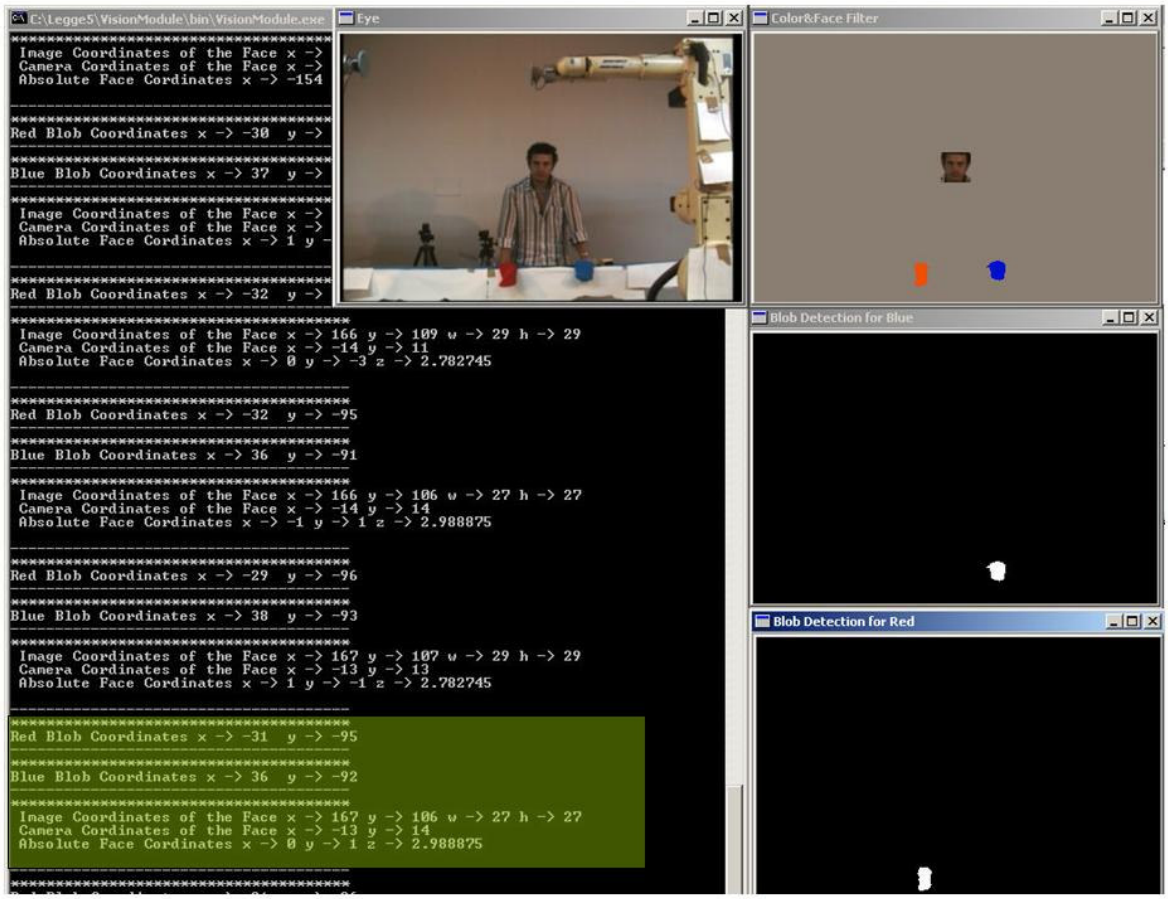


Figura 8 – altro esempio con utente diverso

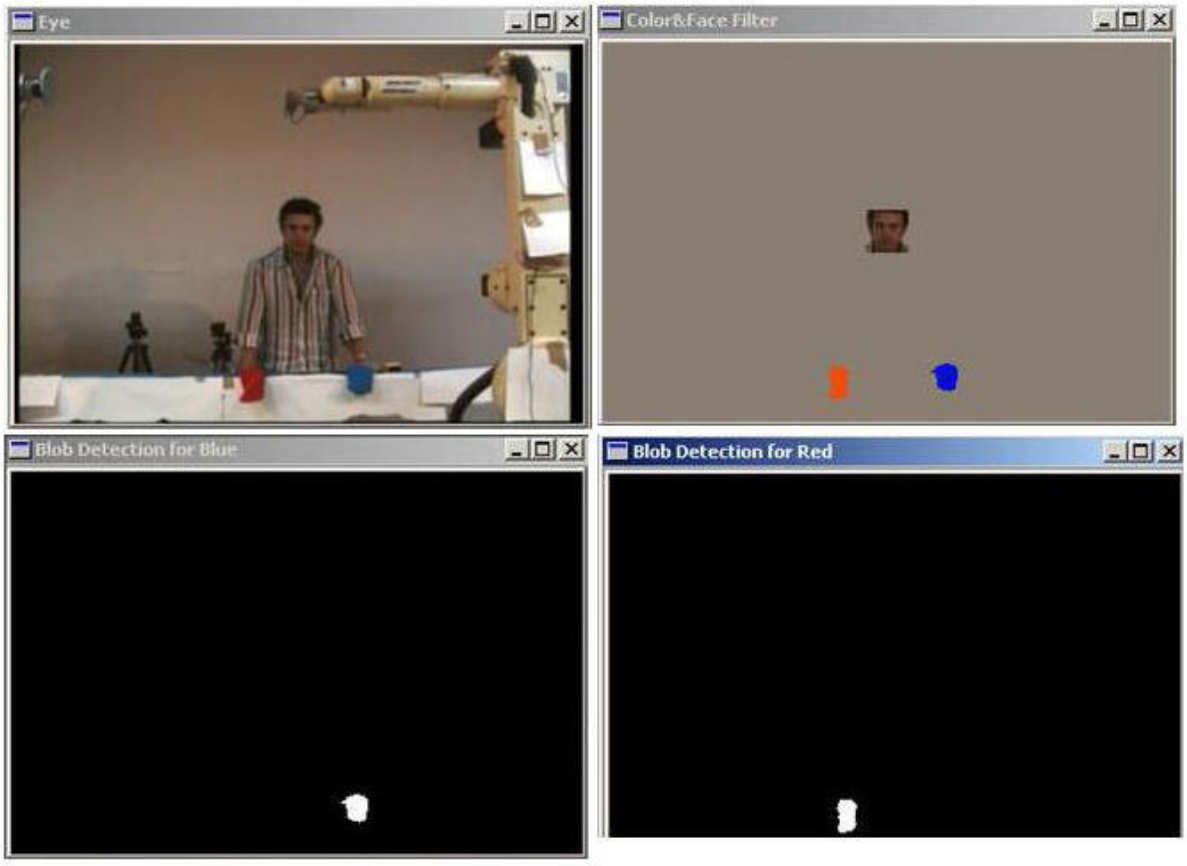


Figura 9 – immagine riassuntiva

Figura 10 –risultati del tracking

3.7.2.3. Creazione di volumi di sicurezza

Una volta definite le caratteristiche dell'utente più importanti per l'interazione e dati gli algoritmi che ne permettono il monitoraggio nel modo più efficiente possibile, è possibile

utilizzare questa informazione per ottenere che l'interazione avvenga in maniera "sicura". Risulta, infatti, conveniente che i punti così ricavati siano racchiusi in volumi (che possono essere per comodità di calcolo assimilabili a delle sfere) che li contengono. Si creano, così, dei cosiddetti "*volumi di sicurezza*" che proteggono il corpo dell'utente da possibili collisioni con il manipolatore (in figura 11 è schematizzato tale meccanismo).

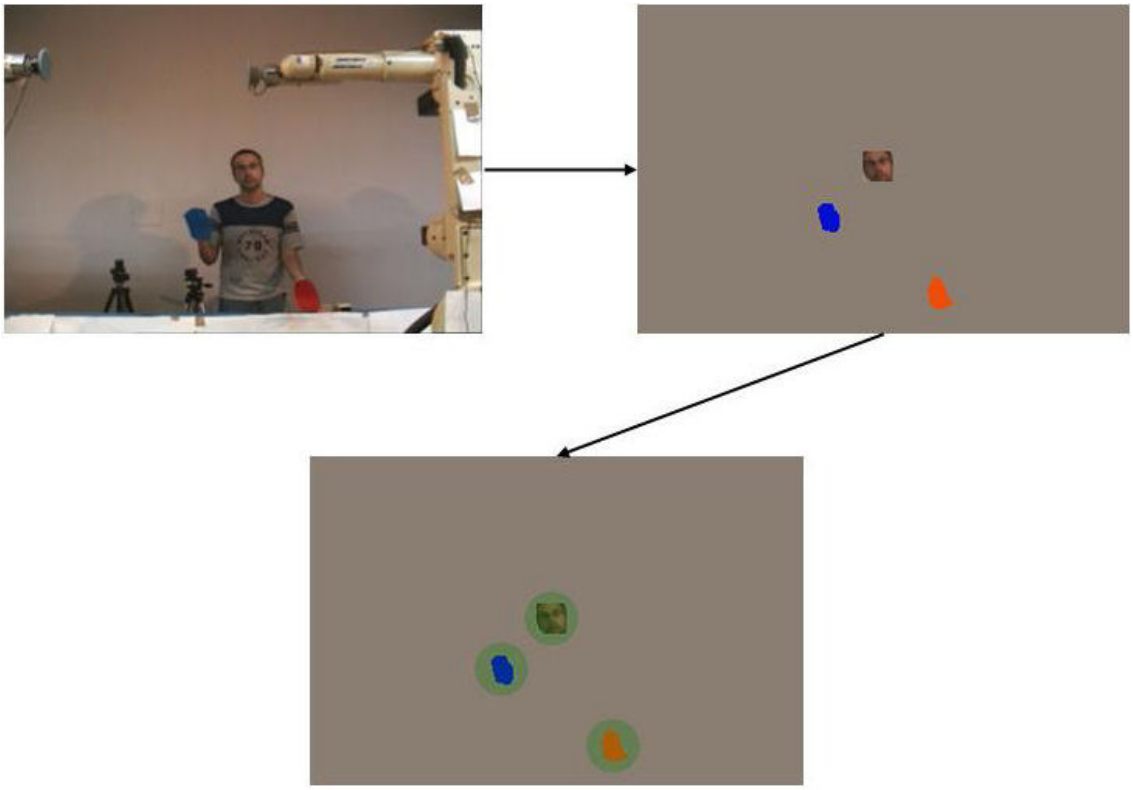


Figura 11 –schematizzazione della creazione dei volumi di sicurezza

Tale procedimento può essere utilizzato anche rispetto alla modellazione del manipolatore. Affinché la struttura del manipolatore possa muoversi nel suo ambiente in maniera sicura, si deve dotare il sistema di un modulo che, in tempo utile, generi delle traiettorie di allontanamento per il punto più vicino ad un volume di sicurezza su ciascuno dei bracci. Un problema centrale con le strutture articolate quali i manipolatori sembra essere la difficoltà di ottenere delle forze di repulsione cosiddette "*smooth*" che agiscano con continuità sull'intera struttura del manipolatore. Tale forza dovrebbe essere il risultato di un campo di potenziale ottenuto attraverso il calcolo (valutazione) di una distanza analitica e una corrispondente azione su una coppia ai giunti. Dal punto di vista della struttura del manipolatore, un algoritmo che eviti degli ostacoli in real time deve possedere le seguenti caratteristiche:

- considerare il moto di ogni punto della struttura per un suo possibile movimento reattivo e non solo l'end effector;
- effettuare il calcolo analitico delle distanze in tempo reale dell'intera struttura robotica dai possibili punti di collisione;
- generare delle forze di repulsione continue per il evitare le collisioni;
- calcolo delle matrici Jacobiane[18] opportune necessarie per valutare le coppie di forze nominali corrispondenti da trasmettere ai giunti per evitare la collisione;
- possibilità di sommare i comportamenti reattivi con ogni tipo di moto attuale durante l'interazione del robot con l'ambiente e le persone.

La modellazione del manipolatore diventa un problema cruciale per poi costruire un algoritmo di controllo che possegga tali caratteristiche. Come anticipato in precedenza, si può scegliere di assimilare la struttura del manipolatore a uno scheletro formato da più segmenti bracci. In generale, si può derivare lo scheletro direttamente da una descrizione appropriata della cinematica attraverso, ad esempio, la tavola di Denavit-Hartenberg [Sciavicco e Siciliano, 2000] ma risulta più semplice, diretto e immediato costruirlo empiricamente come concatenazione di segmenti come in figura 12 (sulla sinistra si ha il manipolatore utilizzato, al centro il suo scheletro, a destra il volume di sicurezza relativo al braccio *d*).

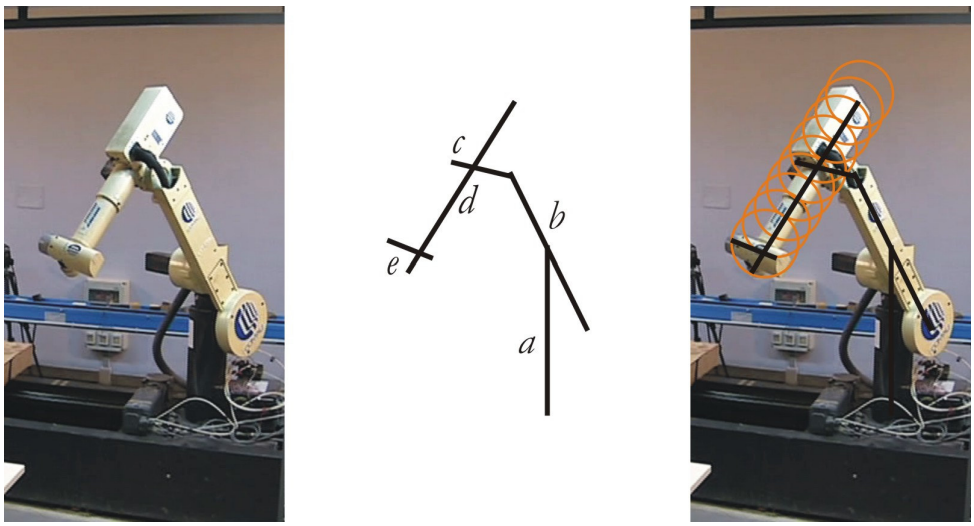


Figura 12 – modellazione dei punti di controllo di un manipolatore antropomorfo

[18] Per una definizione di Jacobiano di un manipolatore e per le relazioni tra Jacobiano e variabili di giunto si rimanda all'appendice B

Utilizzando una tale modellazione del manipolatore è possibile controllare ogni singolo punto del manipolatore [De Santis et al., 2007] ed è inoltre possibile semplificare il compito di navigazione sicura controllando la posizione di un sottoinsieme di punti della struttura [De Santis et al., 2006] che sono "spinti" opportunamente da forze virutali per evitare collisioni. Nel capitolo successivo, si illustrerà in dettaglio l'integrazione visuo-motoria che permette la navigazione sicura del manipolatore. Riassumendolo, l'algoritmo di monitoraggio delle possibili collisioni per il *path planner* può essere schematizzato come segue:

1. Costruzione di un modello appropriato del robot (lo scheletro), utile per il calcolo analitico delle distanze;
2. trovare i punti più vicini ad una collisione possibile lungo lo scheletro (i punti di collisione);
3. generare le forze di repulsione;
4. calcolare i comandi per le coppie che effettuano l'aggiramento che devono essere sommate alle coppie nominali per il controllore.

Il punto nodale dell'algoritmo dal punto di vista del controllo del manipolatore risiede nel fatto che vengono considerati solo i due punti più vicini tra il manipolatore e l'oggetto in questione e dunque si ottiene una semplificazione nella generazione della velocità desiderata (o della forza) per un solo punto della struttura del manipolatore che, a sua volta, è trasformato in una opportuna traiettoria dei giunti attraverso un appropriato schema di inversione cinematica [Sciavicco e Siciliano, 2000].

Nella sua interazione con l'essere umano, il sistema robotico deve essere in grado di evitare le collisioni con l'utente e, dunque, di poterlo rilevare istante per istante individuando la sua posizione nello spazio (coordinate x,y,z). Come si è visto nel paragrafo precedente, con gli algoritmi di face e color detection, si è in grado di avere una stima di queste posizioni per il volto (la testa), e le mani (se si suppone che l'utente utilizzi dei guanti). Tale stima è, per sua natura, approssimata: il volto è comunque racchiuso in un rettangolo individuato dalla procedura che individua il volto così come il centro delle mani è dedotto dall'ampiezza e larghezza di un insieme di pixel classificati di un dato colore.

3.8. Compendio al Capitolo 3

In questo capitolo è stato introdotto e discusso un semplice compito di un sistema robotico industriale nel quale è prevista la cooperazione tra uomo e robot. Per l'esecuzione di

questo task è possibile individuare due fasi distinte nelle quali ci può essere interazione fisica tra il sistema e i suoi utenti. Nella prima fase, il robot deve evitare una collisione potenzialmente pericolosa (il viso), nella seconda deve avvicinarsi in maniera appropriata verso una specifica parte del suo corpo (le mani). Gli algoritmi preposti al monitoraggio dell'utente umano devono quindi godere di alcune capacità indispensabili quali un tempo di elaborazione non elevato salvaguardando la robustezza della capacità di rilevazione. Le tecniche di visione adottate rispondono a questi requisiti, risultando sufficientemente robuste per un ambiente, quale quello industriale, strutturabile *ad hoc*. Per l'integrazione visuomotoria, ossia come generare le traiettorie del manipolatore partendo dall'elaborazione visiva, è necessario avere un modello del manipolatore che potesse elaborare adeguatamente le informazioni visive. La creazione dei volumi di sicurezza sembra poter rispondere ai requisiti dell'interazione sicura i entrambe le fasi individuate. Si rende quindi necessaria individuare una metodologia di sperimentazione che dimostri l'efficacia delle proposte del presente capitolo. Tale metodologia e alcuni dei risultati ottenuti saranno oggetto di discussione del Capitolo 4.

CAPITOLO 4

Integrazione VisuoMotoria

4.1. Introduzione

Nei precedenti capitoli sono state introdotte metodologie di progettazione per un compito di interazione sicura uomo/robot e alcune tecniche di visione e controllo che permettono una modellazione dell'utente umano e del sistema robotico appropriata. In questo capitolo verrà descritto un set up sperimentale per il compito descritto nel capitolo 3 (cfr paragrafo 3.7 "Il progetto PAVA – Percezione Attesa & Visione Attiva). L'analisi sarà circoscritta al compito di navigazione sicura del manipolatore, distinguendo sperimentalmente tra la fase in cui la struttura del robot deve evitare i volti presenti nella scena (fase di *collision avoidance*) e la fase di approccio alla mano dell'essere umano presente nella scena. Sono questi, infatti, i due compiti principali nei quali si esplica l'interazione fisica uomo/robot sia che questa debba evitare il contatto tra i due agenti cooperativi – fase di navigazione del manipolatore evitando il volto dell'utente – sia quando, invece, ciò deve avvenire – fase in cui il manipolatore deve porgere un utensile. In questo capitolo sarà, inoltre, descritta in dettaglio l'integrazione dell'algoritmo di visione per il riconoscimento e il tracciamento del volto, quello riguardante il color detection con quello per il controllo del manipolatore attraverso l'uso dei volumi di sicurezza descritti nel capitolo precedente.

4.2. Controllo del manipolatore per la navigazione sicura

Il sistema propriocettivo del manipolatore fornisce le posizioni dei giunti mentre il suo sistema esterocettivo rileva le posizioni del volto e delle mani. Tali informazioni sono usate per evitare il contatto o permettere un approccio "*soft*" con gli esseri umani presenti nello spazio

operativo durante l'interazione. Per ottenere questo obiettivo, sono state proposte, in letteratura alcune soluzioni che prevedono, ad esempio, la progettazione e l'implementazione di sistemi per la rilevazione e la conseguente reazione a collisioni ottenibile attraverso l'uso combinato delle capacità sensoristiche esterne e interne del robot, delle procedure usate e della componentistica elettronica dal sistema. Se, ad esempio, si suppone che la collisione avvenga in prossimità del punto terminale del manipolatore la ridondanza cinematica del braccio può essere utilizzata per minimizzare l'effetto indesiderato dell'impatto [Walker, 1994]. Mentre si esegue una traiettoria desiderata dell'end effector, il manipolatore può cambiare in maniera continua la configurazione cinematica interna per minimizzare l'inerzia vista dall'end effector. Un framework completo per evitare gli ostacoli è stato proposto in [Brock e Khatib, 2002] dove, però, non si specifica come calcolare la posizione dei punti di controllo e le matrici Jacobiane corrispondenti necessarie per il controllo. L'approccio proposto in [Bon e Seraji, 1996], [Seraji et al., 1996] considera la possibilità di utilizzare delle forze elastiche per la repulsione; inoltre, l'aggiramento degli ostacoli è ottenuto attraverso la perturbazione della posizione desiderata, per poi calcolare l'inversione cinematica per completare il compito. La discretizzazione di parte della struttura di un robot è stata proposta per l'auto collision-avoidance relativa al moto di gambe umanoidi [Kuffner et al., 2002] dove si adottano modelli di poliedri. I campi di potenziale sono stati spesso utilizzati per il moto reattivo di robot che vengono modellati come particelle sotto l'effetto di un campo di forze.

La modellazione proposta dell'utente e del manipolatore, la prima ottenuta attraverso il monitoraggio del volto e delle mani, la seconda con uno scheletro formato da segmenti congiungenti i giunti e avvolti da volumi di sicurezza, permette di implementare una procedura di navigazione che sia sicura e flessibile. Per sicura si intende che è in grado di evitare collisioni potenzialmente pericolose, per flessibile si intende che, a seconda dell'obiettivo del manipolatore, può evitare una mano oppure andare verso essa con una traiettoria e, sopratutto, velocità adeguata. Il punto di forza di tale approccio è che permette il calcolo della distanza tra punti vicini e della forza repulsiva (o attrattiva) da applicare alla struttura in maniera semplice, veloce e intuitiva.

4.2.1. Calcolo della distanza minima tra manipolatore e volto rilevato

Come detto in precedenza, la struttura del manipolatore è stata scomposta con dei segmenti congiungenti i suoi giunti. Una tale rappresentazione facilita i calcoli che possono essere eseguiti in modo parametrico per un generico punto.

Per ogni segmento in cui è decomposta la struttura, la distanza da tutti i segmenti è calcolata con una semplice formula

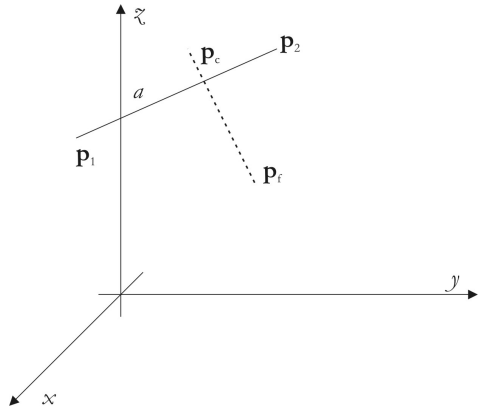


figura 1 – calcolo della distanza tra un punto della struttura e il volto

Sia $\boldsymbol{p}_a$ il punto generico di un segmento di estremi $\boldsymbol{p}_1$ e $\boldsymbol{p}_2$ e $\boldsymbol{p}_f$ il punto dove è stato rilevato il volto[19]. L'equazione parametrica del segmento di estremi $\boldsymbol{p}_1$ e $\boldsymbol{p}_2$ è data da:

$$\boldsymbol{p}_a = \boldsymbol{p}_1 + t_a \hat{\boldsymbol{u}}_a$$

con $\hat{\boldsymbol{u}}_a$ dato da:

$$\hat{\boldsymbol{u}}_a = \frac{\boldsymbol{p}_2 - \boldsymbol{p}_1}{\left\| \boldsymbol{p}_2 - \boldsymbol{p}_1 \right\|}$$

e il parametro t_a che deve soddisfare la relazione:

$$\frac{t}{\left\| \boldsymbol{p}_2 - \boldsymbol{p}_1 \right\|} \in \{0,1\}$$

L'equazione del segmento di estremi $\boldsymbol{p}_f$ e $\boldsymbol{p}_c$ con quest'ultimo il punto di contatto che si vuole stimare è data da:

$$\boldsymbol{p}_c = \boldsymbol{p}_f + r\hat{\boldsymbol{v}}_f$$

Si ottiene, così, il seguente sistema di equazioni:

[19] Il discorso è valido anche se è stata rilevata una mano attraverso il color detection quando questa deve essere evitata. In seguito si tratterà il caso in cui il manipolatore ha, come posizione desiderata, la mano dell'utente.

$$\begin{cases} \boldsymbol{p}_c = \boldsymbol{p}_1 + t_c \hat{\boldsymbol{u}}_a \\ \boldsymbol{p}_c = \boldsymbol{p}_f + r_c \hat{\boldsymbol{v}}_f \\ \hat{\boldsymbol{u}}_a \cdot \hat{\boldsymbol{v}}_f^T = 0 \\ \hat{\boldsymbol{v}}_f \cdot \hat{\boldsymbol{v}}_f^T = 1 \end{cases}$$

sostituendo il valore di $\boldsymbol{p}_c$ della prima equazione nella seconda e moltiplicando a destra per $\hat{\boldsymbol{u}}_a^T$ otteniamo un'espressione per t_c data da:

$$t_c = \left(\boldsymbol{p}_f - \boldsymbol{p}_1\right) \cdot \hat{\boldsymbol{u}}_a^T$$

da cui otteniamo un valore per $\boldsymbol{p}_c$ dato da:

$$\boldsymbol{p}_c = \boldsymbol{p}_1 + \left[\left(\boldsymbol{p}_f - \boldsymbol{p}_1\right) \cdot \hat{\boldsymbol{u}}_a^T\right] \cdot \hat{\boldsymbol{u}}_a$$

I valori di $\hat{\boldsymbol{v}}_f$ e r_c si ricavano come:

$$\hat{\boldsymbol{v}}_c = \frac{\boldsymbol{p}_c - \boldsymbol{p}_f}{\left\|\boldsymbol{p}_c - \boldsymbol{p}_f\right\|}$$

$$r_c = \left(\boldsymbol{p}_1 - \boldsymbol{p}_f\right) \cdot \hat{\boldsymbol{v}}_a^T$$

Il valore di r_c può essere utilizzato per controllare che i segmenti $\overline{\boldsymbol{p}_1\boldsymbol{p}_2}$ e $\overline{\boldsymbol{p}_c\boldsymbol{p}_f}$ siano effettivamente perpendicolari. Se, infatti, è verificata la condizione:

$$\frac{r_c}{\left\|\boldsymbol{p}_c - \boldsymbol{p}_f\right\|} \in \{0,1\}$$

allora, la distanza tra il punto dove è stato rilevato il volto e il segmento della struttura è data da:

$$d = \left\|\boldsymbol{p}_c - \boldsymbol{p}_f\right\|$$

altrimenti, la distanza del volto dalla struttura sarà data dalla distanza del punto $\boldsymbol{p}_f$ con uno degli estremi della struttura.

Tali calcoli vengono ripetuti per ogni segmento della struttura, ricavando, così, la distanza minima tra il volto rilevato e il braccio. Se tale distanza è al di sotto di una determinata soglia, si genera una forza (fittizia) opportuna per evitare la collisione tra il manipolatore e il volto.

4.2.2. Generazione della forza repulsiva

La forza agente sul braccio *a* del manipolatore nel punto di collisione *c* può essere ricavata come segue:

$$\boldsymbol{f}_{a,c} = \frac{h(d, d_0, d_{start})}{d}\left(\boldsymbol{p}_c - \boldsymbol{p}_f\right)$$

dove h è una funzione (che può essere anche non lineare) dei suoi argomenti; d è la distanza minima calcolata in precedenza tra volto rilevato e mani, d_{start} è la distanza di partenza da dove la forza inizia ad agire: i punti più lontani di d_{start} non sono soggetti ad alcuna repulsione. d_0 è la distanza limite attorno allo scheletro dove può avvenire una collisione: nel caso di giunti cilindrici, d_0 è il raggio della sezione del giunto. Si noti che h>0 dà il modulo della forza lungo la direzione tra i due punti di collisione.

Non è difficile mostrare che le forze di repulsione appena descritte possano essere derivate da una funzione potenziale

$$U_c = -\int_{d_{\min}}^{\infty} h(\boldsymbol{\delta}, d_0, d_{start}) d\boldsymbol{\delta}$$

derivando rispetto a $p_{a,c}$ ossia:

$$\boldsymbol{f}_{a,c} = -\frac{\partial U}{\partial \boldsymbol{p}_{a,c}}$$

Nel caso di una funzione lineare si ha:

$$h = \begin{cases} k\left(d_{start} + d_0 - d\right) & \text{se } d < d_0 + d_{start} \\ 0 & \text{altrimenti} \end{cases}$$

con k>0, allora il potenziale repulsivo diviene:

$$Uc = \begin{cases} \frac{1}{2} k(d_{start}, d_0, d)^2 & \text{se } d < d_0 + d_{start} \\ 0 & \text{altrimenti} \end{cases}$$

Inoltre, per ottenere un movimento fluido ("smooth") da parte del manipolatore sotto l'azione delle forze repulsive, è appropriato aggiungere dei termini di smorzamento:

$$\boldsymbol{f}_{a,c} = \frac{h(d_{\min}, d_0, d_{start})}{d_{\min}}\left(\boldsymbol{p}_{a,c} - \boldsymbol{p}_{b,c}\right) - \boldsymbol{D}_a \dot{\boldsymbol{p}}_{a,c}$$

dove $\boldsymbol{D}_a$ è una matrice appropriate definita positiva.

Il modulo della forza dipende dalle scelte, fatte in fase di progettazione e poi tarate in fase sperimentale, sull'intensità della repulsione sul braccio, che influenza i valori di h e $\boldsymbol{D}_a$. Si possono introdurre, inoltre, dei sistemi di inferenza che possono risultare utili per modificare in maniera dinamica le distanze limite e quella di inizio della forza repulsiva e della forma della funzione h. Nell'interazione uomo-robot la modellazione di tali parametri può dipendere dalla parte del corpo

che è più vicina al manipolatore: se il manipolatore deve evitare la testa di un essere umano, si può preferire una scelta di distanze di sicurezza più grandi in modulo. Inoltre, se il compito del manipolatore è quello di raggiungere la mano, si può diminuire (di poco), la forza repulsiva generata dal volto in modo che interferisca di meno con la traiettoria desiderata.

Le forze di repulsione generate in questo modo possono essere utilizzate per il calcolo delle coppie di forze ai giunti del manipolatore attraverso la trasposta dello Jacobiano. Ciononostante, si può notare che è possibile generare delle velocità di repulsione appropriate al posto delle forze di repulsione. In tal caso, queste velocità potrebbero essere utilizzate per il calcolo delle velocità di avoidance attraverso la pseudo-inversa dello Jacobiano [Sciavicco e Siciliano, 2000].

4.3. Il set-up sperimentale

Il set up sperimentale approntato presso il Prisma Lab (www.prisma.unina.it) consiste di un robot industriale Comau SMART-3 S (figura 2) a sette assi con due giunti con offset non nulli (il giunto che rappresenta il gomito e quello che rappresenta la spalla) e polso sferico.

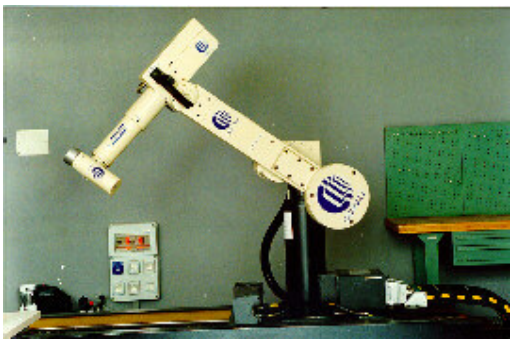


figura 2 –manipolatore a 7 assi antropomorfo

Il robot è gestito da un'unità di controllo C3G 9000 che ha un'architettura basata su VME (*Virtual Machine Environment*) con due schede processori (Servo CPU e Robot CPU). Il Servo CPU è responsabile della generazione delle traiettorie, della cinematica diretta e inversa, della micro-interpolazione del controllo ai giunti e alle posizioni dei giunti. La Robot CPU controlla l'interfaccia uomo-macchina e provvede ad interpretare i programmi dell'utente scritto nel linguaggio di programmazione PDL 2.

Le capacità sensoristiche della cella robotica sono completate da sensori di forza/coppia e un sistema di visione. Più precisamente, il sensore di forza/coppia è un ATI

FT30-100 con un intervallo di misura di ±10 N·m che si può montare sul polso del braccio. Tale sensore è connesso al PC da una scheda di interfaccia parallela che fornisce le letture delle sei componenti della forza con un rate di 1 ms. Il sistema di visione è composto da una telecamera Sony EVI-D31 collegata a un PC con un Pentium IV con processore a 1.7GHz e con un sistema operativo Windows 2000. La telecamera è dotata di movimento pan/tilt ed è stata utilizzata con messa a fuoco fissata. Il robot può montare, infine, dei gripper pneumatici con due ganasce parallele. Il controllo sullo stato del gripper aperto/chiuso è effettuato attraverso sensori che sfruttano l'effetto Hall.

L'ambiente software utilizzato negli esperimenti per il controllo real-time dei manipolatori industriali utilizzati è RePLiCS (REal-time PrismaLab Linux Control System) [Replics, 2005]. Questo ambiente permette di fare da ponte tra il sistema di visione che lavora sulla macchina collegata alla videocamera e il controllore del manipolatore COMAU. RePLiCS facilita la progettazione e il test di schemi di controllo avanzati che includono il controllo in forza e compiti di visual servoing. Replics è strutturato in un modulo real-time, composto di un driver per il kernel di RTAI-Linux, e un insieme di applicazioni non real-time che forniscono l'interfaccia utente per il modulo real-time.

Il modulo real-time di Replics implementa tutte le funzioni richieste per il controllo della cella robotica. Queste funzioni possono essere raggruppate in:

- comunicazione con i controllori C3G-9000;
- letture dei sensori di forza;
- sincronizzazione per il controllo cooperativo;
- controlli sulla sicurezza;
- cinematica del manipolatore;
- controllo del manipolatore;
- pianificazione delle traiettorie;
- comunicazione seriale e parallela;
- immagazzinamento dei dati;
- funzioni di I/O;

Le funzioni usate per la cinematica del robot permettono il calcolo della cinematica diretta e quella inversa e del loro Jacobiano. La cinematica inversa è calcolata attraverso l'algoritmo CLIK [Caccavale et al., 2001]. Sono, inoltre, disponibili delle funzioni per la pianificazione e per il controllo punto–punto del moto che fanno uso di una funzione velocità trapezoidale rispetto al

tempo. È inoltre implementata una funzione che permette la generazione di una traiettoria quando gli si assegna dei punti di attraversamento.

Le porte seriali e parallele permettono il controllo e la comunicazione con dispositivi esterni quali la macchina usata per il sistema di visione.

RePLiCS può interagire con gli utenti attraverso un'interfaccia grafica. Tale interfaccia permette le operazioni di sistema più importanti e, caratteristica molto importante, offre la possibilità di simulare i comandi al manipolatore in modo virtuale. Questa particolarità è importante dato che permette di provare, prima di immetterle sul robot "reale", tutte le funzionalità del controllore in un ambiente di simulazione che rispetta i vincoli real-time e che include la dinamica del robot e l'interazione con l'ambiente virtuale.

4.3.1. Calibrazione della telecamera e riferimento "mondo"

La telecamera, deputata al rilevamento del volto, elabora i frame in un suo sistema di riferimento chiamato "riferimento camera" mentre il manipolatore ha un suo sistema di riferimento nel quale agisce. Si rende quindi necessario dapprima calibrare[20] la telecamera e poi scegliere un riferimento comune ai due sistemi. Per la calibrazione della telecamera è stato utilizzato il *Camera Calibration Toolbox* per Matlab sviluppato dalla Caltech e disponibile gratuitamente su web. Tramite questo programma è stato possibile ricavare la stima dei cosiddetti parametri intrinseci ed estrinseci della telecamera, tra cui la lunghezza focale orizzontale (f_x) e verticale (f_y) in pixel, il "centro ottico" della videocamera o "punto centrale", i parametri di distorsione della lente e le matrici di rotazione per le immagini campione scelte per calibrare la telecamera[21]. Tali parametri permettono di trovare le relazioni tra i punti della scena monitorata dalla telecamera e i punti del suo piano immagine.

L'uso di una sola telecamera non permette di ricavare la profondità dell'oggetto rilevato (la coordinata z). Per avere una prima stima del volto rilevato, si è fatta l'ipotesi che le dimensioni del *box*, ottenuto tramite l'algoritmo di face detection che racchiude il volto presente nella scena, avessero una dipendenza funzionale con la distanza. Sono stati, dunque, presi diversi campioni di volti rilevati a varie distanze (partendo da 3 metri fino a una distanza di 60cm, con un numero maggiore di campioni presi nella zona di lavoro del manipolatore per avere una precisione migliore) e, per ricavare la dipendenza funzionale, usato un algoritmo [De Falco et al.,

[20] Per calibrazione della telecamera si intende, sinteticamente, trovare la relazione tra i punti nella scena (tridimensionale) rilevati dalla videocamera con i punti del suo piano immagine (bidimensionale).
[21] Per una spiegazione dettagliata del procedimento di calibrazione e sul significato proprio di tutti i suoi parametri, si rimanda alla documentazione del programma sul sito ufficiale della Caltech http://www.caltech.edu/

2005,2006] che fornisse i parametri ottimali della serie che meglio approssimava tale funzione. I primi risultati sperimentali sono stati sorprendentemente soddisfacenti, rispetto al metodo empirico utilizzato, dando delle precisioni sulla coordinata z dell'ordine delle decine di centimetri (mediamente quindici centimetri con picchi che, comunque, non superavano i venticinque centimetri). L'espressione ricavata che lega l'ampiezza del box con la distanza è:

$$z(width) = \frac{1}{2}\frac{a}{width} \qquad \text{con a} = 12.7043^2 \qquad (1)$$

Tale stima è stata in qualche modo suffragata utilizzando i parametri ricavati attraverso la calibrazione. Infatti, se X e Y rappresentano le coordinate in pixel di un punto nel piano immagine, X_0 e Y_0 il centro di quest'ultimo (sempre misurato in pixel), f_x e f_y, le lunghezze focali ricavate attraverso la calibrazione, x^c e y^c i punti nel riferimento camera (espressi in metri) di un punto dello spazio allora vale la relazione:

$$\begin{pmatrix} X \\ Y \end{pmatrix} = \begin{pmatrix} X_0 \\ Y_0 \end{pmatrix} + \begin{pmatrix} f_x & 0 \\ 0 & f_y \end{pmatrix} \begin{pmatrix} x^c \\ y^c \end{pmatrix} \cdot \frac{1}{z^c}$$

Se si fa l'ipotesi che, in genere, un volto è racchiudibile in un quadrato di 20 cm di lato, sviluppando quest'ultima relazione si ottiene:

$$A = (x_2 - x_1)(y_2 - y_1) = \frac{f_x f_y (x_2^c - x_1^c)(y_2^c - y_1^c)}{(z^c)^2}$$

Il termine $(x_2^c - x_1^c)(y_2^c - y_1^c)$ rappresenta l'area di riferimento scelta (in metri), mentre $A_{rif} = (x_2 - x_1)(y_2 - y_1)$ rappresenta l'area del box rilevato dall'algoritmo di face detection (ossia l'ampiezza del box, *width*, al quadrato). La dipendenza di z^c dall'ampiezza del box rilevato è dunque:

$$z^c = \sqrt{\frac{f_x f_y A_{rif}}{width^2}} \qquad (2)$$

Confrontandola con quella ricavata empiricamente tramite l'algoritmo [De Falco et al., 2005,2006], si osserva la stessa dipendenza funzionale ($1/width$). Con i parametri ricavati, sperimentalmente, di f_x e f_y si ottengono, praticamente, gli stessi risultati (con uno scarto massimo di 13 cm per z pari a quattro metri e uno di 3 cm per z pari a 40 cm).

Date queste due procedure, si possono fare alcune considerazioni sull'errore di misura su z. L'errore su z è praticamente determinato dall'errore su *width*, ossia dall'ampiezza del box ottenuto dal rilevatore di volti. Tale errore non è determinabile a priori, dato che dipende da troppe variabili indipendenti (illuminazione, angolatura del viso rispetto alla telecamera, dimensioni e caratteristiche del volto stesso etc.) ed è difficile anche ottenere una stima del suo

limite superiore[22]. Il primo procedimento utilizzato (la rilevazione empirica di volti a determinate distanze) risulta, in qualche modo, più accurato, dato che sono stati misurati empiricamente delle ampiezze di box a partire dai volti effettivamente presenti nella scena, e quindi, in una certa misura, contemplano una parte degli errori del face detector. L'algoritmo di interpolazione usato ha, poi, dato la possibilità di generalizzare le misure fornendo anche degli errori di stima su z. Tali errori, però, non possono essere considerati decisivi in quanto le misure sono state prese partendo dall'ipotesi che il face detector lavorasse in condizioni comunque "ideali", ad esempio con la stessa illuminazione e con quasi la stessa postura del viso. La soluzione potrebbe essere quella di aumentare il numero di campioni, nel primo procedimento, che contempli vari fattori variabili. Nella pratica, però, ciò è risultato superfluo dato che una taratura appropriata sui volumi di sicurezza da utilizzare permette di inglobare, in qualche modo, l'errore su z[23].

In definitiva, si è scelto di utilizzare la formula (1), mentre la calibrazione della telecamera è servita comunque per fornire una sorta di confronto e, sopratutto, per mettere in relazione i punti del piano immagine con i punti della scena. Come sistema di riferimento comune, è stato scelto uno solidale alla struttura del manipolatore come mostrato in figura 3: in rosso è evidenziato il sistema di riferimento della telecamera, in verde quello del manipolatore che è anche il riferimento "mondo" scelto.

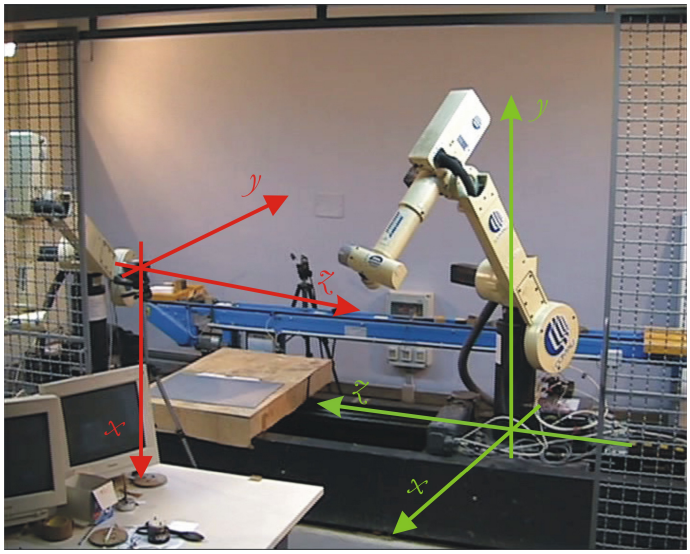


figura 3 – sistema di riferimento impiegato

[22] A ciò si aggiunga il fatto che è non nulla la probabilità di ottenere dei falsi negativi, ossia quando non si rilevava alcun volto anche se questo, invece, è presente. Per ovviare a questo tipo di errore, è stato scelto, durante il funzionamento dell'algoritmo di controllo, di aggiornare la lettura del sistema di visione solo se questa è "non nulla": quando non si rileva alcun volto, il sistema utilizza la lettura precedente per il controllo del manipolatore per almeno due cicli di funzionamento.

[23] Sperimentalmente, l'errore su z non ha quasi mai superato i 30cm ogni qual volta il volto veniva rilevato.

4.4. Fase sperimentale

La sperimentazione del sistema visuomotorio è stata suddivisa in due fasi riguardanti la navigazione sicura del manipolatore. Nella prima fase è stato testata la parte relativa alla rilevazione del volto[24] e della susseguente reazione del manipolatore. Assegnate, quindi, della posizioni iniziali e finali, il manipolatore doveva raggiungere la posizione finale (detta anche posizione target o desiderata) evitando le collisioni potenzialmente pericolose con gli esseri umani presenti nella scena. La seconda fase ha riguardato il compito di avvicinamento alle mani con una traiettoria "sicura" del manipolatore che evitasse collisioni potenzialmente pericolose. In questo caso, assegnate della posizioni iniziali, il sistema rilevava il volto e le mani dell'utente e seguiva queste ultime ricoperte da un guanto di colore blu.

4.4.1. Aggiramento del volto

I primi test sull'integrazione visuomotoria sono stati effettuati sfruttando la modalità virtuale di RePLiCS. Tale fase è servita, sopratutto, a testare il sistema di riferimento comune, la comunicazione tra il sistema di visione e il sistema di controllo e ha potuto fornire una prima indicazione sulla taratura dei volumi di sicurezza.

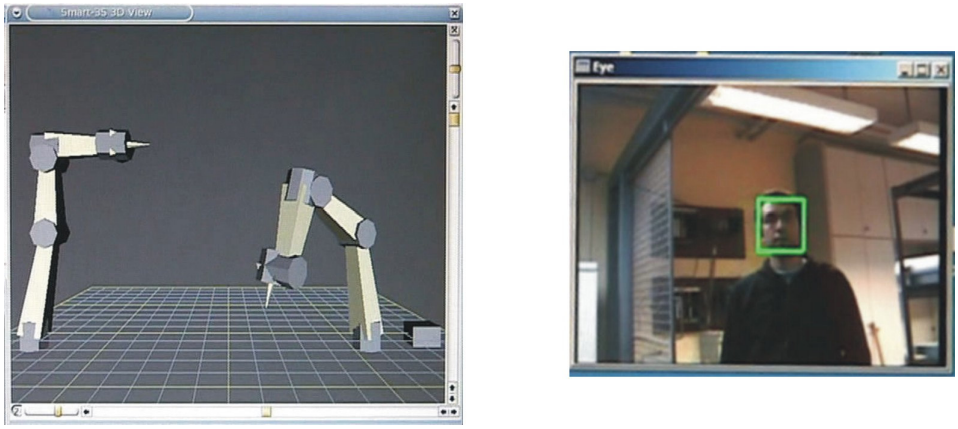


figura 4 – simulazione del sistema visuomotorio

[24] L'aggiramento delle mani, rilevabili con l'algoritmo di color decetion, segue gli stessi principi.

In ogni suo ciclo, il sistema di visione invia le informazioni relative alla posizione della faccia rilevata ossia il vettore tridimensionale $\boldsymbol{p}_f=(x_f,y_f,z_f)$ con coordinate relative al sistema mondo, l'algoritmo di controllo confronta la posizione del volto con quelle dei giunti per verificare che tutti i punti della struttura siano a una distanza maggiore della distanza di sicurezza. Se tale confronto è positivo, il controllo continua con la sua traiettoria pianificata, altrimenti la cambia esercitando una forza sul punto di contatto più vicino. Il sistema di visione dà una valore di fondo scala quando il volto non è rilevato.

Sono stati scelti diversi punti di partenza e diversi punti target con una traiettoria rettilinea che li congiungeva e, inoltre, è stato impostato un tempo fisso per ogni ciclo di operazioni del sistema (1s). Questa scelta è dovuta a questioni di stabilizzazioni della comunicazione tra sistema di visione e di controllo. Il tempo di campionamento è stato scelto per essere sicuri che il face detector elaborasse l'intera immagine. Sebbene, infatti, mediamente il face detection implementato è in grado di elaborare 4 frame al secondo (una ogni 250ms), è stato necessario tenere in conto dei tempi di comunicazione tra i calcolatori dove elaboravano i sistemi di visione e quello di controllo. Inoltre, la visualizzazione di una finestra che mostrasse i risultati del modulo di visione richiede un ulteriore tempo di elaborazione. In questa fase, infatti, si è avuto bisogno di un'interfaccia di immediata comprensione per monitorare il sistema di visione per controllare gli eventuali falsi positivi/negativi che si potevano avere.

In un secondo momento si è passati dalla modalità virtuale a quella reale di funzionamento del manipolatore. Prima, però, di eseguire gli esperimenti utilizzando esseri umani, c'è stata una fase intermedia con l'utilizzo di una *silhouette* formata da una foto a grandezza naturale. La necessità di questa ulteriore precauzione è dovuta a comprensibili motivi di sicurezza, ricordando che in linea di principio è vietata la presenza di esseri umani all'interno della gabbia di protezione che delimita lo spazio operativo di un manipolatore industriale COMAU.

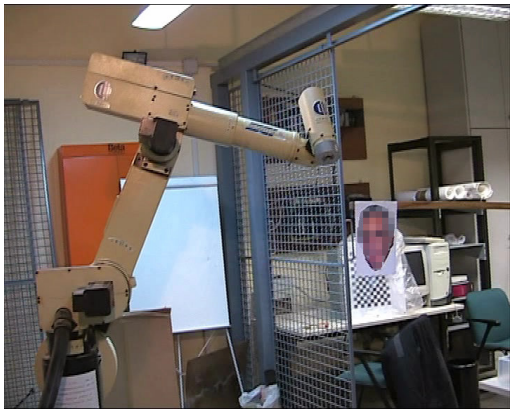


figura 5 – utilizzo di una sagoma per gli esperimenti

Come anticipato precedentemente (paragrafo 4.2.2 – generazione della forza repulsiva) una volta calcolata la forza di repulsione da imprimere al manipolatore è poi possibile utilizzarla per il calcolo delle coppie di forze ai giunti attraverso la trasposta dello Jacobiano. Tutto questo, però, si traduce in accelerazioni non costanti ai giunti che ne possono pregiudicare la sicurezza durante l'interazione. Dagli esperimenti eseguiti è risultato più sicuro lavorare con accelerazioni costanti e velocità limitate (e dunque si utilizza la pseudo-inversa dello Jacobiano per generare le velocità ai giunti).

Questa fase di testing ha permesso di provare diverse configurazioni iniziali e finali del manipolatore e ottenuto risultati soddisfacenti (il volto è evitato praticamente per ogni esperimento eseguito) quando la distanza di sicurezza è stata imposta a 1 metro (la distanza d_0 dello scheletro che modella il manipolatore (cfr. capitolo 3 paragrafo 3.7.2.3.) è di 0.40 metri) e la velocità massima di repulsione (relativa all'end-effector) a 1 m/s. Con questi parametri, si raggiunge una velocità operativa soddisfacente per l'end effector di 0.25 m/s.

Con questa configurazione è stato possibile effettuare alcuni esperimenti per l'aggiramento del volto con esseri umani che potevano muoversi all'interno dell'ambiente.

4.4.2. Raggiungimento della mano

Per la seconda fase di esperimenti, è stata seguita la stessa procedura descritta nel precedente paragrafo. Ai primi esperimenti, eseguiti in modalità virtuale, sono seguiti quelli che utilizzavano una *silhouette* dotata, in questo caso, anche di un guanto di colore blu e infine si è interagito realmente con un essere umano. In questo caso, il punto target del manipolatore non era stabilito a priori ma cambiava di volta in volta a seconda della posizione dell'utente nello spazio di interazione.

Il punto target è scelto, in un primo momento, in base alla posizione nello spazio del volto rilevato. Infatti, data $\boldsymbol{p}_f^t = (x_f^t, y_f^t, z_f^t)$, posizione del volto rilevata dal face detection all'istante *t*, il punto target iniziale stimato dal sistema è posto a:

$$\boldsymbol{p}_{expected}^t = f(\boldsymbol{p}_f^t) = (x_f^t - \Delta x, y_f^t - \Delta y, z_f^t - \Delta z)$$

Ossia il punto target desiderato è una posizione dello spazio dove ci si aspetta che l'utente possa raccogliere l'utensile. Denominata con $\boldsymbol{p}_e^t = (x_e^t, y_e^t, z_e^t)$ la posizione dell'end effector del manipolatore all'istante *t* si confronta tale posizione con $\boldsymbol{p}_{expected}^t$, se la loro differenza è maggiore di una certa soglia $d_{\text{interaction}}$, il punto target desiderato è calcolato ancora a partire dalla posizione

del volto rilevata. L'ipotesi sottintesa è che l'utente sia approssimativamente in posizione frontale rispetto alla videocamera e che l'*end-effector* cercherà di raggiungere la mano destra dell'utente. In questa prima fase di approccio, l'algoritmo di color detection rileva comunque la posizione della mano (anche perché potrebbe urtare la struttura del manipolatore in maniera opportuna), ma è come se il sistema "invitasse" l'utente umano a raggiungere quella determinata posizione dove "interagire".

Quando il manipolatore ha raggiunto approssimativamente la posizione target calcolata come funzione della posizione del volto (ossia $\left\| \boldsymbol{p}^t_{expected} - \boldsymbol{p}^t_e \right\| < d_{interaction}$) se il color detection non rileva la presenza del guanto blu, il manipolatore si arresta (si suppone che il manipolatore abbia raggiunto il suo scopo e che copra, quindi la vista del guanto alla telecamera), altrimenti il nuovo punto target diventa il punto della mano stimato dal sistema di visione. In questo ultimo caso, la traiettoria del manipolatore "insegue" la mano dell'utente.

figura 6 – fase sperimentale del raggiungimento della mano

Questo meccanismo è stato modellato secondo lo schema di interazione sicura proposto nel capitolo 2 (cfr. paragrafo 2.5). Riferendosi a tale schema, infatti, data la posizione del volto $\boldsymbol{p}_{face}(t)$ e della mano $\boldsymbol{p}_{hand}(t)$ rilevata dalla telecamera, l'*action planner* calcola calcola la traiettoria dal punto attuale dove si trova l'end-effector $p_{\text{end-eff}}(t)$ al punto target $p_{\text{expexted}}(t)$. Tale traiettoria è suddivisa, per ogni ciclo del sistema, in cosiddetti via-point ossia posizioni intermedie tra la posizione iniziale e quella finale. L'*action executer* calcola i comandi Il punto $p_{\text{expexted}}(t)$ è calcolato dal modulo *user intention interpreter*: esso è funzione del punto dove è stato rilevato il volto se la posizione attuale dell'end-effector è ancora lontana dal punto stimato dove dovrebbe esserci la mano, altrimenti il nuovo punto target è la posizione rilevata della mano. l'*action executer* calcola il punto intermedio attuale da raggiungere da parte dell'end-effector $\boldsymbol{p}_{end\text{-}effector}(t+1)$ e i comandi motori necessari per il raggiungimento del prossimo via-point. Queste informazioni sono passate al modulo di *real time safety control* che calcola la distanza minima tra i punti rilevati e la struttura del

volto che controlla che il comando calcolato non porti a una collisione tra la struttura del manipolatore e il volto o il viso: sia $\boldsymbol{p}_i(t+1)$ il punto della struttura più vicino al volto data la posizione dell'end effector $\boldsymbol{p}_{end\text{-}effector}(t+1)$. Se non c'è pericolo di collisione, viene eseguita l'azione appropriata per spostare l'end effector del manipolatore in $\boldsymbol{p}_{end\text{-}effector}(t+1)$. Inoltre, le informazioni sulla posizione del volto, delle mani e dell'end-effector sono passate al modulo *user intention interpreter* che controlla se l'end-effector è in prossimità della posizione desiderata. Se ciò non accade, la nuova posizione target è calcolata a partire dalla posizione rilevata del volto. Questa fase di funzionamento è schematizzata in figura 7 (il ciclo va letto cominciando dal comando generato dall'*action executer*).

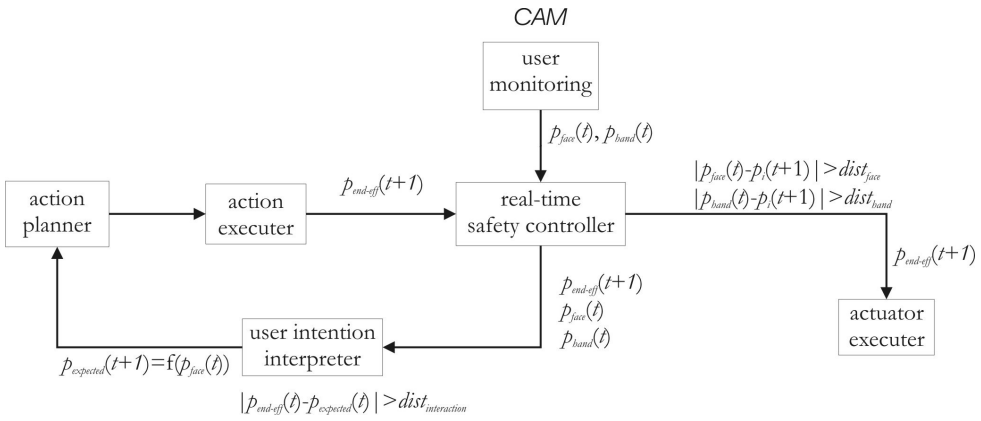


figura 7 – schema per l'interazione sicura nel caso di "soft landing"

Quando la posizione dell'end-effector ha raggiunto la posizione desiderata, si fa un'ulteriore test: si calcola la distanza tra la posizione dell'end-effector e la mano: se questa distanza è minore di una determinata soglia oppure la mano non è stata rilevata, allora il prossimo comando del manipolatore è quello di fermarsi. Questo comando è interpretabile con il fatto che il manipolatore ha raggiunto l'obiettivo di entrare in contatto con l'utente: se la distanza è minore di una certa soglia, sarà compito dell'untente "toccare" l'end-effector. Se non si rileva il guanto, il sistema interpreta che l'end effector ha "toccato" la mano dell'utente che non è più rilevabile dalla videocamera.

Se, invece, la distanza tra posizione attuale dell'end effector e quella della mano è maggiore della soglia, il nuovo punto target del sistema è la posizione della mano rilevata dalla telecamera. Ciò, però comporta, che il controllo eseguito dal *user intention interpreter* verrà effettuato su tutti i bracci della struttura ad eccezione del polso con l'end effector: ossia le nuove

traiettorie "inseguiranno" la mano evitando di urtarla con gli altri assi della struttura (ovviamente si monitora sempre il volto). In figura 8 è schematizzato il comportamento del sistema (il ciclo va letto cominciando dalla generazione della nuova traiettoria da parte dell' *user intention interpreter*).

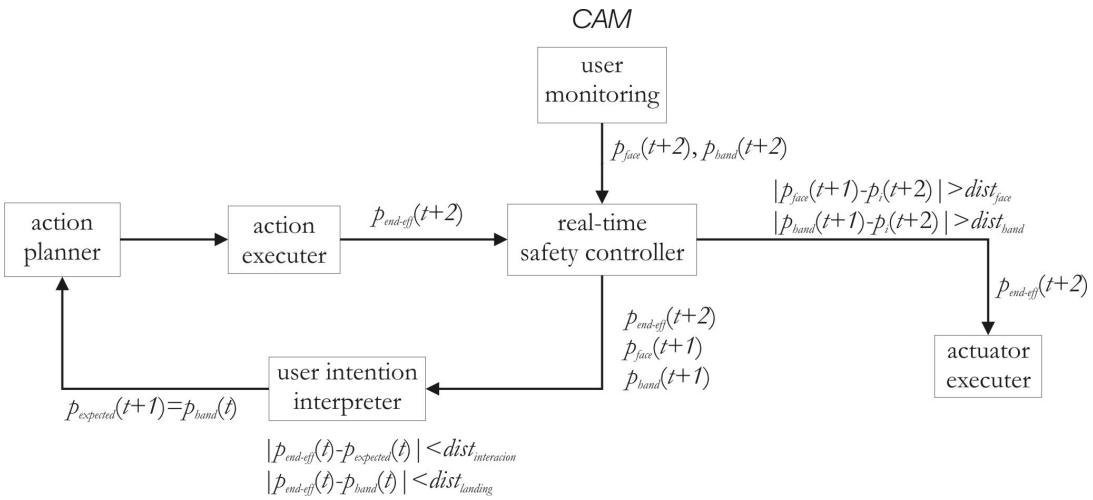


figura 8 – interazione sicura: approccio alla mano

Gli esperimenti eseguiti hanno confermato l'adeguatezza dei parametri ricavati nella fase di testing precedente. In questo caso, infatti, qualora la distanza tra il manipolatore e il volto divenga minore della soglia di sicurezza, si genera una velocità di repulsione ai giunti che, sommata alla velocità attuale del robot per il raggiungimento del target, ne cambiava la traiettoria preservando la sicurezza dell'utente e non pregiudicando, quando possibile, il raggiungimento della posizione target.

4.5. Compendio al Capitolo 4

In questo capitolo si è discussa la costruzione effettiva del sistema progettato nei capitoli precedenti e alcuni esperimenti che è stato possibile condurre. Il sistema robotico risultante consiste di un sistema di visione formato da una videocamera collegata a un calcolatore si eseguivano gli algoritmi di visione e un manipolatore COMAU come attuatore; la comunicazione tra i due sistemi è assicurata da RePLIcs (paragrafo 4.3).

È stato poi istituito un protocollo di sperimentazione che prevedesse che la verifica del sistema fosse eseguita innanzitutto in simulazione, poi attraverso l'uso di un manichino e, infine, l'effettiva interazione con un utente reale.

Tramite gli esperimenti eseguiti si sono ricavati i parametri per le velocità di funzionamento dell'apparato robotico in funzione di volumi di sicurezza che permettessero un'interazione sicura. Una scelta troppo conservativa dei volumi di sicurezza (ossia delle distanze di sicurezza tra struttura del manipolatore e volto troppo grandi) porterebbe, in alcune situazioni, all'incapacità, da parte del manipolatore, di espletare il suo task. È quindi necessario trovare il giusto equilibrio tra la necessità di sicurezza (comunque primaria) e il soddisfacimento dell'obiettivo.

CONCLUSIONI

L'interazione uomo-robot è un campo relativamente giovane nell'ambito della ricerca. La degli argomenti di cui si occupa richiede uno sforzo congiunti di ricercatori provenienti da diverse discipline. In questo lavoro di tesi si è analizzata l'interazione fisica sicura uomo-robot. La possibilità che la struttura di un robot in grado di prendere decisioni autonome entri in contatto con degli esseri umani, comporta diverse problematiche. Tra queste c'è il come gestire il contatto salvaguardando, innanzitutto, la sicurezza degli utenti del sistema robotico. Questo problema è particolarmente avvertito, ad esempio, in ambito industriale; infatti nelle fabbriche sono installati robot che, anche a causa della loro struttura, non possono entrare in contatto con gli operai e agiscono, quindi, in ambienti obbligatoriamente separati.

In questo lavoro di tesi si è cercato di individuare nuove metodologie per la progettazione di sistemi robotici, con particolare attenzione ai manipolatori industriali; questi ultimi dovrebbero consentire lo svolgimento di compiti di supporto con utenti umani, mediante il possibile contatto tra robot e utenti umani. Lo schema di interazione sicura proposto è risultato applicabile per risolvere questa problematica. Tale schema ha evidenziato la necessità di una modellazione dell'utente e della struttura robotica che permetta un comportamento sicuro del robot, ossia capace di riconoscere ed evitare situazioni potenzialmente catastrofiche. Una modellazione accurata dei due agenti della cooperazione risulta troppo onerosa e computazionalmente intrattabile per avere dei comportamenti real-time efficaci. In particolare, si è cercato di modellare l'utente umano monitorando i suoi punti salienti: per l'ambiente di lavoro nel quale opera il robot (e per l'obiettivo di cooperazione primario), si è scelto di monitorare l'utente a partire dal volto e dalle mani, protagonisti principali dell'interazione cooperativa sicura.

Gli algoritmi implementati sono stati progettati considerando la necessità di avere un sistema di visione affidabile (elevate percentuali di riconoscimento) e veloce (tempi di elaborazione quanto minori possibile), e un controllo sul manipolatore che permettesse di gestire le sue traiettorie in modo reattivo al verificarsi di una situazione di pericolo. Per l'algoritmo deputato al monitoraggio del volto si è scelto quello che, in letteratura, offriva le migliori prestazioni minimizzando i tempi di esecuzione. Tale algoritmo è stato opportunamente adattato al caso in esame garantendo le caratteristiche di robustezza necessarie.

La fase sperimentale ha verificato che il sistema proposto evitasse, sotto determinate condizioni, collisioni potenzialmente pericolose, permettendo, al contempo, un approccio appropriato nei confronti dell'utente. Il tipo di modellazione utilizzato ha reso possibile che le

condizioni di funzionamento sicuro non fossero eccessivamente restrittive e che il sistema risultante potesse lavorare in autonomia in tempi di esecuzione ragionevoli. Tutto questo utilizzando una sensoristica minima (un'unica telecamera come sensore esterocettivo).

Il sistema robotico proposto può essere perfezionato, aggiungendo altri sensori che permettano un monitoraggio più appropriato e affidabile degli utenti. Ciò comporta, però, lo studio e lo sviluppo di tecniche di fusione sensoriale e analisi dati che forniscano, in ogni caso, degli algoritmi che elaborino in tempo utile le risposte del robot. Una possibile soluzione potrebbe essere l'aggiunta di un sistema inferenziale basato su regole, che potrebbe anche migliorare la flessibilità del sistema.

Gli sviluppi possibili per questo lavoro di tesi vanno in due direzioni. Da un lato si dovrebbe esaminare quanto il sistema risulta robusto per task più articolati in ambienti vieppiù complessi. Dall'altro, un altro possibile obiettivo è quello di generalizzare ad altre piattaforme robotiche (quali i robot mobili) i risultati ottenuti per il manipolatore industriale. In questo ultimo caso, è possibile prevedere una difficoltà aggiuntiva, rispetto al manipolatore, ossia il problema, per una base mobile, di localizzarsi precisamente nel proprio spazio operativo. Infatti, se per un manipolatore il calcolo della stima della propria posizione è facilitato dal fatto di avere un'estremità fissa (quindi l'unica fonte di errore sulla sua posizione può essere dovuta a imprecisioni dei suoi *encoder*), una base mobile ha uno spazio operativo che è (potenzialmente) illimitato. Inoltre, sorgono ulteriori difficoltà dovute a nuove sorgenti di errore sul calcolo della posizione della struttura del robot. Infatti una base mobile, per localizzarsi nel proprio ambiente, non può fare affidamento esclusivamente al suo sistema di posizionamento interno ma, di solito ha bisogno di una mappa dell'ambiente. I problemi di localizzazione in robotica mobile sono attualmente studiati in maniera approfondita e sarebbe interessante indagare come coniugare i risultati ottenuti in questo campo con quelli ottenuti nell'ambito dell'interazione uomo-robot sicura.

APPENDICE A

Algoritmi di Visione

In questa appendice saranno approfonditi gli algoritmi di visione implementati per il progetto PAVA.

A.1. Algoritmo SIFT

L'algoritmo SIFT trasforma i dati di un'immagine in caratteristiche (*feature*) invarianti rispetto alla scala (Scale Invariant Feature Transform). Uno degli aspetti più interessanti di questo approccio è la generazione di un numero significativo di feature che ricopre in modo denso l'immagine in un intervallo ampio di scala e posizioni.

Nell'algoritmo SIFT il costo di estrazione delle feature viene minimizzato attraverso un filtraggio a cascata dell'immagine di input da analizzare. In questo modo, le operazioni computazionalmente più onerose sono applicate solo a porzioni dell'immagine che superano un'analisi iniziale e che sono state candidate a essere possibili feature salienti dell'immagini (dette anche *keypoint*).

Per ottenere le caratteristiche SIFT di un'immagine, si eseguono i seguenti passi:

1. *rilevamento degli estremi dell'immagine*: si analizzano i pixel dell'immagine a diversi fattori di scala per identificare potenziali punti di interesse che dovranno essere invarianti in scala e orientazione;
2. *localizzazione dei keypoint*: si costruisce un modello dettagliato per determinare la posizione e scala per ciascun punto candidato a essere una feature;
3. *assegnamento dell'orientazione*: si assegna a ciascuna posizione dei keypoint un'orientazione basata sulla direzione dei gradienti locali all'immagine. Le operazioni successive verranno, quindi, effettuate su dati dell'immagine che sono stati trasformati e resi invarianti, per ogni feature, all'orientazione, alla scala e alla posizione assegnate;
4. *creazione di descrittori dei punti chiave*: i gradienti locali dell'immagine vengono misurati rispetto alla scala selezionata in una regione intorno a ciascun punto chiave. I

keypoint sono così trasformati in una rappresentazione che consente livelli significativi di distorsione locale e cambiamenti di illuminazione.

A.1.1. Rilevamento degli estremi dell'immagine

Il primo passo della trasformata SIFT è la ricerca dei punti di interesse dell'immagine. L'approccio del filtraggio a cascata permette di determinare le posizioni e la scala delle caratteristiche dell'immagine candidate a essere i punti chiave e che, in un secondo momento, verranno analizzate in maggior dettaglio.

La posizione e la scala devono caratterizzare la feature in modo da poter essere assegnate in modo ripetuto anche in caso di vista differente del medesimo oggetto. Utilizzando una funzione di scala nota come *scale space* (spazio scala) [Witkin, 1983], è possibile ricercare punti dell'immagine che siano invarianti rispetto a tutte le scale possibili. Lo spazio scala di un'immagine è definito come una funzione $L(x,y,\sigma)$ data dalla convoluzione di x e y di una funzione Gaussiana $G(x,y,\sigma)$, variabile in scala, con l'immagine $I(x,y)$.

$$L(x, y, \sigma) = G(x, y, \sigma) \otimes I(x, y)$$

dove:

$$G(x, y, \sigma) = \frac{1}{2\pi\sigma^2} e^{-(x^2+y^2)/(2\sigma^2)}$$

Come funzione spazio-scala è possibile utilizzare una funzione cosiddetta "differenza di Gaussiane" (*difference of Gaussians* o *DoG*) ottenuta dalla differenza di due Gaussiane separate in scala da un fattore k:

$$D(x, y, \sigma) = (G(x, y, k\sigma) - G(x, y, \sigma)) \otimes I(x, y) = L(x, y, k\sigma) - L(x, y, \sigma)$$

I vantaggi nell'utilizzo di una tale funzione sono molteplici. Innanzitutto, risulta particolarmente efficiente da calcolare perché le immagini cosiddette *smoothed*, che corrispondono alla funzione $L(\cdot)$, devono comunque essere calcolate per ottenere una descrizione di tipo spazio-scala delle feature: la funzione $D(\cdot)$ è facilmente ottenibile come loro sottrazione. Inoltre, la funzione DoG costituisce una buona approssimazione del Laplaciano normalizzato in scala di Gaussiane e, come dimostrato in [Lindberg, 1994], la normalizzazione del laplaciano con un fattore σ^2 è necessaria per ottenere la proprietà di invarianza su "giusta scala". Mikolajczyk

[Mikolajczyk, 2002] ha stabilito sperimentalmente che il massimo e il minimo di $\sigma^2\nabla^2 G$ producono le feature dell'immagini più stabili in confronto ad altre possibili funzioni per le immagini quali il gradiente, l'Hessiano, o la *corner function* di Harris.

La relazione tra D e $\sigma^2\nabla^2 G$ può essere ricavata dall'equazione di diffusione del calore (parametrizzata in termini di σ piuttosto che del più usuale $t = \sigma^2$):

$$\frac{\partial G}{\partial \sigma} = \sigma \nabla^2 G$$

Da questa espressione, si può vedere che $\nabla^2 G$ può essere calcolato con un'approssimazione finita della differenza di $\partial G / \partial \sigma$, usando la differenza delle due scale più vicine $k\sigma$ e σ:

$$\sigma \nabla^2 G = \frac{\partial G}{\partial \sigma} \approx \frac{G(x, y, k\sigma) - G(x, y, \sigma)}{k\sigma - \sigma}$$

ottenendo, infine:

$$G(x, y, k\sigma) - G(x, y, \sigma) \approx (k-1)\sigma^2 \nabla^2 G^2$$

Questa equazione mostra che, quando la funzione doG ha scale che differiscono per un fattore costante, essa già incorpora il fattore di normalizzazione di scala σ^2 richiesto per il Laplaciano invariante per scala. Il fattore (k-1) nell'equazione risulta costante su tutte le scale e dunque non influenza la posizione degli estremi. L'errore di approssimazione tende a zero quando k tende a 1, ma, in pratica, si trova sperimentalmente che l'approssimazione non influenza la stabilità della rilevazione degli estremi.

Un approccio efficiente per costruire la $D(x,y,\sigma)$ è il seguente: l'immagine di partenza viene convoluta in maniera incrementale con delle Gaussiane per produrre immagini separate da un fattore costante k nello spazio della scala. Si può scegliere di dividere ogni ottava dello spazio delle scale (ossia per ogni raddoppio di σ) per un numero intero s di intervalli, sicché $k = 2^{1/s}$.

Per rilevare i massimi e minimi locali di $D(x,y,\sigma)$, ogni punto campione è confrontato con i suoi otto punti vicini dell'immagine corrente e i nove vicini dell'immagine con scala, rispettivamente, inferiore e superiore (fig. 1). Il punto viene selezionato solo se risulta o più grande o più piccolo di tutti i suoi vicini. Il costo computazionale di tale controllo risulta ragionevolmente basso dato che la maggior parte dei punti campione verrà eliminato con i primi controlli.

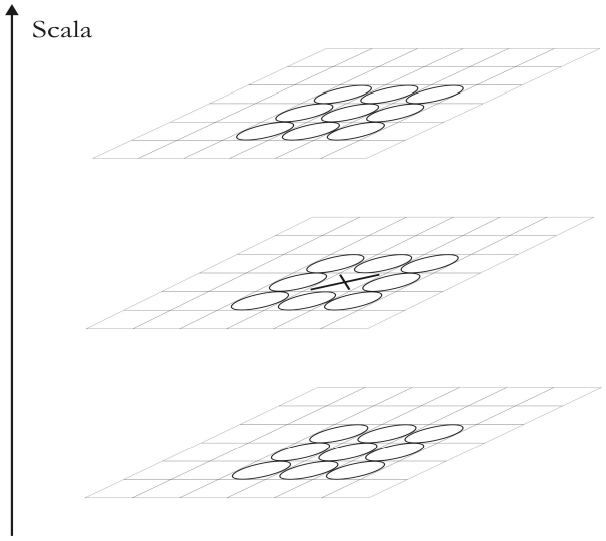


figura 1 – confronto tra pixel per il rilevamento dei massimi e minimi di immagini DoG

A.1.2. Localizzazione accurata dei keypoint

Una volta trovati i keypoint candidati confrontando un pixel con i suoi vicini, il passo successivo è quello di realizzare un modello dettagliato dei dati per la posizione, il fattore di scala, e la regione delle curvature principali. Questa informazione permette di rigettare punti che hanno basso contrasto e che sono, dunque, sensibili al rumore o che sono scarsamente localizzati lungo un contorno. Il modello scelto è quello parabolico continuo [Brown e Lowe, 2002] che, sperimentalmente, fornisce un miglioramento sostanziale in termini di *matching* e di stabilità. In questo approccio, si utilizza un'espansione in serie di Taylor (fino al termine quadratico) della funzione di spazio-scala D(x,y,σ) collocandone il punto origine nel punto campione:

$$D(\vec{x}) = D + \frac{\partial D^T}{\partial \vec{x}} \vec{x} + \frac{1}{2} \vec{x}^T \frac{\partial^2 D}{\partial \vec{x}^2} \vec{x}$$

dove D e le sue derivate sono valutate nel punto campione e $\boldsymbol{x} = (x,y,\sigma)^T$ è l'offset da quel punto. La posizione dell'estremo, x, è determinata prendendo la derivata di questa funzione rispetto a x e ponendola a zero. Si ottiene, così:

$$\hat{x} = -\frac{\partial^2 D^{-1}}{\partial x^2} \cdot \frac{\partial D}{\partial x}$$

Il valore della funzione agli estremi, $D(\hat{x})$, è utile per rigettare gli estremi instabili che hanno un basso contrasto. Ciò può essere ottenuto per sostituzione tra le due formule precedenti e ottenendo:

$$D(\hat{x}) = D + \frac{1}{2} \cdot \frac{\partial D^T}{\partial x} \cdot \hat{x}$$

In figura 2 sono mostrati i keypoint relativi a un'immagine 320x320 pixel dove sono stati trovati 1900 keypoint

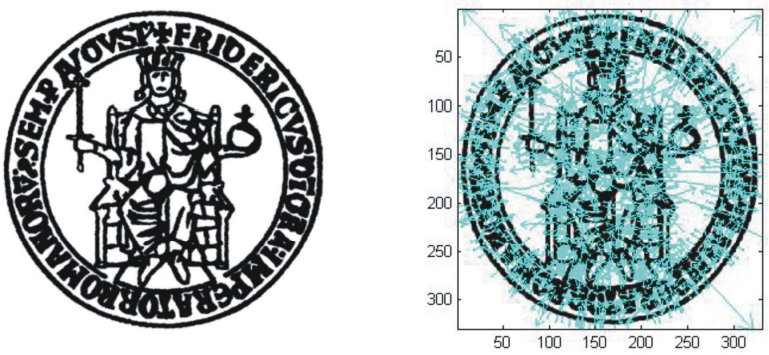


figura 2 – keypoint distintivi per un'immagine

Per la stabilità, risulta insufficiente rigettare keypoint con basso contrasto. La funzione DoG avrà, infatti, buone risposte lungo i bordi anche se la posizione lungo tali bordi non è determinata accuratamente risultando, perciò, instabile per piccole quantità di rumore.

Un picco debolmente definito nella funzione DoG avrà una curvatura principale grande lungo il contorno ma un valore piccolo lungo la direzione perpendicolare. Le curvature principali possono essere calcolate dalla matrice Hessiana 2x2, *H*, calcolata nella posizione e scala del keypoint:

$$H = \begin{bmatrix} D_{xx} & D_{xy} \\ D_{xy} & D_{yy} \end{bmatrix}$$

Le derivate sono stimate prendendo le differenze dei vicini dei punti campione.

Gli autovalori di *H* sono proporzionali alle curvature principali di *D*. Dato che si è interessati solo al rapporto tra autovalori, è possibile evitare di calcolarli esplicitamente. Sia α l'autovalore più grande in modulo e β il più piccolo. Allora si può calcolare la somma degli autovalori dalla traccia di H e il loro prodotto dal determinante:

$$Tr(H) = D_{xx} + D_{yy} = \alpha + \beta$$
$$Det(H) = D_{xx}D_{yy} - (D_{xy})^2 = \alpha\beta$$

Nel caso in cui il determinante risulti negativo, le curvature hanno segno opposto e dunque il punto viene rigettato dato che non è un estremo. Sia r il rapporto tra l'autovalore più grande e quello più piccolo in modulo, in modo che $\alpha = r\beta$. Allora,

$$\frac{Tr(H)^2}{Det(H)} = \frac{(\alpha+\beta)^2}{\alpha\beta} = \frac{(r\beta+\beta)^2}{r\beta^2} = \frac{(r+1)^2}{r},$$

che dipende solo dal rapporto degli autovalori piuttosto che dai loro valori individuali. La quantità $(r+1)^2/r$ è un minimo quando due autovalori sono uguali e aumenta con *r*. Perciò, per controllare che il rapporto di due curvature principali è al di sotto di una certa soglia, r, bisogna solo controllare la quantità:

$$\frac{Tr(H)^2}{Det(H)} < \frac{(r+1)^2}{r}.$$

Questo test risulta computazionalmente vantaggioso dato che sono necessarie meno di venti operazioni floating point per controllare ogni keypoint. Con una soglia $r=10$ si è visto sperimentalmente che si eliminano keypoint che hanno un rapporto tra le curvature principali maggiori di dieci.

A.1.3. Assegnamento dell'orientazione

Se si attribuisce, basandosi sulle proprietà locali dell'immagine, una orientazione corretta a ciascun keypoint, questi può essere rappresentato con questa orientazione ottenendo, così, un'invarianza rispetto alla rotazione dell'immagine.

Per assegnare un'orientazione locale, si utilizza la scala del keypoint per scegliere un'immagine Gaussiana *smoothed*, *L*, con il fattore di scala più prossimo, in modo tale da eseguire i calcoli ottenendo un'invarianza rispetto al fattore di scala. Per ogni immagine campione, $L(x,y)$, a questa scala, si calcolano l'ampiezza del gradiente, $m(x,y)$, e la sua orientazione, $\theta(x,y)$, usando differenze tra pixel:

$$m(x, y) = \sqrt{(L(x+1, y) - L(x-1, y))^2 + (L(x, y+1) - L(x, y-1))^2}$$

$$\theta(x, y) = \tan^{-1}\left[\frac{L(x, y+1) - L(x, y-1)}{L(x+1, y) - L(x-1, y)}\right]^2$$

Si forma così un *istogramma di orientazione* formato dalle orientazioni dei gradienti dei punti campione attorno a una regione che circonda il keypoint. Tale istogramma ha 36 bin che coprono tutto l'intervallo dei 360 gradi di orientazioni possibili. Ogni campione aggiunto all'istogramma è pesato dall'ampiezza del suo gradiente e da una finestra circolare pesata da una Gaussiana con σ pari a una volta e mezza la scala del keypoint.

I picchi nell'istogramma dell'orientazione corrispondono alle direzioni dominanti dei gradienti locali. Si rileva il picco più alto e ogni altro eventuale picco locale presente entro l'80% del valore del picco massimo per creare un keypoint con quella determinata orientazione. Solo al 15% circa dei punti si assegnano più di una orientazione, ma ciò contribuisce alla stabilità del *matching*. Infine, si adatta una parabola ai tre valori di istrogramma più vicini ad ogni picco per interpolare la posizione del picco e ottenere, così, una migliore accuratezza.

A.1.4. Descrittori dell'immagine locale

Tramite le operazioni precedentemente descritte, si assegnano a un'immagine la posizione, il fattore di scala e l'orientazione di ogni keypoint Questi parametri definiscono un sistema di coordinate 2D locali nel quale descrivere la regione dell'immagine locale fornendo, così, un'invarianza rispetto a tali parametri. Il passo successivo consiste nel calcolo di un descrittore della regione dell'immagine locale che risulta altamente distintiva anche rispetto a cambiamenti di illuminazione e punti di vista 3D.

L'approccio utilizzato si ispira ai lavori di Edelman, Intrator e Poggio su un modello di rappresentazione basato su una visione biologically-inspired. Da studi sulla corteccia visiva primaria, si è notato che i neuroni presenti in tale zona rispondono al gradiente di una particolare orientazione e frequenza spaziale, ma la posizione del gradiente della retina può muoversi su un piccolo campo recettivo piuttosto che essere localizzato precisamente. Secondo tale approccio si ipotizza che la funzione di questi neuroni sia quella di consentire il matching e il riconoscimento di oggetti 3D in un intervallo di punti di vista.

Per il calcolo dei descrittori dei punti chiave si campionano, innanzitutto, l'ampiezza del gradiente dell'immagine e le orientazioni attorno alla posizione dei punti chiave, utilizzando la scala del punto chiave per selezionare il livello della Gaussiana dell'immagine. Per ottenere

invarianza rispetto alla rotazione, le coordinate del descrittore e l'orientazione del gradiente sono ruotati relativamente all'orientazione dei punti chiave. Si utilizza, poi, una Gaussiana con deviazione standard pari alla metà della lunghezza della finestra del descrittore per pesare l'ampiezza di ciascun punto. Tale Gaussiana serve per evitare improvvisi cambiamenti del descrittore per piccoli spostamenti sulla posizione della finestra e per dare meno enfasi ai gradienti che sono più lontani dal centro del descrittore, dato che questi sono maggiormente affetti da errore.

È importante evitare gli effetti che si hanno ai bordi in cui il descrittore cambia in modo brusco quando i campioni variano di poco. A tale scopo, si usa un'interpolazione trilineare per distribuire il valore di ciascun gradiente sulle barre di istogrammi adiacenti. In altri termini, ciascun ingresso della barra viene moltiplicato da un peso pari a $(1-d)$ per ciascuna dimensione, dove d è la distanza dal campione dal valore centrale della barra, misurata in unità di spazio per la barra dell'istogramma.

Il descrittore è formato da un vettore che contiene i valore di tutte le entrate dell'istogramma di orientazione. I risultati sperimentali mostrano che i migliori risultati si ottengono con array 4x4 di istogrammi con otto orientazioni per ogni bin: alla fine si ottengono vettori di caratteristiche a 128 elementi (4x4x8).

Infine, il vettore di caratteristiche viene modificato per gli effetti del cambio di illuminazione. Innanzitutto, si normalizza il vettore in unità di lunghezza. Un cambiamento in contrasto dell'immagine nel quale ogni valore del pixel è moltiplicato per una costante, moltiplicherà i gradienti per quella stessa costante: la normalizzazione del vettore cancellerà l'effetto del cambiamento del contrasto. Un cambiamento in luminosità nel quale si aggiunge una costante ad ogni pixel dell'immagine non influenzerà i valori del gradiente, dato che questi è stato calcolato come differenza tra pixel. Perciò, il descrittore risulta invariante per cambiamenti affini di illuminazione. Ciononostante, si possono verificare dei cambiamenti di illuminazione non lineari dovuti alla saturazione della videocamera o dovuti a cambiamenti di illuminazione che influenzano le superfici 3d con orientazione differente per un valore diverso. Questi effetti possono causare un cambiamento cospicuo nelle grandezze relative di alcuni gradienti, ma non influenzeranno, nella stessa maniera, le orientazioni dei gradienti. Perciò, si può ridurre l'influenza dei valori dei gradienti più ampi ponendo una soglia ai valori nel vettore di feature unitario in modo tale che esso non superi un dato valore (negli esperimenti di Lowe tale valore era 0.2), e poi rinormalizzandolo per unità di lunghezza. Ciò significa che eseguendo un matching per le grandezze dei gradienti più grandi non ha più tanta importanza e che la distribuzione delle orientazioni ha un'enfasi maggiore.

A.1.5. Riconoscimento degli oggetti

Il riconoscimento di oggetti tramite SIFT viene eseguito confrontando indipendentemente ogni keypoint dell'immagine con i keypoint estratti dalle immagini presenti nel database che forma il training set del sistema. Molti confronti iniziali saranno incorretti a causa di feature ambigue o a rumore di sfondo presente nella scena. Perciò, si identificano innanzitutto dei cluster formati da almeno tre feature che hanno un match positivo con l'oggetto e la sua posizione, dato che tali cluster hanno una probabilità molto più grande di essere corretti rispetto a quelli di feature individuali. Poi, viene controllato ogni cluster eseguendo un *fitting* geometrico dettagliato al modello e si usa il risultato per accettare o rigettare l'interpretazione.

A.1.6 Keypoint Matching

Il miglior candidato per ogni keypoint estratto dalla prima immagine viene ricercato identificando il suo vicino più prossimo nel database di immagini di test. I vettori di caratteristiche sono punti di uno spazio euclideo e dunque la vicinanza può essere definita in termini di distanza euclidea standard. È importante sottolineare che in questo contesto non si considera la vicinanza dei keypoint in quanto punti di uno spazio tridimensionale (piano immagine e il fattore di scala) bensì in quanto vettori di feature appartenenti a uno spazio a 128 dimensioni, ovvero pari al numero di componenti del descrittore associato a ogni feature.

Al fine di rendere più veloce la ricerca dei vicini, si è sostituito il calcolo della distanza euclidea con quello del prodotto scalare tra le coppie di vettori. Il rapporto tra gli angoli, infatti, è una buona approssimazione del rapporto tra le distanze euclidee. Dati due vettori normalizzati, $\boldsymbol{x}_1$ e $\boldsymbol{x}_2$, l'angolo α tra essi compreso risulta essere semplicemente:

$$\alpha = \cos^{-1}(x_1 \cdot x_2)$$

Ciononostante, molte feature estratte da una immagine non avranno un match corretto nel database di partenza dato che queste hanno luogo a causa del rumore di sfondo o perché non sono state rilevate nelle immagini di training. Risulta dunque necessario avere un modo per rigettare feature che non abbiano un buon riscontro con quelle presenti nel database. Una soluzione che implichi l'applicazione di una soglia globale sulla distanza dalla feature più vicina non funziona molto bene, dato che alcuni descrittori risultano più discriminanti di altri. Una misura più efficace è ottenuta confrontando la distanza tra le coppie più vicine a quella del secondo vicino più prossimo. Se esistono più immagini dello stesso oggetto nel database di

training, allora si può definire il secondo vicino più prossimo come quello che risulta il vicino più prossimo che si sa che proviene da un oggetto diverso dal primo, usando solo immagini che si sa che contengono oggetti differenti. Questa misura ha buone prestazioni dato che i match corretti hanno bisogno di avere il vicino più prossimo significativamente vicino rispetto al match incorretto più vicino per ottenere un matching affidabile.

In figura 3 è mostrato il matching tra due viste differenti della stessa immagine (la seconda immagine è ruotata e ingrandita. Nella prima immagine sono stati trovati 1900 keypoint, nella seconda 2256 mentre i match stabiliti sono 1230.

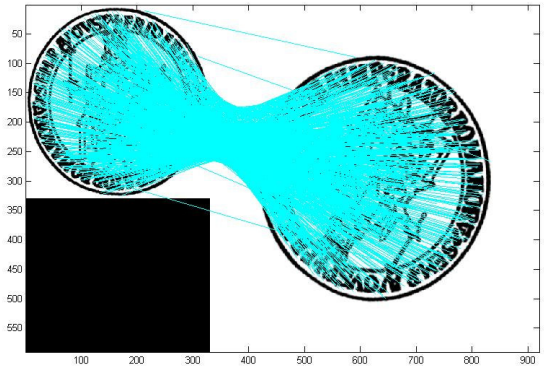


figura 3 – confronto tra due immagini dello stesso oggetto

A.1.7. Rimozione delle false corrispondenze

Al fine di ottenere risultati migliori in fase di matching, si può valutare che tipo di informazione può essere d'aiuto nella valutazione di un possibile match corretto. Tale informazione può essere utilizzata per costruire una procedura di validazione per ogni matching ed eliminare quelle che non soddisfano il criterio prescelto. Con questo approccio in cascata al metodo precedente, si può migliorare la precisione della classificazione eliminando al contempo, dove possibile, i confronti sbagliati.

Se si considera l'intorno spaziale del piano immagine si può supporre che, in generale, i match corretti hanno altri vettori di feature nel loro intorno che hanno caratteristiche simili (ad esempio stessa orientazione o stesso fattore di scala). Per quelli che si rivelano falsi match, invece, è altamente probabile che si verifichino delle incongruenze. Per ogni match dalla prima alla seconda lista di keypoint si possono individuare N elementi più prossimi e controllare che, tra essi, almeno K elementi forniscano match localmente consistenti. Per dare una definizione effettiva di consistenza locale, si ricordi che ogni keypoint è un vettore a quattro dimensioni $\boldsymbol{k} = (R,C,\theta,\sigma)$ dove R e C rappresentano rispettivamente l'indice di riga e di colonna del keypoint

dell'immagine, θ è l'orientazione in radianti e σ il fattore di scala. Il problema principale con i vettori k è che non possono essere confrontati in maniera omogenea. Ad esempio, infatti, la similarità tra angoli in radianti può essere misurata come sottrazione tra essi, mentre per quanto riguarda il fattore di scala è opportuno considerarne il rapporto. Pertanto, se il primo confronto tra due keypoint rileva una differenza in radianti pari a $\Delta\theta$, allora gli altri match dell'intorno sono considerati consistenti se hanno una differenza di orientazione simile. Se si considera il fattore di scala, invece, è rilevante la vicinanza tra i rapporti dei fattori di scala.

Un altro fattore da considerare per definire la consistenza spaziare è la prossimità spaziale dei keypoint: due keypoint che risultano vicini nella prima lista dovrebbero avere due corrispondenti (due vicini più prossimi) che sono anch'essi vicini. Tuttavia, se si confrontassero semplicemente le due distanze, verrebbe trascurato il fatto che l'occorrenza dello stesso oggetto nella seconda immagine può avvenire a una scala differente e si ha quindi il bisogno di scalare opportunamente queste differenze di distanza.

Si può dunque definire la seguente misura di similarità tra keypoint:

$$\delta_{i_{12}} = \delta_i(k_{i_1} - k_{i_2}) = \left[\frac{R_{i_1} - R_{i_2}}{\sigma_{i_1}}, \frac{C_{i_1} - C_{i_2}}{\sigma_{i_1}}, \theta_{i_1} - \theta_{i_2}, \frac{\sigma_{i_1}}{\sigma_{i_2}} \right]$$

Questa misura di similarità non definisce, ovviamente, una metrica ma piuttosto una sorta di vettore di similarità che può essere usato efficacemente per valutare quanto siano vicini due keypoint. Pertanto due keypoint, k e l, appartenenti allo stesso intorno nella lista 1 e con un vettore di similarità $\delta_{1_{kl}}$ si diranno consistenti se i loro corrispondenti nella lista 2 hanno un vettore di similarità $\delta_{2_{kl}}$ tale che la loro differenza è minore di un determinato valore di soglia α.

A.1.8. Proiezione dell'immagine target sull'immagine della scena

Proiettando l'immagine presente nel database di training su quella presente nella scena permette di individuare l'esatta posizione e orientazione dell'oggetto cercato nella scena di interesse. In particolare, se si suppone che la prima immagine raffiguri sempre l'oggetto cercato in primo piano e con le dimensioni tali da ricoprire quasi interamente l'immagine, allora tale proiezione identificherà esattamente l'oggetto nella scena[25]. Se nella scena sono presenti più di

[25] tale ipotesi è del tutto coerente con l'applicazione particolare del presente lavoro di tesi

un'occorrenza dello stesso oggetto, il sistema prende in considerazione solo l'istanza dell'oggetto i cui match sono più consistenti escludendo l'altro.

I passi di tale procedura consistono in:

1. si individua il centro del cluster e le relative orientazioni e fattore di scala medie;
2. si trasla l'origine degli assi della prima immagine in maniera tale da coincidere con il centro del cluster;
3. si applica una trasformazione di rotazione (pari alla differenza di orientazione media tra i cluster della prima e della seconda immagine) seguita da una traslazione per far coincidere il centro del primo cluster con il centro del secondo;
4. si esegue un ingrandimento sulla prima immagine i cui contorni non sono ridisegnati sulla seconda.

In figura 4 si mostra il confronto tra un'immagine e una scena dove è presente tale immagine.

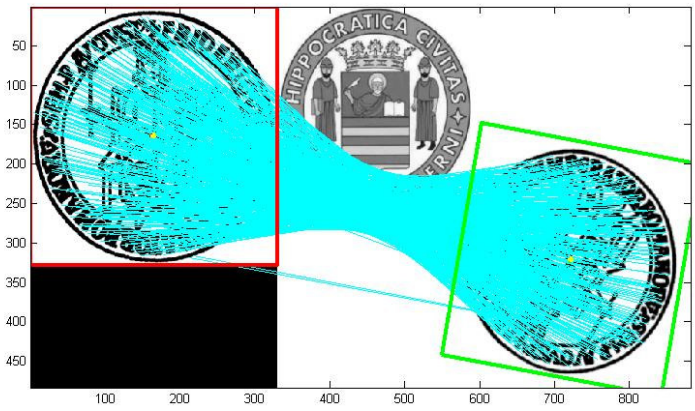


figura 4 – riconoscimento di un oggetto presente in una scena

In questo caso, la risposta del sistema sono le coordinate del centroide dei keypoint (392,320), l'orientazione (circa 1.4 radianti) e il fattore di scala (circa 0.9).

A.2. Algoritmo per il rilevamento di volti

L'algoritmo per il rilevamento di volti implementato nella presente tesi si basa sui lavori di Paul Viola e Michael Jones [Viola e Jones, 2004]. La scelta di tale approccio è motivata dalla sua principale peculiarità: la capacità di ottenere il rilevamento del volto in tempi estremamente rapidi mantenendo, al contempo, un rapporto di falsi positivi equivalente ai risultati migliori conosciuti, finora, in ambito scientifico ([Sung e Poggio, 1998], [Rowley et al., 1998], [Osuna et al., 1997a], [Schneidermann e Kanade, 2000], [Roth et al., 2000]).

I tre contributi principali che rendono vantaggioso tale approccio sono:

1. l'utilizzo di una nuova rappresentazione dell'immagine chiamata *integral image* che permette una valutazione rapida di particolari feature;

2. una selezione delle caratteristiche salienti attraverso l'algoritmo AdaBoost [Freund e Schapire, 1995];

3. un metodo per combinare in cascata classificatori vieppiù complessi focalizzando l'attenzione (e dunque la potenza di calcolo) solo nelle regioni delle immagini più promettenti.

Il sistema implementato raggiunge, quando è utilizzato per l'analisi di un flusso video, un alto valore di frame rate analizzando l'informazione presente in una singola immagine codificata in scala di grigi. Un'analisi delle immagini che si basa sulle tonalità dei pixel, infatti, si rivela particolarmente vantaggiosa in settori quali il riconoscimento del movimento, mentre lo è di meno quando si tratta di riconoscere oggetti.

L'algoritmo di *face detection* proposto da Viola e Jones utilizza un insieme di caratteristiche dette "haar-Like" derivate dalle trasformate di Haar. Tale approccio nasce dagli studi di Papageorgiou [Papageorgiou et al., 1998] per l'analisi di immagini attraverso insiemi ristretti di caratteristiche.

Nel caso dell'algoritmo per la rilevazione dei volti, si definisce un classificatore come un albero decisionale con almeno due foglie che utilizza queste caratteristiche per determinare se una zona dell'immagine di un'immagine di input corrisponda a qualche immagine da lui conosciuta. Un classificatore è addestrato inizialmente con esempi positivi, ossia dei volti in formato 24x24 pixel, e negativi della caratteristica da riconoscere, ossia immagini arbitrarie sempre dello stesso formato. La caratteristica usata in un classificatore è definita dalla sua forma, dalla posizione all'interno della regione di interesse e dal fattore di scala. Nel caso, ad esempio, della caratteristica (2c) di figura 6, la risposta è calcolata come differenza tra la somma dei pixel di immagine coperti da tutta la caratteristica e la somma dei pixel coperti dalla porzione di immagine nera, moltiplicati per 3 per compensare le differenze nel formato delle zone. Le somme dei pixel coperti dalle regioni rettangolari possono essere calcolati in maniera particolarmente rapida se si una rappresentazione dell'immagine chiamata "immagine integrale" (*integral image*) e che costituisce il primo aspetto significativo dell'algoritmo di face detection implementato.

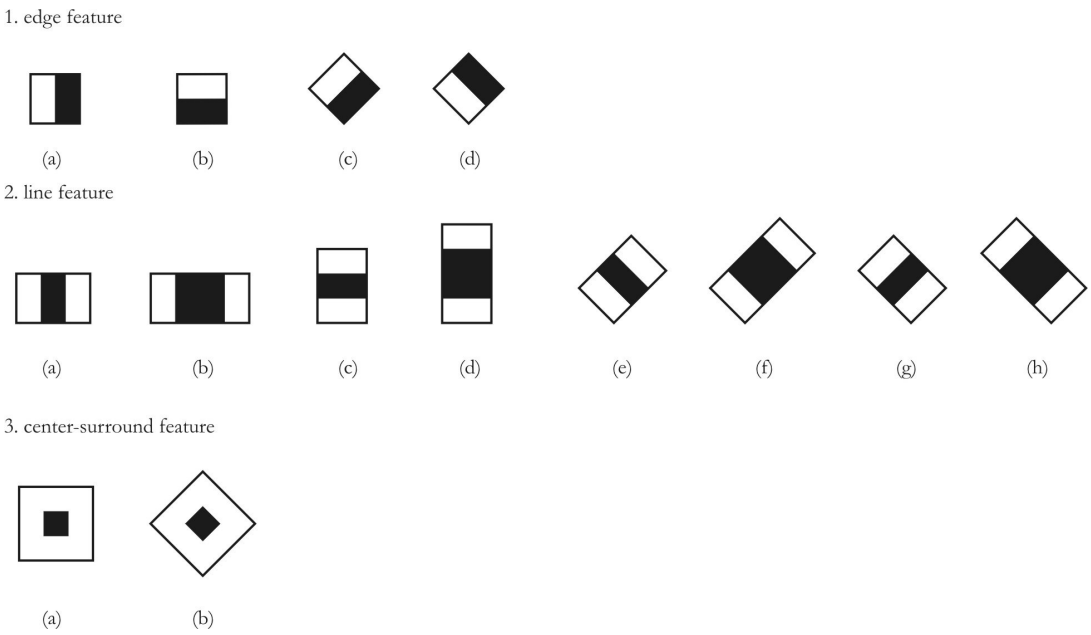


Figura 6 – caratteristiche haar utilizzabili

A.2.1. Immagini integrali

Le feature rettangolari possono essere calcolate in maniera estremamente rapida usando una rappresentazione intermedia per l'immagine chiamata "immagine integrale". L'immagine integrale nella posizione (x,y) contiene la somma dei pixel contenuti al di sopra e alla sinistra del punto (x,y):

$$ii(x, y) = \sum_{x' \le x, y' \le y} i(x', y')$$

dove $ii(x,y)$ è l'immagine integrale e $i(x,y)$ è l'immagine originale. Usando le due seguenti formule ricorsive:

$$s(x, y) = s(x, y-1) + i(x, y)$$
$$ii(x, y) = ii(x-1, y) + s(x, y)$$

dove $s(x,y)$ è la somma cumulativa delle righe, $s(x,-1)=0$, e $ii(-1, y)=0$, è possibile calcolare un'immagine integrale in un solo passo di computazione sui pixel dell'immagine originale.

Utilizzando l'immagine integrale ogni somma di rettangoli presenti in un'immagine può essere calcolata attraverso quattro riferimenti di un array.

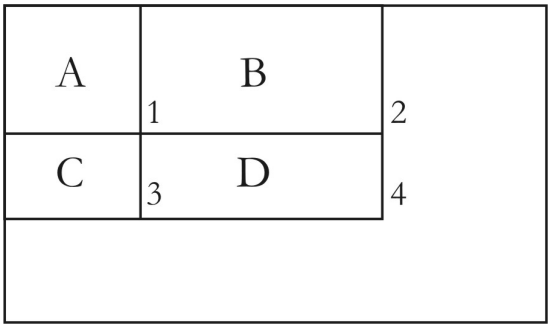


figura 7– riepilogo di un'immagine integrale

Con riferimento alla figura 7, la somma dei pixel nel rettangolo *D* può essere calcolata facilmente con semplici operazioni sui valori dei punti (1,2,3,4). Il valore dell'immagine integrale nel punto 1 è la somma dei pixel presenti nel rettangolo *A*. Il valore nel punto due è dato da *A*+*B*; nel punto 3 è *A*+*C* e, infine, nel punto 4 è *A*+*B*+*C*+*D*. Viceversa per i valori dei vari rettangoli otteniamo:

- *A*=1;
- *B*=2-1;
- *C*=3-1;
- *D*=4-*A*-*B*-*C*

Infine, il valore in D può essere calcolato come *D*=4+1-(2+3).

Per quanto riguarda le feature Haar-like, queste possono essere dunque calcolate rapidamente, dato che sono differenze di rettangoli di pixel, attraverso la rappresentazione descritta nelle equazioni ricorsive per $s(x,y)$ e $ii(x,y)$.

Una motivazione alternativa per l'introduzione dell'immagine integrale si deve agli studi di Simard [Simard et al., 1999] con i quali è possibile dimostrare che l'immagine integrale è, nei fatti, l'integrale doppio dell'immagine.

La scelta di feature "haar-like" rispetto ad altri approcci porta diversi vantaggi rispetto ad approcci tradizionali ([Freeman e Adelson, 1991], [Greenspan et al., 1994]). Le feature rettangolari, ad esempio, sono sensibili anche alla presenza di bordi (*edge*), barre, e altre immagini di strutture semplici; inoltre sono facilmente implementabili e danno una buona stima ancorché abbastanza grezza.

Per apprezzare al meglio il vantaggio computazionale apportato dall'introduzione dell'immagine integrale, si consideri l'approccio tradizionale nel quale si deve calcolare una piramide di immagini. Come nella maggior parte dei sistemi di riconoscimento, il sistema utilizzato analizza l'immagine in ingresso a diversi fattori di scala: si analizza un'immagine 384x288 utilizzando dodici immagini di fattori di scala differenti in maniera crescente partendo da

una risoluzione base nella quale i volti sono ricercati con una finestra di dimensioni 24x24 pixel con una differenza di un fattore pari a 1.25 tra un'immagine della piramide e l'altra. L'approccio convenzionale prevede, invece, il calcolo di una piramide di dodici immagini, ognuna delle quali con un fattore 1.25 più piccola. Si utilizza, poi, un rilevatore che analizza tutte queste dodici immagini nella loro interezza Un tale approccio esaustivo del calcolo della piramide, sebbene sia semplice e diretto, richiede dei tempi di elaborazione considerevoli. Di contro, si può definire un insieme di feature rettangolari che sia significativo e che goda della proprietà che una singola feature possa essere valutata per ogni fattore di scala e posizione in poche operazioni. Per fare ciò, come si vedrà in seguito, basterà assemblare in maniera opportuna diversi classificatori.

A.2.2. Costruzione del classificatore – AdaBoost

Dato un insieme di feature e un training set di esempi positivi e negativi, si possono utilizzare diversi approcci per costruire una funzione che li classifichi. Ad esempio, l'approccio seguito da Sung e Poggio utilizza un modello di misture di Gaussiane, nel lavoro di Rowley [Rowley et al., 1998] si utilizza un insieme semplice di feature e una rete neurale; in [Osuna et al., 1997b] si utilizza, invece, una support vector machine.

Con l'insieme di feature proposto, si devono considerare 160000 feature rettangolari associate a ogni sottofinestra di un'immagine di dimensione 24x24, un numero largamente superiore al numero di pixel presenti nell'immagine stessa e dunque una procedura che analizzasse tutte le caratteristiche in maniera esaustiva comporterebbe costi di calcolo proibitivi. Si ipotizza, quindi, che solo un numero ridotto di queste feature possano essere combinate in maniera effettiva per formare un classificatore. Nel riconoscitore di volti implementato, si utilizza l'algoritmo di AdaBoost proposto in [Freund e Schapire, 1995] sia per analizzare l'immagine in input che per addestrare i classificatori di base. Ad ogni ciclo di computazione, AdaBoost seleziona una caratteristica tra le 160000 presenti nell'immagine 24x24 basandosi sulle caratteristiche "haar-like" descritte in precedenza. Per fare questo l'algoritmo prende in input immagini di esempi "positivi" e "negativi" 24x24 in tonalità di grigio e vi cerca caratteristiche haar. Analizzando tutte quelle trovate, seleziona quella che compare più volte, ovvero quella che in un processo di riconoscimento è una caratteristica frequente e può dunque generare meno errori possibili.

Nel sistema proposto, quindi, l'algoritmo di apprendimento debole (quello che forma il classificatore debole) è progettato per scegliere la singola feature rettangolare che meglio separa gli esempi positivi da quelli negativi. Per ogni feature, l'algoritmo di apprendimento "debole"

determina la miglior funzione classificazione che funge da soglia in modo tale che un numero minimo di esempi sono classificati in maniera errata. Un classificatore debole ($h(x,f,p,\theta)$) consiste quindi di una feature (f), una soglia (θ) e una polarità (p) che indica il verso della disuguaglianza:

$$h(x, f, p, \theta) = \begin{cases} 1 \text{ se } pf(x) < p\theta \\ 0 \text{ altrimenti} \end{cases}$$

In questo caso, x è una sottofinestra 24x24 pixel dell'immagine.

Nella pratica accade che nessuna feature singola può eseguire il task di classificazione con un errore trascurabile. Le feature che sono scelte inizialmente nel processo hanno delle percentuali di errore che variano tra 0.1 e 0.3. Le feature scelte negli ultimi stadi, quando il task apprendimento diventa più difficile, hanno percentuali di errore comprese tra 0.4 e 0.5.

Lo pseudocodice dell'algoritmo è il seguente:

date le immagini di esempio (x_1,y_1), ... (x_n,y_n) dove $y_i=0,1$ a seconda che sia rispettivamente un esempio negativo o positivo;

inizializzo i pesi $w_{1,i} = 1/2m$, $1/2l$ per $y_i=0,1$ rispettivamente, dove m e le sono il numero di esempi negativi e positivi;

per t=1,...,T

1. normalizzo i persi,

$$w_{t,i} \leftarrow \frac{w_{t,i}}{\sum_{j=1}^{n} w_{t,j}}$$

   in modo tale che w_t possa essere interpretata come una distribuzione di probabilità
2. per ogni feature, j, addestra il classificatore h_j che è ristretto a usare una singola feature. L'errore è valutato rispetto a w_t[26];
3. scelgo il classificatore h_t con i parametri che minimizzano l'errore;
4. aggiorno i pesi:

$$w_{t+1,i} = w_{t,i}\beta_t^{1-e_i}$$

   dove ei=0 se l'esempio xi è stato classificato

[26] più precisamente, l'errore è valutato come $\varepsilon_j = \sum_i w_i \left| h_j(x_i) - y_i \right|$

correttamente, altrimenti ei=1, e

$$\beta_t = \frac{\varepsilon_t}{1 - \varepsilon_t}$$

il classificatore "forte" finale è:

$$h(x) = \begin{cases} 1 & \text{se } \sum_{t=1}^{T} \alpha_t h_t \geq \frac{1}{2} \sum_{t=1}^{T} \alpha_t \\ 0 & \text{altrimenti} \end{cases}$$

dove $\alpha_t = \log(1/\beta_t)$

.

In figura 8 sono visualizzati i risultati dei primi nove cicli dell'algoritmo. La prima caratteristica scelta è quella "più presente" in tutte le immagini: essa evidenzia la differenza di tonalità tra gli occhi e la fronte e le guance.

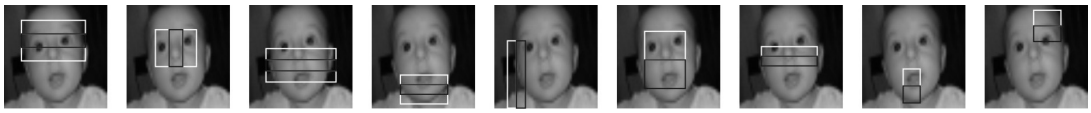


figura 8 – caratteristiche rilevate nei primi nove cicli di AdaBoost

La parte degli occhi risulta più scura della parte che li circonda, e questa caratteristica è ricorrente in tutte le immagini. Alla seconda iterazione, l'algoritmo rileva la differenza di tonalità fra gli occhi e il naso, che risulta essere una zona più chiara in tutte le immagini. Proseguendo con le iterazioni, le caratteristiche rilevate saranno le differenze tra la bocca e le zone comprendenti il naso e il mento per poi soffermarsi sui contorni del viso, collo, sopracciglia fino a discriminare dettagli sempre più piccoli.

A.2.3. La combinazione a cascata dei classificatori

Per ogni passo di *boosting*, si crea un classificatore cosiddetto "debole" (*weak classifier*). Per individuare la presenza di un volto in un'immagine arbitraria bisognerebbe considerare tutti i classificatori e comporli, ad esempio attraverso la loro somma, per creare un classificatore generale (detto "forte") che sia valido per il riconoscimento effettivo del volto e verificarne la presenza in tutte le possibili sottofinestre 24x24 dell'immagine da analizzare. Anche in questo caso, un procedimento esaustivo sarebbe troppo oneroso dal punto di vista computazionale.

Un'idea alternativa è quella di utilizzare una struttura a cascata in vece del classificatore generale (una sua rappresentazione schematizzata è realizzata in figura 9).

Per tutte le sottofinestre dell'immagine sono applicate una serie di classificatori: per ognuna di queste si passa all'analisi di un secondo classificatore solo in caso di risposta positiva da parte del primo classificatore. Il meccanismo a cascata prosegue finché il risultato dei vari test associati ai classificatori incrementali ha un esito positivo, mentre si elimina la finestra corrente alla prima occorrenza di un risultato negativo. Con questo approccio si economizzano le risorse che verranno, così, utilizzate solo nelle aree più promettenti dell'immagine e che quindi possono condurre all'individuazione del volto riducendo drasticamente il numero di ricerche nell'immagine.

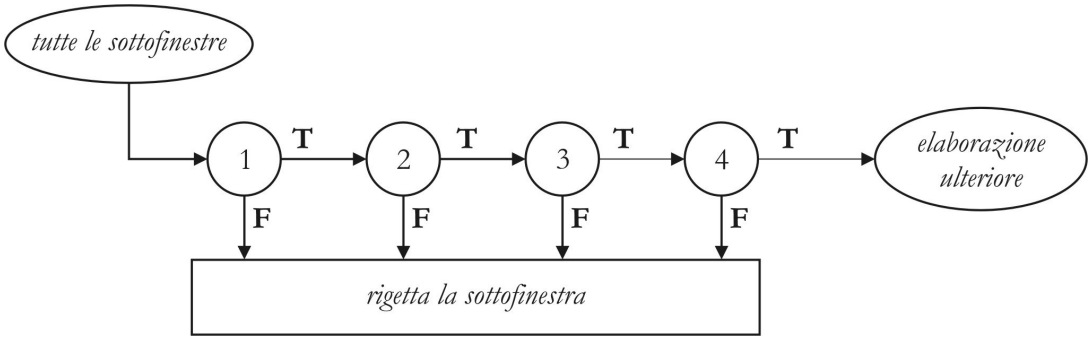


figura 9 – schema dell'approccio a cascata dei classificatori

Se ci si riferisce al primo classificatore rilevante per l'AdaBoost dell'esempio di figura 3, il primo passo dell'algoritmo sarà ricercare, attraverso l'intera immagine, la differenza di tonalità fra le zone del volto che comprendono gli occhi, la fronte e le guance. Si itera il processo a diversi fattori di scala eliminando le finestre in cui non compare la caratteristica in esame e continuando la ricerca nelle rimanenti porzioni di immagine (con lo stesso fattore di scala della prima caratteristica trovata).

Aumentando il numero di stati della cascata, ossia aumentando il numero di verifiche, aumenta di conseguenza l'attendibilità del responso. In questo caso, però, aumenta anche il carico computazionale dell'applicazione: è possibile dunque, in questo caso, trovare un bilanciamento tra la richiesta di tempi di elaborazione bassi e affidabilità nel rilevamento dei volti (riduzione delle false occorrenze dei volti o loro mancato riconoscimento).

A.2.4. Il sistema risultante

Il sistema di face detection è stato realizzato utilizzando la libreria OpenCV della intel [OpenCV]. Questa libreria è un'utile collezione degli algoritmi più utilizzati dalla comunità scientifica che si occupa di Image Processing. Essa è stata creata per essere usata da ricercatori e sviluppatori di software e segue la filosofia del codice libero (*open source*). La libreria punta a creare una infrastruttura aperta e soprattutto gratuita dove i diversi servizi singoli della *computer vision* possono essere consolidati e ottimizzati nelle prestazioni.

Il training set dei volti consiste di quasi cinquemila immagini di facce, raccolte tramite una navigazione random del Web, con un fattore di scala ridefinito e con una risoluzione base di 24x24 pixel. Il rilevatore finale è una cascata di classificatori (cfr. paragrafo A.2.3.) composto da 38 layer con un totale di 6060 feature. Il primo classificatore della cascata è costruito usando due feature e rigetta circa il 50% delle zone dell'immagine che non sono volti mentre rileva il 100% dei volti. Il classificatore seguente ha dieci feature e rigetta l'80% dei non-volti conservando una percentuale prossima al 100% di successi. I due layer successivi sono costituiti da classificatori con 25 feature seguiti da tre classificatori con 50 feature seguito da classificatori con un numero di feature differenti scelti in base a un algoritmo di addestramento per la costruzione di una cascata di rilevatori [Viola e Jones, 2004].

La velocità del riconoscitore è legata direttamente al numero di feature valutate per tutte le sottofinestre analizzate mentre il numero di feature valutate dipende dall'immagine da analizzare. Dato che molte sottofinestre sono rigettate nei primi due stadi della cascata, si ha una media di 8 feature valutate per sottofinestra [Rowley et al., 1998]. Nella sua implementazione finale, il rilevatore analizzava immagini da 284x288 pixel in circa 0.067 secondi elaborando su un processore Pentium III a 700Mhz (circa 15 frame al secondo). Nell'implementazione utilizzata nel seguente lavoro di tesi, ossia utilizzando le librerie pubbliche e un codice non del tutto ottimizzato, si è raggiunta una velocità di elaborazione non inferiore a 4 frame al secondo (oscilla tra i 0.2 e 0.25 secondi circa) conservando le prestazioni per quanto riguarda l'efficienza di rivelazione dei volti.

Figura 10 – esempi di rilevamento volti

In figura 10 sono riportati alcuni esempi di esecuzione del sistema. Si può notare come il riconoscitore sia in grado di rilevare volti in maniera robusta rispetto a condizioni di illuminazione differenti, occlusione parziale, inclinazioni diverse dei volti.

APPENDICE B

Cenni sui Manipolatori Robotici

In questa appendice si daranno alcune definizioni sui manipolatori robotici e verranno descritte le principali proprietà riguardanti la loro cinematica diretta, inversa e differenziale.

B.1. Prime definizioni

Definizione B.1 Un *manipolatore* è la struttura meccanica di un robot che consiste di bracci, connessi l'un l'altro per mezzo di giunti, e di un organo terminale (detto *end effector*).

I sensori del manipolatore servono a misurare la posizione del robot, tramite degli *encoders*, mentre gli attuatori sono gli organi, in genere elettrici o idraulici, che attuano il movimento del robot.

Un manipolatore è costituito da:

- una base fissata nell'ambiente di lavoro, o su una piattaforma mobile,
- dei link, cioè corpi rigidi di collegamento,
- giunti, cioè degli snodi che connettono i links,
- end-effector, cioè l'organo terminale,connesso al manipolatore tramite un polso, che gli permette di muoversi liberamente con un orientamento arbitrario

Definizione B.2 Con il termine *spazio di lavoro* si intende il volume raggiungibile dal manipolatore in una sua qualsiasi configurazione. Chiaramente questo dipende dalla configurazione geometrica del manipolatore stesso, dalla dimensione dei links e dai limiti meccanici di alcuni giunti.

Inoltre si individua lo spazio di lavoro raggiungibile come l'intero insieme di punti raggiungibili dal manipolatore e lo spazio di lavoro destro come tutti quei punti che il manipolatore può raggiungere con un orientamento arbitrario dell'end-effector.

B.2. Cinematica dei corpi rigidi

Nello studio dei problemi di robotica mobile sono necessari conoscenze di geometria e algebra lineare che si rivelano strumenti utili per descrivere la meccanica di un corpo rigido nello spazio.

B.2.1. Matrici di Rotazione

Si consideri un corpo rigido *M*: questi è completamente descritto in termini di posizione e orientamento rispetto ad una generica terna di riferimento $O - xyz$. In figura 1 all' oggetto *M* è associata la terna ortonormale di coordinate di riferimento $S_1 \equiv (O' - x_1 y_1 z_1)$, e dove $\boldsymbol{i}_1, \boldsymbol{j}_1$ e $\boldsymbol{k}_1$ sono i versori degli assi della terna. Per *O* si definisce anche il sistema di riferimento $(O - x_0 y_0 z_0)$ di versori $\boldsymbol{i}_0, \boldsymbol{j}_0$ e $\boldsymbol{k}_0$. Si vogliono ottenere le relazioni tra le coordinate del punto *p*, espresso in S_1, con le coordinate dello stesso punto espresso nel sistema $S_0 \equiv (O - x_0 y_0 z_0)$.

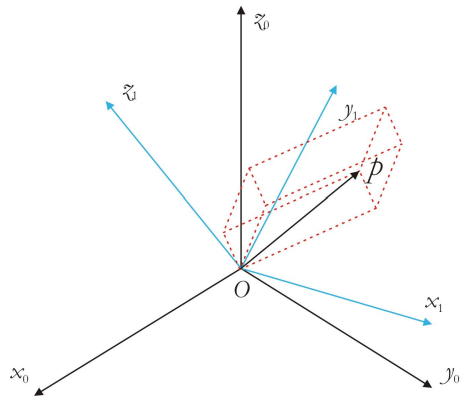


figura 1 – il punto p esprimibile nei due riferimenti

Il punto $\boldsymbol{p}$ espresso nel sistema di riferimento S_1 può essere espresso nel sistema S_0 come:

$$\boldsymbol{p}_0 = \boldsymbol{R}_0^1 \boldsymbol{p}_1$$

Dove:

$$\boldsymbol{p}_0 \triangleq \begin{bmatrix} p_x^0 \\ p_y^0 \\ p_z^0 \end{bmatrix} \quad \boldsymbol{p}_1 \triangleq \begin{bmatrix} p_x^1 \\ p_y^1 \\ p_z^1 \end{bmatrix} \quad \boldsymbol{R}_0^1 \triangleq \begin{bmatrix} \boldsymbol{i}_1 \cdot \boldsymbol{i}_0 & \boldsymbol{j}_1 \cdot \boldsymbol{i}_0 & \boldsymbol{k}_1 \cdot \boldsymbol{i}_0 \\ \boldsymbol{i}_1 \cdot \boldsymbol{j}_0 & \boldsymbol{j}_1 \cdot \boldsymbol{j}_0 & \boldsymbol{k}_1 \cdot \boldsymbol{j}_0 \\ \boldsymbol{i}_1 \cdot \boldsymbol{k}_0 & \boldsymbol{j}_1 \cdot \boldsymbol{k}_0 & \boldsymbol{k}_1 \cdot \boldsymbol{k}_0 \end{bmatrix}$$

$\boldsymbol{R}^1_0$ rappresenta la matrice di trasformazione per esprimere un vettore $\boldsymbol{p}_1$ (espresso nel sistema ($O - x_1y_1z_1$)) nel vettore equivalente $\boldsymbol{p}_0$ (espresso nel sistema ($O - x_0y_0z_0$)).

B.2.2. Composizione di rotazioni.

Un punto $\boldsymbol{p}$ può essere rappresentato in modo equivalente come p_0,p_1 o p_2, ovvero lo si può esprimere nel sistema S_0,S_2 o nel sistema di riferimento intermedio S_1. Indicando con $\boldsymbol{R}_0^1$ la matrice che esprime la trasformazione dei punti del sistema S_1 nel sistema S_0 e con $\boldsymbol{R}_1^2$ la rotazione del sistema S_1 nel sistema S_2 si ottiene:

$$\boldsymbol{R}_0^2 = \boldsymbol{R}_0^1 \boldsymbol{R}_1^2$$

In sostanza, per esprimere un punto p_2 (nel sistema $O - x_2y_2z_2$) nel sistema $O - x_0y_0z_0$, si deve prima passare dal sistema di riferimento intermedio $O - x_1y_1z_1$. La rotazione complessiva è dunque esprimibile come successione di rotazioni parziali, ciascuna definita rispetto all' esito della rotazione precedente. La terna rispetto alla quale avviene la rotazione in atto viene definita "terna corrente".

In genere le matrici di rotazione forniscono una rappresentazione ridondante dell'orientamento di una terna; esse sono, difatti, caratterizzate da nove elementi che non sono indipendenti, ma legati tra di loro da sei vincoli, dovuti alle condizioni di ortogonalità $\boldsymbol{R}\boldsymbol{R}^T = \mathbf{I}$. Questo indica che il numero di parametri indipendenti è 3. Per rappresentare una rotazione arbitraria è possibile utilizzare uno dei seguenti metodi:

1. Rappresentazione Asse/Angolo
2. Rappresentazione con Angoli di Eulero
3. Rappresentazione in Roll,Pitch e Yaw

B.2.3. Trasformazioni omogenee

Si consideri il sistema di coordinate $O - x_0y_0z_0$ ed il sistema $O' - x_1y_1z_1$ traslati tra loro di un vettore $\boldsymbol{d}$ (centrato in S_0) di modulo $|\mathrm{d}|$, come rappresentato figura: la terna S_1 ha gli assi paralleli a S_0.

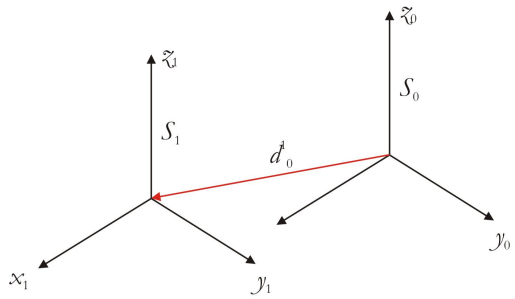


Figura 2 – traslazione dei due sistemi S_0 e S_1

Inoltre $\boldsymbol{d}$ è un vettore che permette di esprimere un punto del sistema S_1 nel sistema S_0; cioè vale la relazione:

$$\boldsymbol{p}_o = \boldsymbol{p}_1 + \boldsymbol{d}_0^1$$

Generalizzando al caso di moto roto-traslatorio, risulta che la relazione più generale tra due sistemi di riferimento $O - x_0 y_0 z_0$ e $O - x_1 y_1 z_1$ può essere espressa come la combinazione di una rotazione e di una traslazione, detta "moto rigido" o "trasformazione di coordinate" (traslazione+rotazione).

Per ottenere una rappresentazione compatta del legame esistente tra le rappresentazioni delle coordinate di uno stesso punto rispetto a due terne differenti, si introduce la cosiddetta matrice di Trasformazione Omogenea:

$$\boldsymbol{A}_0^1 = \begin{pmatrix} \boldsymbol{R}_0^1 & \boldsymbol{d}_0^1 \\ 0^T & 1 \end{pmatrix} \in \mathbb{R}^{4\times 4}$$

Un moto rigido può essere rappresentato da un insieme di matrici, dette Matrici di Trasformazioni Omogenee, del tipo:

$$\boldsymbol{H} = \begin{pmatrix} \boldsymbol{R} & \boldsymbol{d} \\ 0 & 1 \end{pmatrix}$$

con $\boldsymbol{R} \in$ SO(3) e H$\in \mathbb{R}^{4\times 4}$.

B.3. Cenni sulla cinematica diretta, inversa e differenziale di un manipolatore

La struttura meccanica di un manipolatore è caratterizzata da un numero di gradi di mobilità che ne determinano la configurazione. Ogni grado di mobilità viene tipicamente associato ad una articolazione di giunto e costituisce una cosiddetta "variabile di giunto" $\boldsymbol{q}$.

La cinematica diretta si occupa di determinare la posizione e l'orientamento dell'organo terminale del manipolatore, in funzione dei valori assunti dalle variabili di giunto.

Il problema cinematico inverso consiste nello studio di alcune proprietà e tecniche di calcolo per ottenere informazioni utili sulle variabili di giunto una volta nota la posizione dell'end-effector nello spazio.

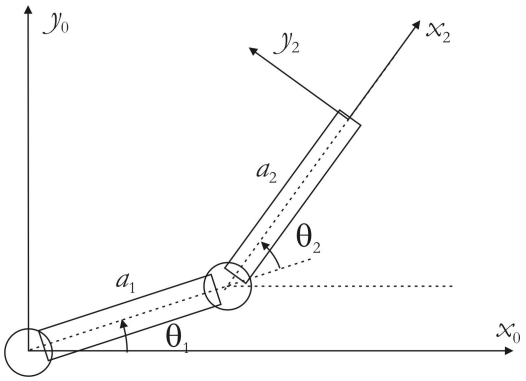


Figura 3 – cinematica diretta in 2D

Nel caso della cinematica diretta, come prima cosa, si fissa un sistema di coordinate, detto sistema base S_0. Così facendo è possibile esprimere anche le coordinate dell'end-effector nel sistema S_0. Nel seguito ci si riferisce al caso bidimensionale SO(2).

Per ricavare la posizione dell'end-effector si scrive:

$$\begin{cases} x = a_1 \cos\theta_1 + a_2 \cos(\theta_1 + \theta_2) \\ y = a_1 \sin\theta_1 + a_2 \sin(\theta_1 + \theta_2) \end{cases} \qquad \text{(B.1)}$$

mentre la matrice che esprime l'orientamento del sistema S_2 rispetto al sistema S_0 risulta:

$$\begin{bmatrix} \cos(\theta_1 + \theta_2) & -\sin(\theta_1 + \theta_2) \\ \sin(\theta_1 + \theta_2) & \cos(\theta_1 + \theta_2) \end{bmatrix}$$

Nel caso in cui si abbia un manipolatore a più gradi di libertà non è immediato ricavare le equazioni cinematiche. Gli strumenti che si usano sono pertanto:

1. Coordinate omogenee
2. Trasformazioni omogenee
3. Convenzione di Denavit-Hartenberg

La soluzione del problema della cinematica inversa, invece, può non esistere e, se la soluzione esiste, può essere anche non unica. Le equazioni sono funzioni non lineari nelle variabili dei giunti e le soluzioni sono molto difficili da ottenere in forma analitica.

La cinematica differenziale si occupa di relazionare la velocità dei giunti con quella dell'end effector. La soluzione viene di solito data sotto forma di una matrice detta Jacobiano del

manipolatore. Come esempio, si considerino le equazioni (B.1) viste per la cinematica di un manipolatore planare a due bracci. Derivando in funzione del tempo (sia θ_i che x e y sono funzioni del tempo t) risulta:

$$\dot{x} = -a_1 \sin\theta_1\dot{\theta}_1 - a_2 \sin(\theta_1+\theta_2)(\dot{\theta}_1+\dot{\theta}_2)$$

$$\dot{y} = a_1 \cos\theta_1\dot{\theta}_1 + a_2 \cos(\theta_1+\theta_2)(\dot{\theta}_1+\dot{\theta}_2)$$

E, utilizzando la notazione vettoriale, si può passare alla notazione matriciale:

$$\begin{pmatrix}\dot{x}\\ \dot{y}\end{pmatrix} = \begin{pmatrix}-a_1\sin\theta_1 - a_2\sin(\theta_1+\theta_2) & -a_2\sin(\theta_1+\theta_2)\\ a_1\cos\theta_1 - a_2\sin(\theta_1+\theta_2) & a_2\cos(\theta_1+\theta_2)\end{pmatrix}\begin{pmatrix}\dot{\theta}_1\\ \dot{\theta}_2\end{pmatrix} = \boldsymbol{J}\begin{pmatrix}\dot{\theta}_1\\ \dot{\theta}_2\end{pmatrix}$$

Con $\boldsymbol{J}$ che è lo Jacobiano della trasformazione. A questo punto è possibile ricavarsi:

$$\begin{pmatrix}\dot{\theta}_1\\ \dot{\theta}_2\end{pmatrix} = \boldsymbol{J}^{-1}\begin{pmatrix}\dot{x}_1\\ \dot{y}_2\end{pmatrix}$$

Per quanto riguarda l'invertibilità di J, si nota che il suo determinante si annulla per θ_2=0,π. In questo caso si dice che il manipolatore si trova in una "configurazione singolare". In questa configurazione non è più possibile passare da un vettore $\dot{\boldsymbol{x}}$ al corrispondente vettore derivato degli stati tramite la $\boldsymbol{J}^{-1}$ e si dice che il manipolatore perde un grado di libertà. Ciò vuol dire che sono preclusi alcuni movimenti al manipolatore stesso.

B.3.1. Derivazione dello Jacobiano

Nel caso di trasformazioni omogenee è stata ricavata l'espressione:

$$\boldsymbol{H} = \begin{pmatrix}\boldsymbol{R}_0^n & \boldsymbol{d}_0^n\\ 0^T & 1\end{pmatrix}$$

Da cui è possibile anche scrivere:

$$\boldsymbol{H} = \boldsymbol{T}_0^n = \boldsymbol{A}_1(\boldsymbol{q}_1)\ldots\boldsymbol{A}_n(\boldsymbol{q}_n)$$

Dove ogni trasformazione omogenea è nella forma:

$$\boldsymbol{A}_i = \begin{pmatrix}\boldsymbol{R}_{i-1}^i & \boldsymbol{d}_{i-1}^i\\ 0^T & 1\end{pmatrix}$$

e quindi:

$$\boldsymbol{T}_i^j = \boldsymbol{A}_i(\boldsymbol{q}_i)\ldots\boldsymbol{A}_j(\boldsymbol{q}_j) = \begin{pmatrix}\boldsymbol{R}_i^j & \boldsymbol{d}_i^j\\ 0^T & 1\end{pmatrix}$$

Dove:

$$\boldsymbol{R}_i^j = \boldsymbol{R}_i^{i+1} \ldots \boldsymbol{R}_{j-1}^j \quad \text{e} \quad \boldsymbol{d}_i^j = \boldsymbol{d}_i^{j-1} + \boldsymbol{R}_i^{j-1} \boldsymbol{d}_{j-1}^j$$

Se si considera, ora, un manipolatore a n bracci, con $\boldsymbol{q}_1, \ldots, \boldsymbol{q}_n$ le variabili di giunto, e sia

$$\boldsymbol{T}_0^n(\boldsymbol{q}) = \begin{pmatrix} \boldsymbol{R}_0^n(\boldsymbol{q}) & \boldsymbol{d}_0^n(\boldsymbol{q}) \\ \boldsymbol{0}^T & 1 \end{pmatrix}$$

la matrice di trasformazione che esprime punti riferiti al sistema end effector nel sistema base.

Un'osservazione importante è che, potendo il robot muoversi, sia le variabili di giunto $\boldsymbol{q}_i$ che la posizione dell'end-effector $\boldsymbol{d}_0^n$ che la sua orientazione $\boldsymbol{R}_0^n$ saranno funzione del tempo

È possibile definire il vettore della velocità angolare dell'end-effector $\boldsymbol{\omega}_0^n$:

$$\boldsymbol{S}(\boldsymbol{\omega}_0^n) = \dot{\boldsymbol{R}}_0^n (\boldsymbol{R}_0^n)^T$$

E si indichi con $\boldsymbol{v}_0^n = \dot{\boldsymbol{d}}_0^n$ la velocità lineare dell'end effector. Lo scopo della cinematica differenziale è trovare una relazione lineare della forma

$$\boldsymbol{v}_0^n = \boldsymbol{J}_v \dot{\boldsymbol{q}} \boldsymbol{\omega}_0^n = \boldsymbol{J}_\omega \dot{\boldsymbol{q}}$$

con $\boldsymbol{q}$ che rappresenta la variabili di giunto e dove $\boldsymbol{J}_v$ e $\boldsymbol{J}_\omega$ sono matrici 3×*n*. In forma compatta è possibile scrivere:

$$\begin{pmatrix} \boldsymbol{v}_0^n \\ \boldsymbol{\omega}_0^n \end{pmatrix} = \boldsymbol{J}_0^n \dot{\boldsymbol{q}}$$

dove:

$$\boldsymbol{J}_0^n = \begin{pmatrix} \boldsymbol{J}_v \\ \boldsymbol{J}_\omega \end{pmatrix}$$

La matrice $\boldsymbol{J}_0^n$ è chiama Jacobiano del manipolatore o, semplicemente, Jacobiano. Si noti che $\boldsymbol{J}_0^n$ è una matrice 6×*n* dove *n* è il numero dei bracci.

B.3.2. Velocità angolare e velocità lineare

le velocità angolari possono essere addizionate vettorialmente solo nel caso in cui siano espresse relativamente a un sistema di coordinate comune. Potrebbe essere possibile determinare la velocità angolare dell'end-effector relativamente alla base esprimendo la velocità angolare di ciascun braccio nel sistema di riferimento base, e poi sommarle.

Se si indica con $\boldsymbol{k}_i$ il versore associato all'*i*-esimo braccio, è possibile far vedere che la velocità angolare complessiva dell'end effector è data da:

$$\omega_0^n = \rho_1 \dot{q}_1 k_1 + \rho_2 \dot{q}_2 R_0^1 k_2 + \ldots + \rho_n \dot{q}_n R_0^{n-1} k_n = \sum_{i-1}^{n} \rho_i \dot{q}_i z_{i-1}$$

Dove

$$z_{i-1} = R_0^{i-1} k_i$$

e dove ρ_i=1 se il giunto *i*-esimo è rotazionale, se prismatico è pari a 0 (è naturale che $z_0 = k_1 = [0,0,1]^T$). In base a ciò è possibile concludere:

$$J_\omega = [\rho_1 z_0, \ldots, \rho_n z_{n-1}]$$

La velocità lineare dell'end-effector è data da $\dot{d}_0^n$. In base alla regola di derivazione a catena si ottiene:

$$\dot{d}_0^n = \sum_{i=1}^{n} \frac{\partial d_0^n}{\partial q_i} \dot{q}_i$$

Quindi la *i*-esima colonna di J_v è data da $\partial d_0^n / \partial q_i$.

BIBLIOGRAFIA

[ANSI, 1999] ANSI, 1999, American National Standard for Industrial Robots and Robot Systems – Safety Requirements. American National Standard Institute, New York, NY, RIA/ANSI R15.06.

[Arbib et al., 1981] Arbib M.A., Kfoury A.J., Moll R.N., 1981, A Basis for Theoretical Computer Science, Springer-Verlag, New York.

[Arbib, 1981] Arbib M. A., 1981, Perceprual Structures and Distributed Motor Control, in Handbook of Physiology – The Nervous System II: Motor Control, ed. V.B. Brooks, American Physiological Society, Bathesda, MD, pp. 1449–80

[Arbib, 1992] Arbib M. A., 1992, Schema Theory, in The Encyclopedia of Artificial Intelligence, 2nd ed., ed. S. Shapiro, Wiley-Interscience, New York, N.Y., pp. 1427–43

[Arbib, 1995], Arbib M. A., 1995, Schema Theory, in The Handbook of Brain Theory and Neural Networks, ed. M.A. Arbib.

[Arkin e MacKenzie, 1994] Arkin R.C., MacKenzie D., 1994, Temporal Coordination of Perceptual Algorithms for Mobile Robot Navigation, in IEEE Transaction on Robotics and Automation, vol. 10, No. 3, pp. 276-286

[Arkin, 1987] Arkin R.C., 1987, Motor Schema Based Navigation for a Mobile Robot: An Approach to Programming by Behavior, in Proceedings of the IEEE Conference on Robotics and Automation, Raleigh, NC, pp. 264-271.

[Arkin, 1998] Arkin, R.C. 1998. Behavior-Based Robotics, The Mit Press.

[Bartlett, 1932] Bartlett F. C., 1932, Remembering: A Study in Experimental and Social Psychology, London, Cambridge University Press.

[Bennewitz et al., 2003] Bennewitz, M., Burgard, W., and Thrun, S., 2003, Adapting navigation strategies using motion patterns of people. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA).

[Bicchi e Kumar, 2000] Bicchi A., Kumar V., 2000. Robotic grasping and contact: A review, in *Proc. of the IEEE Int. Conf. on Robotics and Automation*, pp. 348–353

[Blackmon et al., 1999] Blackmon, T., Thayer, S., Teza, J., Broz, V., Osborn, J., and Hebert, M., 1999, Virtual reality mapping system for chernobyl accident site assessment. In *Proceedings of the SPIE, Vol. 3644*, pp 338–345.

[Boccignone et al., 2004] Boccignone G., Ferraro M., Napoletano P., 2004, Diffused Expectation Maximisation for Image Segmentation, Electronics Letters Vol. 40, No. 18.

[Bon e Seraji, 1996] Bon B., Seraji H., 1996, On-line collision avoidance for the Ranger telerobotic flight experiment, IEEE International Conference on Robotics and Automation, Minneapolis, MN.

[Brady et al., 1998] Brady, K., Tarn, T., Xi, N., Love, L., Lloyd, P., Davis, H., and Burks, B., 1998, Remote systems for waste retrieval from the oak ridge national laboratory gunite tanks. *International Journal of Robotics Research*, 17(4):450–460.

[Brock e Khatib, 2002] Brock O., Khatib O., 2002, Elastic strips: a framework for motion generation in human environments, International Journal of Robotics Research, vol. 21, pp. 1031–1052.

[Brooks, 1986] Brooks R., 1986, A robust Layered Control System for a Mobile Robot, IEEE Journal of Robotics and Automation, vol. RA-2m No. 1, pp. 14-23

[Brooks, 1990] Brooks R., The Behavior Language, A.I. Memo No. 1227, MIT AI Laboratory.

[Brooks, 1991a] Brooks R. A., 1991, Intelligence without representation, Atificial Intelligence vol 47, pp. 139-159.

[Brooks, 1991b] Brooks R. A., 1991, Intelligence without reason, In Proceedings of the 12th International Joint Conference on Artificial Intelligence (IJCAI-91), Sydney, Australia, pp. 569-595.

[Brown e Lowe, 2002], Brown M., Lowe D.G., 2002, Invariant features from interest point groups, British Machine Vision Conference, Cardiff, Wales, pp. 656-665.

[Burke et al., 2004] Burke J.L., Murphy R.R., Rogers E., Lumelsky V.J., Scholtz J., 2004 Final Report for the DARPA/NSF Interdisciplinary Study on Human-Robot Interaction.

[Caccavale et al., 2001] Caccavale F., Natale C., Siciliano B., Villani L., 2001, Achieving a cooperative behaviour in a dual-arm robot system via a modularcontrol structure, Journal of Robotic Systems, vol. 18, pp. 691-699.

[Cappelli and Giovannetti, 2003], Cappelli A., E. Giovannetti, 2003, Human-robot interaction, in A. Cesta (ed.), Proceedings of the *First Robocare Workshop*, Rome.

[Christensen et al, 2003] Christensen H.I., Kragic D., Sandberg F., 2003, Vision for Interaction, in Vision for Interaction, in Sensor Based Intelligent Robots, LNCS, Vol 2238/2002, pp 51-73, Springer Eds.

[Cordeschi, 1998] Cordeschi R., 1998, La scoperta dell'artificiale. Psicologia, filosofia e macchine intorno alla cibernetica, Milano, Masson-Dunod.

[De Falco et al., 2005] De Falco I., Della Cioppa A., Passaro A., Tarantino E., 2005, Genetic Programming for Inductive Inference of Chaotic Series, in Fuzzy Logic and Applications, 6th International Workshop, WILF 2005, Revised Selected Papers, LNCS, vol. 3849, pp 156-163.

[De Falco et al., 2006] De Falco I., Della Cioppa A., Maisto D., Tarantino E., 2006, A Genetic Programming Approach to Solomonoff's Probabilistic Induction, in Proceedings of the 9th European Conference on Genetic Programming, LNCS, vol. 3905, pp. 24-35.

[De Santis et al, 2006] De Santis A, Pierro P, Siciliano B The virtual end-effectors approach for human-robot interaction. In: Lenarcic J, Roth B (a cura di) Advances in Robot Kinematics, Springer, Berlin Heidelberg New York

[De Santis et al, 2007] De Santis A, Albu-Schaeffer A, Ott C, Siciliano B, Hirzinger G, The skeleton algorithm for real-time collision avoidance of a humanoid robot. Submitted to: 2007 IEEE International Conference on Robotics and Automation

[Dean e Wellman, 1991] Dean T., Wellman M., 1991, Planning and Control, Morgan-Kaufmann, San Mateo, CA.

[Dempster et al., 1977] Dempster A.P., Laird N.M., Rubin D.B., 1977, Maximum-likelihood from incomplete data via EM algorithm, Journal of the Royal Statistical Society, Vol. 39, pp 1-38

[Fikes e Nilsson, 1971] Fikes R., Nilsson N., 1971, STRIPS: A New Approach to the Application of Theorem Proving to Problem Solving, Artificial Intelligence, vol. 2, pp. 189-208

[Fleuret e German, 2001] Fleuret, F. and Geman, D. 2001. Coarse-to-fine face detection. *Int. J. Computer Vision*, 41:85–107.

[Fok e Kabuka, 1991] Fok K.-Y., Kabuka, M.R. 1991, An Automatic Navigation System for Vision Guided Vehicles Using a Double Heuristic and a Finite State Machine, in IEEE Transaction on Robotics and Automation, vol. 7, No. 1, pp. 181-188.

[Freeman e Adelson, 1991], Freeman W.T., Adelson E.H., 1991, The design and use of steerable filters, IEEE Transactions on Pattern Analysis and Machine Intelligence, 13(9):891–906.

[Freund e Schapire, 1995] Freund, Y. and Schapire, R.E. 1995. A decision-theoretic generalization of on-line learning and an application to boosting. In *Computational Learning Theory: Eurocolt 95*, Springer-Verlag, pp. 23–37.

[Gat e Dorais, 1994] Gat E., Dorais G., 1994, Robot Navigation by Conditional Sequencing, in Proceedings of the IEEE International Conference on Robotics and Automation, pp 1293-1299.

[Giralt et al., 1984] Giralt G., Chatila R., Vaisset M., 1984, An Integrated Navigation and Motion Control System for Autonomous Multisensory Mobile Robot, First International Symposium on Robotic Research, ed. M. Brady and R. Paul, pp. 191-214

[Goodridge e Luo, 1994]. Goodridge S., Luo R., 1994, Fuzzy Behavior Fusion for Reacive Control of an Autonomous Mobile Robot: MARGE, in Proceedings of the IEEE International Conference on Robotics and Automation, pp. 1622-1627

[Greenspan et al., 1994], Greenspan H., Belongie S., Gooodman R., Perona P., Rakshit S., Anderson C., 1994, Overcomplete steerable pyramid filters and rotation invariance, in Proceedings of the IEEE Conference on Computer Vision and Pattern Recognition.

[Hayes-Roth, 1995] Hayes-Roth B., 1995, An Architecture for Adaptive Intelligent Systems, Arificial Intelligence, vol. 72, No.1-2, January, pp. 329-365.

[Head e Holmes 1911] Head H., Holmes G., 1911, Sensory Disturbances from Cerebral Lesions, Brain, vol. 34, p. 102

[Itti et al, 1998] Itti, L., Koch, C., and Niebur, E. 1998. A model of saliency-based visual attention for rapid scene analysis. *IEEE Patt. Anal. Mach. Intell.*, 20(11):1254-–1259.

[Jablonsky e Posey, 1985] Jablonsky J., Posey J., 1985, Robotic Terminology, in Handbook of Industrial Robotics, ed. S. Nof, J. Wiley, New York, pp. 1271-1303

[Kaelbling e Rosenschein, 1991] Kaelbling L., Rosenschein S., 1991, Action and Planning in Embedded Agents, in Designing Autonomous Agents, ed. P. Maes, MIT Press, Cambridge, MA, pp. 35-48

[Khatib, 1985] Khatib O., 1985, Real Time Obstacle Avoidance for Manipulators and Mobile Robots, in Proceedings of the IEEE International Conference on Robotics and Automation, St. Louis, MO, pp. 500-505

[Khatib, 1986] Khatib O., 1986, Real-time obstacle avoidance for robot manipulators and mobile robots, International Journal of Robotics Research, vol. 5(1), pp. 90–98.

[Kidd, 1992] Kidd, P.T. 1992. Design of human-centred robotic systems in Mansour Rahimi and Waldemar Karwowski (Eds.) Human Robot Interaction. Taylor and Francis: London. 225-241.

[Kim e Khosla, 1992] Kim J., Khosla P., 1992, Real-Time Obstacle Avoidance Using Harmonic Potential Functions, IEEE Transaction on Robotics and Automation, vol. 8, No. 3, June, pp. 338-349

[King and Weiman, 1990] King, S. and Weiman, C., 1990, Helpmate autonomous mobile robot navigation system, in Proceedings of the SPIE Conference on Mobile Robots, pp 190–198, Boston, MA. Volume 2352.

[Koller, 1991] Koller D., Daniilidis K., Nagel H. H. 1993, Model-based object tracking in monocular image sequences of road traffic scenes," *Intl.Jour. of Computer Vision*, vol. 10, no. 3, pp. 257–281

[Koren e Borenstein, 1991] Koren Y., Borenstein J., 1991, Potential Field Methods and Their Inherent Limitations for Mobile Robot Navigation, in Proceedings of the IEEE International Conference on Robotics and Automation, Sacramento, CA, pp.1398-1404

[Koy-Oberthur, 1989] Koy-Oberthur R., 1989, Perception by Sensorimotor Coordination in Sensory Substitution for the Blind, in Visuomotor Coordination: Amphibians, Comparisions, Models, and Robots, eds. J.P. Ewert and M.A. Arbib, Plenum, New York, pp. 397-418

[Krog, 1984] Krog B., 1984, A Generalized Potential Field Approach to Obstacle Avoidance Control, SME-RI Technical Paper MS84-484, Society of Manufacturing Engineers, Dearborn, Michigan

[Kuffner et al., 2002], Kuffner J., Nishiwaki K., Kagami S., Kuniyoshi Y., Inaba M., Inoue H., 2002, Self-collision detection and prevention for humanoid robots, IEEE International Conference on Robotics and Automation, Washington, DC.

[Latombe, 1991] Latombe J-C., Robot Motion Planning, Kluwer Academic Publisher, Boston

[Lindberg, 1994] Lindeberg T. 1994, Scale-space theory: A basic tool for analysing structures at different scales, Journal of Applied Statistics, 21(2):224-270.

[Lorenz, 1981] Lorenz K, 1981, The Foundations of Ethology, Springer–Verlag, New York.

[Lowe, 1991] Lowe D. 1991,Fitting parameterized three–dimensional models to images, *IEEE Transactions on Pattern Analysis and Machine Intelligence*, vol. 13, no. 5, pp. 441–450, 1991.

[Lowe, 2004] Lowe D, 2004, Distinctive image features from scale-invariant keypoints, International Journal of Computer Vision, 60(2):91–110, 2004.

[Mataric, 1992] Maratic M., 1992, Behavior Based Control: Main Prospective and Implications, in Proceedings of Workshop on Intelligent Control Systems, International Conference on Robotics and Automation, Nice, France

[McFarland e Bosser, 1993] McFarland D., Bosser U., 1993, Intelligent Behavior in Animals and Robots, MIT Press, Cambridge, MA.

[Merchand et al., 2000] Marchand E., Bouthemy P., Chaumette F. 2000, A 2D–3D Model– Based Approach to Real–Time Visual Tracking, Technical report ISSN 0249-6399, ISRN INRIA/RR-3920, Unit´e de recherche INRIA Rennes, IRISA, Campus universitaire de Beaulieu, 35042 Rennes Cedex, France

[Mikolajczyk, 2002], Mikolajczyk K., 2002, Detection of local features invariant to affine transformations, Ph.D. thesis, Institut National Polytechnique de Grenoble, France.

[Miller e Allen, 1999] Miller A., Allen O., 1999. Examples of 3D grasp quality computations," in *Proc. of the IEEE International Conference on Robotics and Automation*, vol. 2, pp. 1240–1246

[Miller et al., 1960] Miller G., Galanter E., Pribram K., 1960, Plans and the Structure of Behavior, Holt, Rinehart, and Winston, New York, pag. 16.

[Moravec, 1977] Moracev H., 1977, Towards Automatic Visual Obstacle Avoidance, in Proceedings of the Fifth International Joint Conference on Artificial Intelligence, Cambridge, MA, August, p. 584.

[Moravec, 1988] Moravec H., 1988, Mind Children: The Future of Robot and Human Intelligence, Harvard University Press, Cambridge, MA

[Murphy, 2000] Murphy, R. 2000. An Introduction to AI Robotics, The Mit Press.

[Neisser ,1976] Neisser U., 1975, Cognition and Reality: Principles and Implications of Cognitive Psychology, W.H. Freeman, San Francisco.

[Nilsson, 1969] Nilsson N., 1969, A Mobile Automaton: An Application of Artificial Intelligence Techniques, in Proceedings of the International Joint Conference on Artificial Intelligence (IJCAI-69). Washington D.C., May. Reprinted in Autonomous Mobile Robots, vol. 2, eds. S. Iyengar and A. Elfes, IEEE Computer Society Press, Los Alamitos, 1991, pp. 233-244.

[Nilsson,1994] Nilsson N., 1994, Teleo-Reactive Programs for Agent Control, Journal of Artificial Intelligence Research, vol. 1, pp. 139-158

[Norman e Shallice, 1986] Norman D., Shallice T., 1986, Attention to Action: Willed and Automatic Control of Behavior, in Consciousness adn Self-Regulation: Advances in research and Theory, vol. 4, eds. R. Davidson, G. Schwartz, and D. Shapiro, Plenum Press, New York, pp. 1-17

[OpenCV] INTEL RESEARCH LAB. Opencv: Open computer vision library. http://sourceforge.net/projects/opencvlibrary/.

[Osuna et al., 1997a], Osuna E., Freund R., Girosi, F. 1997, Training support vector machines: An application to face detection. In Proceedingsof the IEEE Conference on Computer Vision and PatternRecognition.

[Ott et al, 2006] Ott C., Eiberger O., Friedl W., Bäuml B., Hillenbrand U., Borst C., Albu-Sch¨affer A., Brunner B., Hirschm¨uller H., Kiehl¨ofer S., Konietschke R., Suppa M., Wimb¨ock T., Zacharias F., Hirzinger G., 2006, A humanoid two-arm system for dexterous manipulation, IEEE International Conference on Humanoid Robots, Genova.

[Papageorgiou et al., 1998] Papageorgiou C., Oren M., Poggio T., 1998, A general framework for object detection, in International Conference on ComputerVision

[Payton et al., 1992] Payton D., Keirsey D., Kimble D., Krozel J., Rosenblatt J., 1992, Do Whatever Works: A Robust Approach to Fault-tolerant Autonomous Control, Applied Intelligence, Vol. 2, No. 3, September, pp. 225-250.

[Piaget, 1971] Piaget J., 1971, Biology and Knowledge: An Essay on the Relations between Organic Regulations and Cognitive Processer, University of Chicago Press.

[Replics, 2005] Caccavale F., Lippiello V., Siciliano B., Villani L., 2005, RePLiCS: An environment for open real-time control of a dual-arm industrial robotic cell based on RTAI-Linux, IEEE/RSJ International Conference on Intelligent Robots and Systems, Edmonton, CND, Aug. 2005.

[Riseman e Hanson, 1987] Riseman E., Hanson A., General Knowledge-Based Vision Systems, in Vision, Brain and Cooperative Computation, eds. M. Arbib and A. Hanson, MIT Press, Cambridge, MA, p. 287

[Rosenblatt e Payton, 1989] Rosenblatt J., Payton D., 1989, A Fine-Grained alternative to the Subsumption Architecture for Mobile Robot Control, in

Proceedings of the International Joint Conference on Neural Networks, June, pp. 317-323

[Rosenblatt, 1995] Rosenblatt J., 1995, DAMN: a Distributed Architecture for Mobile Navigation, Working Notes, AAAI 1995 Spring Symposium on Lessons Learned for Implemented Software Architectures for Physical Agents, Palo Alto, CA, March, pp. 167-178.

[Roth et al., 2000] Roth D., Yang M., Ahuja N. 2000. A snowbased face detector, in Neural Information Processing 12.

[Roth et al., 2000] Roth, D., Yang, M., and Ahuja, N. 2000. A snowbased face detector. In *Neural Information Processing* 12.

[Rowley et al., 1998], Rowley, H., Baluja, S., and Kanade, T. 1998. Neural network-based face detection. IEEE Patt. Anal. Mach. Intell., 20:22–38.

[Sacerdoti, 1974] Sacerdoti E., 1974, Planning in a Hierarchy of Abstraction Spaces, Artificial Intelligence, vol. 5, No. 2, pp. 115-135

[Sacerdoti, 1975] Sacerdoti E., 1975, A Structure for Plans and Behavior, Ph.D. Dissertation, Technical Note No. 109, AI Center, SRI International, Menlo Park, CA.

[Saffiotti et al, 1995] Saffiotti A., Konolige K., Ruspini E., 1995, A Multi-Valued Logic Approach to Integrating Planning and Control, Artificial Intelligence, Vol. 76, No. 1-2, July, pp.481-526

[Saffiotti et al., 1993] Saffiotti A., Ruspini E.,Konolige K., 1993, Blending Reactivity and Goal-Directedness in a Fuzzy Controller, in Second IEEE International Conference on Fuzzy Systems, San Francisco, CA, March, pp. 134-139

[Schneiderman e Kanade, 2000] Schneiderman, H. and Kanade, T. 2000. A statistical method for 3D object detection applied to faces and cars. In *International Conference on Computer Vision*.

[Schneidermann e Kanade, 2000], Schneiderman H, Kanade T, 2000, A statistical method for 3D object detection applied to faces and cars, in International Conference on Computer Vision.

[Schulz et al., 2001] Schulz, D., Burgard,W., Fox, D., and Cremers, A., 2001, Tracking multiple moving targets with a mobile robot using particles filters and statistical data association. In Proceedings of the IEEE International Conference on Robotics and Automation, Seoul, Korea.

[Sciavicco e Siciliano, 2000] Sciavicco L., Siciliano B., 2000, Modelling and Control of Robot Manipulators, (2nd Ed.), Springer-Verlag, London, UK.

[Sciavicco e Siciliano, 2000] Sciavicco, L. , Siciliano, B., Modelling and control of robot manipulators, (2nd Ed.), Springer-Verlag, London, UK, 2000.

[Seraji et al., 1996] Seraji H., Bon B., Steele R., 1997, Real-time collision avoidance for 7-DOF arms, IEEE RAS/RSJ International Conference on Intelligent Robots and Systems, Grenoble, F.

[Simard et al., 1999] Simard P.Y., Bottou L., Haffner P., LeCun Y., 1999, Boxlets: A fast convolution algorithm for signal processing and neural networks, Advances in Neural Information Processing Systems, M. Kearns, S. Solla, and D. Cohn eds., vol. 11, pp. 571–577.

[Steels, 1990] Steels L., 1990, Exploiting Analogical Representations, in Designing Autonomous Agents, ed. P. Maes, MIT Press, Cambridge, MA, pp., 71-88.

[Stone, 1980] Stone H, 1980, Introduction to Computer Architecture, 2nd ed., SRA, Chicago

[Sung e Poggio, 1998] Sung K., Poggio T., 1998, Example-based learning for view-based face detection, IEEE Patt. Anal. Mach. Intell., 20:39–51.

[Sussman, 1975] Sussman, 1975 A Computer Model of Skill Acquisition, American Elsevier, New York.

[Tachi e Komoriya, 1985] Tachi S., Komoroiya, K., 1985, Guide Dog Robot, Robotic Research, Second International Symposium, eds. H. Hanafusa and H. Inoue, MIT Press, Cambridge, MA, pp 333-340.

[Tarabanis et al, 1995], Tarabanis K., Allen P., and Tsai R., 1995, A survey of sensor planning in computer vision, *IEEE Transactions on Robotics and Automation*, vol. 11, no. 1, pp. 86–104.

[Thrun et al. 2000] Thrun S., Beetz M., Bennewitz M., Burgard W., Cremers A.B., Dellaert F., Fox D., Hahnel D., Rosenberg C., Roy N., Schulte J., Schulz D., 2000, Probabilistic algorithms and the interactive museum tour-guide robot Minerva, J. Robotics Res. 19, no. 11.

[Thrun et al., 2003] Thrun, S., H¨ahnel, D., Ferguson, D., Montemerlo, M., Triebel, R., Burgard, W., Baker, C., Omohundro, Z., Thayer, S., and Whittaker, W., 2003, A system for volumetric robotic mapping of abandoned mines. In Proceedings of the IEEE International Conference on Robotics and Automation (ICRA).

[Thrun, 2004] Thrun, S. (2004). Toward a framework for human–robot interaction. *Human–Computer Interaction, 19,* 9–24.

[Tianmiao e Bo, 1992] Tianmiao W., Bo Z., 1992, Time-Varying Potential Field Based on Perception Action Behaviors of Mobile Robot, in Proceedings of the 1992 IEEE International Conference on Robotics and Automation, Nice, France, pp. 2549-2554

[Tinberg, 1953], Tinberg N., 1953, Social Behavior in Animals, Methuem, London.

[Traver et al. 2000] V. J. Traver et al. Making Service Robots Human-Safe. Poc. IEEE/RSJ Int. Conf. on Intelligent Robots and Systems, pp. 696 – 701, 2000.

[Tsai e Chen, 1986] Tsai W., Chen Y., 1986, Adaptive Navigation of Automated Vehicles by Image Analysis Techniques, in IEEE Transaction on Systems, Man and Cybernetics, vol. 16, No. 5, September, pp. 730-740.

[Tsotsos et al, 1995] Tsotsos, J., Culhane, S., Wai, W., Lai, Y., Davis, N., and Nuflo, F. 1995. Modeling visual-attention via selective tuning. *Artificial Intelligence Journal*, 78(1/2):507–545.

[U.N. and I.F.R.R., 2002], 2002. *United Nations and The International Federation of Robotics: World Robotics 2002*. United Nations, New York and Geneva.

[Viola e Jones, 2004] Viola P., Jones M.J. 2004. “Robust Real-Time Face Detection” – *International Journal of Computer Vision*. Internationa Journal of Computer Vision 57(2), 237-254. Kluwer Academic Publisher.

[Walker, 1994] Walker I.D., 1994, Impact configurations and measures for kinematically redundant and multiple armed robot systems, IEEE Transactions on Robotics and Automation, vol. 10, pp. 670–683.

[Watson, 1925] Watson J.B., 1925, Behaviorism, People’s Institute Publishing Co., New York.

[Weiss, 1999] Weiss G., Muliagent Systems, a Modern Approach to Distributed Artificial Intelligence, 1999, Gerhard Weiss eds, the MIT Press, Cambridge Massachesetts, pg 584.

[Witkin, 1983] Witkin A.P., 1983, Scale-space filtering, in Proceedings of the International Joint Conference on Artificial Intelligence, Karlsruhe, Germany, pp. 1019-1022.

[Wooldridge, 1999] Wooldridge M., 1999, Intelligent Agents. In Multiagent Systems, edt. Gerhard Weiss, the MIT Press, Cambridge, MA

[Wunsch e Hirzinger, 1997] Wunsch P., Hirzinger G. 1997, Real-time visual tracking of 3D objects with dynamic handling of occlusion, in *Proceedings of the IEEE International Conference on Robotics and Automation, ICRA'97*, vol. 2, pp. 2868–2873,

[Zivanovic and Davies, 2000] Zivanovic, A. and Davies, B., 2000, A robotic system for blood sampling. IEEE Transactions on Information Technology in Biomedicine, 4(1).