POLITECNICO DI MILANO

Scuola di Ingegneria Industriale e dell'Informazione

Corso di Laurea Magistrale in Ingegneria dell'Automazione

COOPERAZIONE UOMO-ROBOT: CRITERI PER LA VALUTAZIONE E IL CALCOLO DEGLI SPAZI MINIMI DI SICUREZZA

Relatore: Prof. Vittorio RAMPA

Correlatore: Ing. Federico Vicentini

Tesi di Laurea Specialistica di:

Matteo GIUSSANI Matr. 786472

Anno Accademico 2013-2014

“Un cassetto di calze di lana. Questo è sicurezza!”

- *Linus, in Charles M. Schulz, Peanuts.*

Ringraziamenti

Desidero innanzitutto ringraziare il mio relatore ing. Vittorio Rampa e, in particolare, il mio correlatore ing. Federico Vicentini, che mi ha seguito con pazienza, sapendo sempre come stimolare la mia curiosità.
Desidero ringraziare anche tutto il gruppo con cui mi sono trovato a lavorare e che mi ha aiutato in questo periodo, in particolare un grazie a Nicola Pedrocchi, Loris Roveda, Paolo Magnoni, Tito Dinon e Alessandro Scano.
Un grazie speciale però va soprattutto a chi mi ha sempre sostenuto e spronato a dare il meglio di me, la mia famiglia, Giorgia la mia paziente fidanzata, gli amici e tutti quelli che hanno reso possibile tutto questo.

Indice

Elenco delle figure

Abstract

Nella presente tesi, sviluppata presso l'Istituto di Tecnologie Industriali e Automazione del Consiglio Nazionale delle Ricerche (ITIA-CNR), viene descritto e implementato un nuovo metodo di calcolo per le distanze di sicurezza, che devono essere mantenute tra uomo e robot per garantire la sicurezza dell'operatore, sulla base di informazioni generate dai sensori e robot.

L'approccio con cui si è affrontato il problema è stato quello di trovare una soluzione che generi delle zone di pericolo attorno al robot che siano calcolabili in tempo reale e che siano funzione di tempi di latenza di rete, velocità di robot e operatore e della posizione relativa tra robot e operatore tenendo conto della configurazione del manipolatore.

Per raggiungere questo scopo è stata necessaria una analisi su una generica rete di comunicazione, si è definito un protocollo di comunicazione attraverso la quale vengono veicolati i dati provenienti dai sensori e robot. Inoltre si è effettuato uno studio dei tempi di latenza delle varie componenti di rete e dei tempi di calcolo necessari.

Si è studiato inoltre come utilizzare canali sensoriali innovativi adeguati ad un tracciamento dell'operatore in tempo reale, e come utilizzare tale informazione per il calcolo di condizioni di sicurezza dinamiche in base alle condizioni operative locali.

E' stato poi costruito un algoritmo basato in grado di discriminare in ogni istante se le condizioni di base di robot e operatore soddisfano dei requisiti di sicurezza.

Per avere una sensibilità a diversi scenari sono stati studiati due casi reali in cui erano presenti sensori e robot aventi caratteristiche diverse. Nel primo caso si è studiato un sistema composto da un robot KUKA LWR4+ e un sensore infrarosso NDI Polaris, mentre nel secondo caso si è studiato un sistema di radiolocalizzazione passiva e robot Comau.

I risultati sono stati analizzati sia in simulazione, attraverso la creazione di Modelli e Simulazioni Matlab, sia sul campo, mediante delle prove sperimentali condotte su robot e sensori reali in cui parti di codice sono state scritte in linguaggio C++.

I risultati mettono in evidenza il miglioramento delle prestazioni del sistema e il vantaggio di avere delle zone di sicurezza che inseguano il moto del robot in modo continuo

Abstract

In this thesis, developed at ITIA-CNR laboratories, a new method for the computation of the minimum safety distances between robot and operator is introduced and developed. This method is based on the information provided by sensors and robots and the safety distance allow the safe-sharing of a shared workspace.

To address this problem, the danger zone around the robot that is calculated in real time. It is a function of latency of the network, speed of the robot and the operator and relative position between robot and operator taking into account the configuration of the manipulator.

To achieve this goal the communication network used to move sensor and robot data to a central node has been analyzed. The communication protocol, the latency of the various components of the network and the computing time has been considered, too.

It has also been studied how to use innovative sensor channels suitable for a real-time tracking of the operator, and how to use this information for the calculation of dynamic safety conditions according to local operating conditions.

The use of danger field has been analyzed by using simulatio tools able to evaluate in real time the status of the robot and the operator to guarantee safety.

To practice with different scenarios, two cases with different robots and sensors with different configurations have been evaluated. The first case describes a system composed of a KUKA robot and an infrared sensor LWR4 + NDI Polaris, while in the second one shows a system consisting of a passive radio-localization sensor network and a Comau robot.

The results has been analyzed by using simulation tools, through the creation of models and Matlab simulations, and by employing experimental tests conducted on real robots and sensors; in this case, parts of the code have been written in the C ++ language.

The results highlight the improvement of system performances and the advantage of having the safety zones that dynamically change by tracking the motion of the robot.

Capitolo 1
Introduzione

Le nuove frontiere della robotica industriale vedono rafforzarsi [1],[2] le modalità cooperative con l'operatore, come definite dagli standard più attuali [3]. Il tema della sicurezza nella condivisione dello spazio di lavoro tra uomo e robot è pertanto sempre più attuale e di vitale interesse sia per la comunità industriale che accademica.

Ci sono molte situazioni in cui la cooperazione di robot con l'uomo può essere interessante e può migliorare le prestazioni di un sistema di automazione, come ad esempio nell'ambito del robot-assisted manufacturing e durante le operazioni di assembly/disassembly cooperativo. Le funzionalità di maggiore rilievo comprendono la possibilità di operazioni concorrenti sul compito tecnologico (ad es. *handling* cooperativo, guida manuale solo per citare alcuni ambiti) o di co-presenza in compiti svolti in parallelo (ad es. allestimento di *pallet* all'interno di una cella).

In tutti i casi, si evidenzia l'esigenza di utilizzare robot in cella aperta, cioè in uno spazio senza barriere di nessun tipo, che l'uomo e i robot condividono. È molto spesso necessario mantenere anche la modalità normale di lavoro dei robot (e.g. piena velocità del compito programmato) durante la condivisione dello spazio di lavoro, ovvero si deve tentare di evitare il più possibile l'abilitazione di modalità degradate (e.g. riduzione della velocità del robot). A fattor comune, è necessario sempre garantire la sicurezza di questi scenari condivisi.

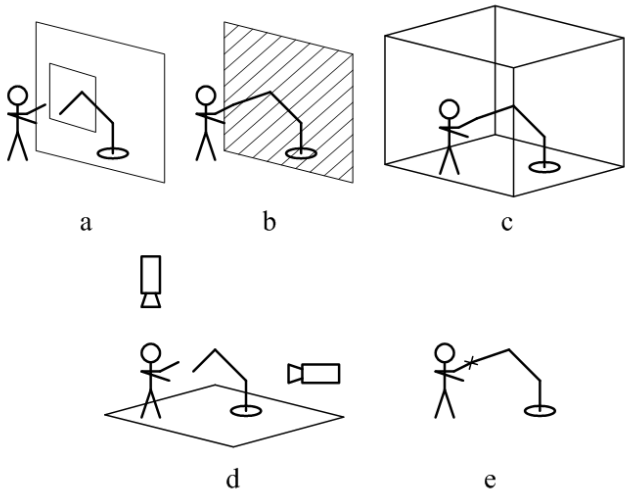


Figura 1.1 – modalità di cooperazione ammesse tra uomo e robot, standard ISO 10218-2.

Il prerequisito comune nelle tecnologie di sicurezza nelle modalità cooperative in robotica industriale (vedi Fig 1.1) è rappresentato dalla possibilità di monitorare in modalità sicura, con un adeguato livello di sicurezza funzionale [4], la posizione e/o il movimento dell'operatore all'interno della cella robotizzata. A sua volta, il monitoraggio e l'esecuzione delle funzioni di sicurezza si basa sui prerequisiti tecnologici relativi al robot come macchina singola, che deve essere dotato delle funzionalità di sicurezza secondo quanto indicato nello standard ISO 10218-1:2011 [3] (Vedi *safe* robot nell'architettura di sistema di Fig. 1.2).

Una delle problematiche più sentite nella sicurezza dei robot è perciò la mancanza di tecnologie sensoriali dotate di livello di sicurezza tale da mantenere il requisito di sicurezza a livello di intera cella robotizzata e la mancanza di flessibilità nel calcolo/verifica delle condizioni di sicurezza della cella, attualmente ristrette a condizione binaria: presenza o non presenza dell'operatore.

L'obiettivo e l'innovazione di questa tesi è pertanto quello di utilizzare canali sensoriali innovativi adeguati ad un tracciamento dell'operatore in tempo reale, e di utilizzare tale informazione per il calcolo di condizioni di sicurezza dinamiche, ovvero determinate con continuità in base alle condizioni operative locali.

Lo scopo tecnologico risiede nell'ampliamento delle sorgenti di informazioni (ad es. sensori o gruppi di sensori) utilizzabili in sicurezza e nell'aumento di produttività derivante dalla flessibilità delle condizioni di monitoraggio dello stato della cella.

Gli obiettivi di localizzazione dell'operatore in modo sicuro con un sistema sensoriale innovativo e di calcolo delle condizioni di sicurezza saranno affrontato analizzando quali requisiti devono rispettare le varie funzioni hardware/software in modo da potersi inserire nell'architettura utilizzata (vedi Fig. 1.2).

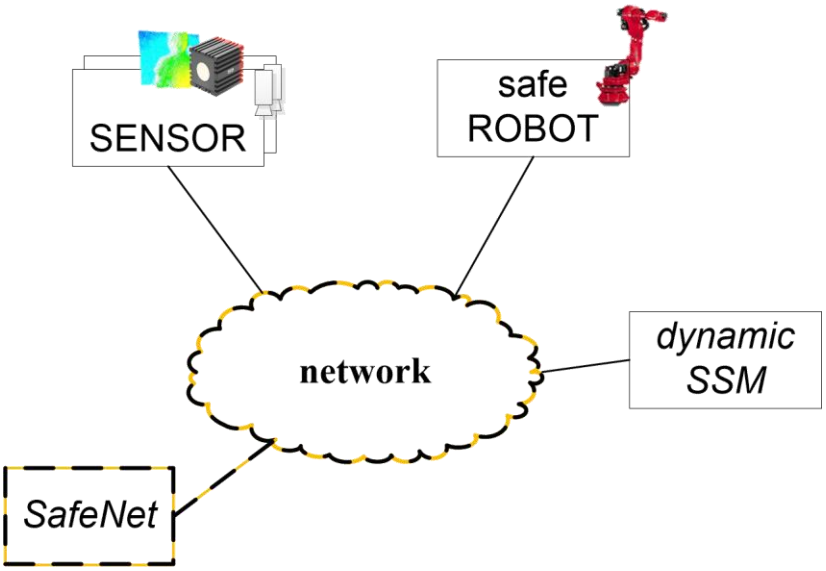


Figura 1.2 – Architettura di sistema creata.

Per fare questo si è utilizzata una architettura di sistema chiamata SafeNet [5], dove il robot e i sensori sono considerati come nodi di una rete che non sono nativamente sicuri, ma dove tutte le operazioni sono coordinate da un insieme di nodi centrali che rappresentano collettivamente la parte relativa alla sicurezza del sistema di controllo (SRP/CS secondo ISO 13489-1[6]), ovvero che gestiscono le comunicazioni, eseguono in tempo reale le operazioni logico-matematiche in modo sicuro e determinano gli output di sicurezza.

Il protocollo di comunicazione di rete utilizzato nell'architettura general purpose di Fig. 1.2 è il protocollo UDP, opportunamente elaborato per poter includere alcuni controlli di integrità del pacchetto richiesti dalle normative di sicurezza, mantenendo la flessibilità e la vasta possibilità di interfaccia dei protocolli IP. In aggiunta ai protocolli IP, sono presenti anche i protocolli di automazione eventualmente utilizzati dai dispositivi specifici (e.g. bus di campo).

I robot e i sensori sono, infatti, interfacciati alla rete attraverso dei controllori programmabili (Programmable Logic Controller o PLC) che gestiscono i segnali in ingresso/uscita sia su protocolli di rete UDP/IP[7] sia sullo specifico bus di campo (X2X e Ethernet POWERLINK nel presente caso) e ritrasmettono l'informazione elaborata in output sicuro su un protocollo dotato delle caratteristiche di sicurezza funzionale [4] adeguate.

Si vuole inoltre determinare lo spazio *minimo* di sicurezza tra uomo e robot sulla base delle condizioni del sistema/architettura sopra descritti e dei compiti (*task*) che stanno eseguendo robot e operatore, in modo da incrementare flessibilità e produttività [8],[9].

Il capitolo 4 riporta quindi una nuova modalità di calcolo della distanza minima di sicurezza dipendente dalla traiettoria in atto, introducendo il concetto di inviluppo di iso-rischio attorno al robot, o all'operatore, calcolato sulla base del tracciamento degli oggetti e delle persone presenti (ad es. mediante il sistema di radio-localizzazione). Il livello di rischio prescelto è determinato sia dalle condizioni di controllo distribuito, attraverso i tempi di reazione del robot genericamente riconducibili alle latenze di trasmissione ed elaborazione, che dalla dinamica del robot. Nel primo caso è necessario identificare le condizioni di sistema (ad es. protocollo, comunicazioni, automazione) in cui viene istanziato ed eseguito il sistema di sicurezza. Emerge che le caratteristiche dei nodi sensoriali rappresentano un contributo determinante alle prestazioni del sistema.

Il passo successivo è stato infatti quello di studiare i tempi di calcolo dei vari sensori utilizzati, i tempi di latenza di rete e il rumore temporale (*jitter*) associato a queste latenze, i tempi di calcolo dei nodi sensore, robot e Check Node. I dettagli sono esposti nel capitolo 4.

Un altro aspetto sviluppato durante questo lavoro di tesi comprende la tecnica passiva di rilevamento dell'operatore basata sulla perturbazione del campo di trasmissione di nodi radio (*wireless*) che permette quindi la localizzazione senza avere bisogno di un sensore attivo indossato dall'operatore.
Inoltre, è stato necessario strutturare ed integrare le informazioni provenienti dai nodi radio, equivalenti ad un unico sensore distribuito, all'interno dell'architettura sicura SafeNet.

Sono presentati infine nel capitolo 5 i risultati preliminari ottenuti mediante simulazione per due scenari applicativi con differenti dispositivi fisici e successiva verifica sperimentale dell'algoritmo di calcolo dello spazio di minima di sicurezza in uno dei due sistemi, in modo da evidenziare sia teoricamente che sperimentalmente le caratteristiche di accuratezza e di prontezza dei sistemi integrati per la sicurezza.

Nel capitolo 6 sono presentate le conclusioni e i possibili lavori futuri.

Capitolo 2
Motivazioni e contesto applicativo (Analisi dello stato dell'arte nelle tecnologie di sicurezza)

2.1 Stato dell'arte e normative ISO

L'interazione uomo robot è una delle tematiche che attira la maggior attenzione sia dal lato industriale che dal lato accademico[1],[2].

Proprio per questo acceso interesse sono già nate diverse soluzioni per la messa in sicurezza di ambienti condivisi tra uomo e robot. Gli standard di sicurezza di componente o funzione (tipo B nella classificazione CEN) utilizzati in robotica industriale prevedono talvolta condizioni molto conservative, di limitato utilizzo nelle modalità cooperative. Ad esempio, lo standard di tipo B ISO 13855 [10] identifica distanze minime tra organi in movimento indipendenti dalle velocità di esecuzione e dalle condizioni operative dei robot. Contemporaneamente, lo standard di prodotto (tipo C) ISO 10218-2 [11] descrive modalità cooperative più flessibili a condizione che tali modalità siano adeguatamente trattate in tutte le fasi della valutazione del rischio (*risk assessment*) e che i sistemi di robot siano dotati di tecnologie di monitoraggio dei guasti di livello di sicurezza funzionale almeno pari a PLd (secondo ISO 13489[6]). Le linee guida dello standard ISO 10218-2[11] stanno correntemente trovando una (ri)definizione specifica (cioè definizione quantitativa) nelle varie modalità attraverso lo standard ISO/TS 15066[12] in fase di completamento. Specificamente, le distanze di sicurezza saranno determinate dalle condizioni locali di traiettoria piuttosto che predeterminate, in modo da superare/specificare nel riferimento di tipo C le correnti limitazioni nel riferimento più generico di tipo B (ISO 13855)[10].

Il vincolo principale, introdotto dalla normativa in corso, risiede nel fatto che il volume di lavoro protetto (*safeguarded*) deve essere calcolato a priori e caricato nel controllore del robot prima dell'avvio o durante una fase di riscontro (*acknowledge*) da parte dell'operatore. Questo implica che, se le condizioni di lavoro cambiassero, bisognerebbe spegnere il sistema, ricalcolare i nuovi volumi di lavoro e ricaricarli nel controllore del robot. Data la normativa e la tecnologie esistente, ogni modifica parametrica richiede il ricaricamento (*reboot*) del sistema di controllo dedicato alle funzioni di sicurezza (SRP/CS o *Safety Related Part of the Control System* [4]). Alternativamente, tutte le condizioni devono essere introdotte a priori, indipendentemente dalla invocazione puntuale di tali condizioni.

Questo vincolo non permette quindi di realizzare delle elaborazioni sui volumi di lavoro sicuri (*safe*) durante l'esecuzione delle operazioni, cioè in tempo reale, bensì costringe ad avere un volume di lavoro *safeguarded* eccessivamente grande, impedendo quindi all'uomo di avvicinarsi troppo, anche in situazioni in cui non c'è un effettivo pericolo (Vedi Fig. 2.1).

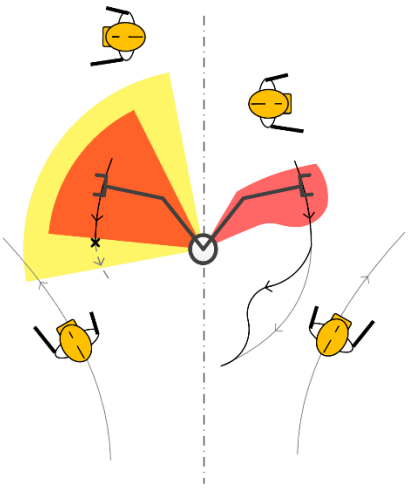


Figure 2.1 – regioni safeguarded statiche a sinistra, calcolo dinamico a destra.

2.2 Soluzioni industriali correnti

Nel seguito di questo paragrafo saranno illustrate diverse soluzioni che sono state implementate in ambito industriale nel rispetto delle normative vigenti.

1. Safety-Eye© prodotto da Pilz[13].
   Questo dispositivo di sicurezza è costituito da un sistema di telecamere che va applicata in una posizione (molto) sopraelevata rispetto alla zona da controllare. Attraverso la configurazione in fase di programmazione, è possibile impostare i volumi software in cui le lavorazioni sono considerate sicure. Questi volumi sono riconducibili a tre diversi livelli di sicurezza (Vedi Fig. 2.2). I volumi devono essere definiti a priori in modo statico. La violazione binaria di tali volumi 2.5D determina l'inserimento di modalità di moto a velocità ridotta (*safe limited speed* – SLS).

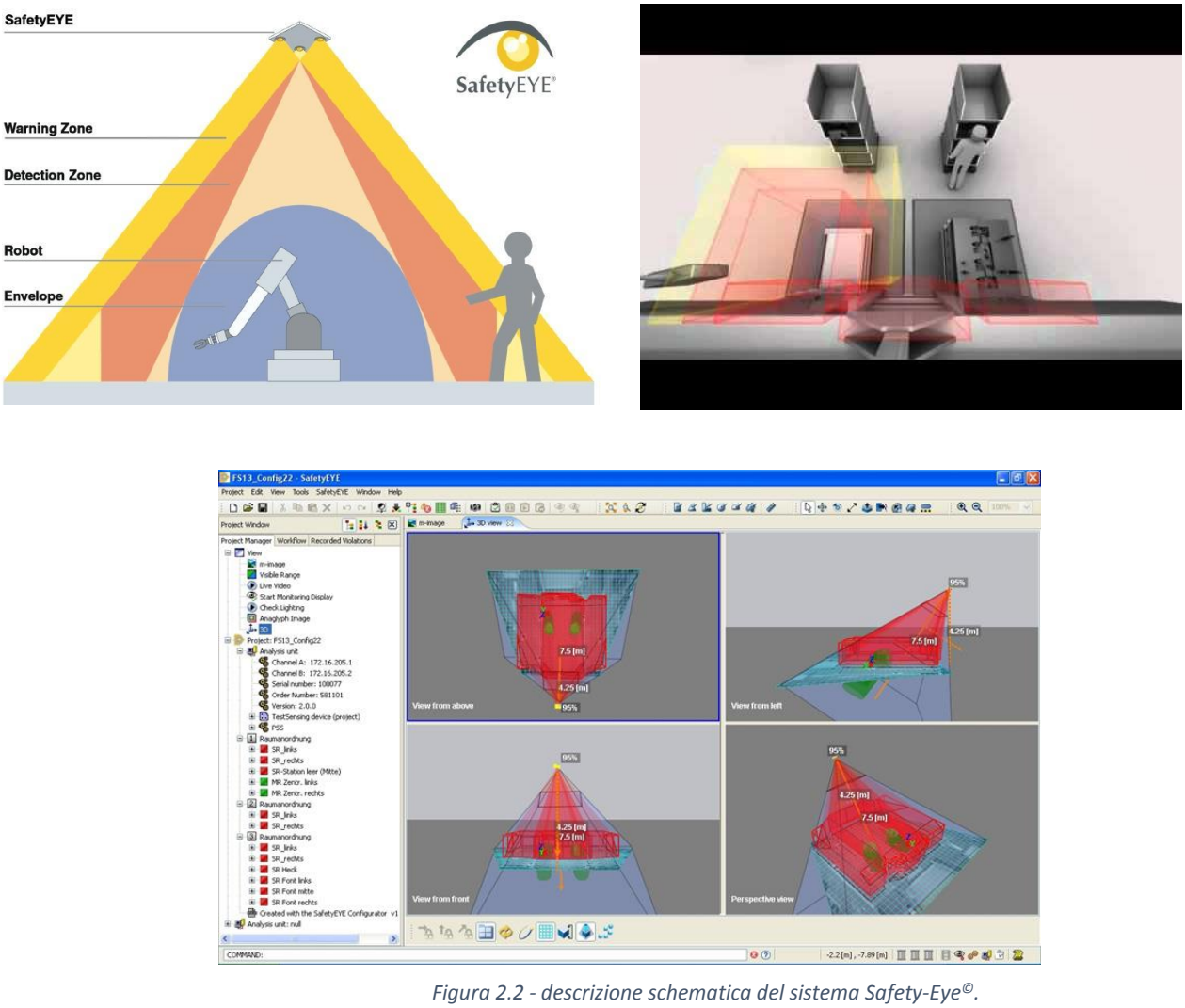


Figura 2.2 - descrizione schematica del sistema Safety-Eye©.

2. SafeMove© prodotto da ABB[14].

   Anche in questa soluzione è prevista una fase in cui si caricano nel controllore del robot dei volumi di lavoro definiti sicuri, dei volumi di lavoro in cui i robot devono rallentare e dei volumi in cui devono fermarsi (ad es. individuati rispettivamente dalla zona verde, gialla e rossa di Fig. 2.3).

   La definizione dei volumi è statica anche in questa soluzione, una volta caricati nel controllore del robot non vi è la possibilità di modificarli in tempo reale. Il vantaggio e l'innovazione introdotta da questa soluzione è dovuta al fatto che viene considerata anche la velocità con cui l'uomo si approccia al robot.

Figure 2.3 – descrizione schematica del sistema ABB SafeMove©.

La maggior parte dei produttori di robot offre tale modalità di definizione dei limiti nello spazio operativo/di giunto del robot in modalità sicura (prerequisito da ISO 10218-1:2011). Esistono poi diverse soluzioni per l'identificazione della violazione degli spazi statici *safeguarded* del volume di lavoro, basati su sensori a soglia. Generalmente, infatti, la sensoristica utilizzata per questo tipo di soluzioni è basata su dispositivi ottici (dalle barriere ottiche alle "*virtual curtain*"[15] da telecamere) oppure dei sistemi di laser scanning (e.g. SICK)[16].

Nel caso di tali sistemi, i sensori sono generalmente di classe sicura (i.e. PLd o PLe), quindi direttamente interfacciabili con un sistema di controllo di pari livello di sicurezza funzionale, ma il segnale provvisto è di natura binaria sulla violazione o meno di soglie predeterminate. Tali sensori pertanto non sono adeguati al monitoraggio continuo della posizione dell'operatore, bensì soltanto al rilevamento di una eventuale violazione di uno spazio predefinito. Nel caso di telecamere, non esistono ad oggi soluzioni di inseguimento ottico (*optical tracking*) che producano dati *output safe* utilizzabili direttamente nel ciclo di controllo.

Le soluzioni di controllo sicuro del moto si basano pertanto su sorgenti non sicure (i.e. di classe non compatibile con i requisiti di sicurezza funzionale atti a garantire l'identificazione di

possibili guasti). Una gamma di sorgenti per l'inseguimento dell'operatore si qualificherebbe invece come potenzialmente assai utile se capace di identificare guasti o segnali corrotti.

2.3 Prospettive future delle modalità cooperative

Le soluzioni illustrate nel paragrafo precedente sono principalmente basate sulla distanza tra operatore e robot. Questa scelta è dettata dal fatto che, in accordo alle norme vigenti, devono essere monitorate le distanze relative uomo-robot.

Nelle definizione della specifica tecnica ISO/TS 15066 [12], candidata a diventare la normativa di riferimento, in quanto fornisce le specifiche per le condizioni definite in ISO 10218-2[11] le modalità cooperative sono:

1. *Safety rated stop* (*halted motion*);
2. *Hand-guided*;
3. *Speed and Separation Monitoring* (SSM);
4. *Power and Force Limiting* (PFL).

Nell'approccio SSM vengono monitorate la velocità e la distanza tra un ostacolo (l'operatore) e il robot, in modo da prevenire qualsiasi impatto. Nel caso PFL vengono invece osservate la Forza e la Potenza del robot per assicurare che siano minimizzate le lesioni all'operatore sulla base di previsioni di collisione tra uomo e robot. La differenza tra le due è fondamentale: nel caso PFL si vogliono limitare i danni **una volta avvenuta la collisione**, mentre nel caso SSM si vuole **prevenire** che si verifichi l'impatto sulla base della distanza e della velocità relativa tra uomo e robot.

La distanza minima di sicurezza che viene calcolata in questo lavoro di tesi, compatibilmente con la norma ISO/TS 15066, è calcolata tenendo conto della **direzione della velocità, oltre che della distanza e della velocità relativa** tra ostacolo e robot, **considerando l'intero corpo del robot.**

Il calcolo della distanza minima nella proposta di aggiornamento della ISO/TS 15066 [12], dove vengono considerate le caratteristiche dinamiche del robot, è data da

$$d_{min} = \int_{t}^{t+Tr} v_r d\tau + \int_{t+T_R}^{t+T_R+T_B} v_r d\tau + \int_{t}^{t+T_R+T_B} v_H d\tau + Z_R + Z_H$$

Dove:

v_H è la velocità di intrusione dell'uomo.

v_R è la velocità del robot e Z_R è la distanza euclidea che rende il robot sicuro.

Il termine Z_H indica la percentuale di intrusione dell'operatore tollerata.

T_R è il tempo di reazione del SRP/CS al rilevamento delle condizioni di rischio

T_B è il tempo di fermata del robot.

I termini integrali sui tempi di frenata rappresentano lo spazio cumulato di frenata del robot, ad una certa velocità e in una determinata configurazione.

Nel contesto di tale definizione (non ancora ufficiale al momento di scrittura della presente tesi) di distanza dipendente dalla traiettoria, si inserisce il contributo di definizione specifica di un algoritmo di sicurezza e della estensione del principio espresso in (1) a casi reali e all'intero corpo del robot.

$$d_{min} = k_H(t_1 + t_2) + k_R t_1 + B + \delta \quad (1)$$

Capitolo 3
Architettura SafeNet

3.1 Introduzione all'architettura SafeNet

Con l'obiettivo di creare uno **spazio di lavoro condiviso in cui uomo e robot possano interagire** il primo passo è quello di creare una **rete ad-hoc** che permetta di connettere più dispositivi di natura diversa, mantenendo il livello di monitoraggio sui protocolli di comunicazione delle linee dedicate alla sicurezza.
Il sistema può quindi essere schematizzato con un Network Control System (NCS), dove tutti i dispositivi vengono collegati mediante una rete di comunicazione.
Questo tipo di rappresentazione è molto comoda perché permette di astrarre il modello del sistema senza vincolarsi al caso specifico, mantenendo la generalità sul numero di sensori, robot e altre apparecchiature.

Per le caratteristiche che la rete dovrà avere si utilizza una architettura di rete a stella (*star network*) [17].

In una architettura *SafeNet* il centro stella incorpora anche le funzionalità del SRP/CS dedicate al monitoraggio (*logic* – L) delle informazioni (input – I) di sicurezza e delle contromisure (output – O). Tale funzionalità è distribuita su più nodi, rendendo ridondanti le funzioni di comunicazione e calcolo del centro stella ed associando un hardware dedicato certificato alle funzioni di sicurezza. Il centro stella diventa pertanto virtualizzato su almeno 3 nodi: 2 *check node* (CN) e un *safe* CPU (Vedi *safe logic* in Fig. 3.1).

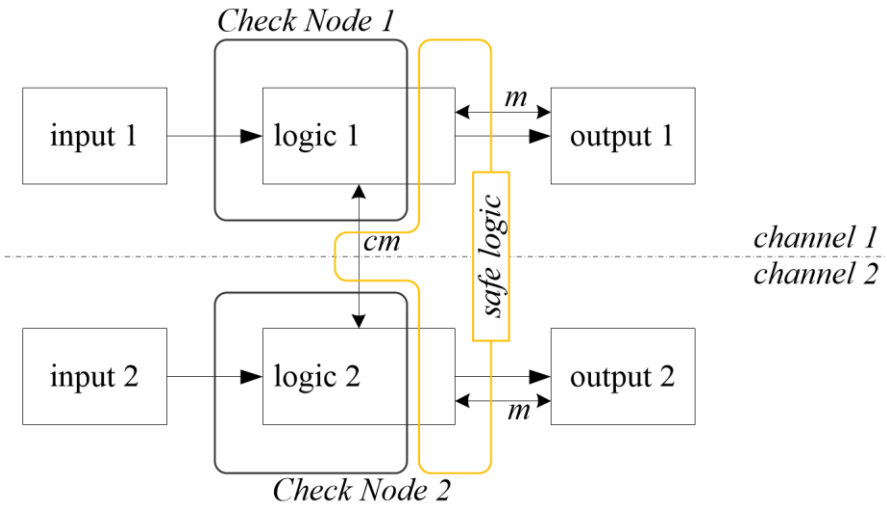


Figura 3.1 – Struttura del nodo posto nel centro-stella.

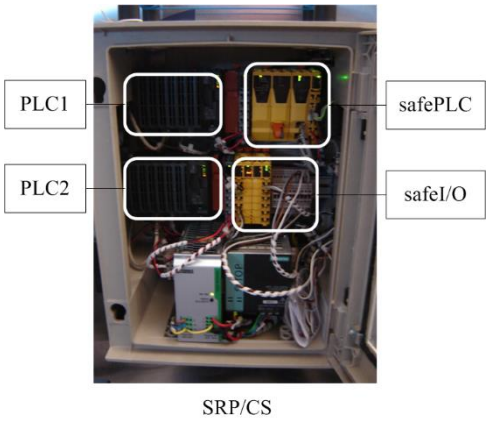


Figura 3.2 – Struttura fisica dell'SRP/CS in un esempio applicativo.

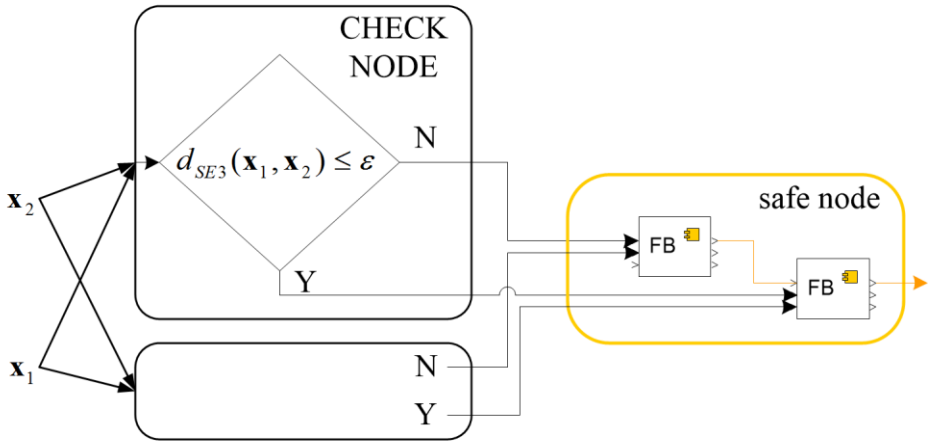


Figura 3.3 – Esempio di logica applicata ad un generico Check Node con due input.

Un esempio di architettura *SafeNet* completo può essere quindi rappresentato dalla Fig. 3.4, in cui si vede la presenza di un numero generico di sensori, un numero generico di robot e il nodo a centro stella composto da due unità ridondate e da una unità *safe* che controlla la consistenza dei dati e applica l'algoritmo.

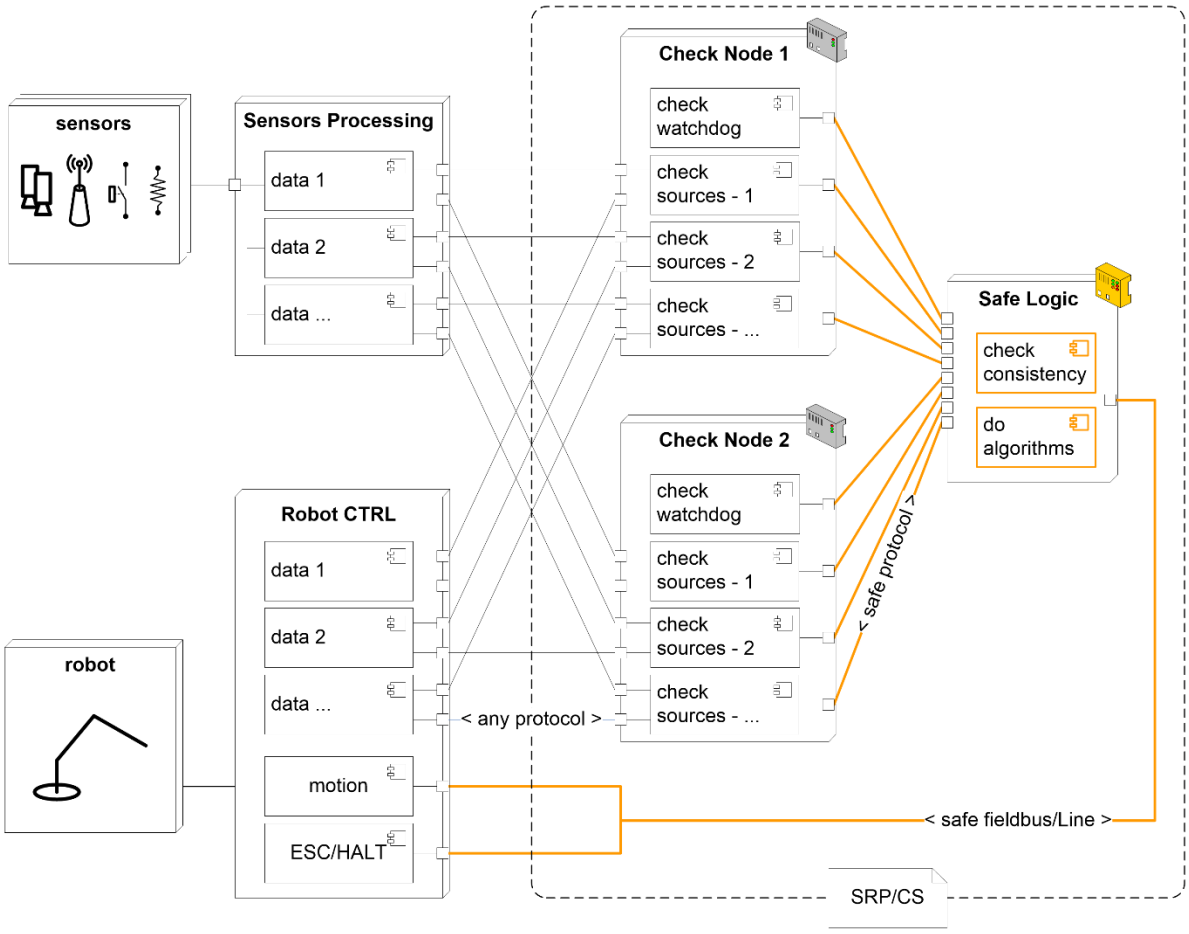


Figura 3.4 – Esempio di architettura SafeNet generalizzata.

La rete a cui si farà riferimento nella trattazione del lavoro seguente sarà una rete di questa tipologia ma con 3 nodi, un sensore (*Sensor Node* SN), un robot (*Robot Node* RN) e un nodo centrale su cui saranno effettuate le operazioni di calcolo e il controllo di consistenza (*Check Node* CN). (vedi Fig. 3.5).

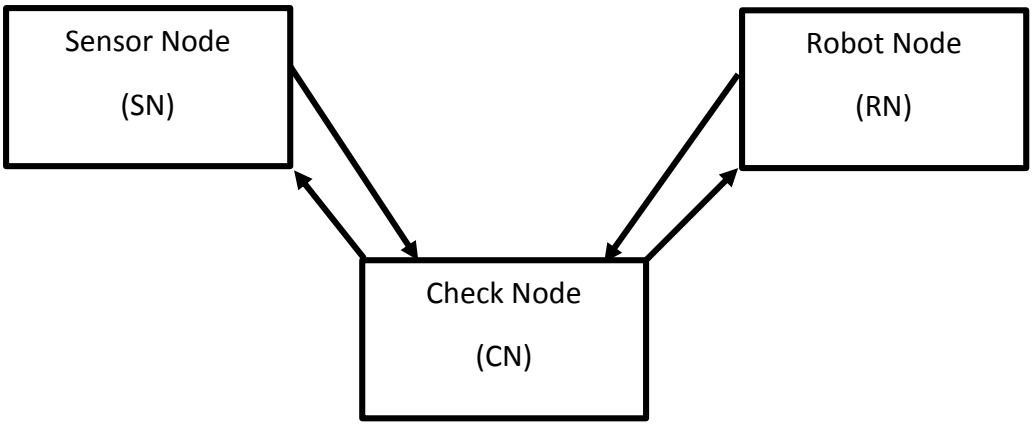


Figura 3.5 – Architettura SafeNet di riferimento per le analisi successive

I nodi remoti sono isolati gli uni dagli altri, in questo modo non si hanno influenze sulle varie misure e si riesce ad avere misure indipendenti nel caso in cui più nodi/sensori dello stesso fenomeno misurato siano collegati al sistema.

Questa architettura di rete permette una adeguata scalabilità nei sistemi robotizzati in quanto il numero di apparecchi da collegare è generalmente ridotto e le performance di rete/calcolo del centro stella (*hub*) sono dimensionate adeguatamente. Inoltre non viene fatta nessuna discriminazione sulla tipologia dei dispositivi che sono connessi.
Il vantaggio è appunto quello di poter sincronizzare al centro stella le misure derivate da fenomeni fisici differenti per la stessa grandezza. Ad esempio, per la posizione, usare sia sensori ottici che sensori radio. Tale accorgimento consente inoltre di gestire ridondanza di informazione al nodo deputato alle operazioni critiche per la verifica delle condizioni di sicurezza.

Il prerequisito è un'interfaccia standardizzata tra i dispositivi remoti e il centro stella. La rete ad-hoc, proposta in questo lavoro di tesi, utilizza il protocollo UDP/IP [7], sopra il quale sono stati introdotti accorgimenti nello strato di applicazione (*application layer*) allo scopo di effettuare verifiche sui messaggi. Infatti, i sistemi di identificazione (ad es. basati su codice di verifica o *checksum*) del protocollo a livello della costruzione del pacchetto dati elementare *(Datagram)* non sono sufficienti a garantire la verifica puntuale degli errori presenti del corpo del messaggio (*payload*).

Come si vede in Fig. 3.5, il canale di comunicazione tra un generico nodo (SN o RN) e il *Safety Controller(CN)* è bidirezionale: non solo il generico nodo invia informazioni al Centro Stella ma anche quest'ultimo invia informazioni di ritorno (*feedback*) ai vari nodi.

I dispositivi utilizzati per interfacciare i sensori fisici e i Robot con la rete Ethernet sono dei computer basati su architettura Unix, capaci di convertire il flusso dati, proveniente dal dispositivo fisico (sensore o robot), in un flusso dati per il protocollo di rete desiderato. Lo

schema in Fig. 3.6 riporta un esempio di architettura congrua con le specifiche introdotte, che è stato utilizzato per il sistema prototipale.
Nel capitolo 5 di questa tesi, verrà mostrato un esempio simile utilizzante sensori radio e robot Comau.
Anche in tal caso, l'architettura di rete utilizzata sarà la medesima mentre cambieranno solo i dispositivi fisici.

Questo avvalora ulteriormente la scelta di questo tipo di **approccio basato sulla rete**, rendendo evidente il vantaggio di avere una infrastruttura generica che viene implementata differentemente, una volta definiti i nodi fisici che si utilizzeranno.

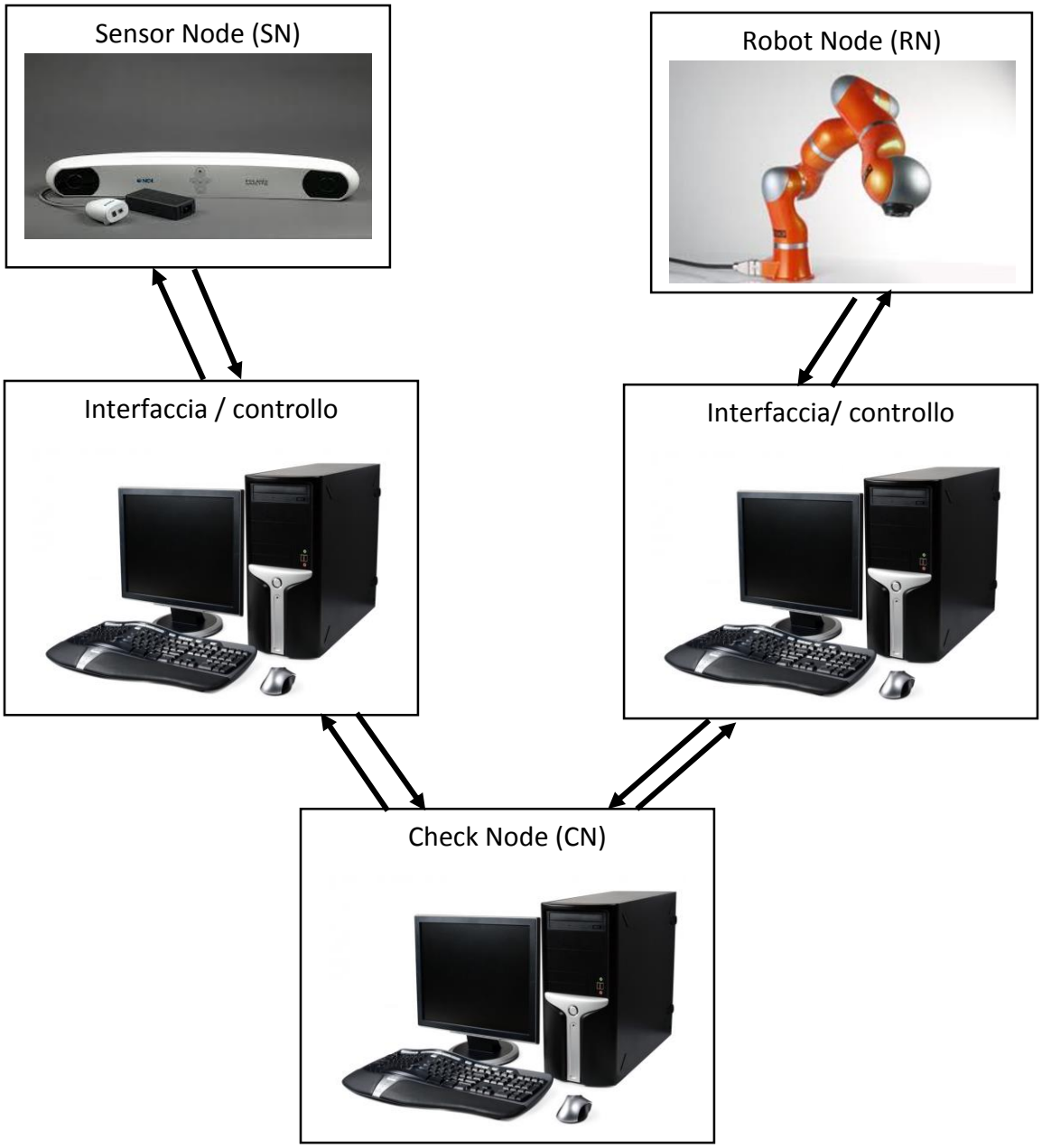


Figura 3.6 – Esempio di architettura NCS nel caso prototipale.

3.2 Utilizzo del black channel

Per raggiungere gli obiettivi preposti, ha un ruolo fondamentale la scelta di un protocollo di rete (*protocol stack*) che permetta di implementare dei protocolli di sicurezza a livello utente (*application layer*).

Data la rilevanza delle prestazioni di protocollo ai fini della sicurezza (e.g. tempi di risposta, tempi di latenza, etc), è determinante procedere alla caratterizzazione delle quantità fondamentali dal punto di vista funzionale associabili al protocollo scelto.

Per mantenere la generalità della soluzione studiata, si è scelto un approccio del tipo *black channel,* ovvero l'utilizzo di un qualsiasi protocollo come supporto fisico sul quale le funzionalità di sicurezza sono introdotte sullo strato di applicazione (livello 7 dello stack ISO/OSI di Fig. 3.8) [18].

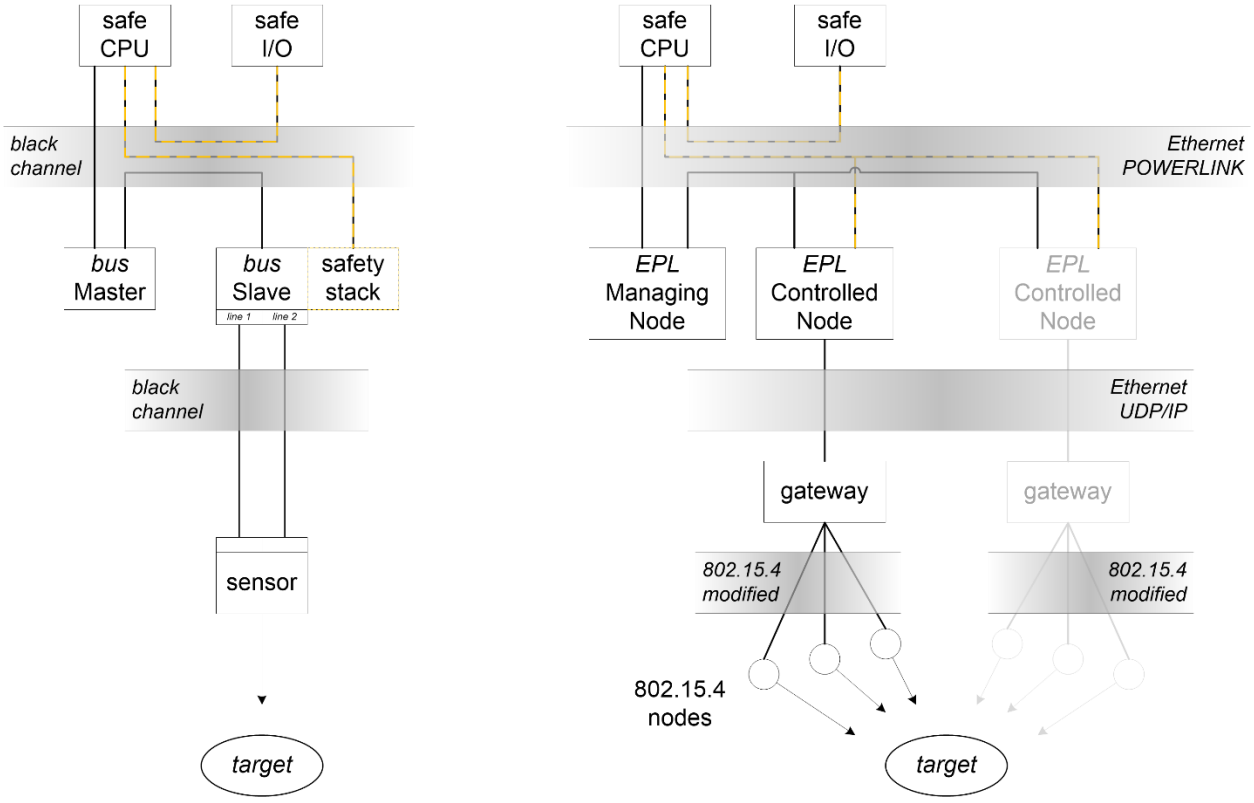


Figura 3.7 – Funzionalità black channel con supporto fisico corrispondente.

Nella parte destra di Fig. 3.7 le funzionalità di sicurezza al livello applicazione sono collegate tra di loro a prescindere dal supporto fisico sottostante. Nell'istanza specifica di Fig. 3.7 a destra, alcuni supporti fisici al black channel sono esplicitati nell'architettura in esame.

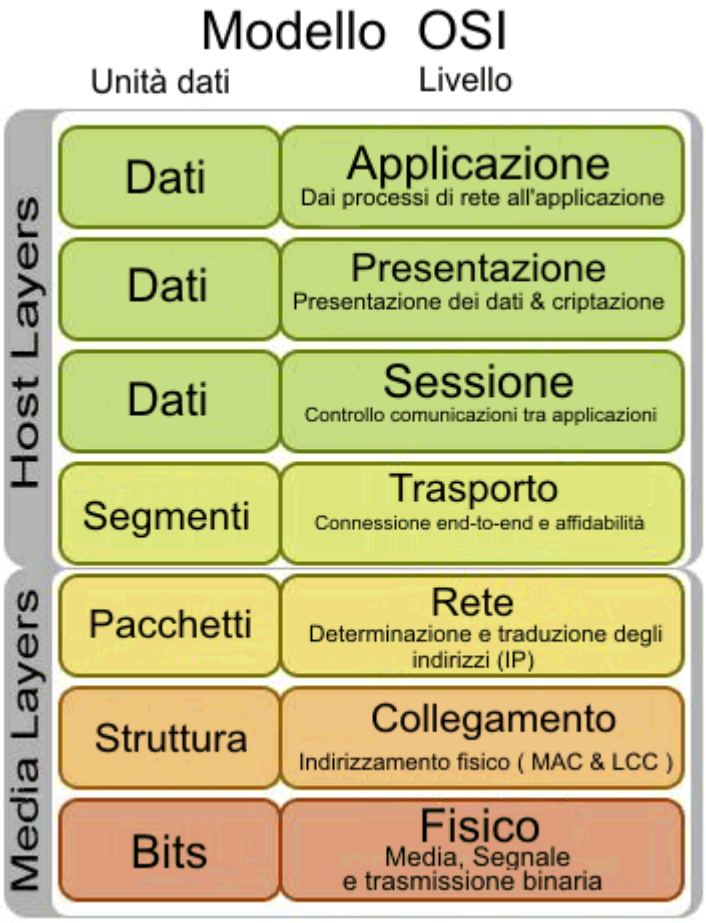


Figura 3.8 – Pila protocollare secondo le definizioni ISO/OSI.

L'approccio *black channel* permette di introdurre un protocollo di sicurezza sopra un protocollo già esistenti ed è molto comodo perché permette la trasparenza rispetto ai terminali collegati i quali utilizzano uno standard conosciuto, senza dover introdurre interfacce dedicate, sia hardware che software, per la comunicazione.

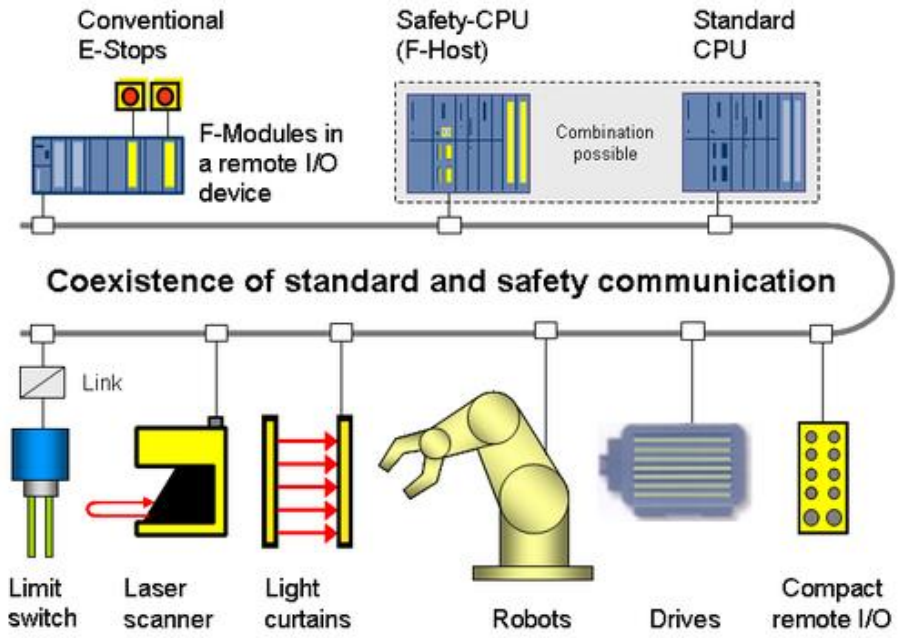


Figura 3.9 – Esempio di architettura industriale.

Con questo approccio, come si vede in Fig. 3.9 è possibile includere altri elementi che ormai fanno parte degli standard industriali e integrarli facilmente. Un aspetto utile ed interessante dal punto di vista della flessibilità delle applicazioni industriali è il fatto che le comunicazioni dei singoli canali possono essere di tipo eterogeneo.

IL medesimo approccio è, ad esempio, perseguito in automazione industriale dal *protocol stack* opensource OpenSafety [19] rilasciato come standard aperto di codifica delle funzioni sul layer applicazione, integrabile sopra qualsiasi bus industriale (*fieldbus*). Tale protocollo è disponibile secondo l'approccio *Open Source*. Qualora l'utente disponesse anche dello stack di trasporto (e.g. lo stack POWERLINK da compilare su una propria piattaforma target), avrebbe la possibilità di creare un nodo personalizzato direttamente interfacciabile con i sistemi SRP/CS di sicurezza (ad es. un SafePLC standard commerciale).

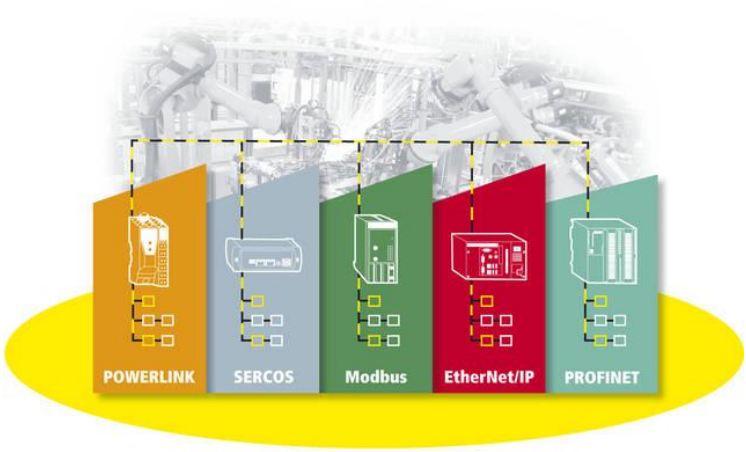


Figura 3.10 – OpenSafety su black channel.

3.3 Strutture dati utilizzate

Il generico pacchetto dati UDP [7] ha la struttura descritta in Fig. 3.11:

+	Bit 0-15	16-31
0	Source Port (optional)	Destination Port
32	Length	Checksum (optional)
64+	Data	

Figura 3.11 – Datagramma UDP

Si noti che c'è anche la possibilità di riempire un campo *checksum* per il controllo di integrità dei dati. Questo però non basta per avere buone proprietà di robustezza e sicurezza necessari da quanto stabilito dalla norma ISO 13489.

Per rendere più sicuri i dati trasmessi tra i vari nodi sono state usate ulteriori tecniche per il controllo d'integrità e per la crittografia dei dati trasmessi; in tal modo è possibile aumentare la protezione dei dati in caso di eventuali attacchi esterni. Per realizzare questo doppio controllo sono utilizzate delle librerie preesistenti di crittografia e di controllo d'integrità.

Nel dettaglio, sono state implementate in linguaggio C++ le seguenti funzioni:

1. HMAC (**keyed-hash message authentication code**). E' un tipo di codice per l'autenticazione dei messaggi basata su una funzione di hash. Tramite HMAC è infatti possibile garantire sia l'integrità che l'autenticità di un messaggio. HMAC utilizza infatti una combinazione del messaggio originale e di una chiave segreta per la generazione del codice. Una caratteristica peculiare di HMAC è quella di non essere legato a nessuna *hash function* (si potrebbe definire come la funzione che permette di ottenere la firma o impronta specifica del messaggio) in particolare, questo per rendere possibile una sostituzione della funzione nel caso non fosse abbastanza sicura.
2. La hash function utilizzata in questa implementazione dell'HMAC è la funzione SHA1 [20] (Secure Hash Algorithm) che è considerata una tra le funzioni più sicure per questo tipo di applicazioni. La funzione SHA1 produce un *message digest*, o impronta del messaggio, di lunghezza fissa, partendo da un messaggio di lunghezza variabile. La sicurezza di un algoritmo hash risiede nel fatto che la funzione non è reversibile (non è possibile cioè risalire al messaggio originale conoscendo solo questo dato) e che non

deve mai essere possibile creare intenzionalmente due messaggi diversi con lo stesso *message digest*. L'algoritmo SHA1 produce in uscita un *message digest* di lunghezza pari a 160bit.

La parte dati del pacchetto UDP sarà costruito in modo tale da avere 2 aree dati: la prima area dati è dedicata al *digest* del pacchetto, che verrà scritto da chi invia e verrà decifrato da chi riceve, nota la chiave di cifratura. La seconda area è chiamata invece *payload* e contiene l'informazione utile inviata tramite il pacchetto.I tipi di pacchetto, o più precisamente i payload specifici caratterizzanti i pacchetti, sono definiti in base all'interfaccia xx-CN: ad esempio per il RN si istanzia un messaggio di classe *robot packet.* Analogamente*, sensor packet e status packet* si utilizzano per messaggi dal sensore al *CN* e dal *CN* a *RN/SN*.

Nel *sensor packet* sono presenti tre campi in cui verranno inserite le pose degli oggetti tracciati mediante sensore. Le pose sono espresse come componenti cartesiane e quaternione.

Il *robot packet* contiene invece due campi posa in cui vengono inserite le pose del gomito e dell'*end effector* del robot. Vengono inserite anche le velocità cartesiane del gomito e dell'*end effector.*

Il terzo tipo di pacchetto, detto *status packet,* è il pacchetto che viene utilizzato come messaggio di risposta (*feedback*) dal *CN* verso i nodi. Contiene un unico campo che rappresenta lo stato (*status*) del sistema. Nell'implementazione delle distanze minime di sicurezza verrà sfruttato questo campo per diffondere in tutto il sistema l'informazione ternaria relativa alla sicurezza.

In tutti i pacchetti inoltre verranno inseriti dei campi come il numero di pacchetto, il nodo che ha generato il dato, e fondamentale l'istante temporale (*timestamp*) in cui è stato generato il pacchetto.

Dato che i pacchetti vengono generati su sistemi diversi, l'asse temporale dei diversi sistemi non farà riferimento ad una stessa base; per tale motivo è necessario effettuare la sincronizzazione tra i riferimenti temporali (*clock*) dei vari sistemi presenti, come indicato nel seguito nel paragrafo 3.4.

In Fig. 3.8 è schematizzata la definizione dei pacchetti a partire dal generico pacchetto UDP modificato con i campi di *hashing*.

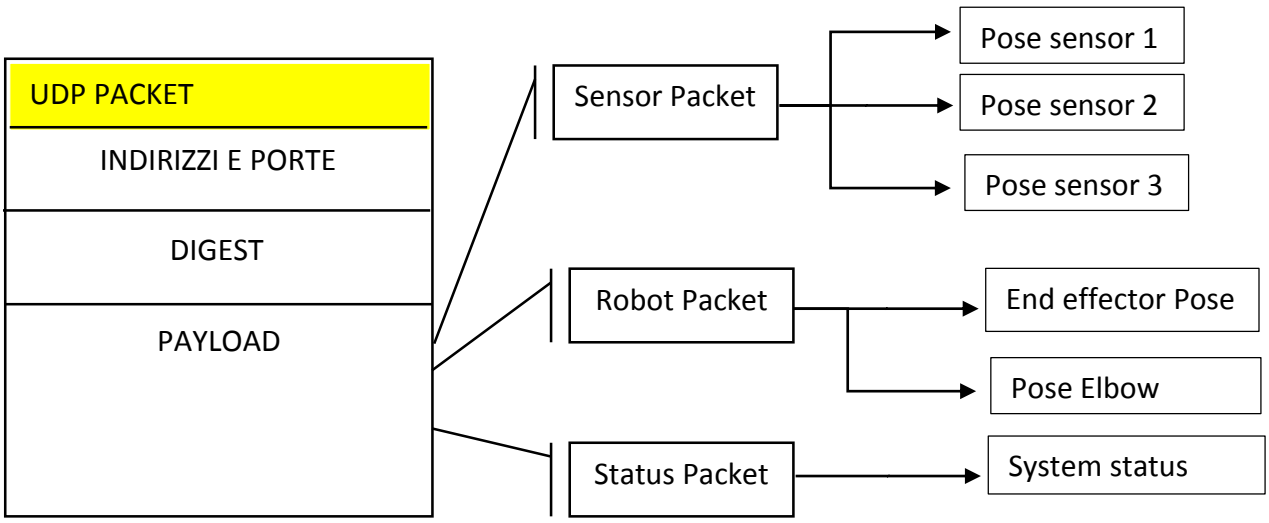


Figure 3.12 – pacchetto UDP con payload specifici

Tali funzionalità garantiscono un utilizzo *fail safe* di un protocollo nativamente non sicuro come l'UDP/IP ai fini della sicurezza funzionale (ISO 13489).

In particolare con il monitoraggio del numero di pacchetto con un contatore, si è anche in grado di identificare errori del tipo pacchetto perso, pacchetto ripetuto. Infatti, il nodo ricevente resterà in attesa di un pacchetto per un periodo di tempo predefinito (*Timeout*) scaduto il quale sarà possibile identificare un pacchetto in ritardo o un pacchetto cancellato.

measures / type of error	message counter	timestamp	timeout	echo	send/recv identifier	check of data consistency
repeated message	✓					
deleted message	✓		✓	!		
insertion	✓			!	✓	
resequenced message	✓					
corrupted message						✓
delayed message		!	✓			

Figura 3.13– Contromisure applicate per l'identificazione di errori.

Nel pacchetto inoltre è inserito un identificatore per stabilire chi ha inviato il pacchetto in modo da rendere nota la fonte di un eventuale problema. Per il controllo dell'integrità e consistenza dei dati si è utilizzata invece la procedura di *hashing* precedentemente introdotta. Si veda anche la Fig. 3.13.

Per avere inoltre nel pacchetto un campo *timestamp*, è stata necessaria l'introduzione della sincronizzazione dei nodi in quanto i tempi devono essere riferiti tutti alla stessa base temporale (cioè quella del CN).

Nel paragrafo successivo verrà illustrato il metodo di sincronizzazione applicato, in modo da aggiungere le funzionalità *timestamp*, rendendo identificabile un pacchetto ricevuto in ritardo.

3.4 Sincronizzazione dei nodi

L'ulteriore passo necessario per la creazione di un'architettura adatta alle analisi temporali che si vogliono realizzare, è quello di ottenere una rete avente nodi sincronizzati rispetto ad un unico *clock*. In questo modo i dati che si ricevono e inviano saranno tutti sincroni.

La tecnica che si è utilizzata per la realizzazione della sincronizzazione è quella illustrata dal Precise Time Protocol [21], ma la sua implementazione è stata fatta localmente senza usare uno dei tanti processi automatici (*daemon*) disponibili in quanto essi sfruttano sempre una connessione internet per il collegamento con uno dei server NTP (Network Time Protocol) disponibili. Quello che si è realizzato è quindi l'implementazione locale dell'PTP.

Il vantaggio di questa soluzione è che si riesce a sincronizzare la propria rete di dispositivi sul riferimento temporale di uno dei nodi. Ovviamente, avendo un minor numero di nodi a disposizione rispetto a una rete estesa la precisione è più limitata, ma per lo scopo preposto una precisione di 100 [ns] è risultata più che sufficiente.

Data la centralità fisica e funzionale del CN il clock di riferimento è proprio quello del CN e il software di sincronizzazione sarà anch'esso implementato su questo nodo centrale.

Queste routines sono sostanzialmente tre:

- routine 1: Ogni nodo invia un pacchetto con il proprio *timestamp* e si aspetta una risposta *acknowledgment* da parte del *Check Node* (CN). Il processo software del CN che riceve i pacchetti del nodo ha un contatore interno che calcola il tempo che intercorre tra l'invio di un *acknowledgment* e la risposta del nodo (*round trip delay*) di ogni pacchetto.
  In questo modo si riesce a determinare il tempo di latenza dovuto alla rete (si veda anche la Fig. 3.17). Il pacchetto che riceve il CN avrà un campo che contiene il *timestamp* del nodo. Il CN confronterà questo tempo con il suo clock interno a cui è sottratto metà del *round trip delay*. In questo modo si ottiene la differenza temporale tra i 2 clock. Tale differenza viene memorizzata. Questa procedura di scambio pacchetti sarà ripetuta per un arco di tempo in cui il CN colleziona dati.
- Routine 2: questa routine prevede che il CN calcoli le medie e le varianze delle differenze temporali calcolate con la routine 1 e che le memorizzi.
- Routine 3: correzione in tempo reale del ritardo/anticipo del *clock* dei vari nodi in base alle stime fatte nel punto precedente.

Il pacchetto inviato dal nodo al CN e viceversa ha la stessa dimensione inoltre la configurazione della rete è fissa durante tale intervallo. Si può ipotizzare che il tempo di trasmissione in andata e in ritorno sia uguale.

Il pacchetto avrà la seguente struttura (visto rispetto al CN):

TS_SN	
TS_CN	
Number	1

Figura 3.14 – esempio di generazione di un pacchetto: inserimento del numero progressivo (Number)

In cui i campi TS_SN sono riempiti alla ricezione del pacchetto da parte del nodo che inserisce il suo valore *timestamp* espresso in secondi.

TS_SN	11452.9878
TS_CN	
Number	1

Figura 3.15 – esempio di generazione di un pacchetto: inserimento del tempo di generazione (TS_SN).

Alla ricezione del pacchetto grazie al campo *Number,* si controlla che il pacchetto ricevuto sia quello atteso, dopodiché viene riempito il campo TS_CN;

TS_SN	11452.987
TS_CN	11453.589
Number	1

Figura 3.16 – esempio di generazione di un pacchetto: inserimento del tempo di ricezione (TS_CN).

In Fig. 3.17 e 3.18 sono riprodotti in modo schematico l'andamento di questa procedura sui diversi assi dei tempi per visualizzare concretamente e distintamente tutte le varie componenti temporali che vanno a costituire il tempo di offset T_{off}.

0

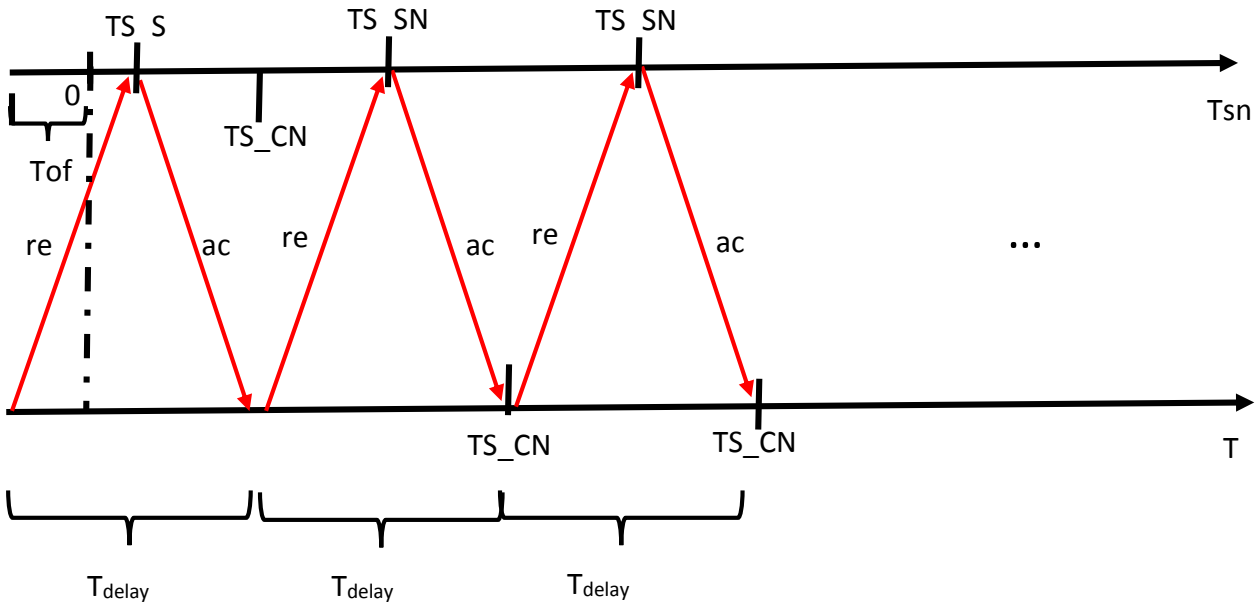


Figura 3.17 – Sfasamenti temporali

In tali figure sono indicati dove:

- T_{off} Indica il ritardo o l'anticipo (*offset*) tra i *clock* del nodo in considerazione (SN nella Fig. 3.7) e il CN;
- TS_CN Indica il *timestamp* del CN;
- TS_SN Indica il *timestamp* del SN;
- T_{delay} Indica il *round trip delay* cioè il tempo che un pacchetto impiega per essere ricevuto dall'altro nodo ed essere rimandato indietro.

Il calcolo del T_{off}, per ogni pacchetto ricevuto, avverrà con la seguente formula:

$$T_{off} \approx TS\_CN - TS\_SN + \frac{1}{2}\,(T_{delay}) \qquad (2)$$

Per migliorare la stima di questo valore si sono poi considerati anche i tempi di calcolo interni per ogni nodo in modo da togliere incertezza alla stima dei tempi. Riportando le grandezze di interesse su un grafico simile al precedente si otterrà lo schema di Fig. 3.18:

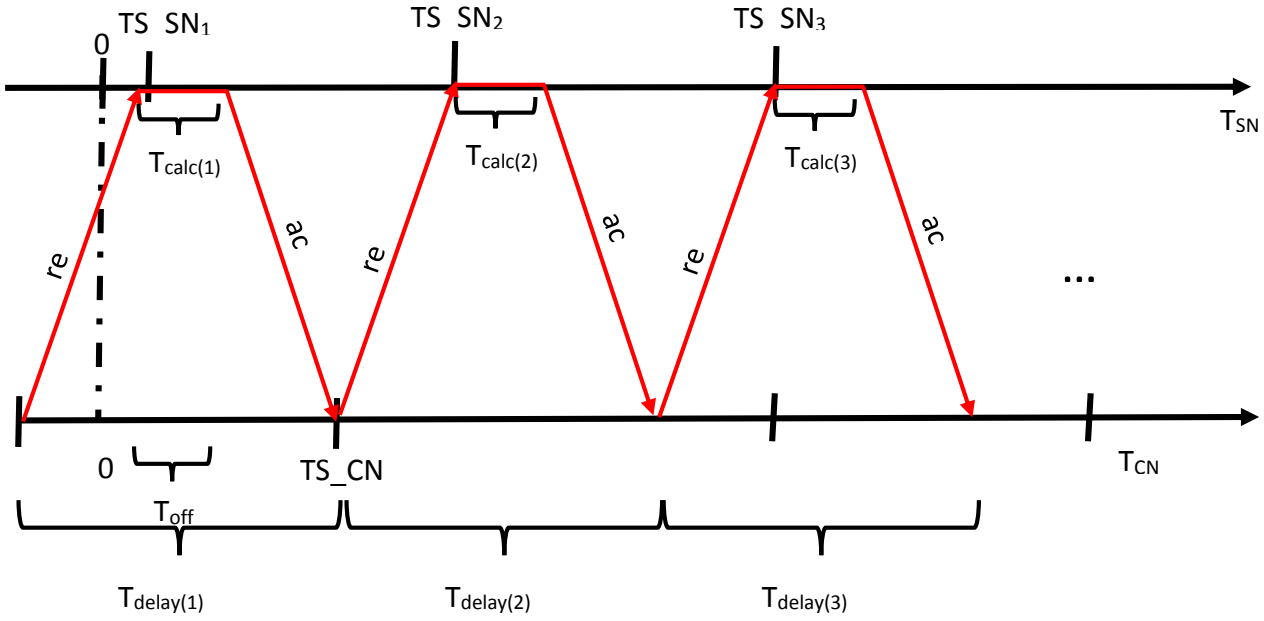


Figura 3.18 – Sfasamenti temporali con l'inclusione del tempo di calcolo

Le grandezze di Fig. 3.18 hanno lo stesso significato del grafico precedente con in aggiunta il tempo T_{calc} che rappresenta il tempo impiegato dal nodo per effettuare le operazioni di calcolo richieste per l'elaborazione del pacchetto.

La formula dell'offset per ogni pacchetto, che sarà utilizzata in questo caso, vale:

$$T_{Off}(i) = TS\_CN(i) - TS\_SN(i) - \frac{1}{2}(T_{delay(i)} - T_{calc(i)}) \qquad (3)$$

La stima di T_{off} è effettuata su *N>>1* pacchetti elaborati durante 15 minuti di traffico:

$$T_{off} = \frac{1}{N} \sum_i T_{off}(i) \qquad (4)$$

Il stima T_{off} così calcolata viene salvata in un file di configurazione esterno che viene poi ricaricato ad ogni avvio del programma.

In questa soluzione di sincronizzazione di rete non si è tenuto in conto il fenomeno di deriva del *clock*, ossia il fatto che due *clock* diversi hanno una tendenza ad accelerare o rallentare il loro ciclo facendo così aumentare e/o diminuire lo sfasamento temporale che esiste tra più nodi.

Per semplicità di modellizzazione non è stata fatta una modifica all'algoritmo usato precedentemente per tener conto di tale effetto di allungamento/accorciamento dell'asse temporale.

Infine, si noti come, essendo il processo di sincronizzazione poco costoso in termini computazionali ed il traffico di rete generato dai pacchetti di sincronizzazione è molto scarso, la procedura è ininfluente sulle prestazioni complessive del sistema.

Con tale algoritmo è possibile, ogni 15 minuti, tenere sotto controllo il lo sfasamento dei *clock.* Si è verificato sperimentalmente che, in 15 minuti, la variazione è molto piccola (200 ns) a meno di guasti su uno dei nodi. Per i dettagli si veda il paragrafo 3.5.

Infine, è stato predisposto un meccanismo di identificazione dei guasti nel caso in cui la differenza dei *clock* ricalcolata sia troppo distante da quella calcolata in partenza.

3.5 Affidabilità canali comunicazione e probabilità di errore

Al fine di caratterizzare correttamente il tempo di latenza di rete nelle varie comunicazioni tra i nodi è stata effettuata una analisi considerando il tempo come una variabile casuale.
E' stata quindi realizzata una campagna di acquisizione dati in cui la rete è stata monitorata per un lungo periodo, prendendo i valori di latenza di tutti i pacchetti in transito. I nodi che sono stati coinvolti sono tre, due nodi real-time e un nodo non real-time, con quindi la presenza di due canali bidirezionali monitorati. Una volta finita l'acquisizione è stata calcolata la latenza media di ogni singolo canale e il relativo *jitter* temporale.
Si ipotizza che l'andamento dei tempi di latenza abbia una funzione di densità di probabilità normale e quindi sia caratterizzabile attraverso media e varianza. In questo modo, si può definire un intervallo di confidenza entro il quale il valore della latenza sarà contenuto con un livello di significatività α.

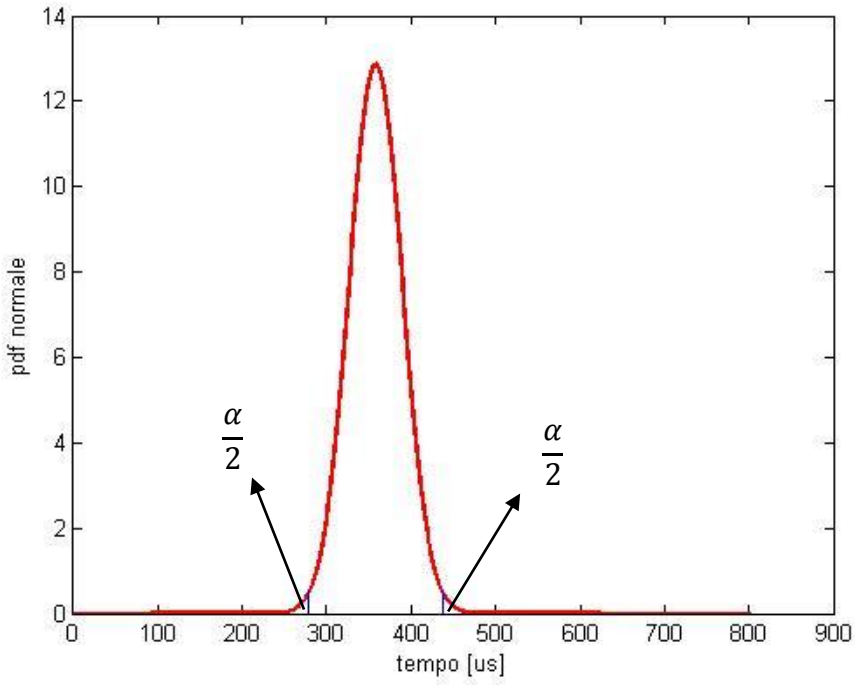


Figura 3.18 – Distribuzione dei tempi di latenza

Il test effettuato è servito quindi a determinare l'ipotesi di Gaussianità, i valori di media e varianza di questa distribuzione. Il campione utilizzato è il seguente:

Canale 1: SN ←→CN.

Numero di pacchetti inviati:	9625680
Numero di pacchetti ricevuti:	9625678
Numero pacchetti persi:	2
Latenza media:	0.357 [ms]
Jitter:	0.031 [ms]
Traffico generato:	2579682240 [Byte]
Alpha:	1%

Canale 2: RN ←→ CN.

Numero di pacchetti inviati:	64171200
Numero di pacchetti ricevuti:	64171199
Numero pacchetti persi:	1
Latenza media:	0.725 [ms]
Jitter:	0.089 [ms]
Traffico generato:	15914457600 [Byte]
Alpha:	1%

Sulla base di questi dati è quindi possibile verificare le ipotesi fatte per le due distribuzioni normali e stimare il valore di significatività.

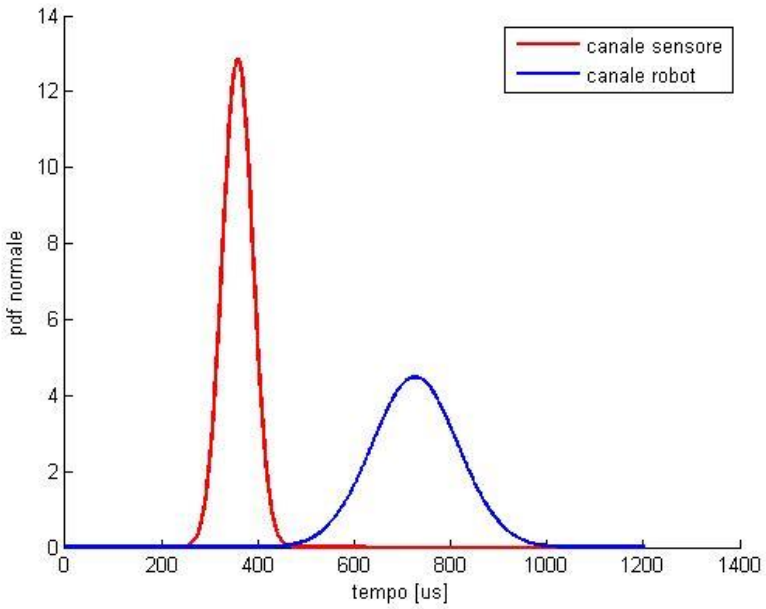


Figura 3.19 – Distribuzione delle latenze nei due canali di comunicazione.

Un altro dato che è stato monitorato nella campagna d'acquisizione è in numero di pacchetti persi per definire la probabilità di errore della rete. Il Packet Error Rate (PER) viene definito come il rapporto tra i pacchetti persi N_e (cioè ricevuti con errore e quindi scartati) e il numero totale di pacchetti N_t inviati:

$$PER = \frac{N_e}{N_t} \tag{5}$$

Si ottengono valori molto bassi, minori di 10^{-7}. Il numero di pacchetti errati è molto basso ed è necessario trasmettere un numero elevatissimo di pacchetti (ovvero sessioni di lavoro molto lunghe) per evidenziare degli errori. Come esempio, per una sessione di lavoro di 200 minuti, si ottiene, per il primo canale:

$$PER = \frac{2}{9625678} = 2.0778 \times 10^{-7}, \tag{6}$$

mentre per il secondo canale si ha invece:

$$PER = \frac{1}{64171200} = 1.558 \times 10^{-8}. \tag{7}$$

Capitolo 4
Spazi minimi di sicurezza

In questo capitolo verrà illustrato l'algoritmo con cui verranno calcolati gli spazi minimi di sicurezza da mantenere.

Verrà illustrata la logica della soluzione e spiegati gli strumenti teorici utilizzati per raggiungere l'obiettivo.

In questo capitolo si farà riferimento all'architettura *SafeNet* che è stata descritta nei capitoli precedenti. Inoltre verrà considerata sempre attiva la procedura di sincronizzazione dei *clock* dei nodi della rete descritta nel capitolo precedente.

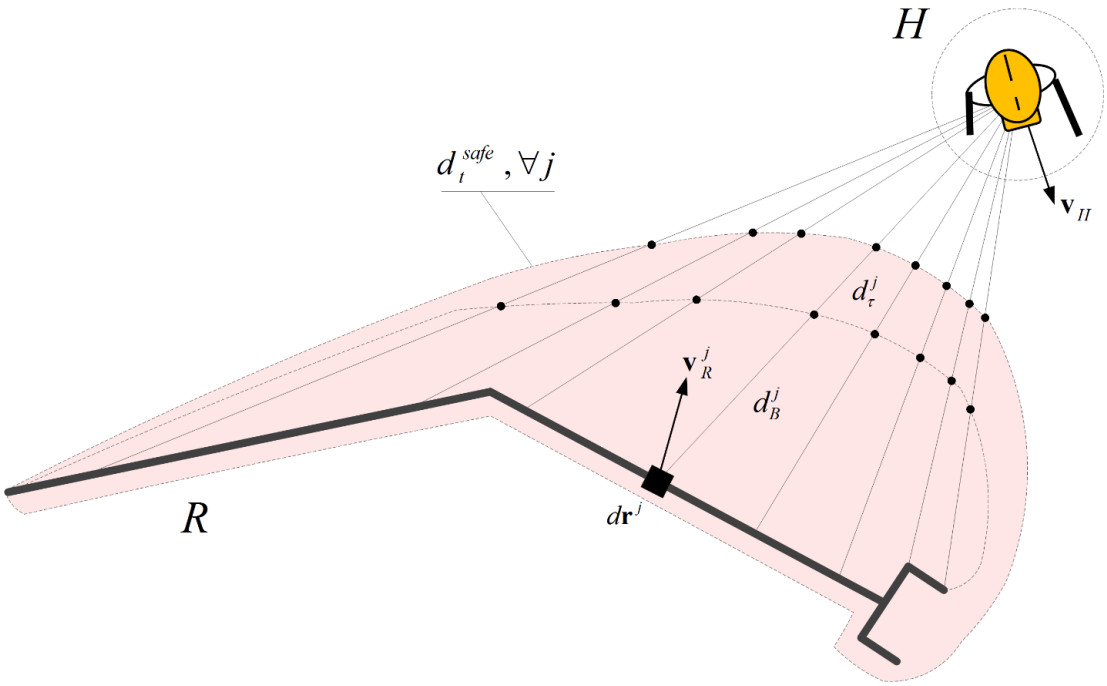


Figura 4.1 – Rappresentazione grafica della distanza di sicurezza composta dai due contributi d_b e d_t.

4.1 Algoritmo

L'algoritmo utilizzato per eseguire il calcolo degli spazi minimi di sicurezza può essere rappresentato con lo schema di flusso di Fig. 4.2.

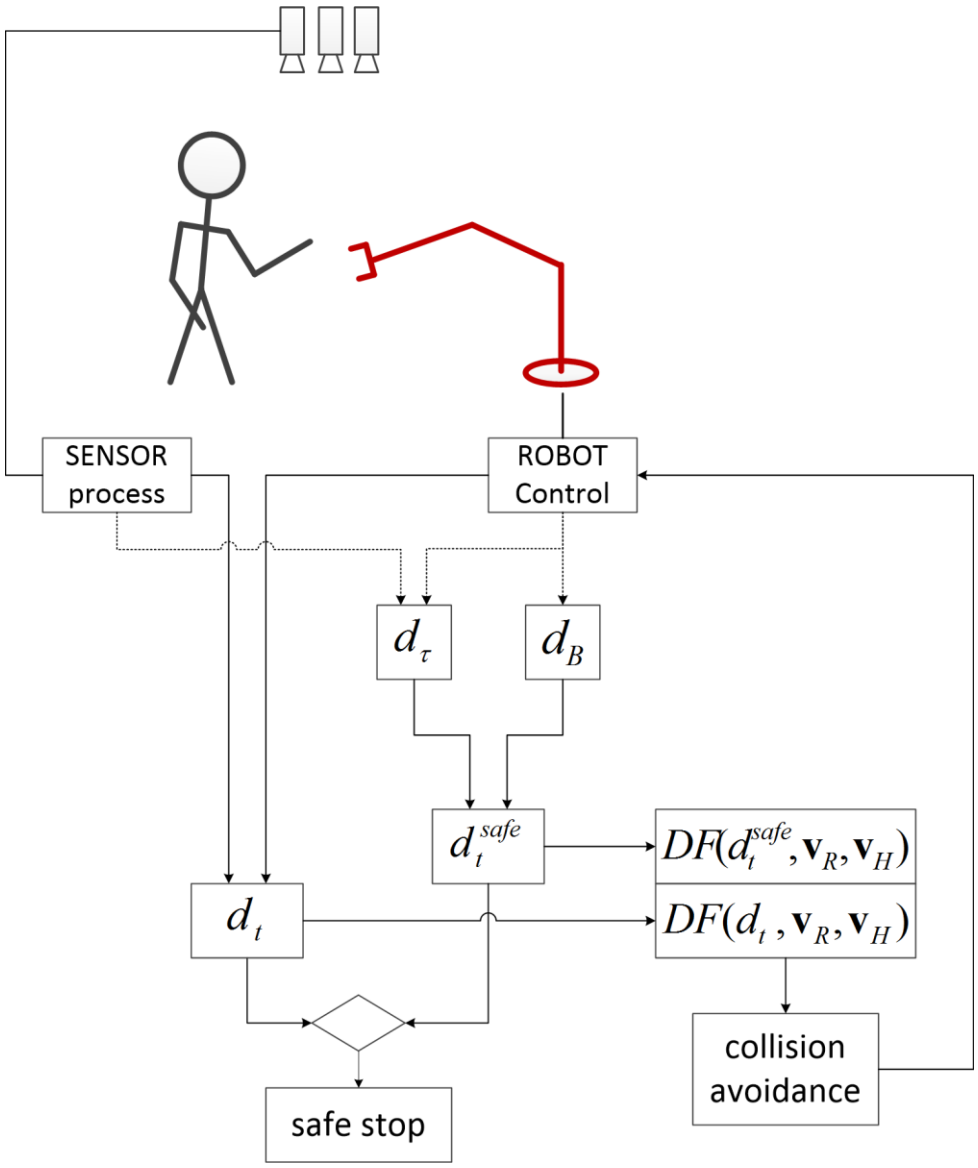


Figura 4.2 – Flow chart dell'algoritmo di calcolo degli spazi minimi di sicurezza

In cui:

- d_τ è l'incertezza spaziale dovuta all'incertezza temporale τ;
- d_B è l'incertezza spaziale legata allo spazio di frenata del robot;
- d_t è la distanza a cui si trovano uomo e robot;
- d_t^{safe} è la distanza minima di sicurezza a cui devono trovarsi uomo e robot affinché il sistema non generi un *safe* stop.
- $DF\left(d_t^{safe}, V_R, V_H\right)$ è il livello di pericolo (*danger field* o DF) associato alle condizioni limite.
- $DF(d_t, V_R, V_H)$ è il livello di pericolo associato alle condizioni misurate.

Il pedice *t* delle variabili precedenti indica che tali variabili dipendono dall'istante temporale *t* considerato. Nei prossimi paragrafi saranno descritti in modo dettagliato i vari modi con cui si sono ottenute queste grandezze, ora si vuole dare una visione d'insieme del funzionamento dell'algoritmo.

Come si vede in cima allo schema di Figura 4.2 il processo del sensore (*sensor process*) e quello del controllo del robot (*robot control*) leggono i dati della posizione dell'uomo e del robot. Da questi valori si calcola la distanza relativa tra uomo e robot.

I dati generati dal sensore e dal robot vengono però anche utilizzati per calcolare l'*input distance penalty* (d_t) e l'*output distance penalty* (d_B) che verranno poi sommati per comporre la distanza minima di sicurezza d_t^{safe}.

La distanza reale e la distanza minima di sicurezza verranno sfruttate per calcolare il livello di pericolo associati. Se calcolato puntualmente fornisce un valore che può essere comparato con un valore di *danger field* di soglia.

A questo punto il valore dei *danger field* vengono confrontati e in base all'esito del confronto si decide se applicare le procedure per evitare la collisione (*collision avoidance*) oppure se la condizione è ammissibile si lascia evolvere liberamente il sistema.

4.2 Composizione distanza minima di sicurezza

In questo capitolo verranno affrontate le tematiche relative al calcolo dei tempi necessari per comporre il massimo tempo di ritardo e come questi tempi vengono usati nelle computazioni dei volumi di sicurezza.

Il fattore temporale infatti gioca un ruolo fondamentale in quanto il calcolo dei volumi di lavoro avviene *in linea* e partendo dai dati raccolti dai sensori e dal robot.

Nel capitolo precedente è stata introdotta la sincronizzazione temporale dei *clock* per poter confrontare direttamente i tempi impiegati (si veda la Fig. 4.2 e 4.3).

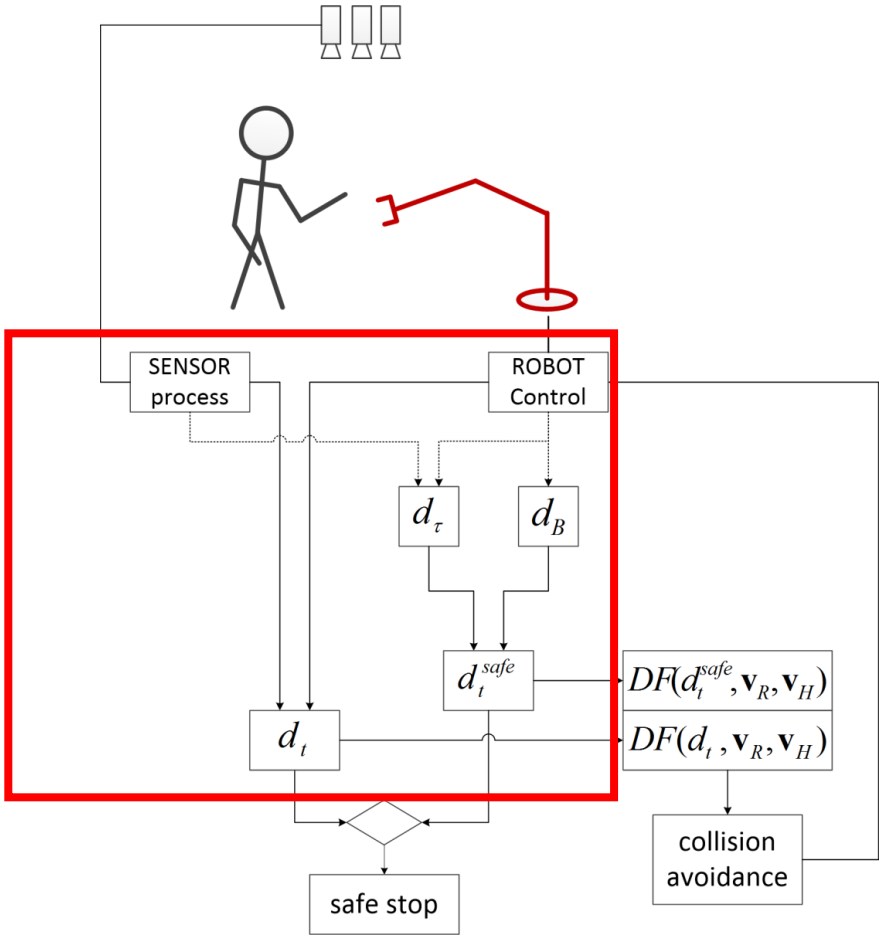


Figura 4.3 – Flow chart relativo al Calcolo degli spazi minimi di sicurezza.

Il problema è rappresentabile in Fig. 4.4. Il sistema acquisisce le posizioni di robot e uomo in determinati istanti, con una frequenza dipendente dai dispositivi fisici impiegati.

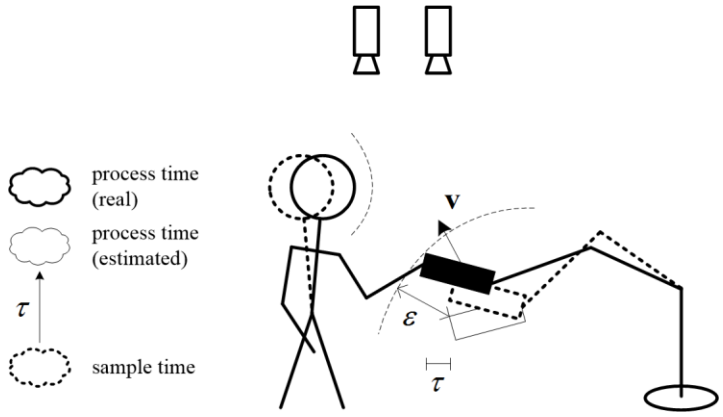


Figura 4.4 – effetto dei ritardi temporali

Il sistema evolve però naturalmente in modo continuo e quindi si avrà che da quando i dati sono acquisiti a quando sono utilizzati il sistema sarà cambiato. Per questo motivo è importate conoscere con esattezza qual è l'incertezza temporale.

Nei prossimi paragrafi verranno spiegati innanzitutto quali sono i tempi di interesse per questo tipo di analisi.

4.2.1 Composizione del ritardo massimo

Per come è stato costruito il sistema con la sua architettura, si avranno diversi componenti collegati alla rete. Questi componenti saranno sensori e robot, ognuno dei quali raccoglierà dei dati ad una frequenza diversa. Questa diversa frequenza di campionamento comporta quindi dei problemi in quanto i dati **non sono sincroni** e si ha un aggiornamento del dato solo ogni periodo di campionamento del sensore e/o robot.

Il tempo di campionamento in questione coinvolge però solo un singolo nodo, non si considera il fatto che questo dato dovrà essere trasmesso, ricevuto da un *CN* e utilizzato insieme a dati provenienti da altri nodi generati a tempi diversi.

Proprio per questo motivo è stato necessario effettuare un'analisi temporale più approfondita.

Posiamo innanzitutto il punto di vista sul nodo centrale *CN* in quanto è questo il nodo di riferimento temporale che raccoglierà tutti i dati e che dovrà poi usarli. E' necessario che esso conosca in modo esatto tutti i tempi che andranno a comporre il ritardo massimo e per fare ciò i termini dovranno essere specificati nei pacchetti provenienti dai singoli nodi.

Per facilitare la comprensione analizziamo il seguente schema temporale di figura 4.5 che va ad esplicitare le varie componenti.

Si considerino pertanto le linee temporali dei nodi SN e CN di Fig. 4.5.

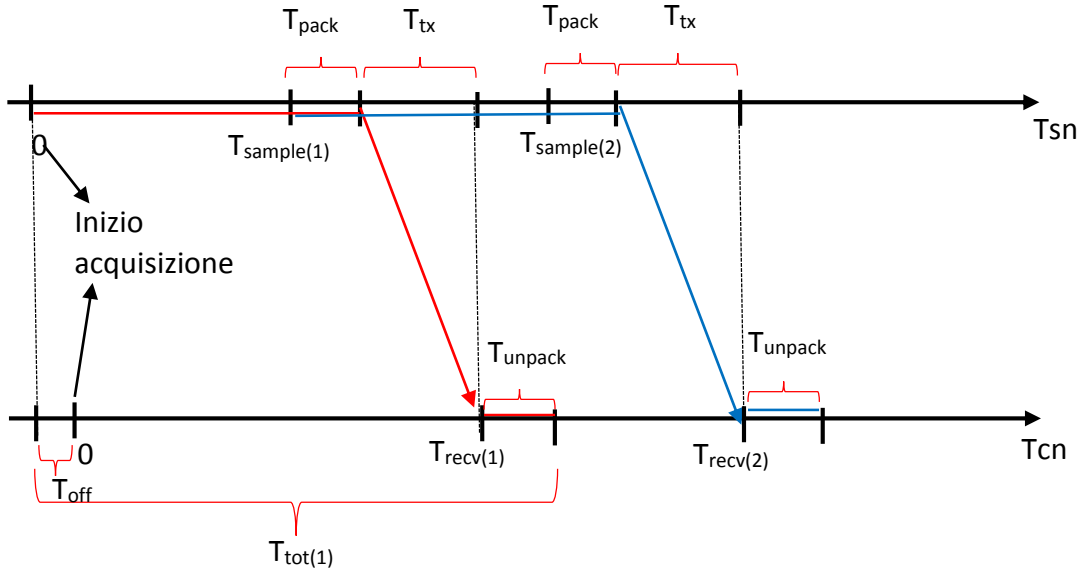


Figura 4.5 – Basi temporali e composizione dei tempi

Le linee colorate rappresentano il percorso temporale che dati diversi percorrono dall'origine al calcolo dove:

- T_{sample}(i) rappresenta il tempo in cui termina l'acquisizione di un dato da parte del sensore. Non essendo dei sistemi con sistema operativo *hard real time*, e dato che la comunicazione tra Tsensore e rete avviene attraverso una porta USB, si deve tenere in conto della variabilità di questo periodo, monitorandolo ad ogni ciclo.
- T_{pack} *rappresent*a il tempo necessario, una volta acquisito il dato, ad effettuare le operazioni per inserire i dati nella forma corretta all'interno della struttura dati.
- T_{tx} è il tempo di trasmissione del pacchetto con i dati provenienti dall'ultima acquisizione inseriti.
- T_{unpack} è invece il tempo necessario, una volta ricevuto il pacchetto lato *CN,* a decriptarlo e a mettere il contenuto informativo del pacchetto nelle variabili locali che saranno usate per i calcoli dei volumi di sicurezza.

T_{off} rappresenta l'*offset* tra i *clock* dei due nodi calcolato in precedenza.

Come risulta evidente dal grafico di Fig. 4.5 avremo quindi che il tempo totale che intercorre tra la richiesta del dato (quando il fenomeno fisico è osservato) e l'istante in cui questo dato è reso disponibile lato CN è pari a:

$$T_{tot}(i) = T_{sample}(i) + T_{pack}(i) + T_{tx}(i) + T_{unpack}(i) + T_{off} + \sum jitter(k) \quad (8)$$

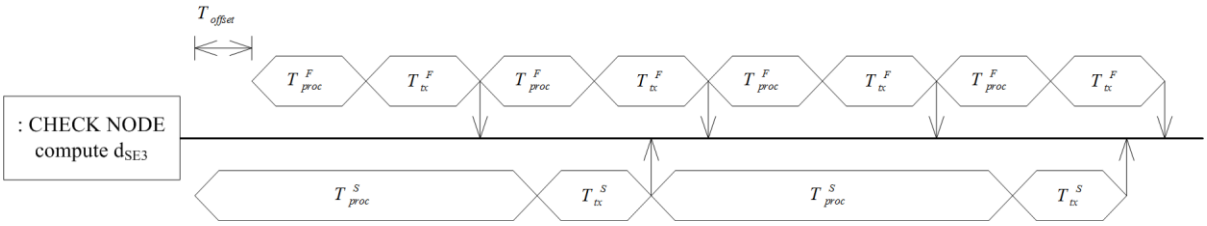


Figura 4.6 – Composizione del tempo di ritardo massimo

Nella computazione viene incluso un termine di sommatoria degli scostamenti (*jitter*) in quanto tutte le grandezze temporali sono grandezze assimilabili ad una variabile casuale.

Un analogo diagramma temporale si avrà anche per il nodo RN, in quanto anche il robot verrà interrogato periodicamente circa la sua posizione, che verrà inviata al nodo CN.

Dopo questa fase di analisi, necessaria per identificare tutte le cause di ritardo in gioco in ogni nodo, bisogna concentrare l'attenzione sulla combinazione dei ritardi percepiti dal CN quando riceve i pacchetti provenienti sia dal sensore che dal robot. Infatti, come accennato in precedenza, si ha che i nodi hanno frequenze di acquisizione diverse, pertanto i pacchetti che arrivano al nodo centrale non saranno sincroni.

Al fine di considerare correttamente il lasso di tempo in cui il sistema non conosce le nuove posizioni di robot e operatore (*blind time*), è necessario considerare la combinazione del diagramma temporale lato sensore e lato robot.

Il risultato è riportato in Fig. 4.7, dove la prima linea temporale è relativa al nodo SN, la seconda a *CN* e la terza a *RN*. Le linee dello stesso colore indicano dei pacchetti costruiti partendo da dati associati allo stesso istante temporale di osservazione.

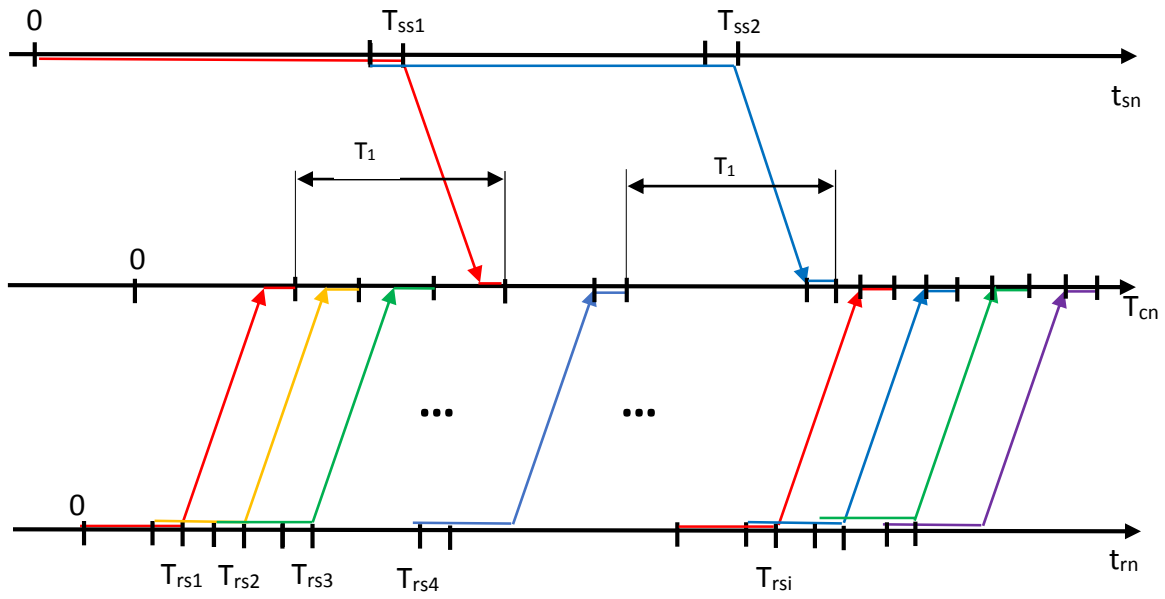


Figura 4.7 – Sfasamenti temporali delle osservazioni dello stesso fenomeno.

In tale figura sono indicati:

•$T_{ss}(i)$ è il tempo di campionamento del sensore (in cui è già incluso il tempo di packaging discusso precedentemente).

•$T_{rs}(i)$ è il tempo di campionamento del robot, in cui è già incluso il tempo di packaging.

Come messo in evidenza dal grafico, i pacchetti che contengono le informazioni relative all'osservazione di un fenomeno fisico di un preciso istante, arrivano al nodo CN in istanti temporali diversi.

Questo provoca una serie di problematiche legate all'utilizzo dei dati in modo consistente. Siccome uno dei due nodi avrà una frequenza maggiore dell'altro, si avrà che un dato viene acquisito più volte rispetto all'altro. Si considera quindi come *blind time* complessivo del sistema il caso peggiore, ossia quello indicato da T_1. In questo modo siamo sicuri che **entro** un periodo di durata T_1 arriveranno almeno dei dati aggiornati con cui fare un calcolo consistente degli spazi di sicurezza.

Il calcolo degli spazi di sicurezza sarà effettuato in tempo reale e periodicamente. Ogni periodo T_{calc} viene calcolato il valore puntuale di pericolo associabile al robot usando i dati più aggiornati che si hanno a disposizione, a questo calcolo sarà applicata una maggiorazione calcolata partendo dal periodo T_1.

Possiamo ora formulare analiticamente la composizione dell'incertezza temporale τ, utilizzata per calcolare dτ.

$$\tau = T_{offset} + \Delta T_1 + \Delta T_{tx} + \Delta T_{pack} + \Delta T_{unpack} + \sum_k jitters(k) \qquad (9)$$

Dove T_{offset} rappresenta la differenza tra i clock di sensore e robot, ΔT_{sample} rappresenta la differenza tra i periodi di campionamento di Robot e Sensore, ΔT_{tx} rapprensenta la differenza tra i tempi di trasmissione mentre ΔT_{pack} , ΔT_{unpack} rappresentano la differenza dei tempi impiegati per trasformare i dati reali in pacchetti in rete.

Il termine di sommatorie dei *jitter* temporale, come nel caso precedente, indica il rumore statistico legato alla variabilità dei tempi considerati.

Nel caso migliore i tempi di trasmissione, *pack* e *unpack* sono trascurabili in quanto le differenze sono dell'ordine dei microsecondi. Questa assunzione è verificata con l'esperienza diretta dalle molteplici campagne di misurazione dei tempi, riportate nel capitolo 3.

4.2.2 Composizione delle incertezze spaziali

Calcolata e misurata questa incertezza temporale τ il prossimo passo è quello di trasformare questa incertezza temporale nell'incertezza spaziale ε (si veda ε in Fig. 4.3).
Questa incertezza spaziale può essere vista come composta da due macro componenti:
il primo componente è l'*input distance penalty* d_τ e rappresenta l'incertezza spaziale che si ha a causa della incertezza temporale τ.

Supponiamo in prima approssimazione di considerare la velocità istantanea del robot, in quanto è un dato facilmente disponibile, quindi avremo:

$$V_R = \frac{d_\tau}{\tau} \qquad (10)$$

da cui si ottiene:

$$d_\tau = V_R \cdot \tau \qquad (11)$$

V_R rappresenta la velocità istantanea del robot (in modulo) mentre τ rappresenta l'incertezza temporale.

Con questa formula si riesce a capire qual è la distanza da mantenere da un punto del robot che si muove con una velocità V_R **in ogni istante**, solo per il fatto che i pacchetti che arrivano al nodo CN sono affetti da questa incertezza temporale.

Nel caso limite in cui la velocità del robot è nulla sarebbe sbagliato considerare una distanza da mantenere anch'essa nulla in quanto il robot nel lasso di tempo in cui non viene monitorato potrebbe eseguire un qualche moto. Per garantire anche in questa situazione una distanza minima, viene utilizzata la seguente formula:

$$d_t = \frac{1}{2} a_{spunto} * \tau^2 \qquad (12)$$

Dove a_{spunto} rappresenta l'accelerazione massima di spunto del robot quando è completamente fermo.

L'altra componente di incertezza spaziale è data dall'*output distance penalty* d_B, che è dovuta all'incertezza temporale legata al fenomeno della frenata del robot. Lo spazio di fermata del robot è definito dallo spazio percorso durante l'intervallo temporale che intercorre tra l'emissione del comando di stop e l'effettiva fermata del robot. Tale distanza è associabile ad un tempo di fermata attraverso una relazione complessa che lega la dinamica del robot, la traiettoria corrente, la configurazione del robot e il carico. In ultima analisi, la

distanza/intervallo di frenata sono determinati dalle caratteristiche dell'interpolatore del robot e dalle caratteristiche del controllo finalizzate alla realizzazione di tale traiettoria.

Il modo in cui è stato ricavato questo spazio in questo lavoro di tesi è invece di tipo sperimentale e verrà illustrato nel seguito. Per avere una mappa degli spazi di frenata bisogna misurare tutti i tempi di arresto in ogni configurazione del robot all'interno del suo spazio di lavoro. In ognuna di queste configurazioni è poi necessario far variare la velocità di partenza.

Sono stati misurati i tempi di frenata e gli spazi solo in alcune configurazioni del robot e con alcune condizioni iniziali.

Una volta ottenuti sia l'input distance penalty (d_τ) che l'output distance penalty (d_B), si devono sommare trovando così la massima distanza che percorrerà il robot se in quell'istante fosse generato un segnale di stop:

$$d_t^{safe} = d_\tau + d_B \qquad (13)$$

Bisogna prestare attenzione perché questa distanza è riferita ad un punto del robot, quindi bisogna procedere per tutti i punti del braccio robotico in modo da conoscere la distanza che si deve mantenere.

Il risultato è quindi quello di disegnare un insieme di punti attorno al robot che hanno una distanza ben precisa dal robot. Questa procedura sarà descritta nel prossimo paragrafo.

4.3 Danger Field

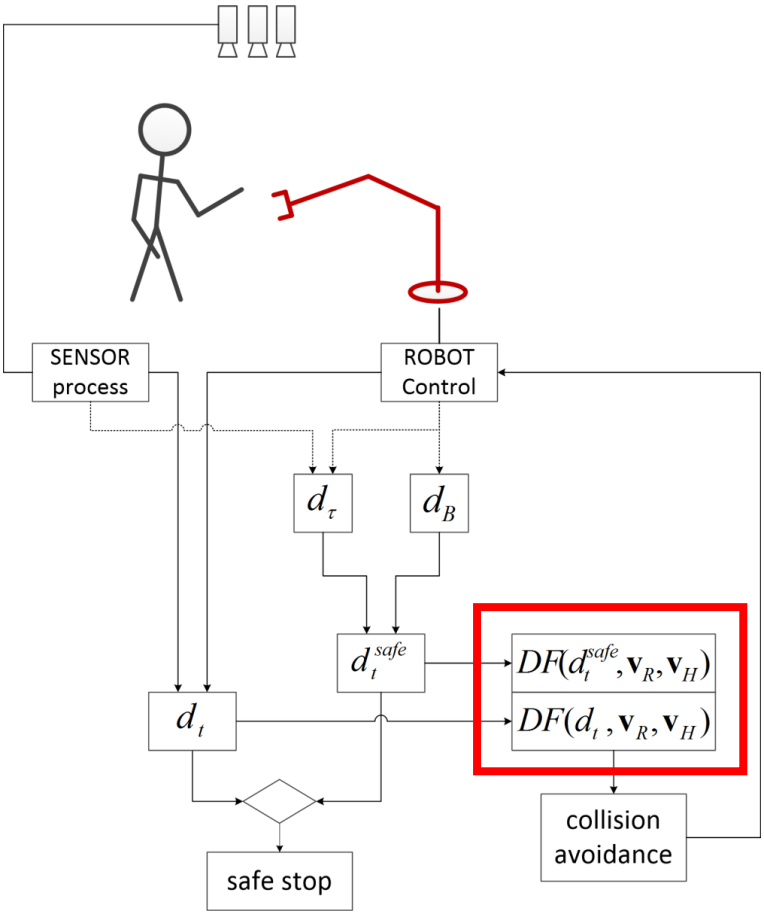


Figura 4.8 – Flow chart. Danger Field

Una volta calcolata la distanza D_t^{safe} e misurata la distanza effettiva tra uomo e robot – passi che rappresentano il nucleo di applicazioni HRC di tipo SSM – è anche possibile utilizzare le informazioni ottenute dal monitoraggio di cella allo scopo di implementare anche politiche di *collision avoidance* o manovre evasive. Tra esse, le tecniche basate su campi di rischio dipendenti dalla traiettoria del robot si prestano molto bene ad integrare l'informazione di livello di rischio associato ad una determinata distanza critica.

Tra queste tecniche, Il *danger field* [22][23] è ad esempio un campo vettoriale che indica il livello di pericolo associato alla vicinanza di un robot, tenendo in considerazione la posizione del robot rispetto a quella dell'ostacolo, alla velocità relativa dei due attori e alla direzione della velocità degli stessi. Il campo è continuo su tutto il dominio e rappresenta una grandezza intensiva della pericolosità del moto del robot in un dato istante. La distanza minima (grandezza estensiva) può quindi essere impiegata per confrontare all'interno di tale campo il valore di

rischio associato a tale distanza. Data la natura traiettoria-dipendente di entrambi i principi cinematici (distanza minima e *danger field*), le grandezze disponibili possono venire impiegate come efficace misura di ripianificazione del moto allo scopo di mantenere adeguatamente lontana la condizione di rischio.

In seguito vengono riportate e riassunte da [22],[23] le definizioni matematiche dalle quali sono stati sviluppati gli algoritmi utilizzati nello sviluppo di questa tesi. In appendice A sono state inserite le definizioni matematiche.

Verrà ora introdotta un'istanza del *danger field*, ossia la funzione che rispetta tutte le proprietà che si sono elencate prima, che sarà ad utilizzata ai fini pratici. Consideriamo una funzione, valida per il singolo punto descritta mediante l'equazione:

$$DF(\boldsymbol{r}, \boldsymbol{r_t}, \boldsymbol{v_t}) = \frac{k_1}{\|\boldsymbol{r}-\boldsymbol{r_t}\|} + \frac{k_2\|\boldsymbol{v_t}\|[\gamma+\cos(\boldsymbol{r}-\boldsymbol{r_t},\boldsymbol{v_t})]}{\|\boldsymbol{r}-\boldsymbol{r_t}\|^2} \qquad (14)$$

dove k_1 , k_2 e γ sono delle costanti positive che possono essere scelte arbitrariamente, **r** il vettore che identifica un punto nello spazio in cui viene calcolato il danger field mentre $\boldsymbol{r_t}$ *e* $\boldsymbol{v_t}$ sono la posizione e la velocità di un elemento infinitesimo del *link* robotico che si sta considerando.

Il ***cumulative kinetostatic danger field*** indotto dal movimento dell'intero *link* robotico sarà ottenuto mediante integrazione della funzione sopra indicata, ossia:

$$CDF(\boldsymbol{r}, \boldsymbol{r_i}, \boldsymbol{r_{i+1}}, \boldsymbol{v_i}, \boldsymbol{v_{i+1}}) = k_1 \int_0^1 \frac{dt}{\sqrt{at^2+bt+c}} + k_2\gamma \int_0^1 \frac{\sqrt{At^2+Bt+C}}{at^2+bt+c}dt + k_2 \int_0^1 \frac{Mt^2+Nt+P}{(at^2+bt+c)^{\frac{3}{2}}}dt$$

(15)

E' fondamentale che questi integrali siano tutti risolvibili analiticamente: in questo modo, una volta ottenute le primitive, il valore del *danger field* in ogni punto sia ricavabile per via algebrica, senza dover calcolare ogni volta l'integrale stesso.

Per verificare la correttezza dei calcoli, e per essere sicuri del significato fisico delle grandezze che si stanno calcolando, è stata compiuta un analisi su un manipolatore a 2 *link*. Inoltre, per semplicità di calcolo andremo a calcolare il *danger field* in uno spazio 2D quindi su un piano di lavoro.

La prima cosa necessaria per questo calcolo è la determinazione delle primitive dei tre integrali indicati in precedenza. Una volta determinate queste primitive si ha a disposizione la formula algebrica che sarà utilizzata in ogni punto dello spazio per calcolare il *danger field*.

Si introduce quindi una discretizzazione in quanto non è possibile ovviamente calcolare le grandezze continue, ma con un passo di discretizzazione sufficientemente piccolo questa approssimazione è accettabile. La discretizzazione è relativa sostanzialmente ai punti considerati, sia dello spazio di lavoro che del robot. Lo spazio di lavoro è discretizzato con una

griglia di passo $\Delta x = \Delta y = 0.1\ cm$, in cui ogni elemento sarà memorizzato in una matrice, in cui gli indici rappresentano la posizione.

Per visualizzare in modo grafico questo campo nello spazio si è utilizzata una griglia bi-dimensionale con granularità pari a quella della discretizzazione dello spazio. In questo modo gli indici della griglia corrispondono ad una data posizione spaziale mentre il valore del *danger field* viene scritto nella cella corrispondente a quegli indici.

Quindi si andrà a discretizzare anche il robot, in cui ogni *link* è visto come una sequenza di punti.

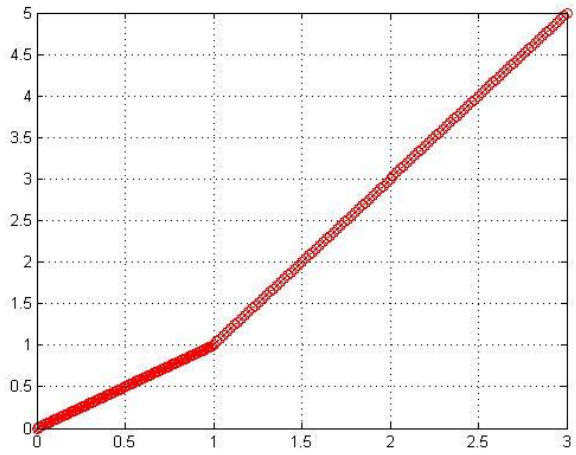


Figura 4.9 – rappresentazione 2d del robot discretizzato

Il passo di discretizzazione è identico, sia per lo spazio di lavoro che per il robot. In Fig. 4.10 sono state riprese le grandezze, già introdotte in precedenza, che saranno utilizzate nel calcolo, anche per visualizzare fisicamente le quantità indicate.

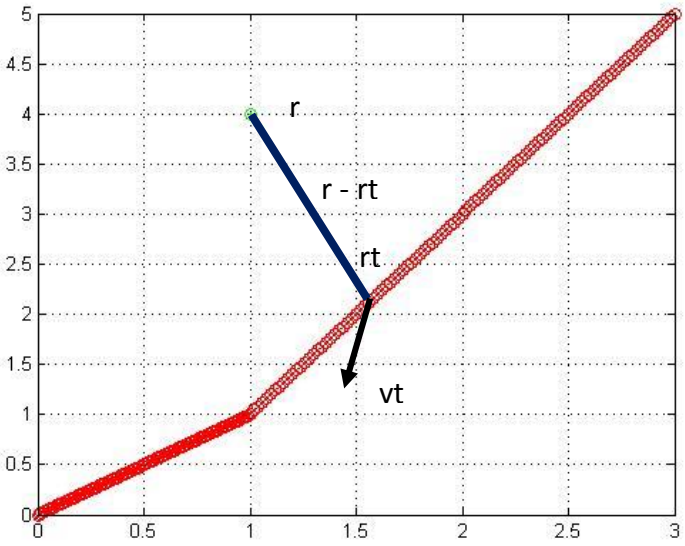


Figura 4.10 – grandezze rappresentate nell'ambiente di simulazione dopo la discretizzazione

Per il calcolo del *danger field* si è poi utilizzata una specializzazione della funzione, in particolare saranno utilizzati i seguenti valori per i parametri:$k_1 = 90, k_2 = 100\ e\ \gamma = 100$.

$$k_1 = 90, k_2 = 100\ e\ \gamma = 100.$$

Il valore di questi parametri è stato ottenuto empiricamente, ricavandoli sulla base del modello del robot utilizzato.

Il robot è quindi stato discretizzato in un numero ben definito di punti, di cui conosciamo la posizione ad ogni istante.
Per il calcolo delle coordinate di questi punti che costituiscono il robot si è proceduto nel seguente modo: data la posizione delle estremità dei *link*, si ricostruiscono i segmenti analitici e l'insieme dei punti secondo lo schema di Fig. 4.11.

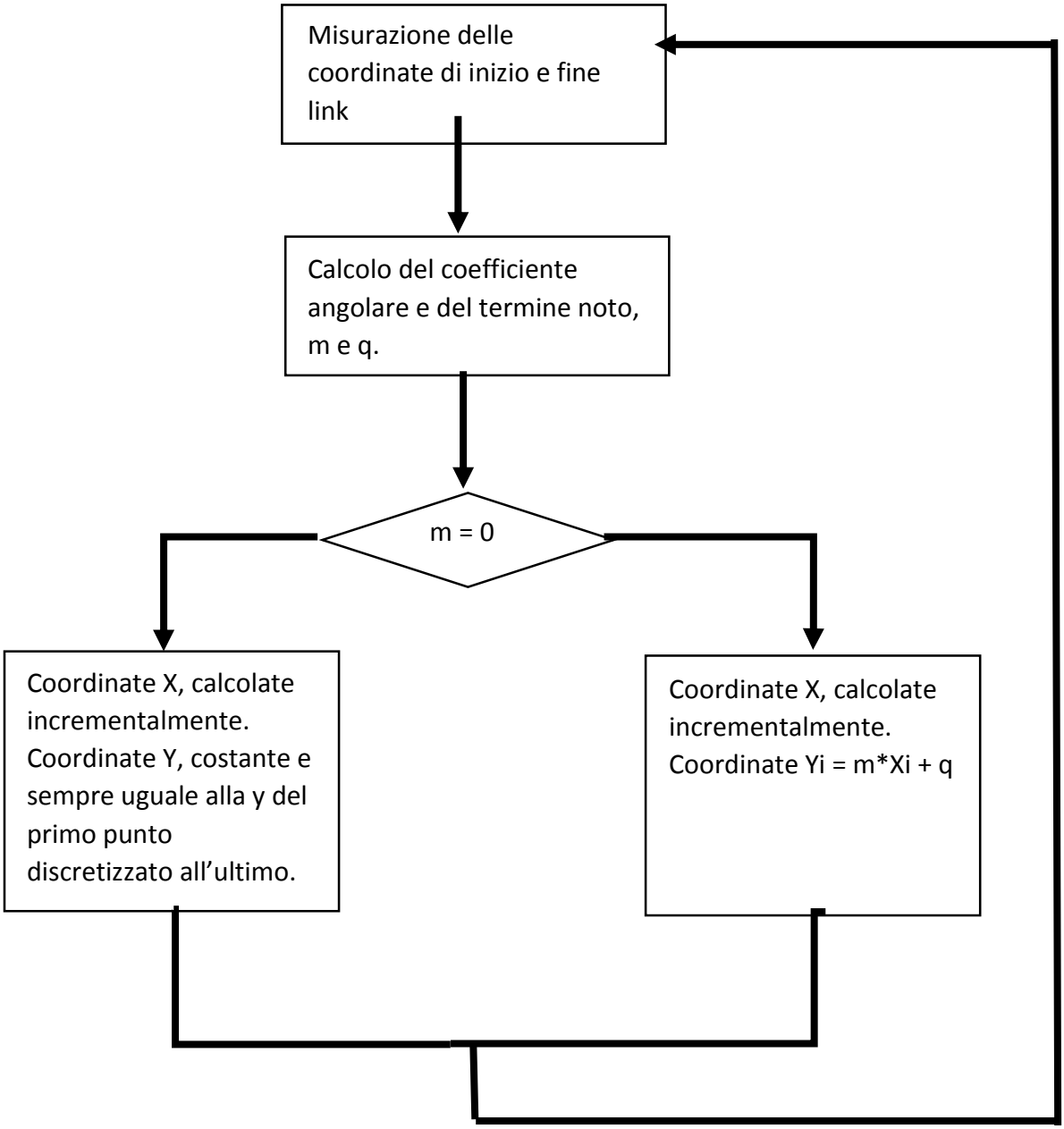


Figura 4.11 – Algoritmo di discretizzazione e di calcolo delle coordinate.

A questo punto una volta calcolate le coordinate degli elementi che compongono i *link* del robot, ad ognuno di questi punti deve essere associata la propria velocità. Come per le coordinate del gomito e del polso si possono calcolare le velocità associate, o per misura diretta o per differenziazione delle posizioni misurate in due istanti successivi a breve distanza di tempo.

Per associare ad ogni punto dei segmenti ricostruiti precedentemente una velocità, si è utilizzato l'approccio del profilo di velocità triangolare come indicato in Fig. 4.12. Quindi, una volta nota la velocità degli estremi dei link è possibile ricreare il profilo di velocità.

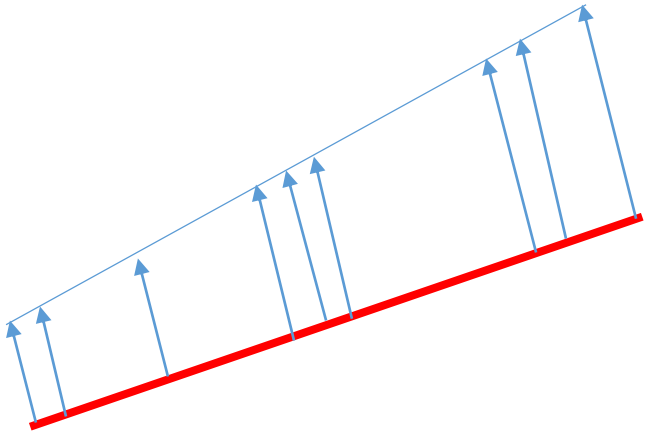


Figura 4.12 – profilo di velocità di un link

Della velocità per ogni punto del *link*, non si conosce solo l'intensità ma anche le diverse componenti sui tre assi cartesiani.

In questo modo si conosce la direzione e il verso della velocità in questione, riuscendo così a calcolare l'angolo che questo vettore velocità forma con il vettore distanza tra punto e elemento del link.
A questo punto tutti gli elementi necessari per il calcolo del *danger field* sono disponibili.
E' importante tenere presente che il calcolo che viene eseguito serve per calcolare il *danger field* di **tutto lo spazio di lavoro**. Questo calcolo è stato eseguito per comprendere il comportamento e la velocità con cui si modifica il *danger field* rispetto alla velocità di movimento del robot.

Lo schema di flusso di Fig. 4.13 rappresenta l'algoritmo che è stato implementato in Matlab ed è stato utilizzato per le verifiche di consistenza. In questo algoritmo è fondamentale che tutti gli elementi siano discretizzati con lo stesso passo, in modo da non avere le matrici rapprensentanti lo spazio di lavoro e i valori del *danger field* di dimensioni diversa. I vettori rappresentanti i singoli elementi dei link e le relative velocità verranno poi rappresentate in questo ambiente.

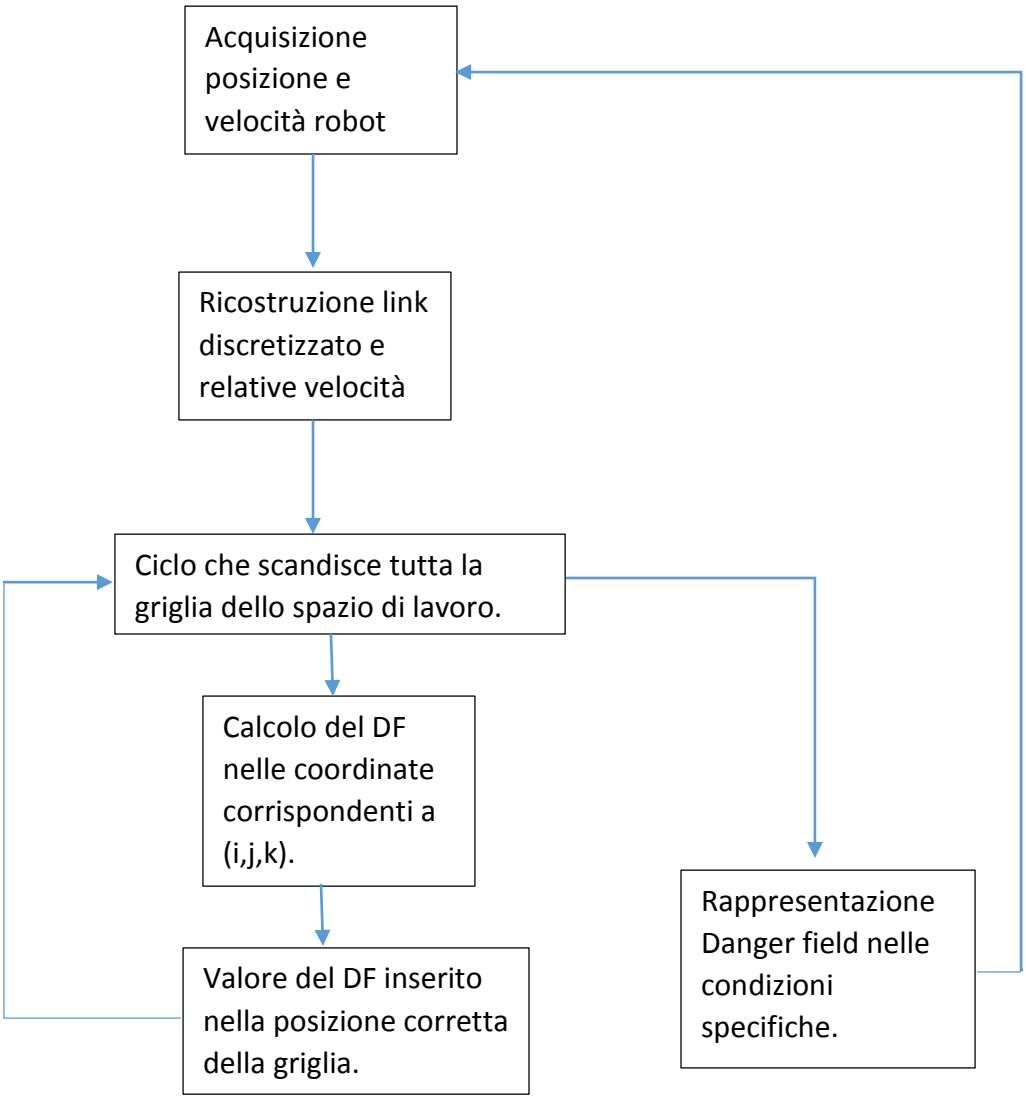


Figura 4.13 – Flow chart dell'algoritmo di generazione dei danger field

Le immagini in seguito riportate sono quelle relative al caso 2D, in cui si seleziona una quota dello spazio di lavoro e si vanno a rappresentare le curve di livello del *danger field* e i *link* del robot. Le matrici risulteranno quindi bidimensionali così come le coordinate dei *link* saranno espresse solo in funzione delle coordinate cartesiane.

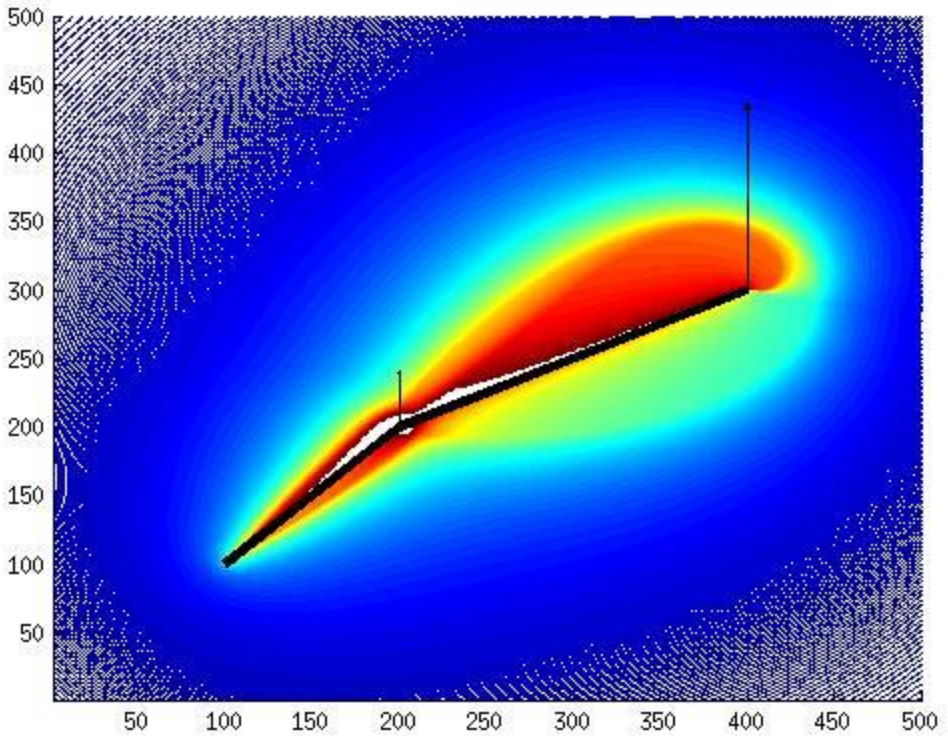


Figura 4.14 – danger field bidimensionale di un manipolatore costituito da 2 link

Come si può vedere dall'immagine 4.14 le curve di livello creeranno una sorta di mappa di colori a seconda dell'intensità del loro valore.

La scala cromatica utilizzata è dal blu al rosso dove le colorazioni blu indicano valori inferiori crescendo fino ai colori rossi. Il *danger field* tiene quindi conto della posizione dello spazio di lavoro rispetto a cui è calcolato, ma soprattutto è fortemente dipendente dalla velocità e dalla direzione della velocità. I vettori rappresentati sono le velocità agli estremi di ogni *link* e la bolla rossa costruita attorno al robot sarà deformata secondo questi vettori.

E' fondamentale vedere che, nel verso opposto di questi vettori velocità, il *danger field* è praticamente nullo in quanto il movimento del robot sarà nella direzione opposta, eccezion fatta per una piccola zona che viene mantenuta con un livello di elevato pericolo in quanto per effetto delle latenze di rete non si può sapere il comportamento del robot in quell'intervallo di tempo, per cui potrebbe anche invertire il moto.

E' stata poi realizzata la simulazione di una traiettoria compiuta dal robot per vedere la variabilità del *danger field* in funzione della posizione ma soprattutto della velocità.

La simulazione è stata effettuata sfruttando un modello del manipolatore KUKA LWR:

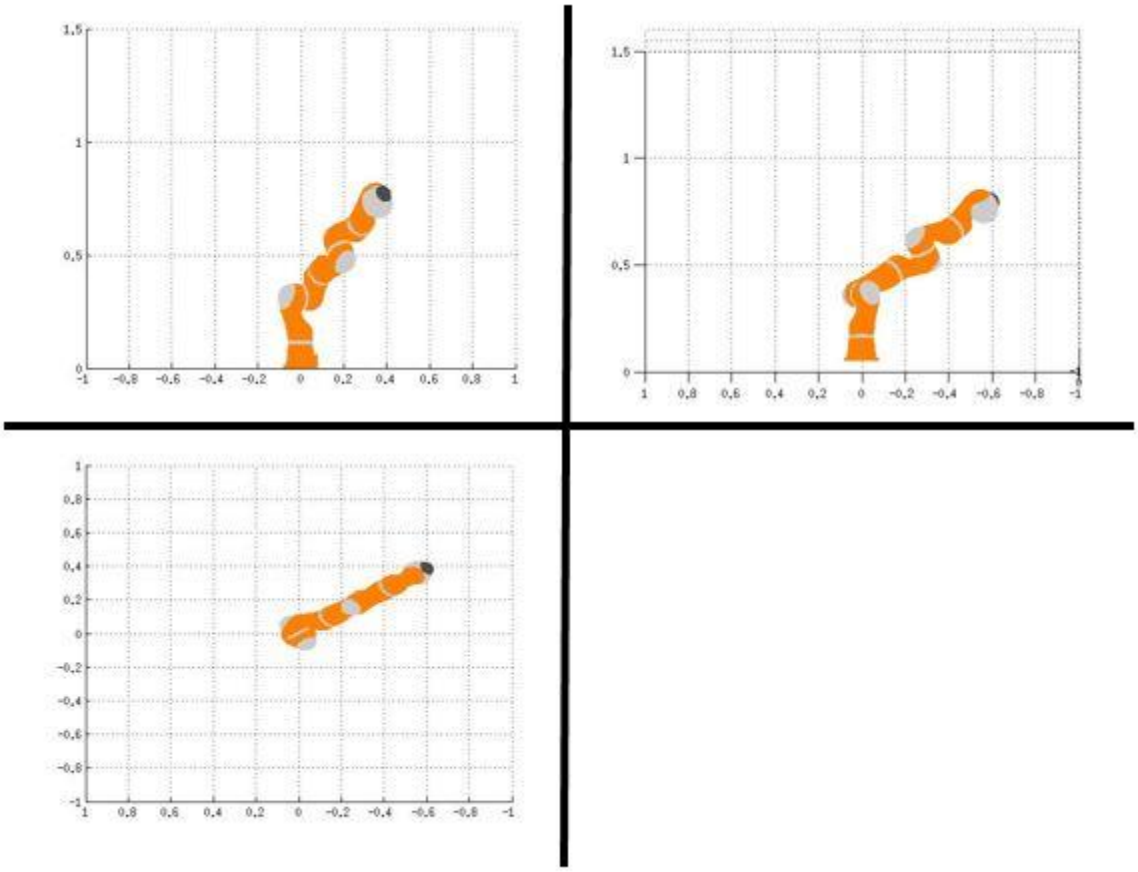


Figura 4.15 – proiezioni del modello robot

Volendo utilizzare un modello 2D, verrà quindi considerata la vista dall'alto; in questo modo il robot è assimilabile a 2 *link*: dalla base al gomito, e dal gomito all'*end-effector*, senza considerare i gradi di libertà aggiuntivi.

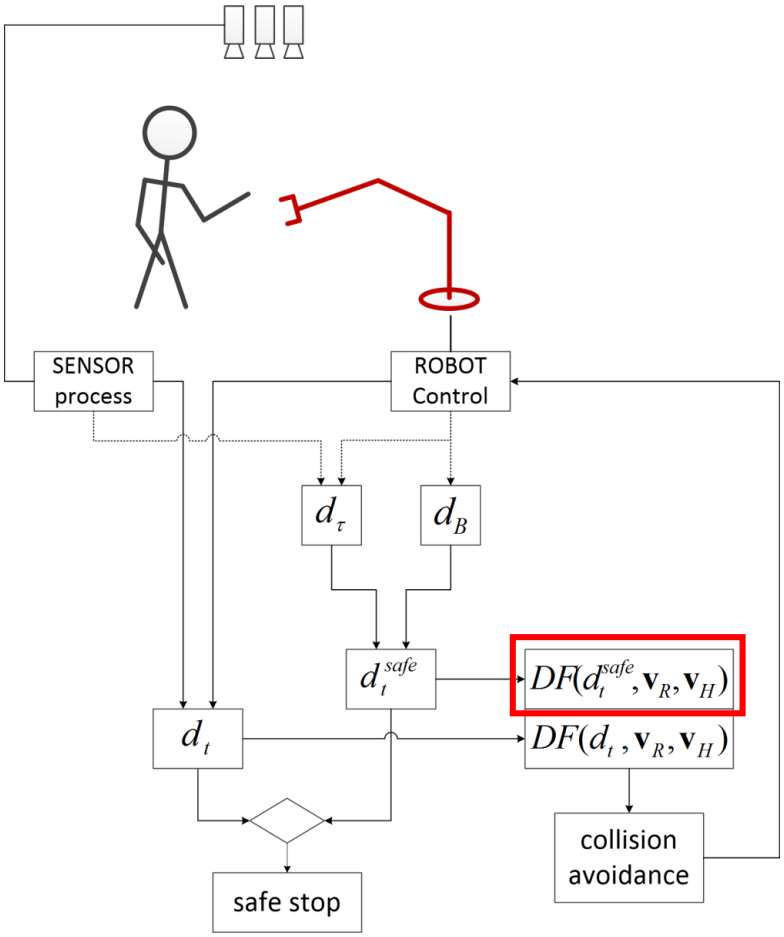


Figura 4.16 – Determinazione di curve iso-rischio.

Concentriamo ora l'attenzione sulle curve di livello (blocco rosso in Fig. 4.16): esse rappresentano un insieme di punti tutti allo stesso potenziale, cioè un insieme di punti che hanno lo stesso valore di *danger field*.

Questo risultato può essere osservato da 2 punti di vista duali:

- fissato un valore di *danger field* si estrae una curva di livello che rappresenta tutti i punti a quel valore ;
- fissata una distanza, si può calcolare un valore del *danger field* associato a quella distanza.

E' proprio su questa seconda interpretazione del risultato che si è fissata l'attenzione.

Infatti con i calcoli precedentemente introdotti si è giunti alla conclusione che in ogni istante, note la velocità relativa e la configurazione si è in grado di calcolare quant'è la distanza minima che si deve mantenere dal robot.

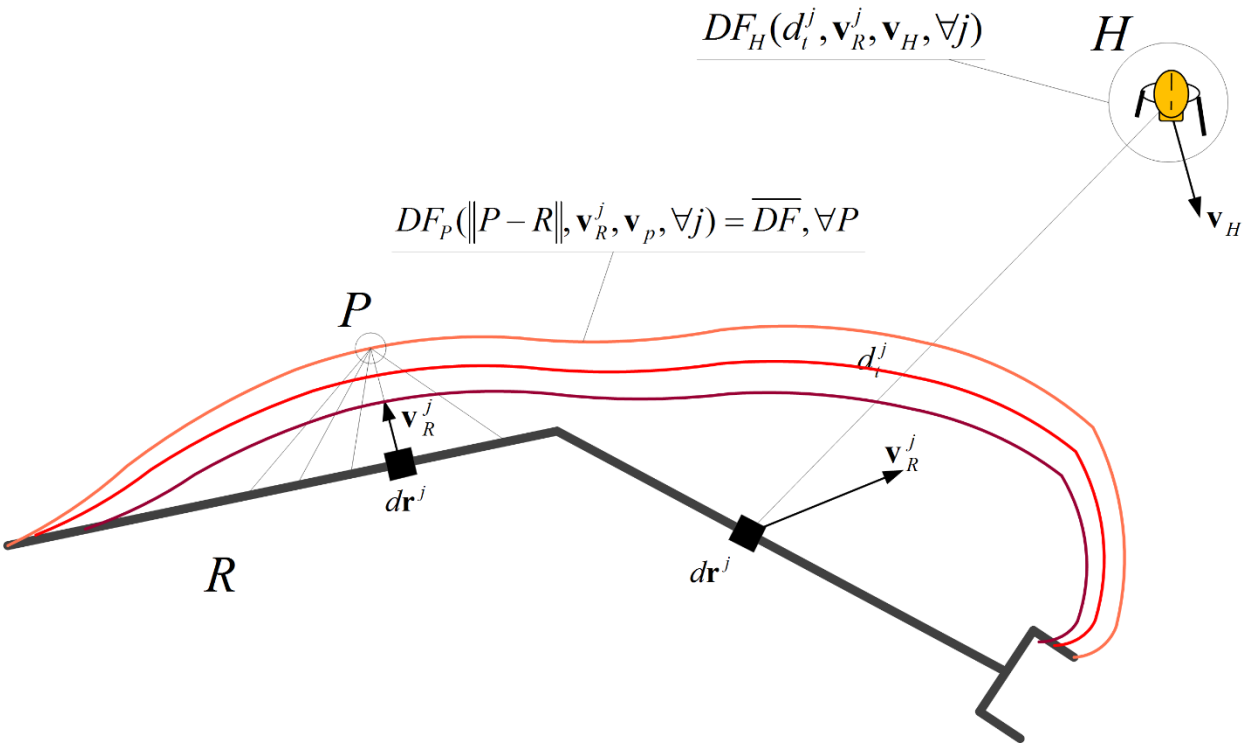


Figura 4.17 – Rappresentazione della curva iso-rischio corrispondente alla distanza minima di sicurezza.

A questo punto questa distanza è quella che useremo nel calcolo del valore del *danger field* di soglia. Non verranno più inserite nella funzione le coordinate di un punto esterno generico, ma verrà inserita direttamente la distanza che si deve rispettare alla luce dei calcoli effettuati, cioè $DF(d_t^{safe}, \boldsymbol{v_r}, \boldsymbol{v_H})$.

L'algoritmo è stato modificato come specificato nella figura 4.18.

Si noti che in questo caso le uniche informazioni sono relative alla distanza e alla velocità, vettoriale in modo da conoscerne la direzione, con cui si muove il robot. Il risultato sarà dunque un **valore scalare** che rappresenta la soglia in un dato istante di tempo.

Il calcolo, a differenza di quello effettuato per il *danger field* su tutto lo spazio di lavoro, è un conto algebrico che risulta poco dispendioso computazionalmente. Questo è fondamentale per mantenere al minimo le componenti temporali di ritardo.

Questo procedimento ci consentirà quindi di calcolare una soglia dinamica, che evolve con l'evoluzione del sistema, evitando così di dover fissare a priori un valore che impedire così un buon funzionamento al sistema.

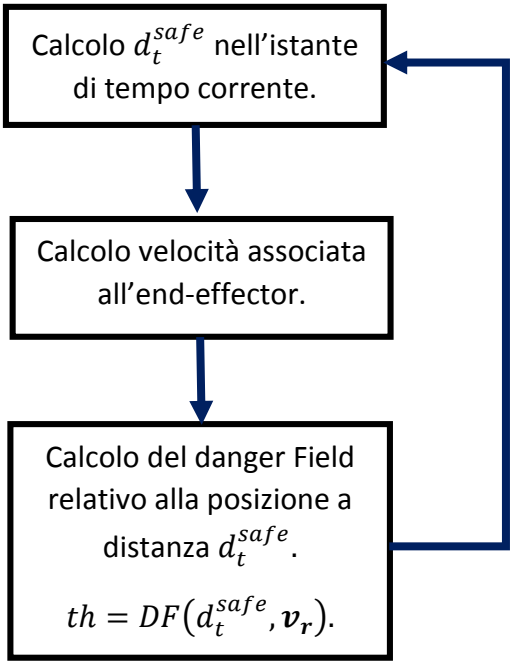


Figura 4.18 – Algoritmo per il calcolo del danger field

Per quanto riguarda il calcolo del *danger field* relativo all'ostacolo, questo altro non è che una semplificazione del caso studiato in precedenza su tutto lo spazio di lavoro. Infatti si avrà che l'ostacolo non è visto che come un punto di uno spazio di lavoro e quindi, con riferimento allo schema a blocchi in figura 4.18, una sola iterazione del ciclo sullo spazio di lavoro. Associato a questo ostacolo ci sarà però una velocità con cui esso si muove.

E' quindi importante tenerla in considerazione per avere un sistema in cui si consideri non solo la velocità del robot ma la **velocità relativa** uomo-robot. Le velocità saranno scomposte quindi nelle tre componenti cartesiane e si considererà una velocità relativa calcolata come:$\boldsymbol{v_{rel}} =$

$$\boldsymbol{v_{rel}} = \begin{bmatrix} \boldsymbol{v_{relx}} \\ \boldsymbol{v_{rely}} \\ \boldsymbol{v_{relz}} \end{bmatrix}$$

Dove ogni componente $v_{reli} \quad per\ i = x, y, z$ viene calcolata nel seguente modo:

$$v_{reli} = \begin{cases} v_{Hi} + v_{ri} & se\ v_{Hi} * v_{ri} < 0 \\ v_{hi} - v_{ri} & se\ v_{Hi} * v_{ri} > 0 \end{cases} \qquad (16)$$

Cioè le componenti si sommano se sono di verso opposto mentre si sottraggono se sono dello stesso verso.

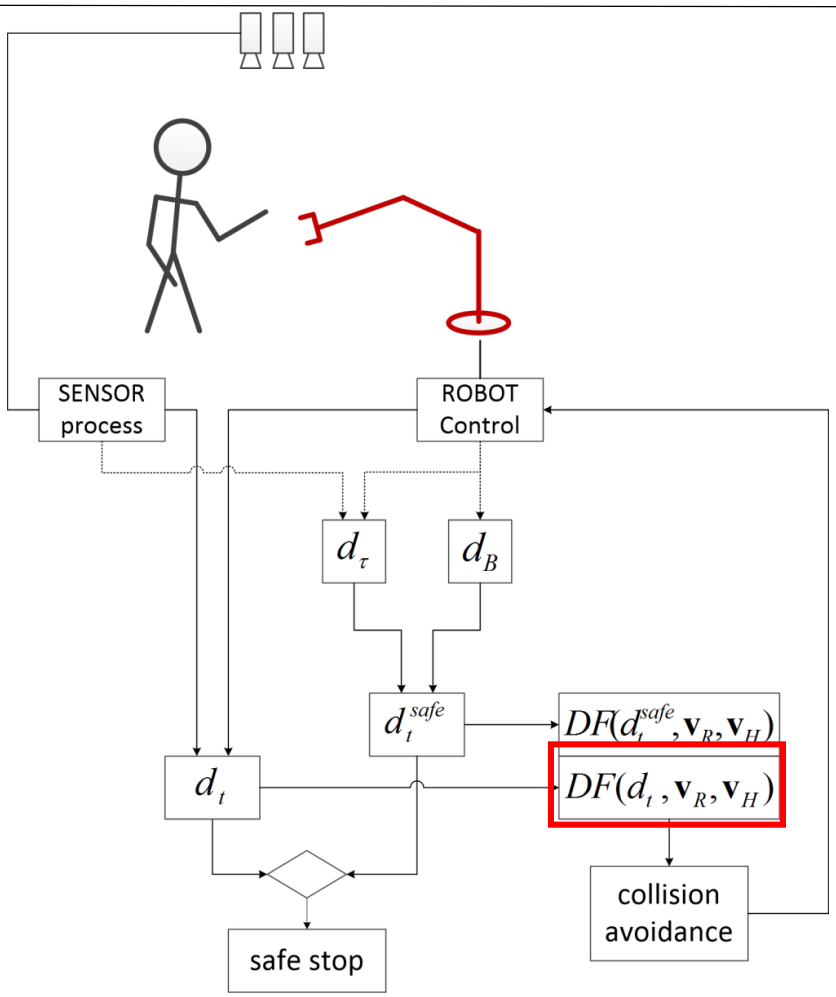


Figura 4.19 – Calcolo danger field associato all'ostacolo.

Si modifica dunque l'algoritmo di Fig 4.18 in quello di Fig. 4.20.

L'ostacolo in prima approssimazione è considerabile come un punto materiale, però nella realtà avrà un ingombro dovuto all'errore di *tracking* dello strumento e alle dimensioni fisiche dell'ostacolo stesso.
Viene quindi considerata, per una maggiore precisione, un insieme di punti in particolare una circonferenza, per rappresentare questo ingombro. Nel calcolo del valore del *danger field* verrà considerato un solo punto dell'ostacolo e per considerare sempre il caso peggiore sarà scelto il punto a distanza minima.
Anche in questo caso, come per il calcolo del valore di soglia, avremo che il calcolo è un operazione algebrica in quanto la primitiva che costituisce la funzione *danger field* è stata estrapolata in precedenza. Si determina istante per istante il **valore scalare** che viene associato al singolo ostacolo.
Nel caso fossero presenti più ostacoli il calcolo viene effettuato una volta per ogni ostacolo.
Il carico computazionale dovuto alle operazioni è comunque irrilevante e si riesce a mantenere così il calcolo in tempo reale.

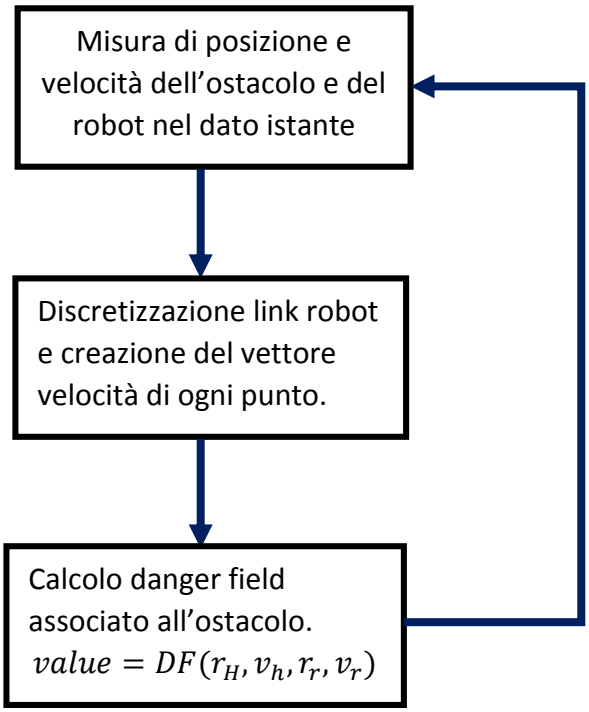


Figura 4.20 – algoritmo per il calcolo danger field associato all'ostacolo

Completata questa fase di analisi e di impostazione dei calcoli è stata realizzata una simulazione su una traiettoria in Matlab, in cui è stato simulato anche il moto di tre ostacoli nello spazio di lavoro.

Per semplicità e chiarezza di rappresentazione dei risultati è stato realizzato il calcolo nel caso bidimensionale.

Il robot eseguirà una rototraslazione ad una quota fissata mentre i tre ostacoli si muoveranno di movimenti puramente traslatori.

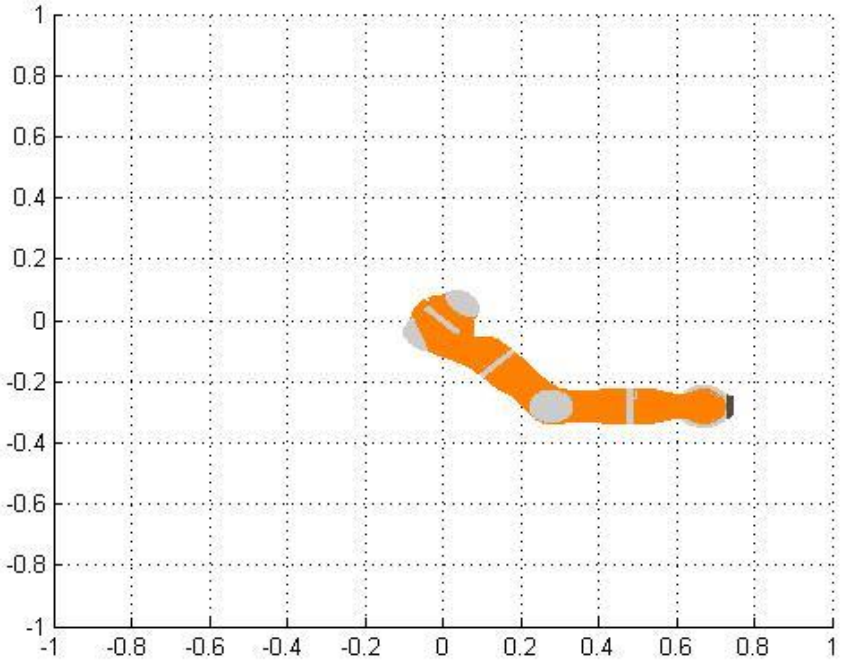


Figura 4.21 – Posizione del robot all'inizio della traiettoria planare

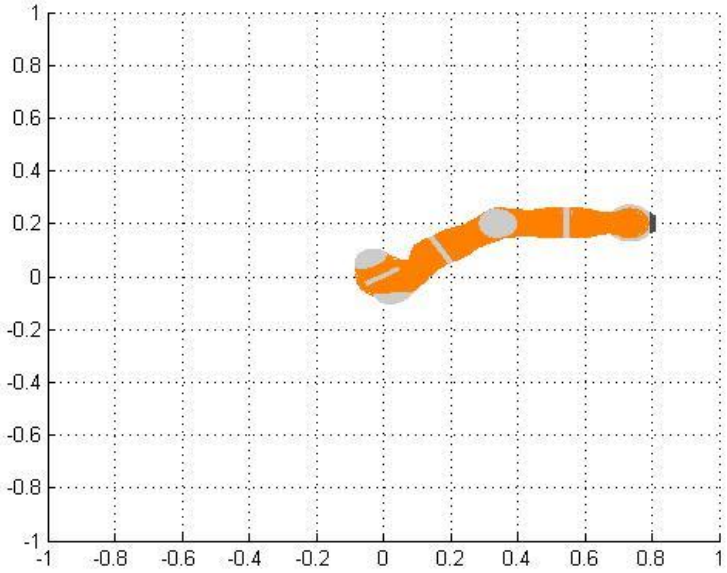


Figura 4.22 – Posizione del robot al termine della traiettoria planare

In seguito vengono riportate le immagini che rappresentano delle *istantanee* della simulazione di traiettoria.

La linea rossa rappresenta la distanza minima di sicurezza, ossia l'insieme dei punti che costituisce la soglia entro il quale l'ostacolo non deve entrare, oltre la quale è garantita l'assenza di collisioni.

Le immagini sono divise su due colonne. Nella colonna di sinistra sono rappresentati gli ingombri degli ostacoli, i *link* del robot e la rappresentazione dello spazio minimo di sicurezza nelle condizioni indicate nell'intestazione.

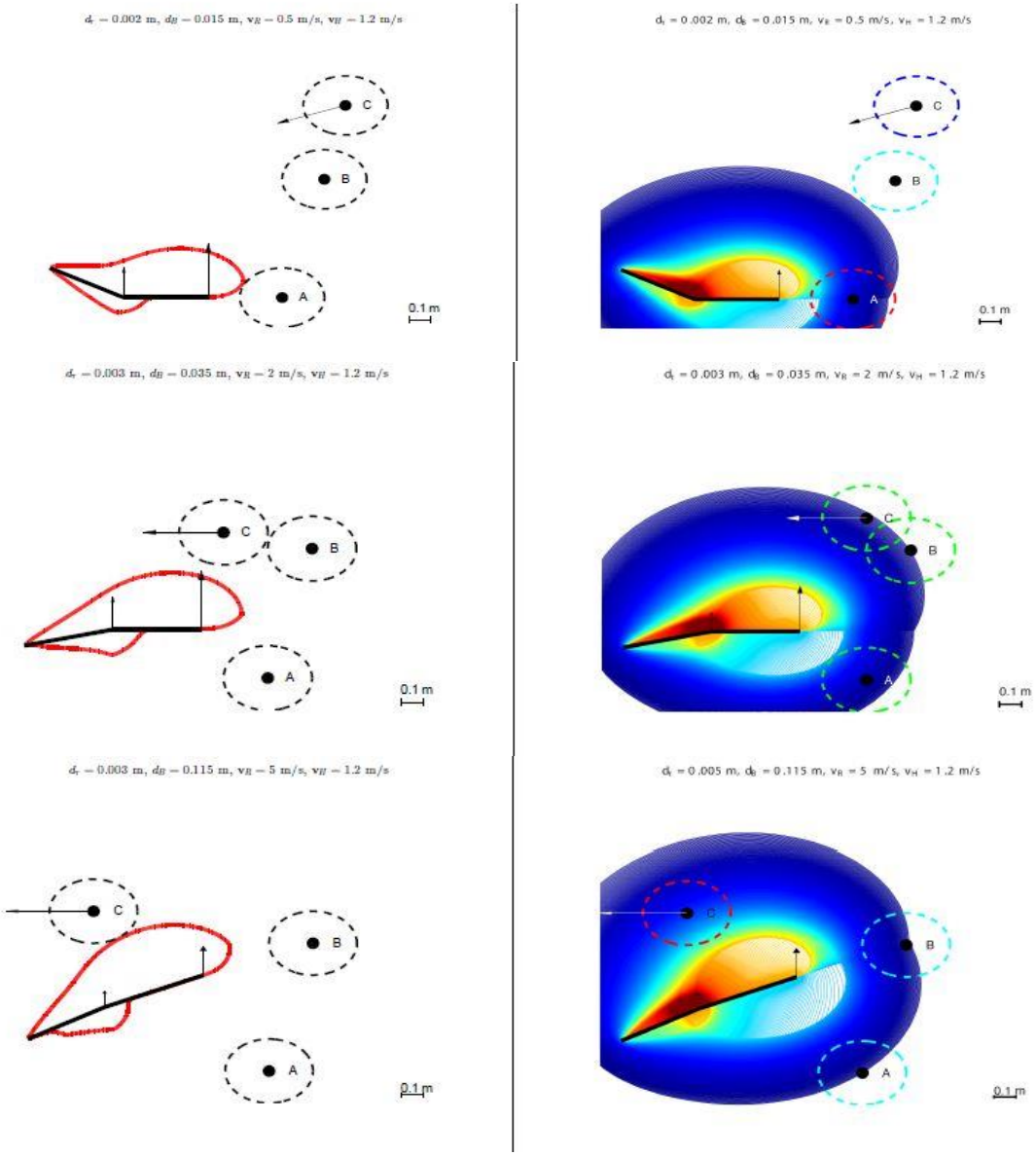


Figura 4.23 – simulazione traiettoria planare con rappresentazione spazi minimi di sicurezza

Nella colonna di destra sono rappresentati il *danger field* dell'intero spazio di lavoro e i gli ostacoli che hanno un colorazione dipendente dall'intensità del *danger field*.
La scala cromatica utilizzata è la stessa sia per lo spazio di lavoro che per gli ostacoli. Si parte dal blu per indicare situazioni di sicurezza elevata, poi ciano, verde e infine rosso nelle zone in cui la sicurezza è al limite in quanto l'ostacolo è al limite dello spazio minimo di sicurezza.

Nella simulazione è stato considerato un ingombro degli ostacoli pari a 0.6 m, considerando che l'ostacolo possa essere quindi un uomo. Essendo così ampia la dimensione dell'oggetto ed essendo "ideale" lo strumento di misura, in quanto sono note le posizioni, la dimensione della circonferenza coincide con quella dell'ingombro stesso.

Si sottolinea il fatto che in ogni istante della simulazione viene eseguito l'algoritmo di figura 4.2 in cui viene fatta una comparazione tra il valore di soglia e il valore attuale.
Se l'esito del confronto restituisce un valore di soglia maggiore del valore attuale allora il sistema viene lasciato evolvere liberamente.
Viceversa se la soglia diventa minore o uguale del valore attuale allora si avrà l'applicazione di un semplice algoritmo di per.

Il caso di soglia minore del valore, in realtà, non si raggiunge mai in quanto per come è costruito l'algoritmo quando il confronto restituisce un valore uguale si ha già una ripianificazione della traiettoria del robot. Questa nuova traiettoria sarà fatta in modo tale da abbassare il valore attuale di *danger field* in modo che ritorni sotto la soglia. Il modo più semplice è quello di far allontanare il robot dall'ostacolo in direzione opposta alla velocità dell'ostacolo stesso in modo da far decrescere il valore in modo significativo ed aumentare contemporaneamente il valore della soglia in quanto cambieranno le grandezze relative al moto del robot.

Capitolo 5
Simulazioni e Test

In questo capitolo verranno presentati i test effettuati sul sistema fisico reale, in fase di prototipazione, e le simulazioni fatte sulla base delle misurazioni effettuate sul campo per quanto riguarda il secondo ambiente di sviluppo.

Il primo sistema fisico su cui sono stati eseguiti i test è composta da un manipolatore a 7 gradi di libertà KUKA LWR4+, un sistema di *tracking* ottico infrarosso a marker passivi NDI Polaris SPECTRA ®, e i relativi PC di asservimento (Vedi Fig. 5.1)

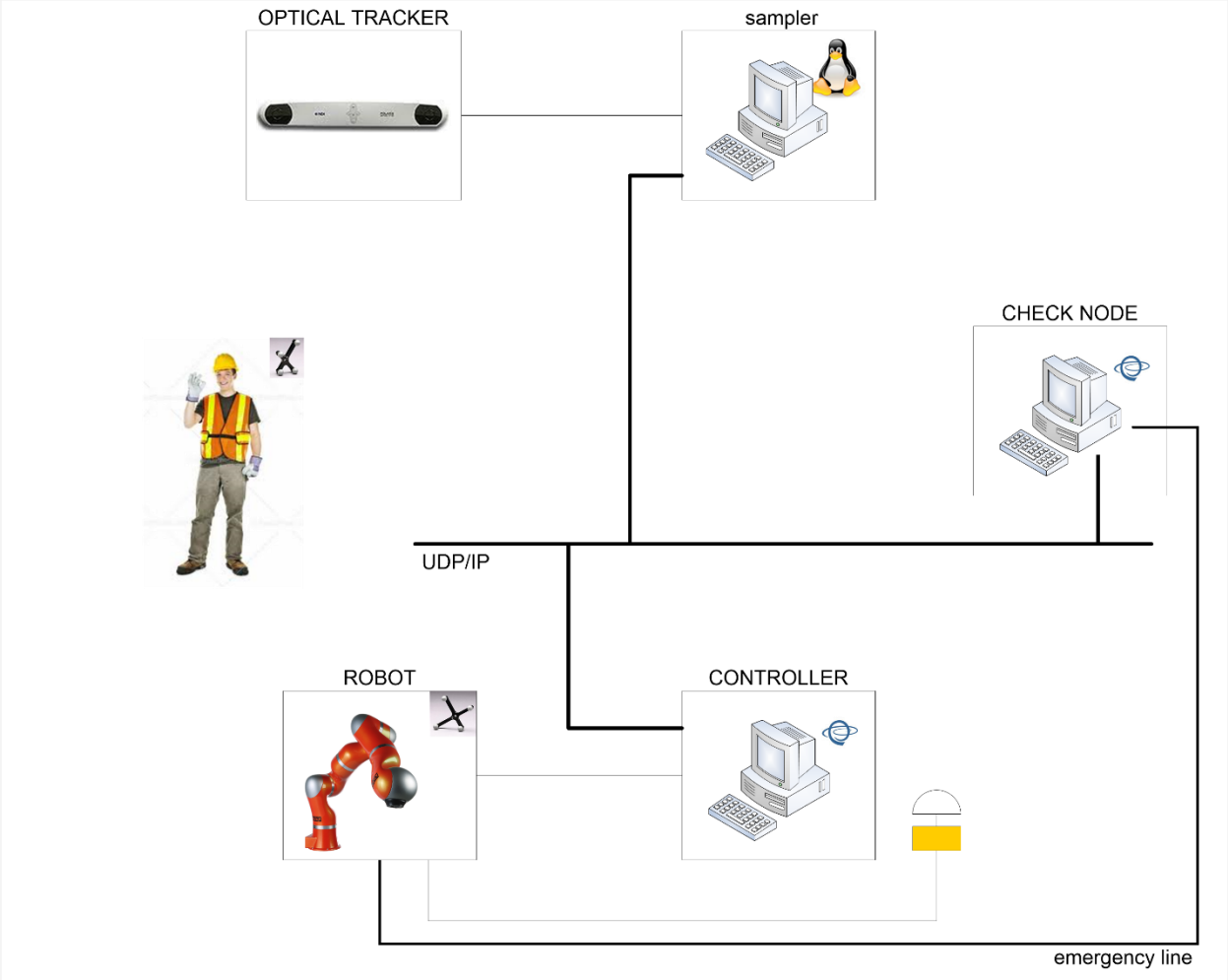


Figura 5.1 – Architettura del primo sistema fisico.

Rispetto alla Fig. 5.1 il *CheckNode* è implementato su una piattaforma Linux Real Time (Xenomai), analogamente al Robot Node, mentre il Sensor Node è implementato su una piattaforma Linux standard.

Il secondo sistema fisico è invece composto da 2 robot COMAU NS16, un sistema di sensori radio non real-time che sfruttano il principio di localizzazione radio passiva (*device free localization* o DFL).

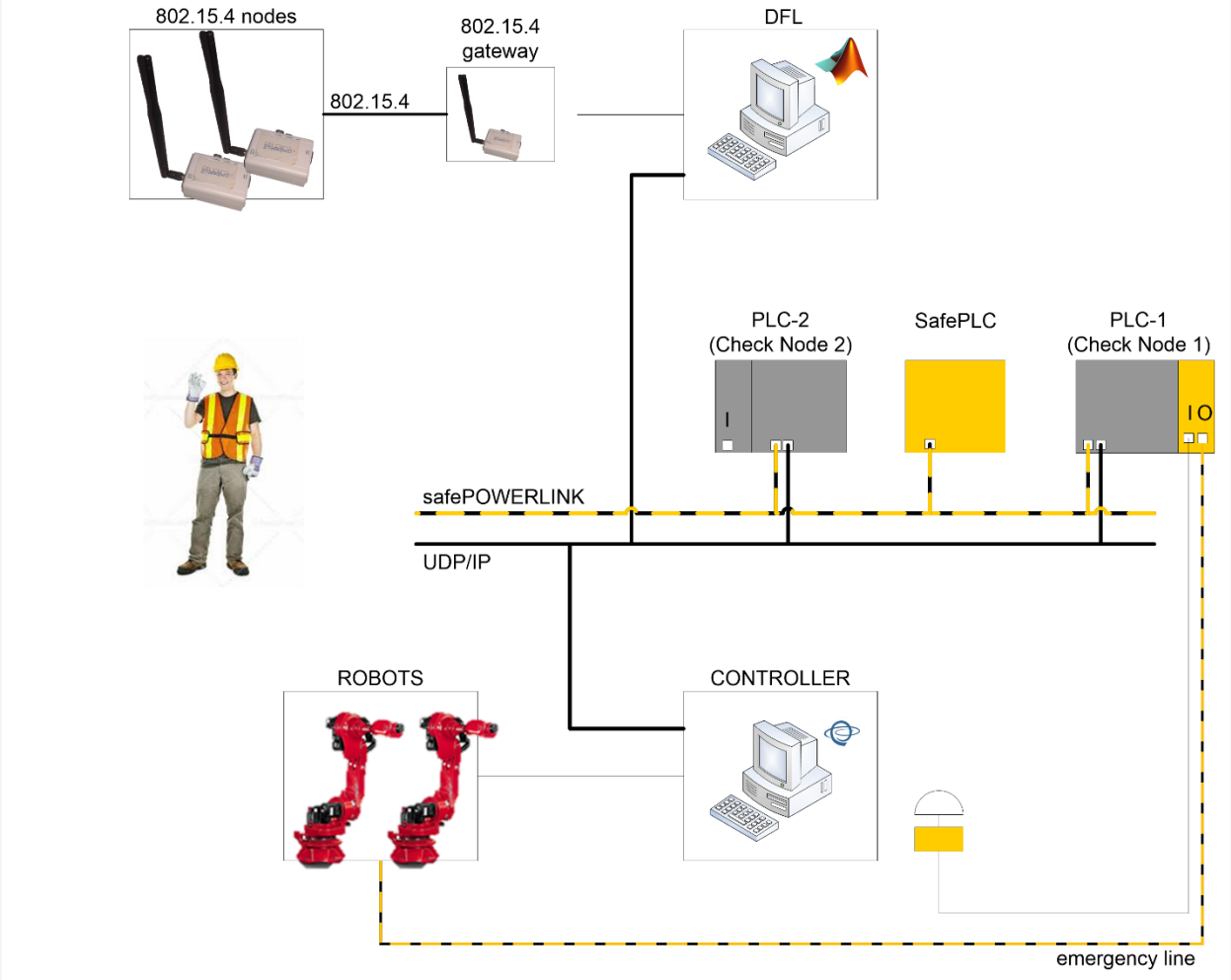


Figura 5.2 – Architettura del secondo sistema fisico.

La versione completa dei dispositivi (*Robot Node* e *Sensor Node*) integrati in una SafeNet (Vedi Fig. 5.2) sono occasionalmente sostituiti, per fini di facilità di sperimentazione, dalla architettura semplificata di Fig. 5.3.

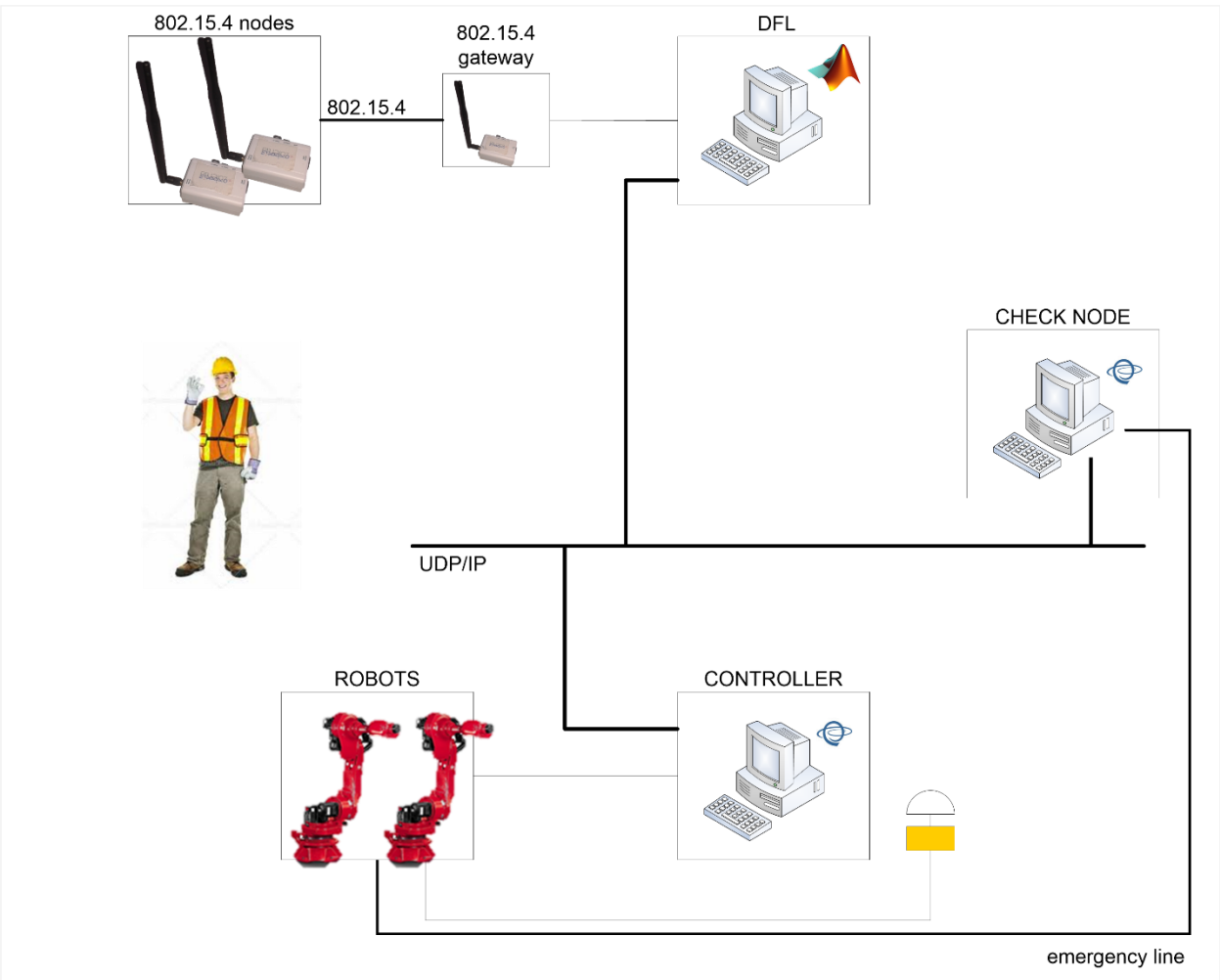


Figura 5.3 – Architettura semplificata utilizzata nel secondo sistema fisico analizzato.

Nel seguito verranno illustrati nel dettaglio i due sistemi e i test effettuati.

5.1 Sistema Prototipale

Verrà ora fatta una panoramica sugli elementi che compongono il sistema che è stato utilizzato come sistema con cui validare le simulazioni svolte in precedenza.
Saranno poi presentati i risultati delle prove sperimentali.

La configurazione del sistema è quella descritta in Fig. 5.4.

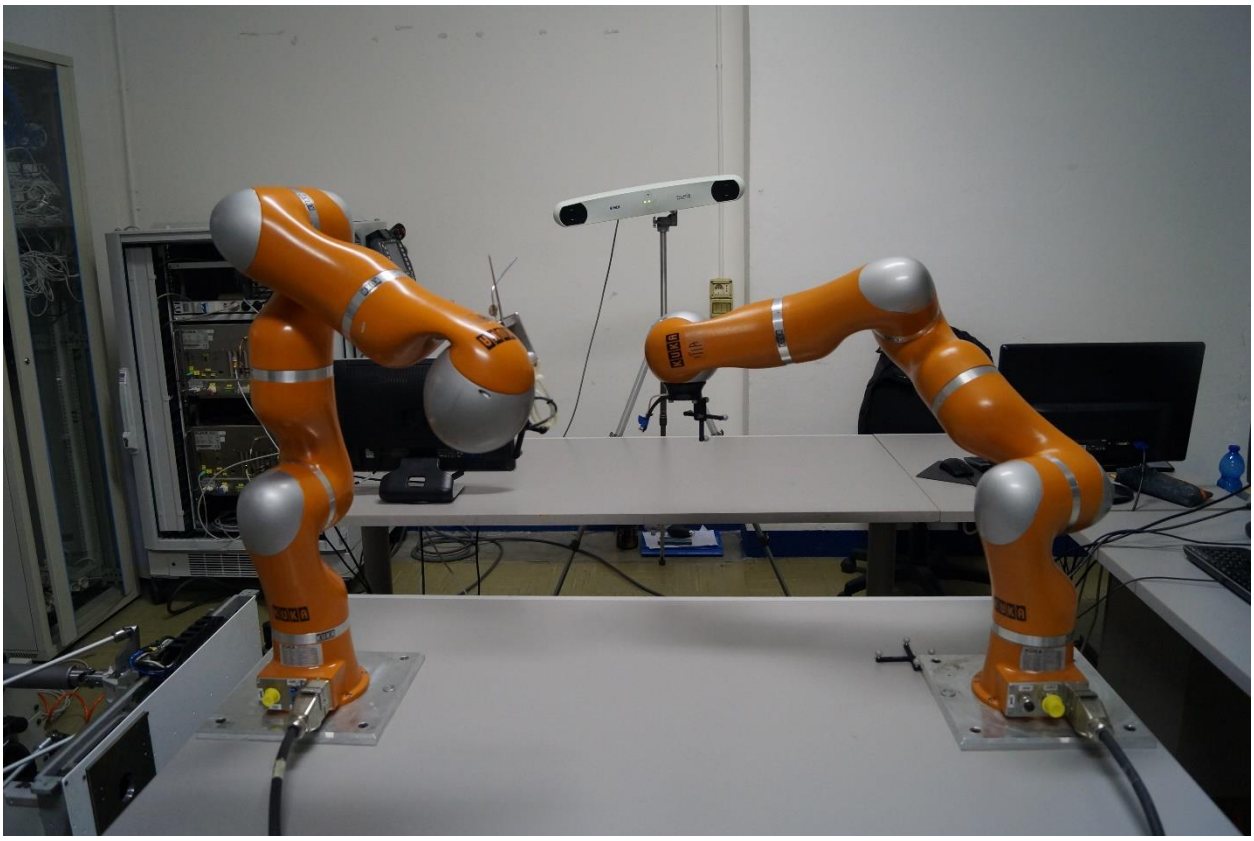


Figura 5.4 – Configurazione sistema prototipale di validazione

In tale figura sono presenti e utilizzati attivamente i seguenti dispositivi:

- un manipolatore kuka modello LWR4+;
- un sensore ottico IR (infrarossi);
- un pc embedded che funziona da nodo RN;
- un sistema real-time che funziona da nodo CN;
- un sistema non real time che funziona da nodo SN.

Il secondo robot KUKA LWR 4+, a sinistra in Fig. 5.1, è stato usato come ostacolo e quindi un utilizzo passivo.

Negli sviluppi futuri, si vuole fare qualche test con l'ostacolo mobile, percui verrà usato attivamente anche il secondo manipolatore.

Andiamo ora ad analizzare nel dettaglio le caratteristiche dei vari elementi.

5.1.1 Manipolatore Kuka LWR4+

Il Kuka LightWeight Robot (LWR) [20], [21] (si veda la Fig. 5.5), sviluppato presso l'Istituto di Robotica e Meccatronica del Centro Aerospaziale Tedesco (DLR), fa parte della nuova generazione di robot leggeri.

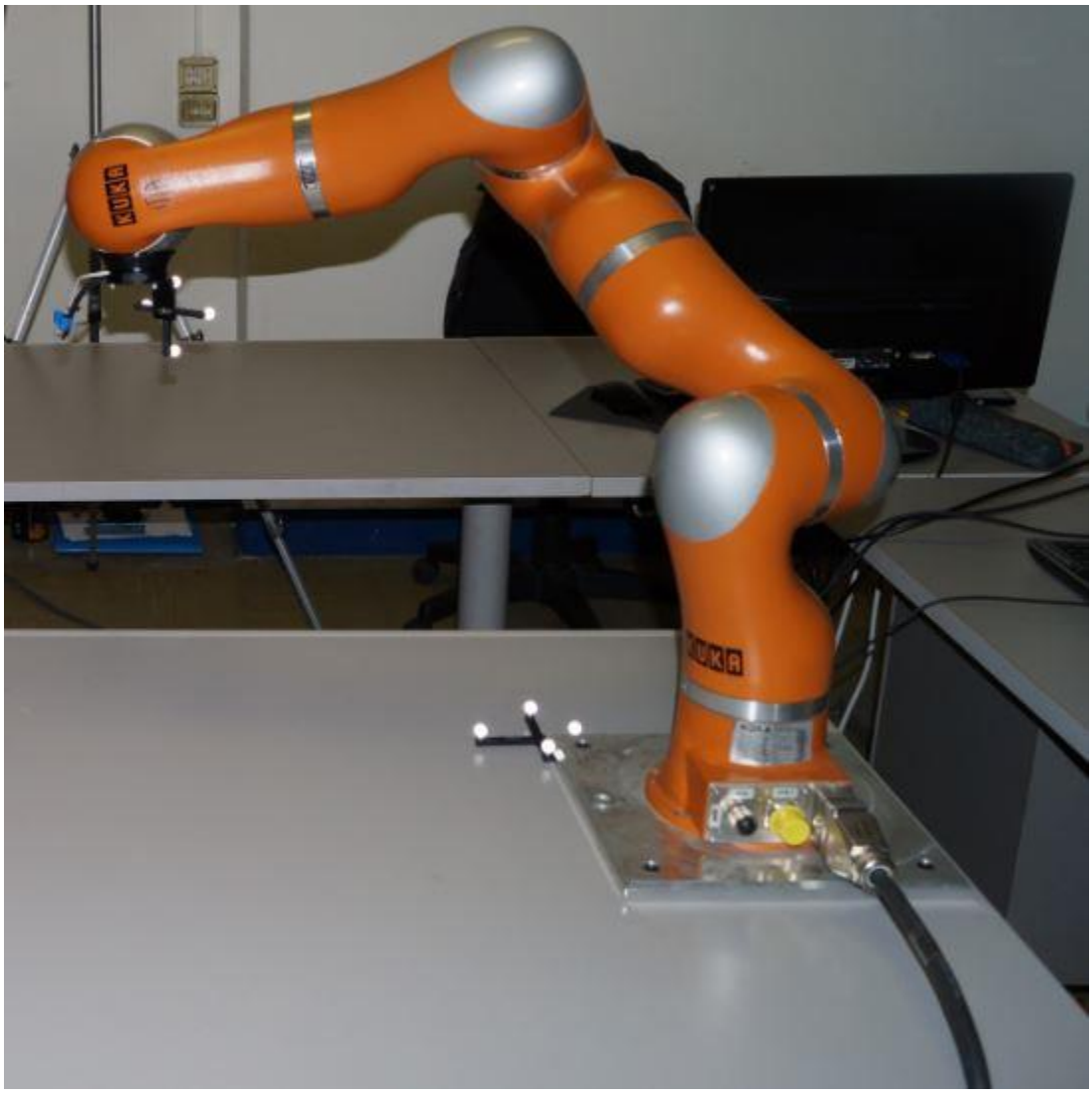


Figura 5.5 - Robot KUKA LWR 4+

Il principale vantaggi di questo manipolatore sono

- controllo veloce in impedenza al giunto, il quale consente il controllo in interazione;
- smorzamento delle vibrazioni al giunto elastico;
- rilevamento distribuito del contatto lungo l'intera catena cinematica, il quale consente al robot di reagire a forze non solo applicate all'*end-effector*;

- regolazione di soglie di impatto per garantire i requisiti di sicurezza, lato robot e lato operatori, richiesti.

Il Kuka LWR è stato ideato per la realizzazione di applicazioni profondamente differenti da quelle ottenibili con le precedenti generazioni di robot industriali.
Se i robot industriali trovano i lori punti di forza in elevate accuratezze di posizionamento, elevata velocità di esecuzione del moto, elevata robustezza associata al controllo e relativo basso costo, il LWR consente l'esecuzione di operazioni in ambienti non strutturati, in cui può avvenire l'interazione con operatori.

Proprio per questa attitudine del robot, ad essere utilizzato in spazi di lavoro condivisi le prove sperimentali sono state pensate su questa piattaforma.

Questo modello di robot mette inoltre a disposizione un metodo per la comunicazione veloce tra il controllore del robot e un pc remoto,chiamato Fast Research Interface (FRI) [26]).

Grazie alla FRI è possibile comunicare con il controllore e scambiare dati. All'interno del controllore del robot sono memorizzate una serie di variabili relative allo stato del robot, come posizioni, velocità e altre informazioni come ad esempio Jacobiano e matrici di massa.
Grazie all'FRI, si riesce ad accedere a questi dati e ad inviarli verso un'altra destinazione.

Il controllore del robot è equipaggiato con un pannello di controllo (Kuka Control Panel - KCP), che consente all'utente di interagire con il robot. Il KCP consente l'utilizzo di diverse modalità operative che permettono il test delle operazioni programmate e l'esecuzione in sicurezza.

Il controllore dispone di un linguaggio di programmazione (Kuka Robot Language - KRL) che consente di programmare l'esecuzione di un'attività. Il KCP può quindi essere utilizzato per la visualizzazione, l'esecuzione e la generazione di script KRL.

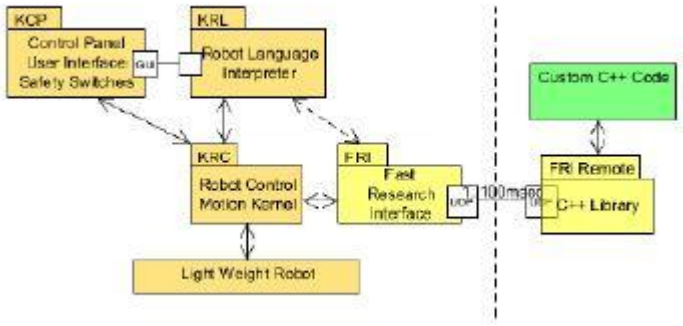


Figura 5.6 – Architettura software di controllo del manipolatore KUKA LWR 4+.

I programmi da far eseguire al robot possono essere quindi scritti direttamente nel linguaggio KRL, oppure grazie alla FRI si può creare una libreria di funzioni che interagisca con le strutture del controllore del robot.

E' proprio questa modalità che permette di ottenere le migliori prestazioni e la possibilità di realizzare molte applicazioni per il robot.

Per la realizzazione delle prove necessarie per le prove sperimentali, si è reso necessario creare ed integrare dei codici con le pre-esistenti librerie del *framework* ITIA, scritte in linguaggio c++. Queste librerie, gestiscono il collegamento con il controllore del robot permettendo di accedere ai dati. All'interno delle librerie, sono stati integrate ulteriori librerie e procedure scritte in linguaggio C++, il cui scopo è quello di controllare il robot nel modo desiderato. Uno schema rappresentante questa architettura software di controllo è rappresentata in Fig. 5.7:

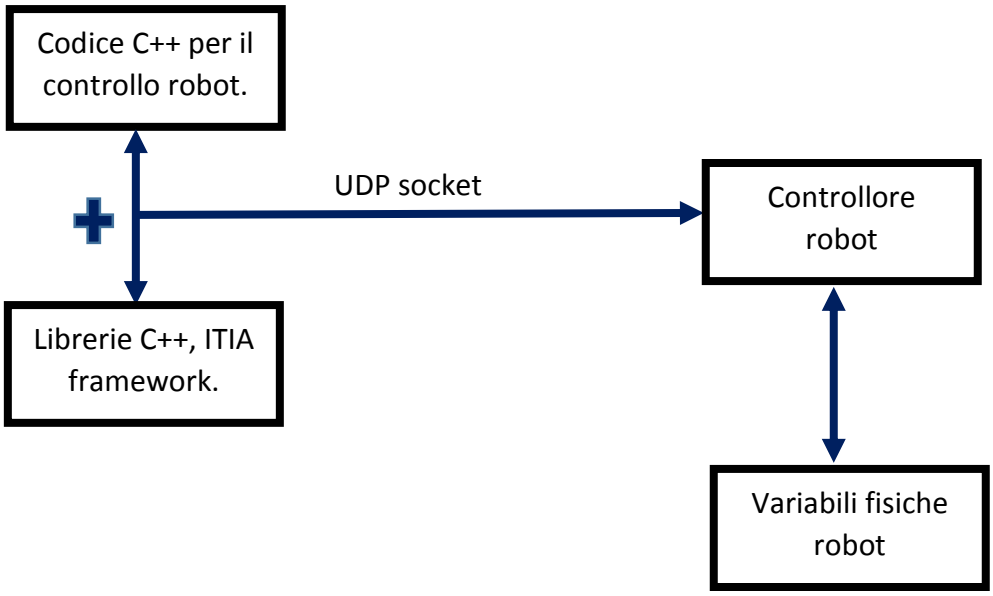


Figura 5.7 – Struttura del codice e legame con le variabili fisiche

Il punto fondamentale è che l'FRI permette controllare il robot nel modo desiderato. Sono nativamente disponibili diversi tipi di controllo[24]: impedenza ai giunti, impedenza nello spazio di lavoro e posizione. E' sfruttando il controllo di posizione che viene comandato il robot in questo caso.

Infatti ad ogni ciclo del controllore, viene misurata la posizione del robot, che è espressa in forma di vettore composto da tre componenti corrispondenti alle coordinate cartesiane dell'*end-effector* e quattro componenti che corrispondono al quaternione[27].

In questo modo si sa la posizione del robot in ogni istante. Tramite codice che viene eseguito sul pc in remoto vengono acquisiti questi dati, e vengono modificati in modo opportuno per far

compiere al robot il movimento all'istante successivo. Verrà quindi inviato al controllore del robot la posizione Comandata, cioè quella a cui il robot dovrà portarsi.

Se il controllore, mediante l'interpolatore di moto interno, valuterà che questa posizione è raggiungibile dalla posizione attuale, verrà inviato il comando effettivo al robot di compiere il movimento. In caso contrario verrà segnalato l'errore.

Questa modalità permette quindi di generare una traiettoria da far percorrere al robot in modo semplice e veloce.

Per fare in modo che siano garantite tutte le misure di sicurezza e sia rispettata l'architettura del controllore è necessario avere una macchina a stati finiti che supervisioni il controllo del robot attraverso pc remoto. Il codice che è stato scritto per questo controllo è quindi un programma che apre più processi (*thread*) per l'esecuzione contemporanea di diversi compiti. In particolare è necessario avere un processo che gestisca le operazioni di controllo del robot, con la determinazione delle coordinate da inviare alla posa comandata successiva, un processo che gestisca le comunicazioni con il CN e infine un processo che gestisca le richieste di comandi in arrivo dall'esterno. Ognuno di questi *thread* avrà quindi una funzione dedicata che ne gestisce e specifica il funzionamento. Il meccanismo per lo scambio dati è ottenuto mediante la creazione di un socket di rete.

Il grande vantaggio di questa procedura *multi-threaded* è data dal fatto che, in fase di inizializzazione, è possibile creare *thread* che condividano una struttura dati e quindi è possibile rendere noti a tutti i *thread* in esecuzione determinate informazioni. Bisogna prestare molta attenzione alla condivisione di queste strutture dati, in quanto bisogna garantire che siano condivise sì, ma utilizzabili solo un *thread* alla volta.

Per realizzare questa condivisione è quindi necessario disporre di un meccanismo di mutua esclusione che garantisca l'accesso ad un solo *thread* alla volta alla struttura dati condivisa.Per fare ciò sono state utilizzate delle logiche di mutua esclusione , le quali proteggono i dati e garantiscono l'impossibilità alla modifica/accesso mentre un dato è letto/scritto da un *thread*.

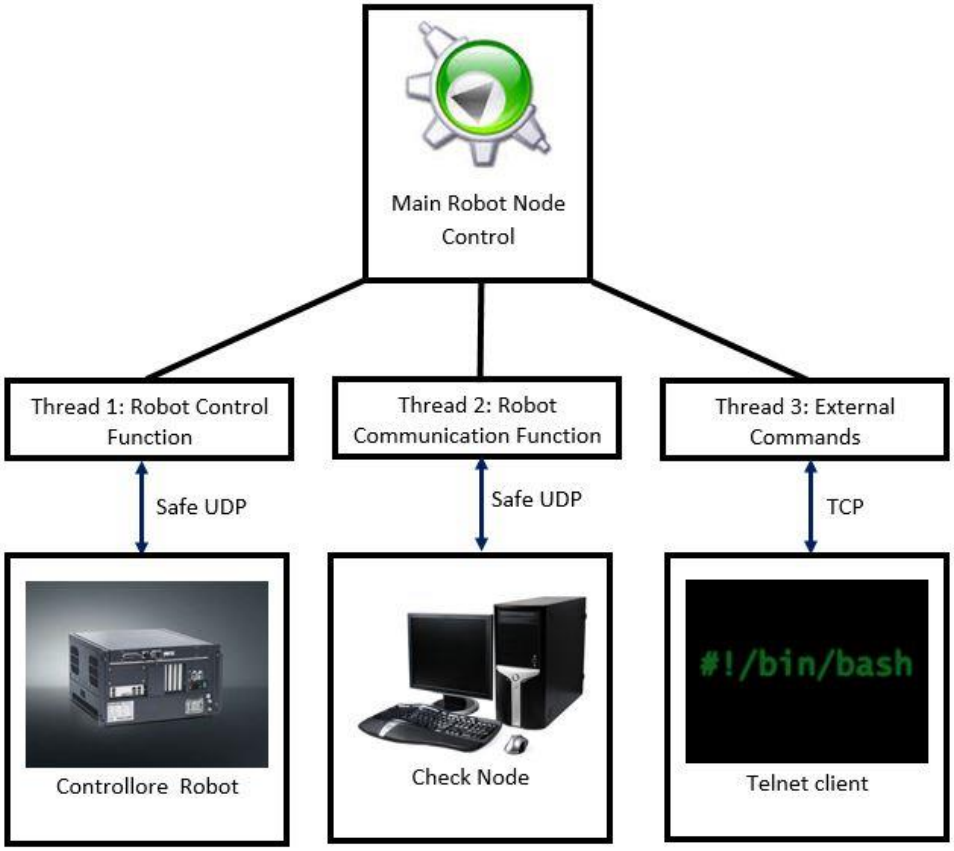


Figura 5.8 – Rappresentazione della struttura multi-threading parallela del main

La struttura dati condivisa prenderà il nome di *CNdata* ed avrà al suo interno i seguenti campi:

- robotTcpPose (vettore di 7 elementi composto da componenti cartesiane e quaternione relativo all'end-effector);
- robotElbowPose (composizione uguale al precedente ma relativo alla posizione del gomito del robot);
- markerPose1 (vettore costruito come i precedenti relativo al primo elemento tracciato dal sensore);
- markerPose2 (vettore costruito come i precedenti relativo al secondo elemento tracciato dal sensore);
- markerPose3 (vettore costruito come i precedenti relativo al terzo elemento tracciato dal sensore);
- Altre variabili di servizio;

Quindi quando i thread verranno creati verrà passato a loro il puntatore della struttura dati CNdata condivisa che sarà gestita mediante un meccanismo di mutua esclusione.

5.1.2 Sensore IR NDI Polaris Spectra

Figura 5.9 – Sensore NDI Polaris Spectra

Il sensore utilizzato per la fase di prototipazione della soluzione è un sensore ottico a infrarossi (Polaris Spectra System) che permette di misurare la posizione 3D di un marcatore (*marker)*, attivo o passivo (Vedi Fig. 5.9).

Lo strumento è molto preciso, ripetibile e accurato. Permette di effettuare misurazioni di posizione con errore dell'ordine di due decimi di millimetro. L'utilizzo di questo strumento è stato importante in quanto presenta una sensibilità, precisione ed accuratezza molto più elevate rispetto agli altri sensori poi utilizzati, permettendone così la loro calibrazione.

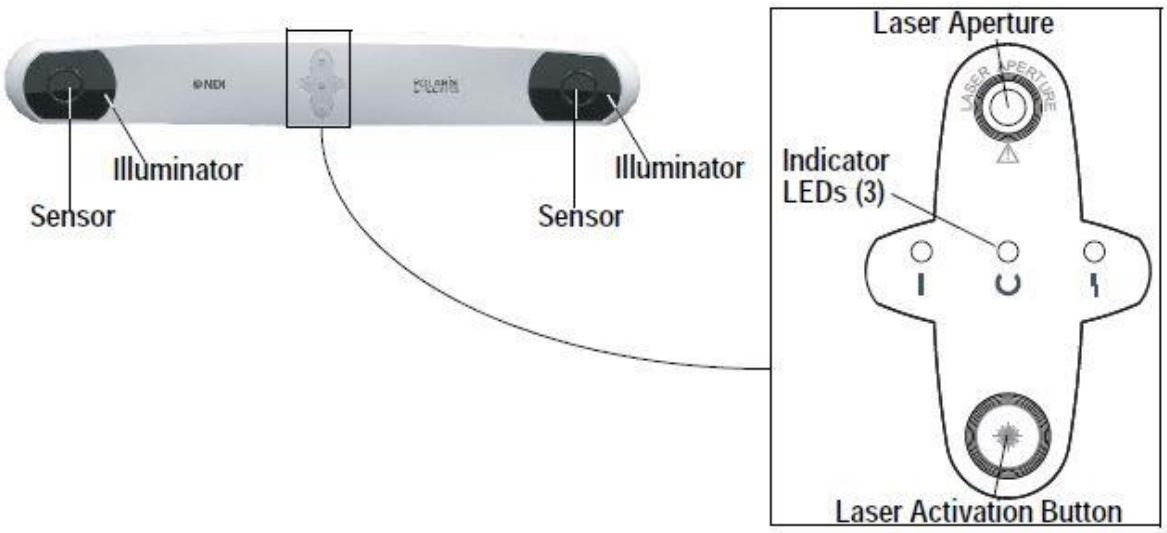


Figura 5.10 – Elementi che compongono il sensore

Come si vede dall'immagine mostrata in Fig. 5.7 il sensore è composto da 2 sensori a infrarossi, 2 illuminatori a infrarossi e da un puntatore laser che aiuta a capire la zona inquadrata dagli infrarossi.

Usando le informazioni ricevute il sistema di misurazione è in grado di determinare posizione ed orientamento dell'oggetto in un determinato volume di lavoro (si vedano le Fig. 5.8 e 5.9).

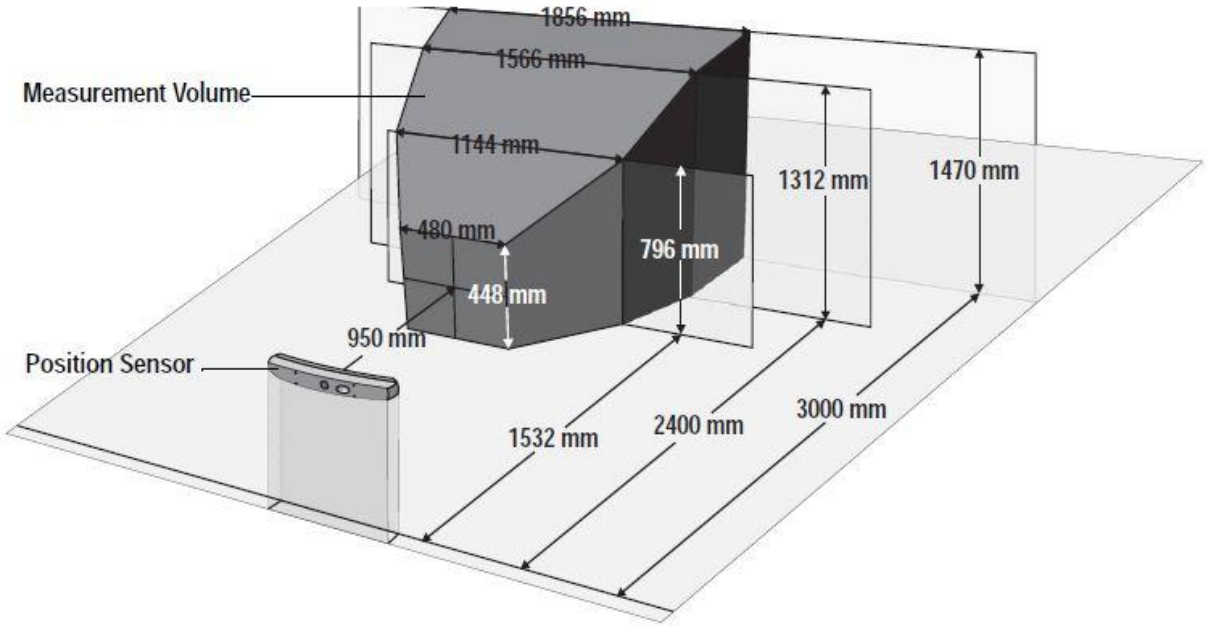


Figura 5.11 – Volume di misurazione

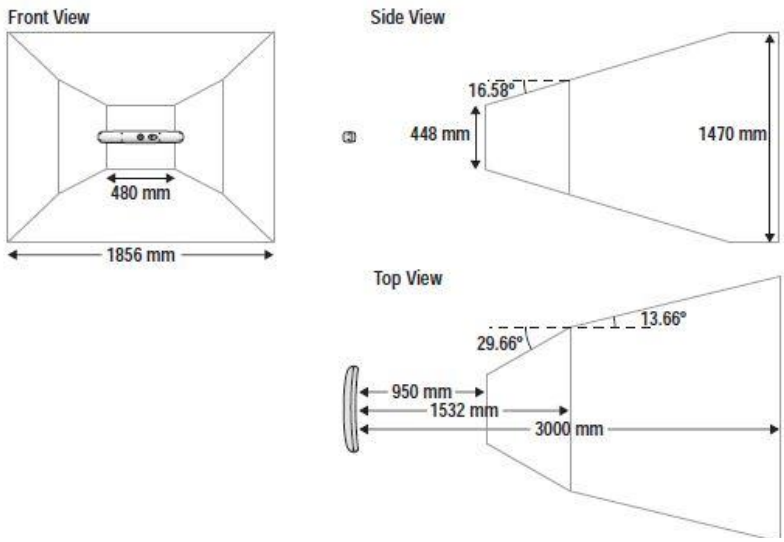


Figura 5.12 – Viste del volume di misurazione

Le coordinate e l'orientamento del *marker* vengono identificate grazie alla triangolazione delle immagini catturate dalle due telecamere ad infrarossi (si veda la Fig. 5.13). L'oggetto viene illuminato attraverso degli illuminatori posti a fianco delle telecamere, in questo modo si riesce ad evitare il disturbo che potrebbe essere introdotto da luci esterne o cattiva illuminazione.

Figura 5.13 – Triangolazione telecamere a infrarossi

I *marker*, oggetti riflettenti passivi o attivi, possono essere di tre tipi:

- sferette riflettenti passive di materiale metallico;
- corpi rigidi, di geometria ben definita, aventi alle estremità sferette del tipo precedente;
- *marker* attivi che inviano segnali di posizione al sensore.

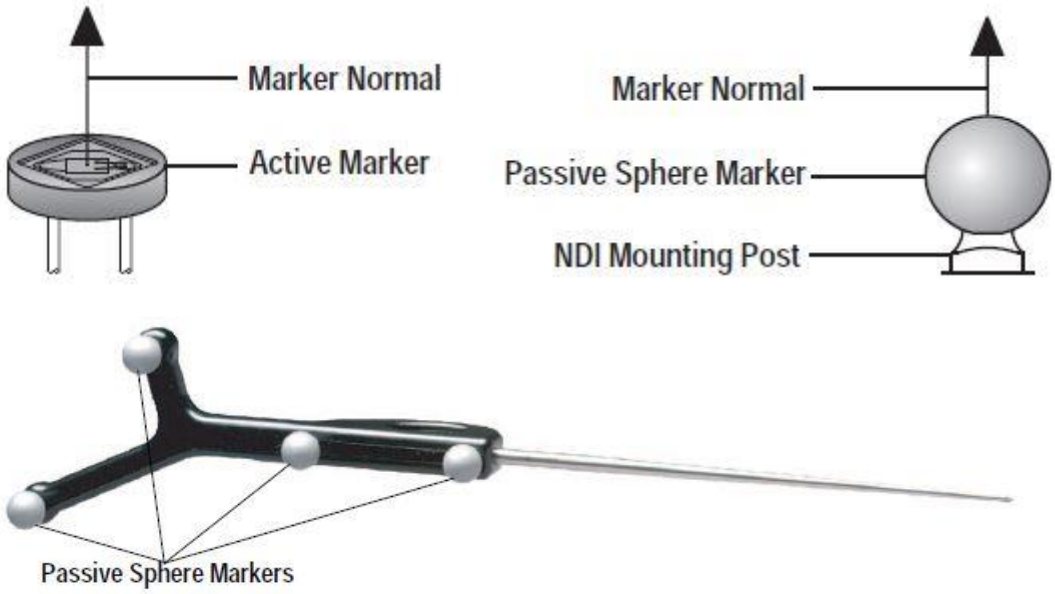


Figura 5.14 – Tipologie differenti di markers

Per gli esperimenti condotti, sono stati utilizzati corpi rigidi con sfere passive fissate alle estremità. In particolare sono utilizzati 3 *marker*, ognuno con quattro sfere.

Il sistema di misura, ha in memoria le geometrie dei corpi rigidi: queste geometrie sono uniche e quindi l'identificazione è univoca, senza possibilità che un oggetto venga scambiato con un altro.

Nell'appendice B vengono riportate ulteriori specifiche sul sensore e le applicazioni scritte per poterlo utilizzare nelle modalità desiderate.

Una delle importanti caratteristiche che nativamente possiede questo sensore è la sua frequenza di acquisizione variabile che permette di avere dei tempi di aggiornamento del dato diversi, in modo da poter analizzare la sensitività del sistema a questa quota parte di tempo.

Inoltre è possibile far funzionare il sensore con diverse velocità di scambio dati con il Sistema Linux che lo controlla: in questo modo si può far variare ulteriormente la velocità di campionamento del sensore.

5.1.3 Check Node e applicazione dell'algoritmo

Verrà ora illustrata l'applicazione descritto nel capitolo 4, mediante l'uso degli strumenti sopra indicati.

Anche in questo caso l'infrastruttura del codice realizzato è simile a quelle precedenti, in cui si ha un programa principale (*main program*) che gestisce più processi, in particolare, in questo caso si tratta di tre processi. Due di essi avranno il compito di gestire la comunicazione rispettivamente con robot e sensore, il terzo sarà dedicato al calcolo dello spazio minimo di sicurezza da utilizzare.

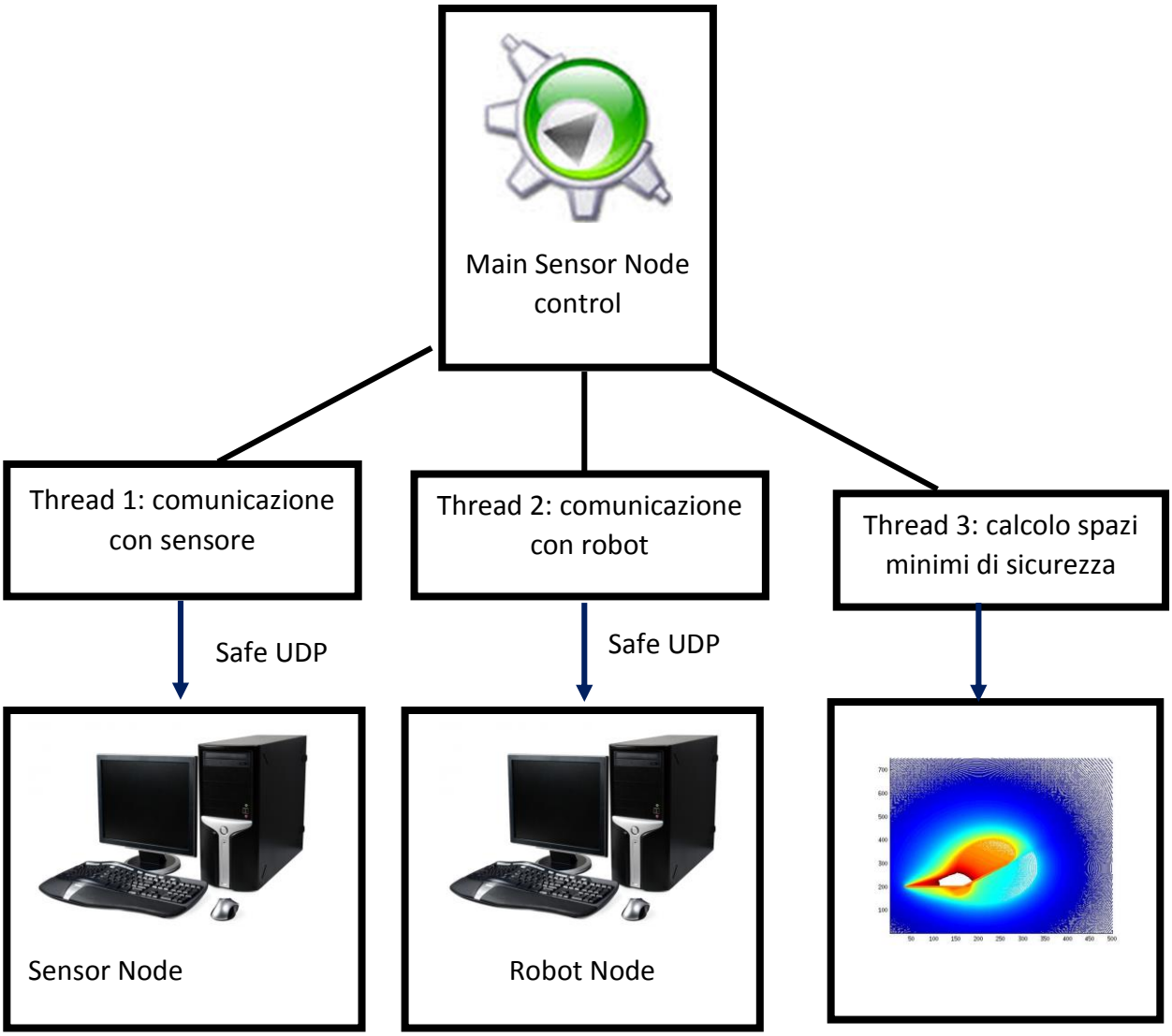


Figura 5.15 – Infrastruttura del codice realizzato per il Check Node

Fondamentale quindi, anche qui, una struttura dati condivisa, protetta con meccanismi di mutua esclusione in cui scrivere i dati provenienti dall'esterno e i risultati delle computazioni.

La prima attività che è stata svolta è quella di sincronizzazione dei clock dei vari nodi connessi in rete. In questo modo tutte le grandezze sono riportate sula stessa base temporale.

Dopo questa fase in cui sono state calcolati gli offset, è stata condotta una lunga campagna di acquisizioni, relativa ai vari tempi che andranno a stimare il valore di τ che è utilizzato per il calcolo di d_t^{safe}. (si veda formula 13).

Le prove sono state condotte facendo variare i tempi di acquisizione del sensore e il tempo-ciclo del controllore del robot. In questo modo si ha una buona sensitività su quali termini vadano ad incidere maggiormente sul risultato finale.

Le frequenze nominali con cui sono state fatte le acquisizioni dal sensore sono le seguenti:

1. 3 Hz → 333.3 millisecondi;
2. 13 Hz → 76.9 millisecondi;
3. 20 Hz → 50 millisecondi;
4. 30 Hz → 33.3 millisecondi;

Le frequenza nominale con cui viene eseguito il ciclo di controllo del controllore del robot sono invece le seguenti;

A) 200Hz → 5 millisecondi;
B) 66 Hz → 15 millisecondi;

Sono state monitorate attraverso opportuni contatori e *timestamp* le singole componenti temporali nelle varie prove.

I risultati riportati in Tab. 5.1 indicano i valori stimati per tau (ultima colonna a destra) insieme con i singoli contributi. I valori riportati sono dei valori medi in quanto per ottenere questi valori sono stati acquisisti diecimila valori, che sono stati successivamente mediati e di cui si è calcolato il range di scostamento massimo.

†	T^R_{proc}	T^R_{TX}	T^S_{proc}	T^S_{TX}	T_{offset}	τ
A.1	5.41±ϵ	0.015± 0.001	273.23± 0.012	0.014± 0.001	5.22	273.1
A.2	5.23±ϵ	0.019± 0.001	77.79± 0.013	0.015± 0.001	5.14	77.7
A.3	5.00±ϵ	0.023± 0.001	50.31± 0.014	0.022± 0.001	4.87	50.2
A.4	5.10±ϵ	0.021± 0.002	33.02± 0.014	0.020± 0.001	5.51	33.4
B.1	16.00±ϵ	0.015± 0.001	274.55± 0.014	0.016± 0.001	5.20	263.8
B.2	15.49±ϵ	0.015± 0.001	77.94± 0.0112	0.026± 0.002	5.46	67.9
B.3	15.86±ϵ	0.019± 0.001	51.55± 0.014	0.025± 0.002	5.34	41.0
B.4	15.41±ϵ	0.020± 0.002	32.11± 0.013	0.014± 0.002	5.20	21.9

Tabella 5.1 – Tabella dei tempi componenti τ in ms.

Questo intervallo è espresso come ε ed è contenuto in un intervallo pari a

$$\varepsilon = [8.2 \cdot 10^{-6}, 9.0 \cdot 10^{-6}]$$

dove:

- T^R_{proc} è il tempo effettivo del ciclo di controllo del robot;
- T^R_{tx} è il tempo di trasmissione del canale di comunicazione del robot;
- T^S_{proc}è il periodo effettivo di campionamento del sensore
- T^S_{TX}è il tempo di trasmissione del canale di comunicazione del sensore;
- T_{offset} è lo sfasamento del clock relativo tra il nodo sensore e il nodo robot.

Tutte le grandezze riportate sono espresse in [ms].

In questo caso, una volta ottenuti i dati è stata verificata la gaussianità dei termini temporali. Infatti i termini temporali sono considerabili come delle variabili casuali, necessitano quindi di una verifica per stabilire qual è la loro funzione di distribuzione.

Questa verifica è stata fatta una volta ottenuto un campione di dati rappresentanti la variabile casuale in questione.

E' stato scelto di applicare il test di gaussianità di Shapiro-Wilk[43] in quanto in letteratura viene indicato come uno dei test più potenti per effettuare la verifica di gaussianità.

Il test statistico prevede l'accettazione dell'ipotesi nulla (variabile casuale in esame distribuita normalmente) quando assume valori lontani dallo zero.

Questo test può essere fatto a diversi livelli di significatività, in questo caso si è considerato un $\alpha = 1\%$ (ovvero con confidenza pari al 99%).

Il test ha dato esito positivo per tutte le variabili su cui è stato utilizzato, pertanto le variabili temporali sono considerabili come variabili normalmente distribuite.

Come si può notare dai valori numerici ricavati in tutte le prove, il termine dominante è costituito dal tempo di acquisizione del sensore, considerati gli opportuni offset.

Il risultato ha un importante senso fisico, in quanto più è grande il periodo di campionamento, più il sistema resta cieco e quindi la penalità che deve essere tenuta in considerazione deve essere grande. Se si avesse invece un sistema di acquisizione delle posizioni con un periodo di campionamento molto piccolo, il termine τ sarebbe dominato dai tempi di trasmissione, e quindi la penalità spaziale sarebbe trascurabile.

Una volta completata questa fase, sono state effettuate diverse prove per misurare lo spazio di frenata del robot in una data configurazione a diverse velocità. Sono per ora stati indagati gli spazi di frenata relativi a tre differenti configurazioni (quelle utilizzate per le verifiche sperimentali). In ogni caso, sarebbe opportuno caratterizzare tali spazi di frenata per le varie configurazioni ammissibili dal robot.

I risultati relativi a due differenti velocità e a tre configurazioni sono indicato in Tab. 5.2.

	v_R	
	0.5 m/s	1.0 m/s
configuration 1	0.0150± 0.010	0.035± 0.02
configuration 2	0.0170± 0.013	0.042± 0.04
configuration 3	0.0196± 0.011	0.033± 0.03

Tabella 5.2 – Spazi di frenata del robot in funzione delle differenti configurazioni.

I risultati sono riportati in metri e sono anch'essi valori medi, ottenuti su venti prove distinte.

Questo risultato è riportato in questa sezione perché gli spazi di frenata sono stati misurati attraverso il sistema di visione ottico, sfruttando così la sua alta precisione.

Una volta ottenute queste grandezze, vengono monitorate costantemente durante l'esecuzione del programma i termini temporali, per determinare eventuali errori e malfunzionamenti, oppure semplicemente per aggiornare le informazioni raccolte dalla rete.

Il successivo passo è stato quello di tradurre il codice utilizzato nelle simulazioni del capitolo 4 in una classe C++, che implementi il calcolo del *danger field*.

L'algoritmo utilizzato è sempre quello presentato nel capitolo 4: sono state implementate le funzioni di calcolo del *danger field* relativo all'ostacolo e al robot. In questo caso non si è considerato uno spazio di lavoro 2D ma un volume di lavoro 3D. Il calcolo, è stato effettuato per i valori puntuali della posizione dell'ostacolo e del robot, tenendo conto i parametri di distanza minima calcolati sulla base dei tempi e dello spazio di frenata, oltre che delle velocità.

Per ciò che riguarda il sistema software, il *thread* che si occupa del calcolo delle soglie, sarà un processo realizzato in tempo reale che sfrutta un ciclo di calcolo pari a 5ms. Una volta calcolate le soglie ed effettuato il confronto, viene memorizzata in una variabile il risultato.
Il *thread* che comunicano con gli altri nodi, leggeranno questa variabile e invieranno un pacchetto *status* di risposta ai nodi, come descritto in Fig. 5.16.

Se il contenuto dello status è *safe_condition* allora il moto del robot continuerà sulla sua traiettoria. Se viceversa il contenuto dello status è *unsafe_condition* allora verrà applicata una politica di *collision avoidance*, facendo variare la traiettoria del robot. Si cercherà di allontanare il più possibile il robot dall'ostacolo, in modo che la soglia ritorni superiore al valore attuale di danger field, che tradotto in termini spaziali significa che l'ostacolo esce dalla zona di pericolo tracciata attorno al robot.

La traiettoria che il robot deve compiere è una traslazione lungo un asse, da un punto di partenza a un punto d'arrivo. L'ostacolo in questa prova è mantenuto fermo, quindi non possiede velocità.

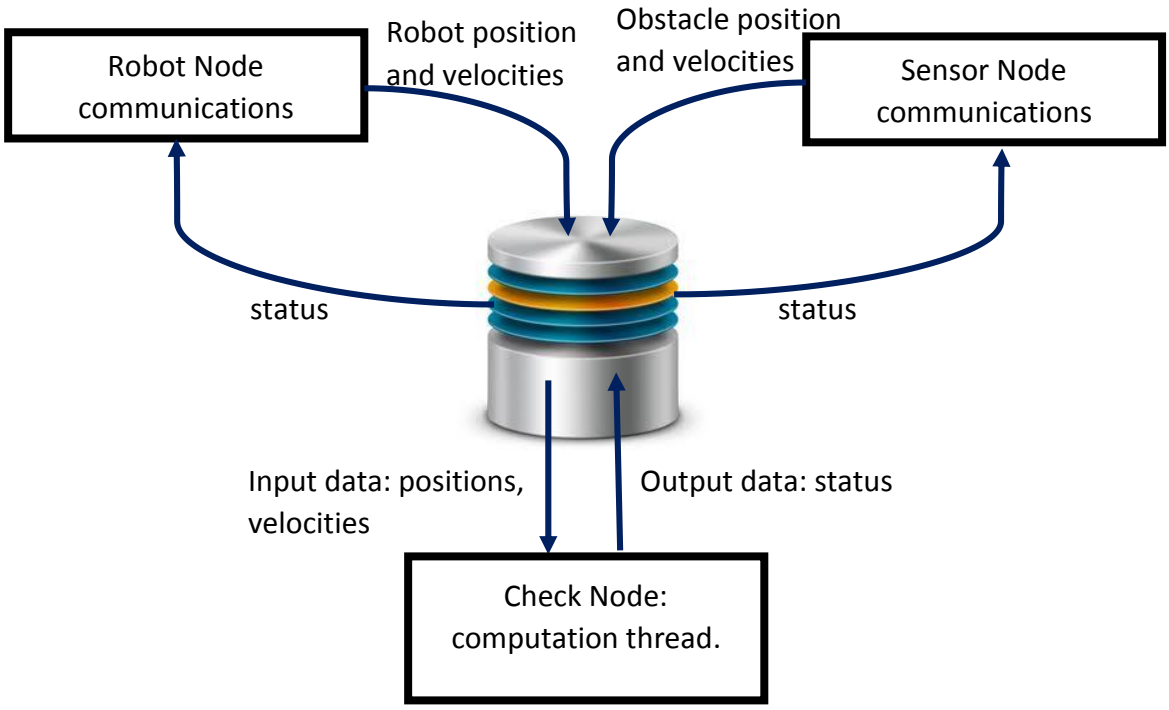


Figura 5.16 – Input e Output data

Il robot, quando riceve lo status *unsafe_status,* cambierà traiettoria, andando a muoversi lungo un altro asse per un dato intervallo di tempo. Anche durante questo movimento vengono monitorati il valore di *danger field* associato all'ostacolo e quello di soglia.

Quando il valore risulta essere inferiore del 5 % della soglia allora il robot tornerà ad eseguire la sua traiettoria fino a quando lo status che riceve è *safe_status*. Nel caso ricevesse altri messaggi di tipo *unsafe_status*, si riapplicherebbe il cambio di moto.

Una volta arrivato al termine del movimento prefissato, a causa delle deviazioni di traiettoria, il robot potrebbe essere ad una quota diversa da quella desiderata.
Si esegue quindi un movimento opposto a quelli introdotti dall'algoritmo di *collision avoidance*, che riporta il robot alla quota desiderata, sempre se il valore monitorato resta sotto la soglia.

Questo esperimento è stato condotto tre volte, in tre condizioni differenti:

- Sensore e Robot alla massima frequenza di campionamento/ controllo;
- Sensore e Robot lenti, quindi con basse frequenze di aggiornamento dati;
- Sensore e Robot veloci, utilizzo di una soglia statica di distanza.

Si riporta l'andamento della quota Z del moto del robot, in quanto variazioni di questa quota implicano che le politiche di *collision avoidance* sono state applicate, quindi si è entrati con l'ostacolo nello spazio minimo di sicurezza del robot.

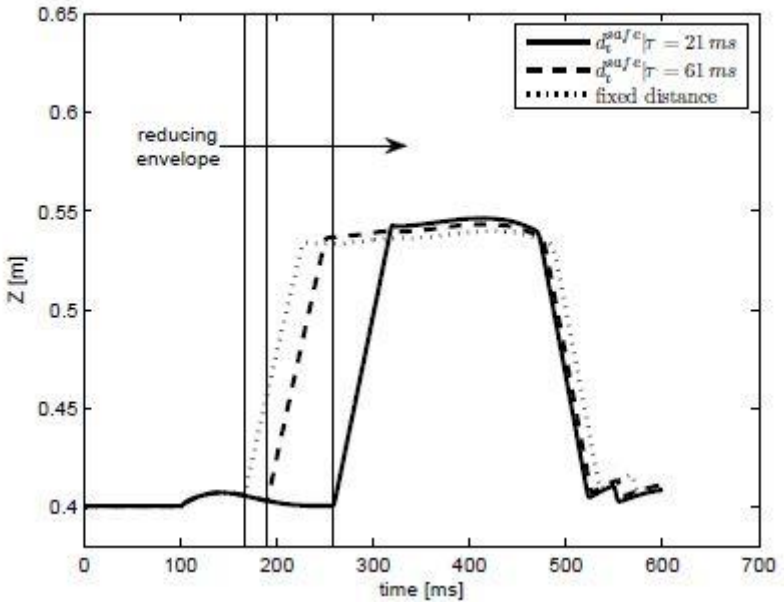


Figura 5.17 – Quota Z del movimento del robot nelle diverse condizioni

Si nota che la deviazione dalla traiettoria originale avviene dopo, quindi sempre più prossima all'ostacolo, se si utilizzano delle soglie dinamiche.

In particolare si vede come la condizione di robot e sensore caratterizzata da alta frequenza di campionamento permette di minimizzare questo spazio.

Sono in seguito riportate alcune immagini significative estratte dal video eseguito durante la prova.

Essendo state memorizzate posizioni e velocità del robot, è stata calcolata con gli strumenti sviluppati in precedenza, la bolla di sicurezza attorno al robot reale.

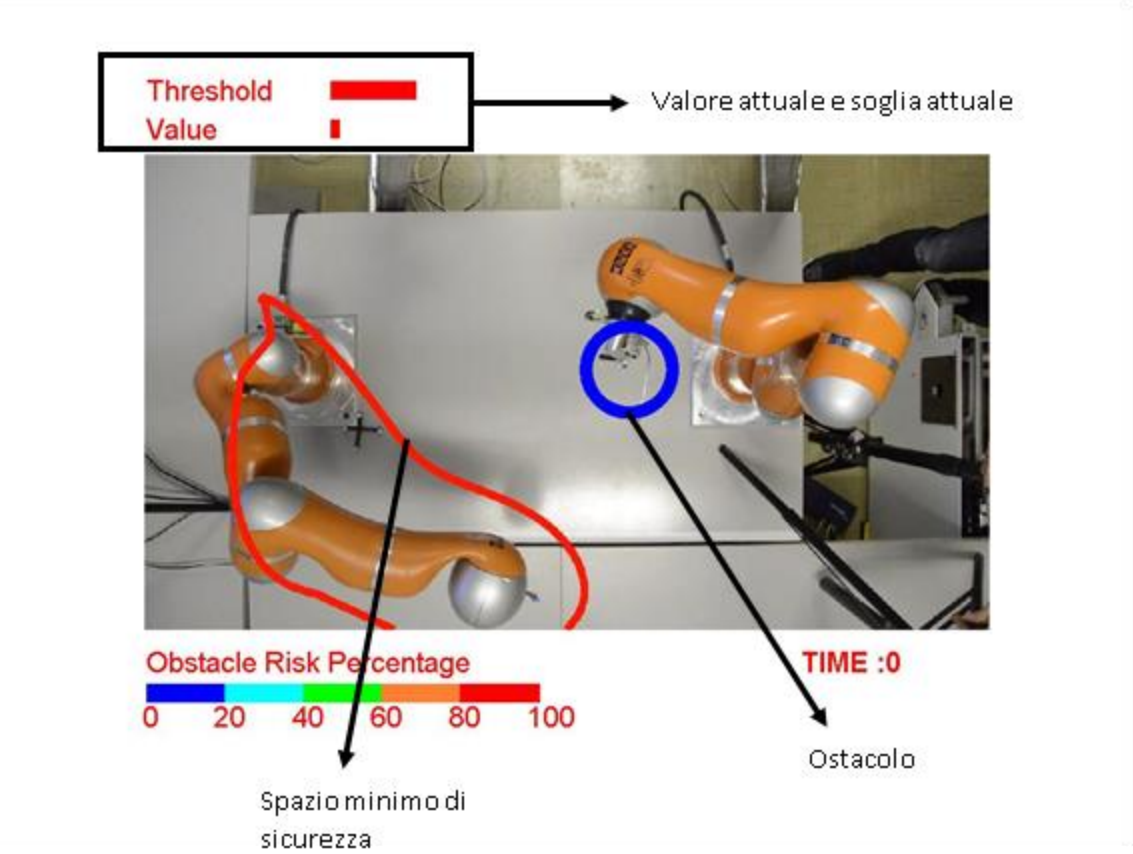


Figura 5.18 – Descrizione elementi grafici presenti nel video del test sperimentale

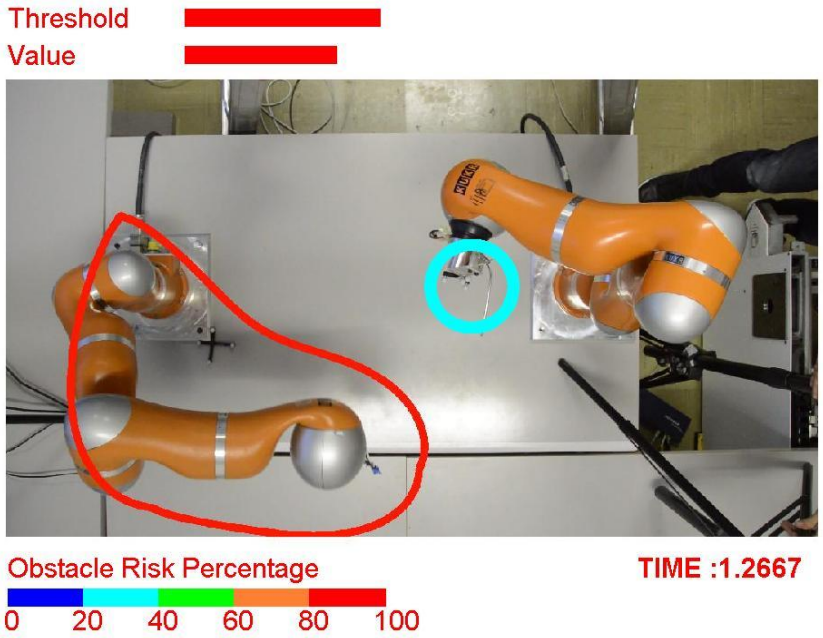
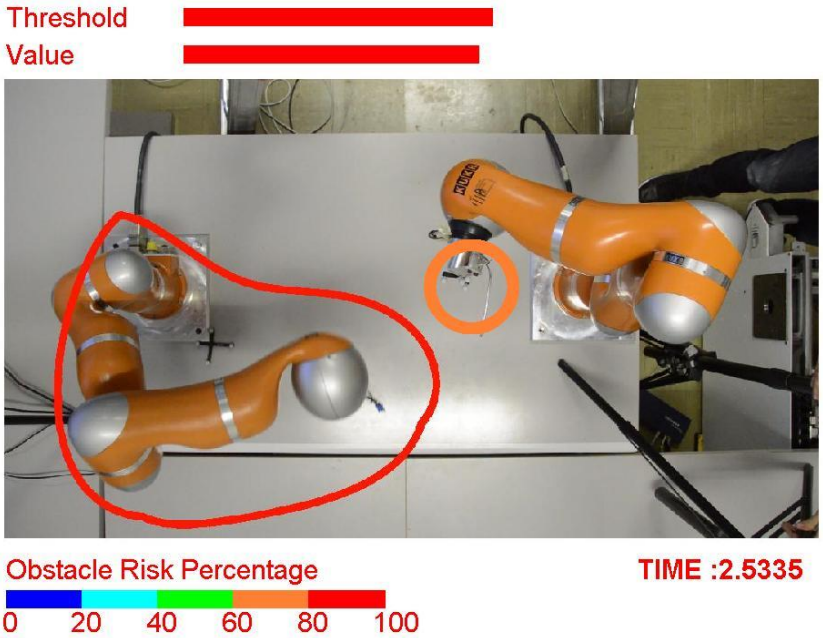


Figura 5.19-a,b – Fotogrammi estratti da video test sperimentale



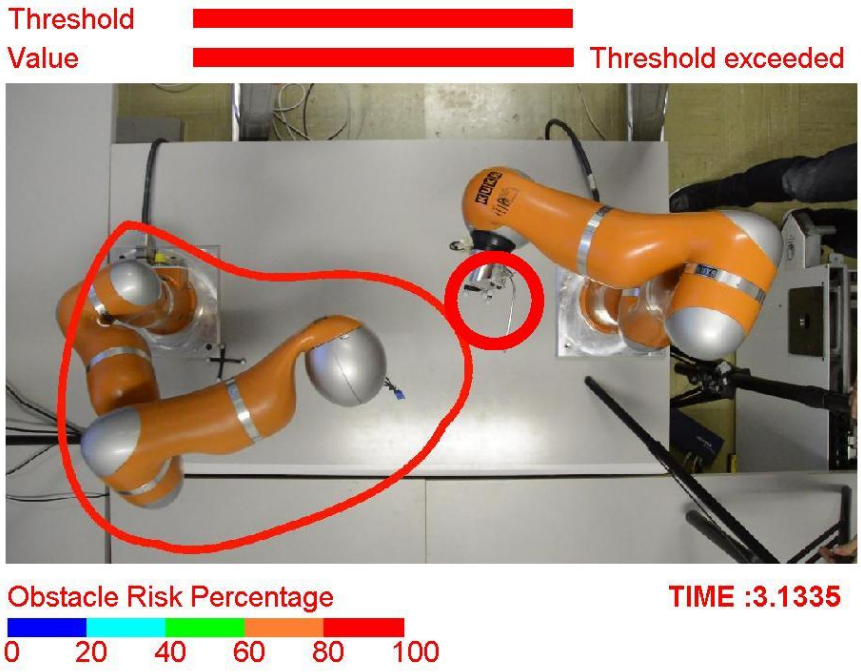
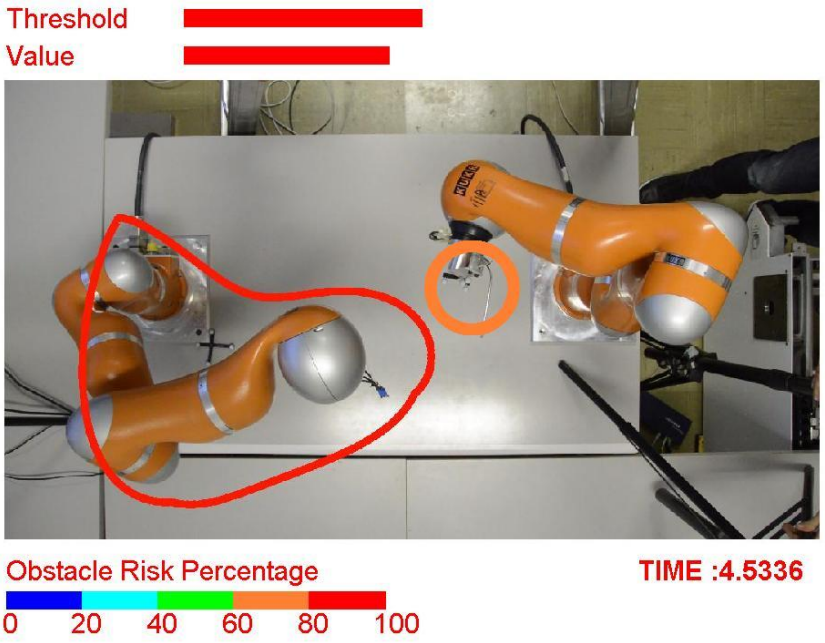


Figura 5.19-c,d – Fotogrammi estratti da video test sperimentale



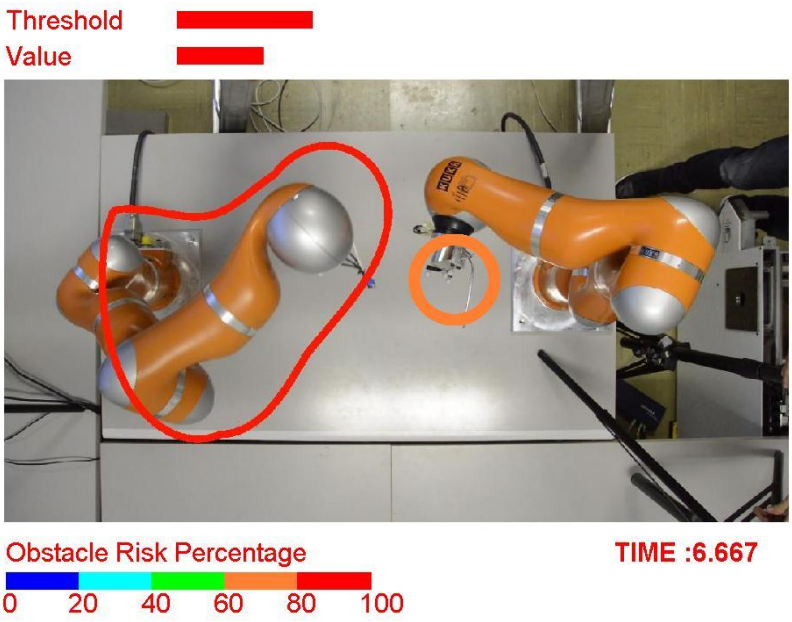


Figura 5.19-e,f – Fotogrammi estratti da video test sperimentale

5.2 Impianto Pilota

Il secondo ambiente, su cui sono stati effettuati i test sperimentali per il calcolo dello spazio minimo di sicurezza, è l'Impianto Pilota presso l'istituto ITIA-CNR (si veda la figura 5.20)

Di questo impianto è stata analizzata la prima cella e le sue zone adiacenti.

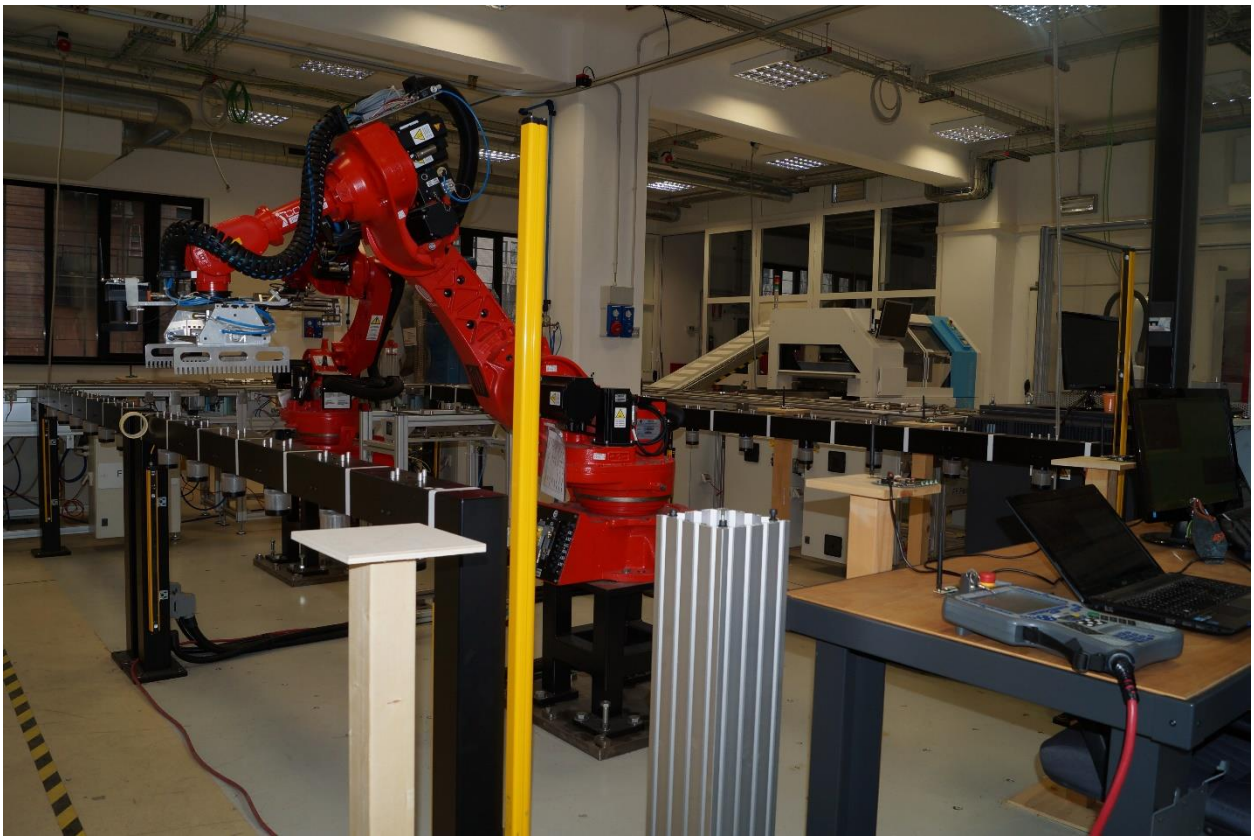


Figura 5.20 – Impianto pilota, laboratori ITIA-CNR

Cambiano gli oggetti fisici che compongono il sistema ma gli algoritmi e le soluzioni adottate sono gli stessi di quelli utilizzati nella soluzione precedente. In questo caso il sistema sarà composto da:

- robot COMAU;
- sensori radio per la localizzazione mediante tecniche DFL;
- computer per controllo Comau con ruolo di Robot Node;
- computer per comunicazione sensori con ruolo di Sensor Node;
- computer con ruolo di Check Node;
- rete i comunicazione Ethernet.

L’architettura è schematizzata nell’ immagine 5.2 dove sono evidenziati in giallo i componenti che sono classificati *safe* dal costruttore.

5.2.1 Robot COMAU

In questo caso, il manipolatore utilizzato (si veda figura 5.21) è molto diverso da quello precedente in quanto a caratteristiche.

Figura 5.21 – Manipolatore COMAU tipo NS-16.

Esso infatti è un robot che si colloca nella fascia dei “manipolatori industriali” di grossa taglia. Questo manipolatore permette di raggiungere delle alte velocità con un alta ripetibilità e precisione nel posizionamento.

E’ una macchina che possiede 6 gradi di libertà ed è pensato per un impiego in ambito industriale, per operazioni di assemblaggio, movimentazione pallet e saldatura.
Il controllore di questo robot (che controlla anche il secondo robot presente nella cella), necessita che il robot sia in sicurezza prima di attivare la movimentazione dello stesso.

La cella è infatti delimitata da una serie di barriere ottiche (indicate in rosso in Fig. 5.22) per identificare intrusioni da parte di operatori ed oggetti che possano andare a interferire con il moto dei robot.

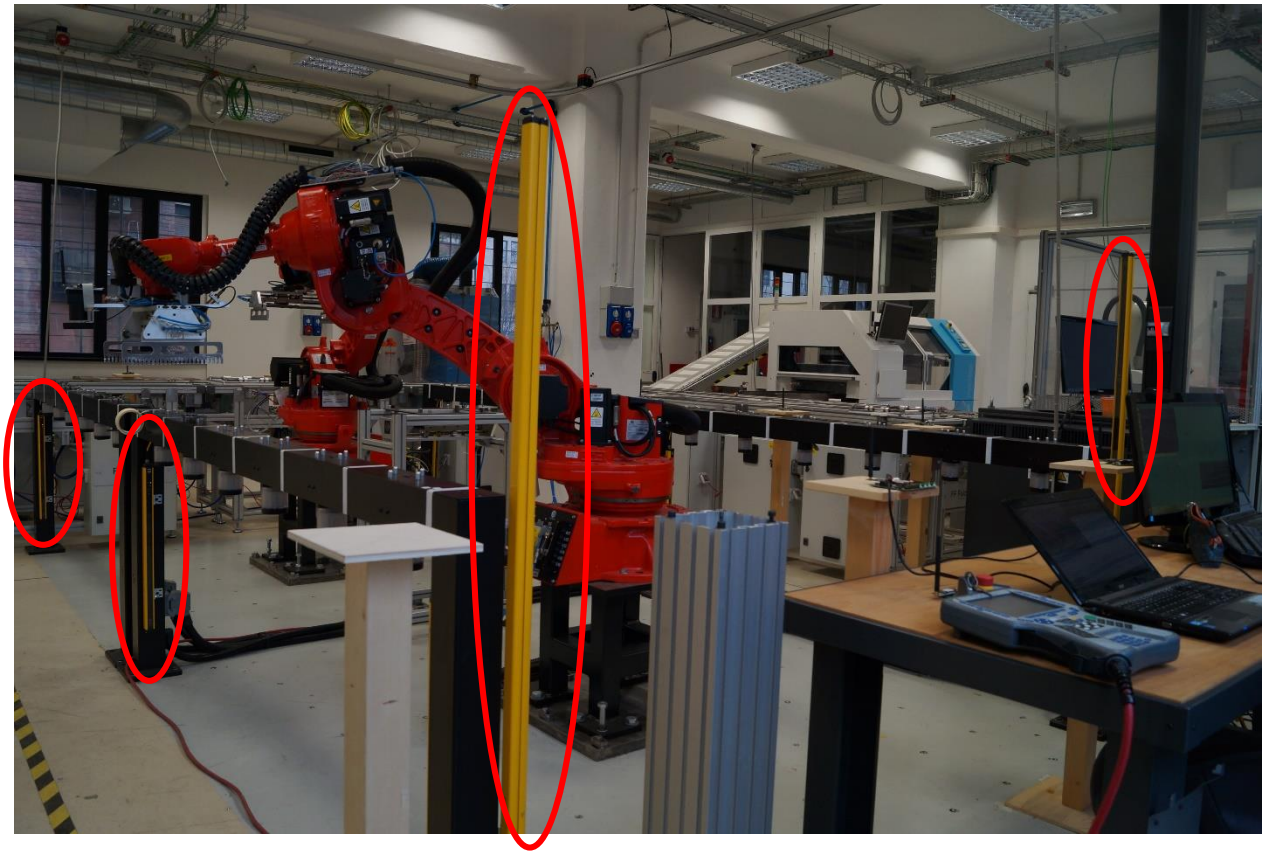


Figura 5.22 – barriere ottiche nell'impianto presso l'area di ricerca di Milano

Per rendere possibile l'interazione di operatori, in modo sicuro, con ambienti di questo tipo è quindi interessante andare a studiare l'applicazione degli spazi minimi di sicurezza. In questo modo, un intrusione di un operatore in una zona lontana a quella in cui il robot sta lavorando attualmente potrebbe essere concessa, senza mandare in stop l'intera cella.

Anche in questo caso, il moto del robot è controllato in remoto tramite un computer, che emula il controllore del robot e impartisce primitive nel linguaggio del controllore, in questo caso il PDL2. Il linguaggio, simile al linguaggio C, permette di eseguire un gran numero di operazioni di moto, per i test eseguiti si è semplicemente aperto un canale di comunicazione che permette lo scambio di informazioni.

Le coordinate del robot (*end-effector* e gomito) sono espresse in un vettore contenente le coordinate cartesiane e il quaternione, permettendo così di conoscerne anche l'orientamento.

Questi dati verranno poi inviati al *Check node*, che li utilizzerà, insieme alle velocità per la determinazione del *danger field* associato[28], come già descritto nel capitolo 4.

5.2.2 Sensore radio DFL

Il sensore utilizzato in questo caso è realizzato mediante una rete di sensori radio che coprono una superficie di lavoro. Recentemente, è stato dimostrato che analizzando le fluttuazioni delle onde elettromagnetiche a radio frequenze (RF), generate da una rete densa di sensori wireless, è possibile determinare la presenza di persone e quindi di tracciarne la loro posizione in maniera passiva senza che debbano indossare dispositivi attivi[29].

Questo sistema è composto da piccoli sensori che emettono onde radio, formando così una rete pervasiva.

I termini *sensor-less localization* (SLL) e *Device Free Localization* (DFL), enfatizzano il fatto che l'operatore non porta con se nessun device wireless attivo e non usa nessun sensore speciale (*sensor-less sensing*).

La presenza dell'operatore, disturba le onde radio nella zona coperta dalla rete wireless[29], rendendo possibile localizzarlo in un area senza la necessità di sensori aggiuntivi. L'uso di questa rete di sensori produce una serie di benefici negli spazi di lavoro industriali [30] come il basso costo della soluzione e la lunga vita dei sensori, che possono essere migliorati e riutilizzati in eventuali soluzioni migliorate.

La soluzione utilizzata si concentra sull'adozione dei sensori radio in quanto questa tecnica permette di ottenere dati in modo continuo da ambienti critici dal punto di vista della sicurezza. In contrasto con i sistemi di localizzazione basati su sensori fisici, la tecnica DFL permette di utilizzare dispositivi in tecnologia IEEE 802.15.4 che sfruttano le porzioni di spettro radio libere da vincoli di licenza (*Bande ISM- Industrial Scientific and Medical*)[31].

Il grande vantaggio rispetto alle tecnologie basate sugli infrarossi è dovuto al fatto che le onde radio possono penetrare corpi non metallici e possono operare in zone rischiose, caratterizzate da scarsa visibilità, come può essere un ambiente industriale.

Il Sistema DFL utilizzato in questo scenario sembra quindi essere una soluzione promettente, da utilizzare eventualmente ad altri sistemi di localizzazione, per l'identificazione di oggetti e operatori in applicazioni industriali di automazione.

Un ulteriore vantaggio che presenta questa tecnologia è quello di essere applicabile in reti wireless già esistenti nel mondo delle reti industriali (ad es. WiFi, ZigBee, WirelessGart,ISA SP100 [32]). Infatti, i sensori radio DFL possono essere usati per trasmettere altre informazioni (ad es. dati di temperatura, pressione, umidità, concentrazione di inquinanti) durante la fase di localizzazione.

Alcuni sistemi DFL sono descritti in letteratura e i principi che sfruttano sono sostanzialmente tre:

- metodo basato su RSS (*Received Signal Strenght*), basato sulle distanze stimate in base alla potenza ricevuta secondo la legge di *path-loss* [33];
- metodo basato su tecniche di *fingerprinting*[34], che sfrutta sempre il principio della potenza ricevuta ma anziché basarsi sulla distanza si basa su delle mappe calibrate);
- metodo tomografico [35] permette di visualizzare una accurata immagine radio dell'area di interesse.

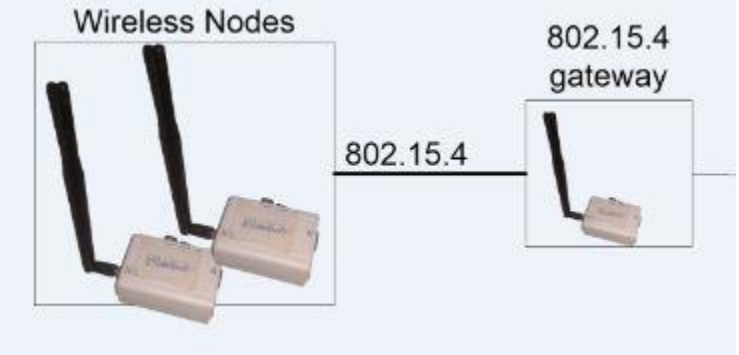


Figura 5.23 – Dispositivi radio utilizzati durate le prove sperimentali.

Nel caso attuale sono stati usati dispositivi basati su tecniche di *fingerprinting* che sono più adatte a sistemi *indoor*.

Figure 5.24 – Piano di lavoro dei sensori radio e dei loro corrispondenti link.

Nella successiva fase quindi sono stati posati i singoli moduli *wireless*, delimitando in questo modo l'area di lavoro che si intende monitorare (si veda la Fig. 5.24).

La rete di sensori, forma un intreccio di collegamenti radio (*radio link*) che giace su un piano, per questo motivo tutti i dispositivi sono stati posizionati in cima a dei sostegni aventi tutti la stessa altezza.

La posizione di questi sostegni è stata calibrata all'interno dello spazio di lavoro attraverso il misuratore ottico ad elevata precisione introdotto prima, in modo da avere una posizione precisa di riferimento (si veda la Fig. 5.27).

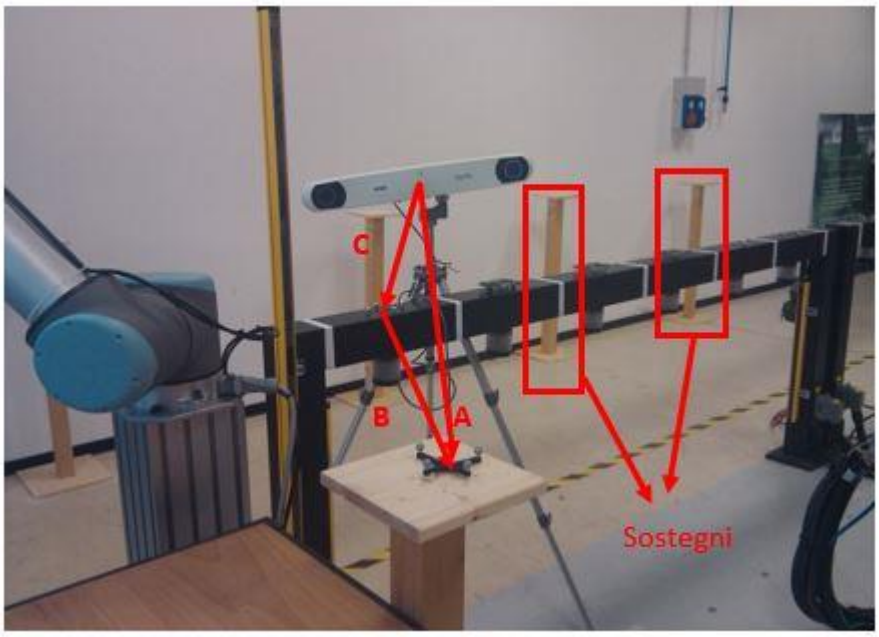


Figura 5.25 – calibrazione posizioni sostegni dispositivi radio

La posizione del marker sul sostegno è quindi ricavabile con delle semplici operazioni di inversioni di matrici:

$$B = A^{-1} \cdot C \qquad (17)$$

Effettuata questa operazione per tutte le posizioni in cui sono stati disposti i sostegni si ottiene la Fig. 5.28, che rappresenta il la disposizione dei dispositivi a RF.

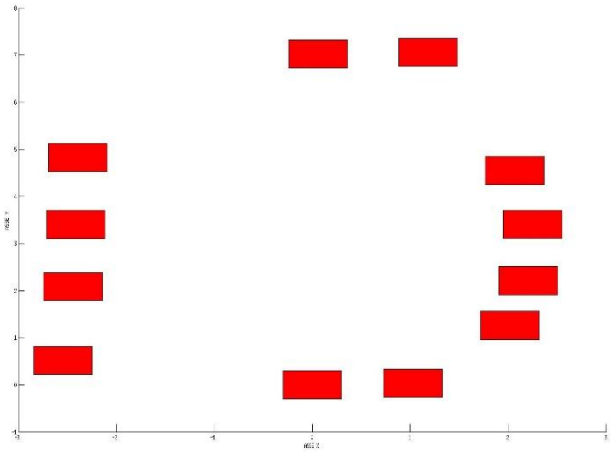


Figura 5.26 – Layout posizionamento dei sensori. Asse x e y espressi in metri.

Oltre ad aver calibrato le posizioni dei dispositivi fisici, sono state calibrate anche diverse altre posizioni, in modo da formare una griglia, in cui l'operatore può essere riconosciuto (come indicato in Fig. 5.27).

La localizzazione dell'operatore infatti non potrà assumere qualsiasi valore, ma sarà discreta relativa alla griglia di posizioni calibrate.
Procedendo con lo stesso metodo utilizzato in precedenza si ottiene la rappresentazione delle zone coperte dalla rete di sensori (si veda la Fig. 5.27).

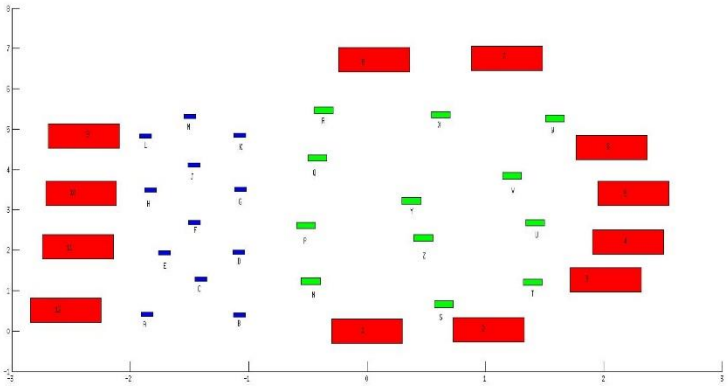


Figure 5.27 – Layout complessivo impianto con posizioni calibrate

Si può quindi suddividere il volume di lavoro in tre zone:

- Zona esterna, corrispondente alla posizione dei sostegni (rettangoli ***rossi).***
- Zona corridoio, corrispondente alla griglia nel corridoio (punti **blu).**
- Zona spazio di lavoro, corrispondente alla griglia nella zona di lavoro dei robot, (punti ***verdi).***

Dopo la calibrazione inziale che permette di definire le mappe di potenza (media e varianza per ogni *radio link*), si procede con l'inizializzazione della rete di sensori.
Di tutti i dispositivi fisici, uno funziona da gateway che si interfaccia con il pc mediante porta USB. Il protocollo di questa particolare rete prevede che i dispositivi accedano secondo la modalità a divisione di tempo (TDMA) dove ad ogni nodo è associato un tempo di trasmissione (*timeslot*) [36].

L'ordine di accensione (a partire dal *gateway* fino all'ultimo nodo radio) determina implicitamente l'ordinamento dei vari dispositivi che sono identificati da un differente indirizzo logico. Una volta completata questa inizializzazione, la prima misurazione serve a identificare l'ambiente.

Considerando che in ambienti industriali è presente molto rumore di fondo, questa procedura consente di rendere le misure successive, necessarie per il tracciamento della posizione, più precise ed affidabili.
Il problema principale per quanto riguarda la cella dell'impianto pilota presa in considerazione è la presenza dei robot che introducono un notevole disturbo[37]. E' quindi necessario capire quale è il contributo che la loro presenza comporta a livello di perdita di potenza di segnale (ad es. si veda la Fig. 5.28).

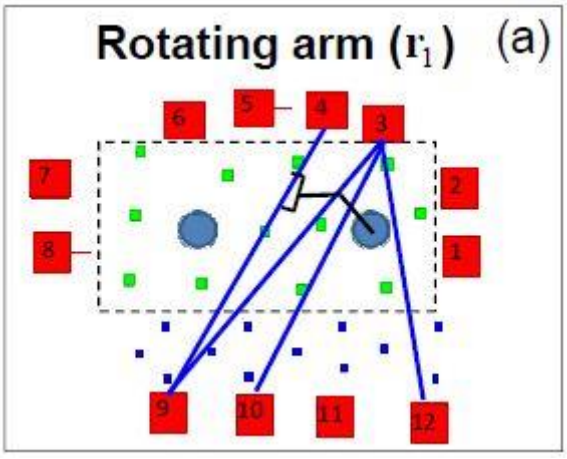


Figura 5.28 – Link interessati alla presenza del robot.

Si vede che non tutti i *radio link* che vengono misurati dalla rete di sensori, sono interessati alla presenza del robot, per cui andando a analizzare le potenze ricevute dai link interessati si riesce a capire qual è l'effetto del robot sulla potenza dei segnali.

Lo stesso procedimento può esser applicato anche ad altri elementi presenti nella cella che disturbano la rete impedendo la trasmissione

Dai dati relativi a questa misurazione si può estrapolare la matrice delle distanze relativa alle varie posizioni (si veda la Tab. 5.3).

D =

0	1.0281	2.3814	3.1222	4.0844	4.9955	6.8654	6.7298	5.3905	4.1740	3.2090	2.5966	0	0
1.0281	0	1.5834	2.4796	3.5934	4.6365	6.7356	6.7701	5.8951	4.8217	4.0318	3.6056	1.0281	1.0281
2.3814	1.5834	0	0.9658	2.1559	3.2836	5.5615	5.8054	5.6718	4.9234	4.5350	4.6243	2.3814	2.3814
3.1222	2.4796	0.9658	0	1.1963	2.3393	4.6641	5.0001	5.2896	4.7709	4.6481	5.0428	3.1222	3.1222
4.0844	3.5934	2.1559	1.1963	0	1.1548	3.5232	3.9806	4.8586	4.6688	4.8790	5.6023	4.0844	4.0844
4.9955	4.6365	3.2836	2.3393	1.1548	0	2.3902	2.9668	4.4714	4.6288	5.1417	6.1279	4.9955	4.9955
6.8654	6.7356	5.5615	4.6641	3.5232	2.3902	0	1.1259	4.0640	4.9238	5.9217	7.2757	6.8654	6.8654
6.7298	6.7701	5.8054	5.0001	3.9806	2.9668	1.1259	0	3.1025	4.1469	5.2797	6.7391	6.7298	6.7298
5.3905	5.8951	5.6718	5.2896	4.8586	4.4714	4.0640	3.1025	0	1.4281	2.7501	4.3186	5.3905	5.3905
4.1740	4.8217	4.9234	4.7709	4.6688	4.6288	4.9238	4.1469	1.4281	0	1.3228	2.8919	4.1740	4.1740
3.2090	4.0318	4.5350	4.6481	4.8790	5.1417	5.9217	5.2797	2.7501	1.3228	0	1.5699	3.2090	3.2090
2.5966	3.6056	4.6243	5.0428	5.6023	6.1279	7.2757	6.7391	4.3186	2.8919	1.5699	0	2.5966	2.5966
0	1.0281	2.3814	3.1222	4.0844	4.9955	6.8654	6.7298	5.3905	4.1740	3.2090	2.5966	0	0
0	1.0281	2.3814	3.1222	4.0844	4.9955	6.8654	6.7298	5.3905	4.1740	3.2090	2.5966	0	0

Tabella 5.3 – Matrice delle distanze

Una volta completate queste fasi, il sistema è pronto per identificare la presenza di un operatore all'interno del suo volume di lavoro. Per l'algoritmo si veda il riferimento [38]. Sono state condotte preliminarmente, una serie di misurazioni per poter verificare il corretto funzionamento del sistema.

L'operatore in queste prove mantiene la propria posizione per qualche secondo, in modo che si abbiano parecchi campioni del segnale ricevuto recandosi in una certa posizione.

Il nodo *gateway* interroga gli altri nodi ogni 60 ms, memorizzando la potenza del segnale proveniente da essi. In questo modo per confronto con il caso di misura d'ambiente si determina se c'è la presenza di un operatore mediante un metodo statistico.

Dopo diverse prove, si è convenuto che il numero minimo di interrogazioni per avere un'informazione affidabile sulla posizione è pari a 3, quindi ogni 180 ms si ha una informazione di posizionamento. L'informazione che viene inviata dal Sensor Node al Check Node, in questo caso è un vettore composto da 24 elementi. Questi elementi rappresentano una posizione della griglia tracciata a terra, nel corridoio o nella zona di lavoro del robot e descritta in Fig. 5.29. La stima della posizione viene realizzata valutando un coefficiente di affidabilità calcolato mediante stime a Massima Verosimiglianza (Maximum Likelihood o ML) [39].

Si determina così quanto affidabile è un valore di potenza, discriminando casi in cui questa informazione non è così certa da quelli in cui è nettamente distinguibile. Vengono considerati i 4 massimi di questo vettore, che corrisponderanno alla posizione in cui è presente l'operatore e altre attigue.

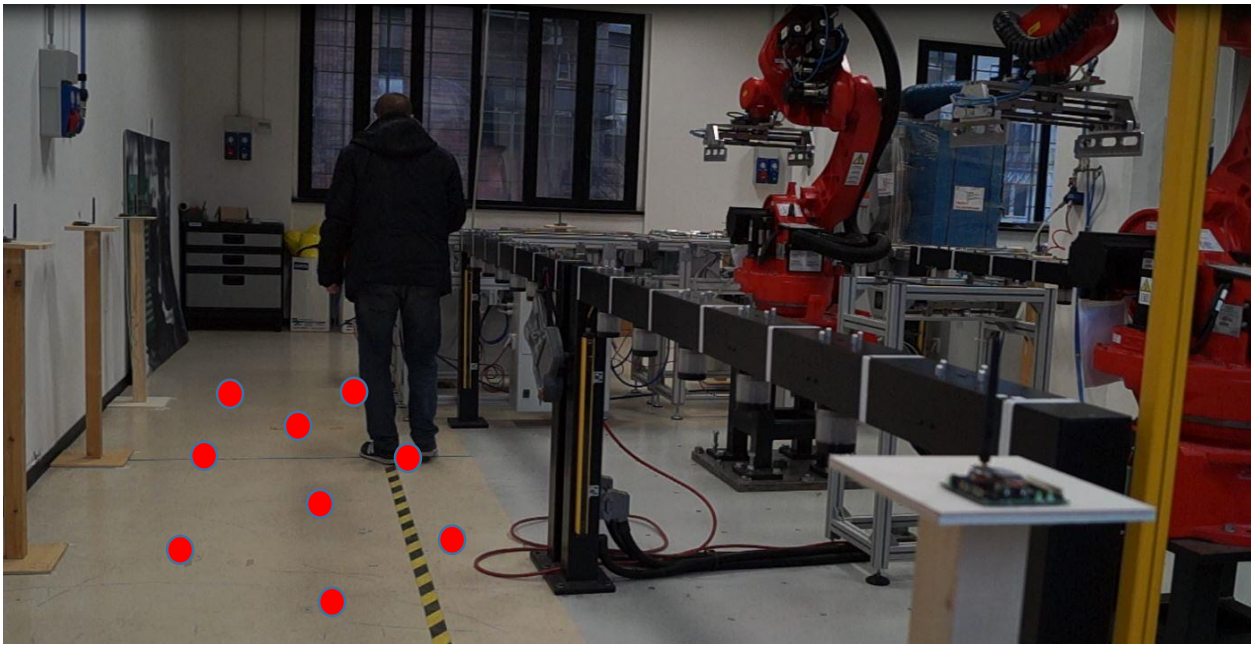


Figura 5.29 – Rilevamento posizione operatore

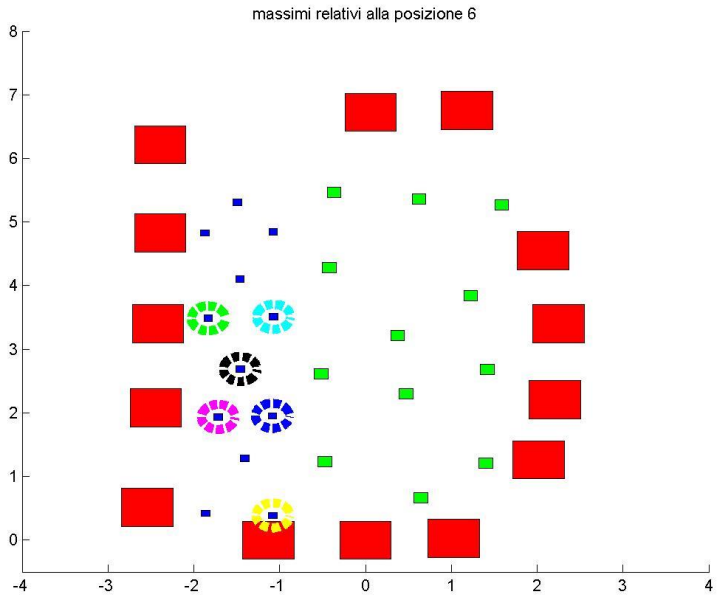


Figure 5.30 – Massimi estratti dal vettore di potenza inviato

Da questi 4 valori massimi si può quindi ricavare la probabilità che l'operatore sia in una delle posizioni, in base al valore del coefficiente di affidabilità a loro associato.

Queste operazioni di calcolo sono a carico del *CN*, il quale riceverà l'intero vettore di 24 elementi. La differenza sul traffico di rete generato su UDP, nel caso di invio dell'intero vettore piuttosto che le sole 4 componenti massime è irrisorio, per cui si preferisce non caricare computazionalmente il nodo sensore.

5.2.3 Applicazione algoritmo

In questo caso i risultati presentati sono relativi a una simulazione effettuata in Matlab, partendo da valori reali misurati sperimentalmente di posizione e velocità del robot e di *tracking* dell'operatore.

In questo caso quindi i dati collezionati dal *CN* e i risultati non sono effettuati online, ma sono raccolti durante più misure.

E' stata quindi realizzata e misurata una traiettoria con il robot, di cui si sono osservate le posizioni di gomito e polso, e le relative velocità.
E' stata poi tracciato un percorso effettuato da un operatore all'interno del corridoio, mediante il sensore radio, al confine tra la zona di corridoio e la zona operativa dei robot.

Anche in questo caso il calcolo del *danger field* è effettuato per il solo valore di soglia e del valore attuale.
In questo caso avremo che l'ingombro dell'operatore considerato, è maggiorato rispetto all'ingombro reale a causa della sensibilità dello strumento che ne misura la posizione (0.8 m).

Infatti, per quanto detto in precedenza si può avere un errore di posizionamento, pari alla granularità della griglia di posizioni presenti nel corridoio e nell'area di lavoro dei robot.

Questa imprecisione dello strumento porterà ad avere uno spazio minimo di sicurezza attorno al robot meno sensibile al modulo della velocità e alla direzione della velocità stessa, ma sarà molto più dipendente dalla distanza.
Per quanto riguarda i termini relativi ai ritardi temporali dovuti alle trasmissioni di rete si è verificato che i risultati sono equiparabili a quelli relativi alla prova precedente.
L'unico termine che non è stato indagato in questo caso è lo spazio di frenata del robot, ma data la predominanza degli altri termini, in questo caso può essere considerato ininfluente ai fini del calcolo dello spazio minimo.

La logica nel controllo del robot è la medesima di quella usata nel caso precedente: si procede lungo una traiettoria finché non viene violata la condizione sulla soglia del *danger field*.
A quel punto viene applicata una politica di *collision avoidance* che permette di ritornare in un funzionamento *sicuro*, dopodiché il robot riprende la sua traiettoria originale.

Vengono riportati in seguito alcuni frame del video che rappresenta la simulazione.

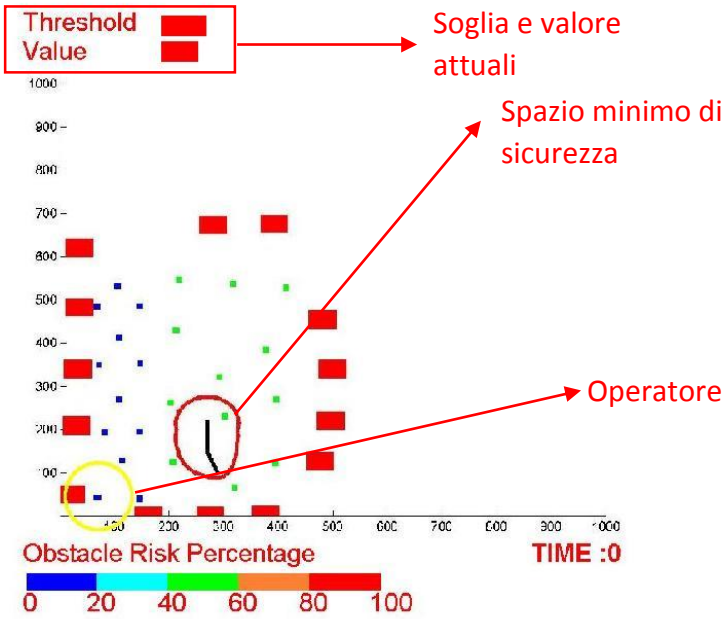
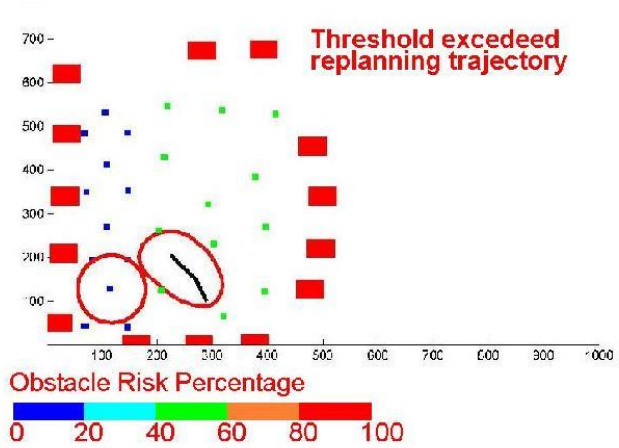


Figura 5.31 a,b – Fotogrammi estratti dal video di simulazione



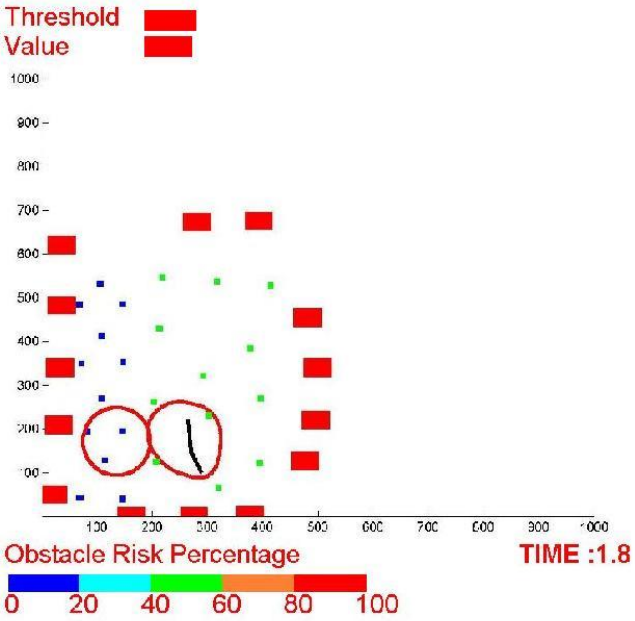
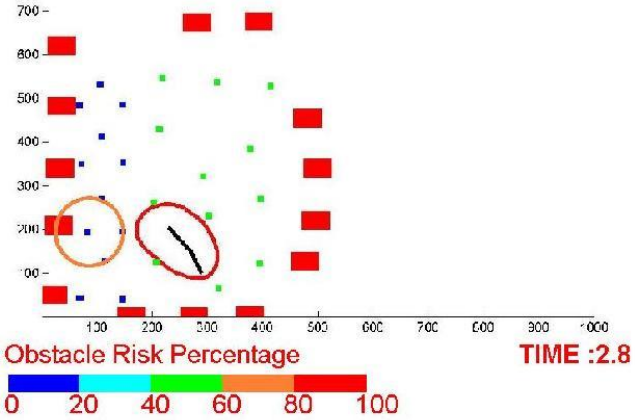


Figura 5.31 c,d – Fotogrammi estratti dal video di simulazione



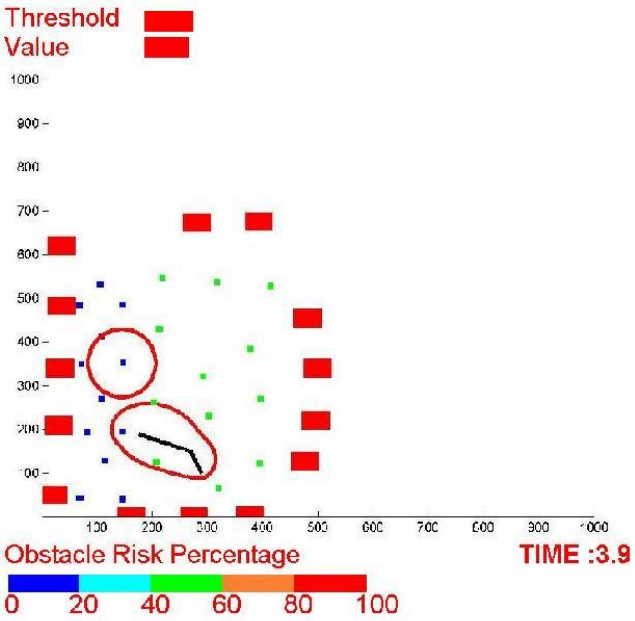
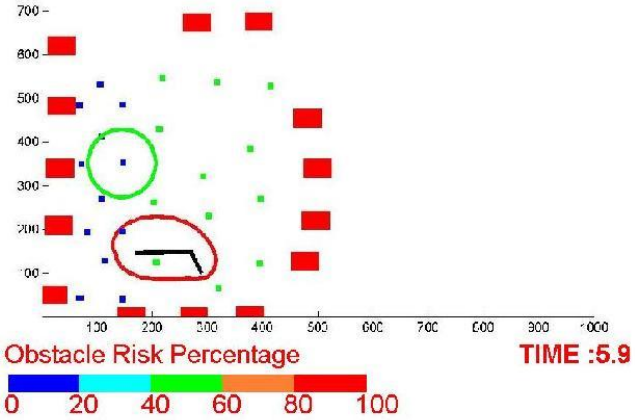


Figura 5.31 e,f – Fotogrammi estratti dal video di simulazione



E’ stata realizzata inoltre, una simulazione nel caso di utilizzo di zone di sicurezza calcolate e prestabilite a priori. Nell’ambiente industriale, solitamente vengono utilizzati dei sensori, come le tende ottiche, che delimitano un area all’incirca simile all’area di lavoro dei robot.

Il risultato è quindi una zona vasta in cui non è permesso accedere all’operatore a meno di fermare l’impianto per permettere operazioni a robot inattivi.
Di seguito sono riportati alcuni fotogrammi del video della simulazione.

Il compito che il robot deve eseguire è il medesimo del caso precedente.
Viene simulato anche in questo caso un percorso per l’operatore (vedi pallino rosso in Fig. 5.32 a,b,c,d) che preveda l’ingresso e l’uscita nella zona di sicurezza.

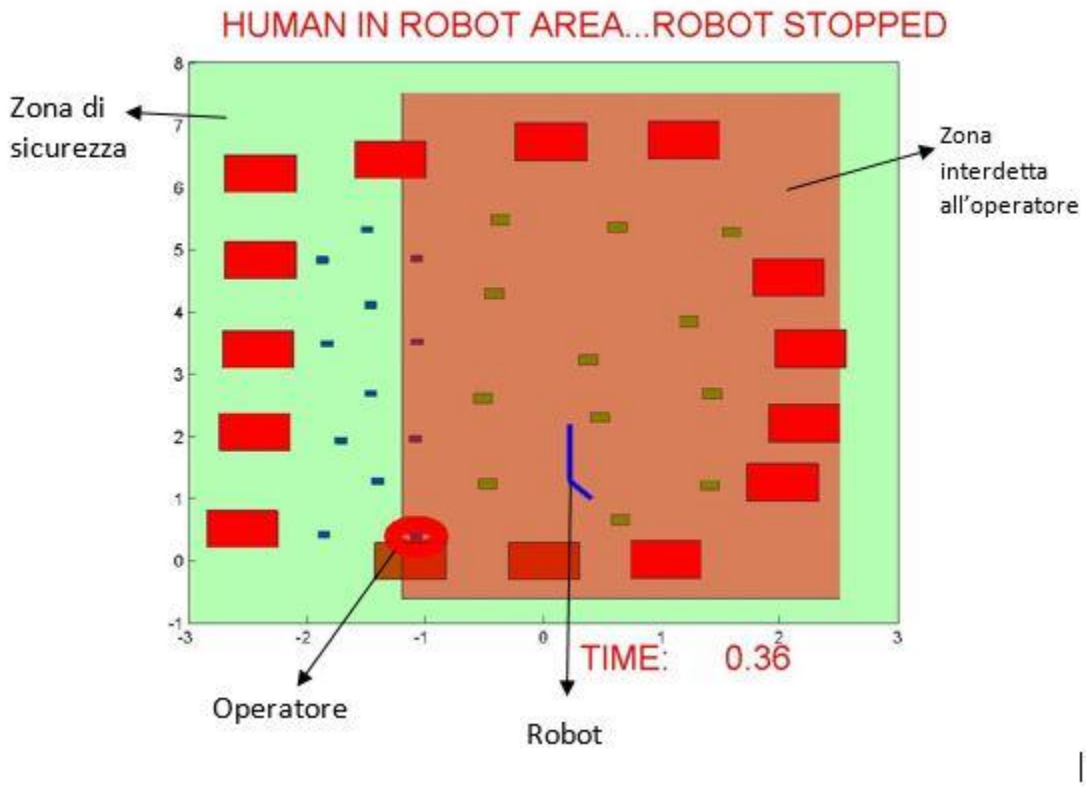


Figura 5.32 a – Simulazione con zone di sicurezza statiche e definite a priori.

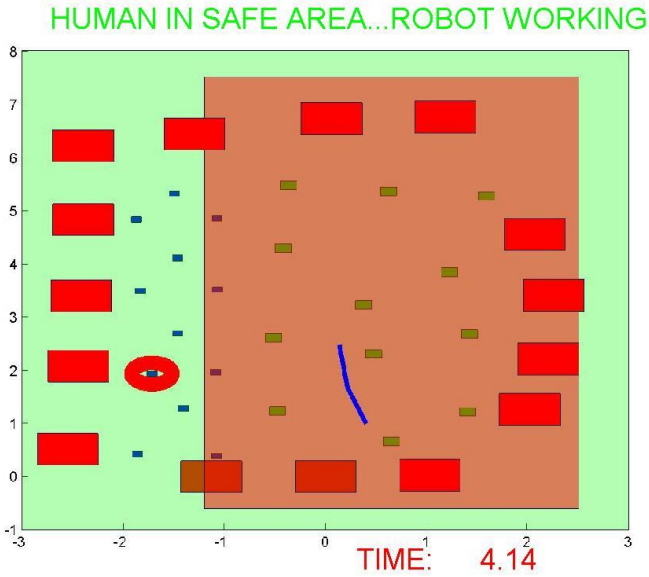
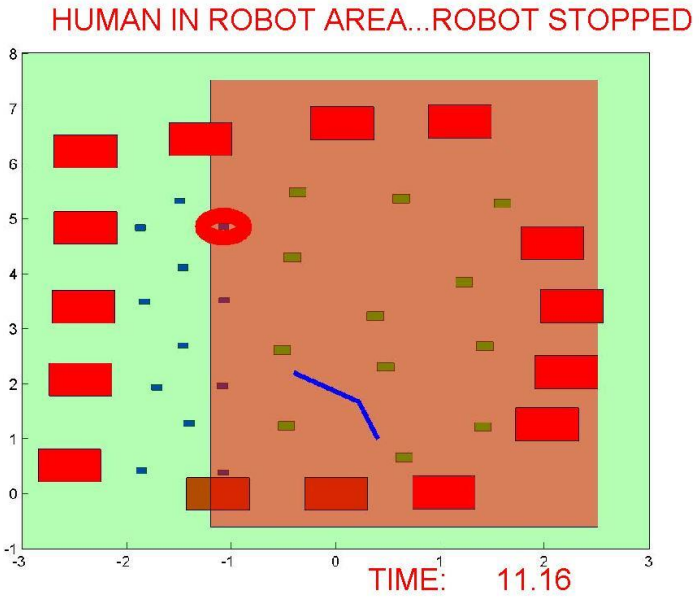


Figura 5.32 b,c– Simulazione con zone di sicurezza statiche e definite a priori.



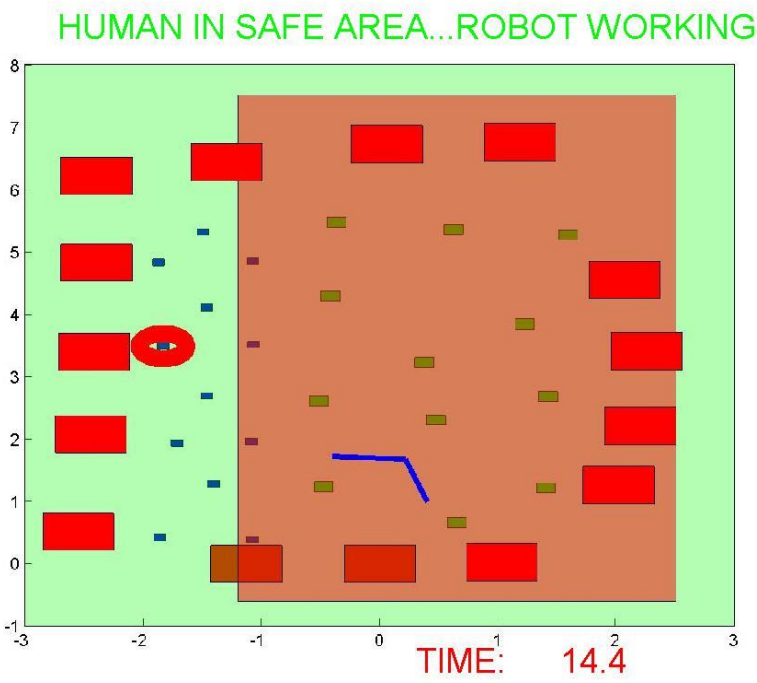


Figura 5.32, d – Simulazione con zone di sicurezza statiche e definite a priori.

Si può notare, anche in questa soluzione il vantaggio che si ottiene con una soluzione dinamica anziché quella statica.
Per evidenziare ulteriormente il guadagno temporale è stata confrontato il tempo stimato per il completamento del compito assegnato al robot, con le due differenti strategie.

Nel grafico di seguito sono riportati gli andamenti della percentuale di completamento in funzione del tempo. La linea rossa indica la strategia con gli spazi minimi di sicurezza calcolati dinamicamente, mentre la linea blu si riferisce al caso con aree di sicurezza fisse.

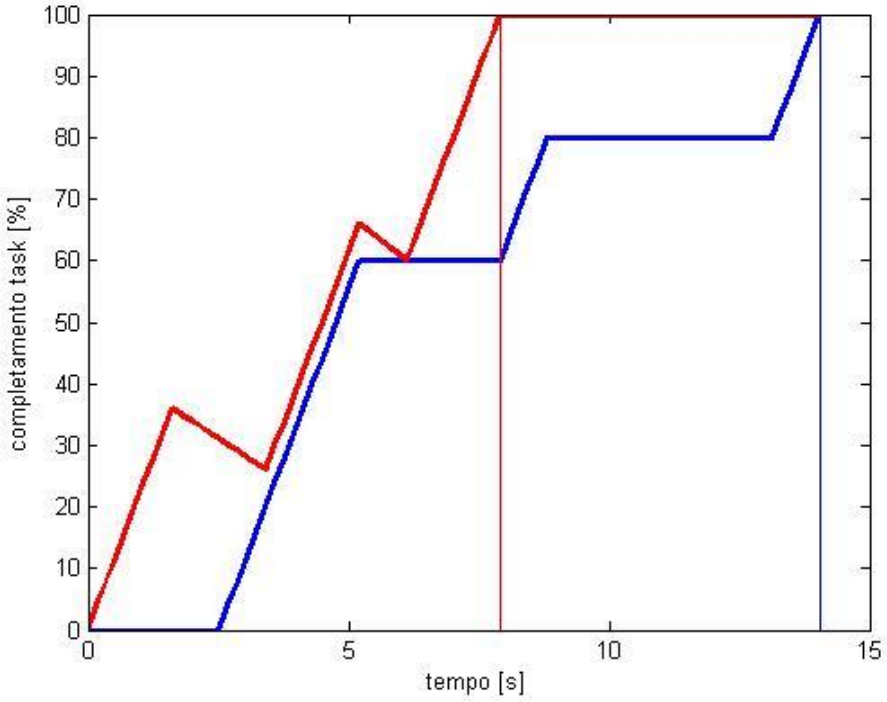


Figura 5.3– confronto durata esecuzione

5.3 Valutazione prestazioni

Per valutare le prestazioni dell'algoritmo, un indice che potrebbe essere utilizzato è il rapporto tra le aree statiche e dinamiche. Possiamo infatti mettere a confronto la proiezione dello spazio minimo di sicurezza su un piano xy, come rappresentato dalla Fig. 5.38.

L'area attorno al robot in cui l'operatore non può entrare, nel caso della normativa vigente, avrà una superficie calcolabile come una porzione del cerchio.

In particolare sarà:

$$A_s = \int_0^{\beta} d\vartheta \cdot \int_0^{D} \rho d\rho \qquad (18)$$

In cui β è l'angolo al centro che definisce la porzione del cerchio da considerare (ovvero $\pi/2$ nel caso dell'esempio di Fig. 5.34) e *D* è la distanza minima che deve essere mantenuta per il rispetto della normativa.

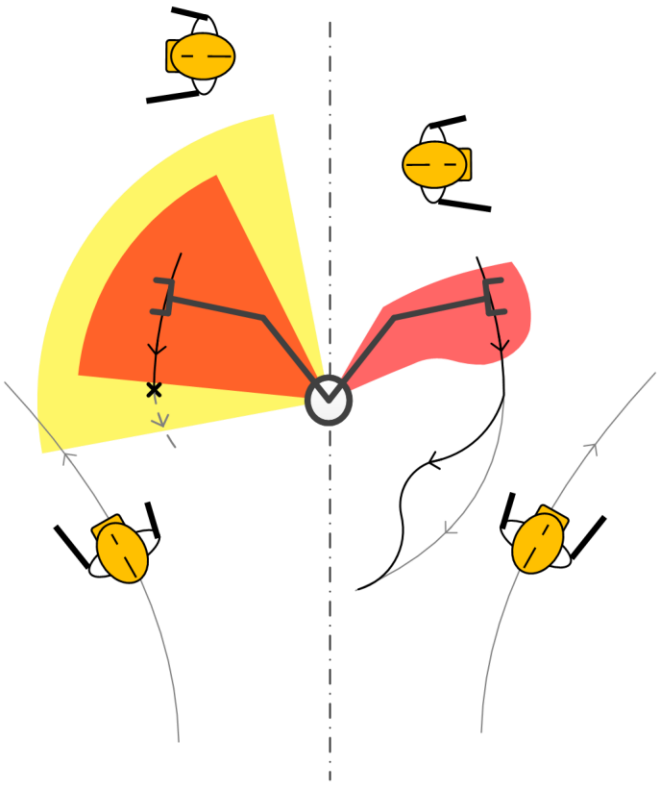


Figura 5.34– Aree di sicurezza a confronto. Standard iso 10218 a sinistra, calcolo zone dinamiche a destra.

Per quanto riguarda il calcolo dell'area che produce l'inviluppo del robot prodotto con l'algoritmo presentato in questa tesi, si può utilizzare la formula dell'area di Gauss.
La curva iso-rischio trovata, di cui sono note tutte le coordinate dei punti che la compongono, può essere vista come un poligono con n lati. In questo modo si applica la formula

$$A_d = \sum_{i=1}^{n}(x_i \cdot y_i - x_{i+1} \cdot y_i) \qquad (19)$$

Dove x_i e y_i sono le coordinate dei punti che fanno parte della curva.

In questo modo, ottenute le due aree è possibile rapportarle e ricavare qual è il fattore di guadagno applicando l'algoritmo illustrato in questa tesi.

$$G = \frac{A_d}{A_s} \qquad (20)$$

Si noti che questo indice dà anche informazioni sulla bontà del sensore che si sta utilizzando, in quanto si è visto che la tempistica che determina il corretto e significativo dimensionamento dell'inviluppo di sicurezza attorno al robot è legata alla frequenza del sensore.

Se la frequenza del sensore non dovesse essere sufficientemente elevata, causerebbe un ingrandimento della zona di sicurezza, e quindi aumenterebbe il valore della sua area.

Sono state effettuate delle simulazioni con valori variabili di frequenza del sensore, in modo da identificare i valori che rendono più performante il calcolo dello spazio di sicurezza in modo dinamico anziché quello statico.

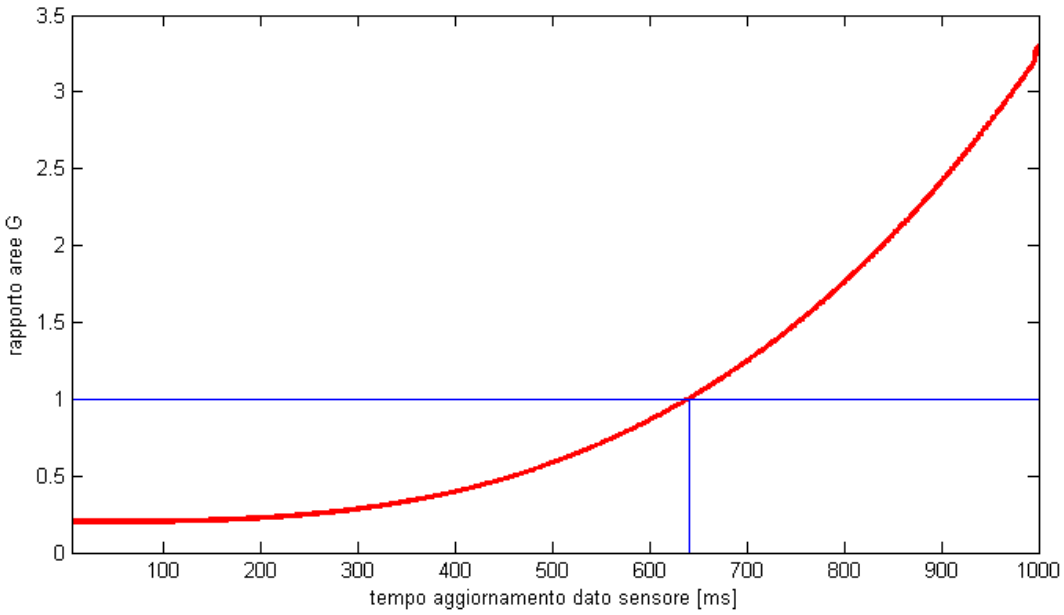


Figura 5.35 – Andamento del rapporto delle aree G in funzione del tempo di aggiornamento del sensore.

L’andamento del rapporto aree è approssimabile con un andamento cubico in funzione dei tempi di ritardo del sensore. Si identifica attorno a 640 ms, la condizione limite che discrimina la funzionalità del sensore.
Si può quindi affermare un sensore che produce il dato di posizionamento dell’operatore con una frequenza di campionamento almeno maggiore di 1/0.640=1.57 Hz, può essere utilizzato in questo tipo di architettura. L’algoritmo presentato produrrà una zona di sicurezza più piccola tanto più è veloce il sensore.

Un altro parametro da tenere in considerazione è la precisione di identificazione del sensore. La curva rappresentata in Fig. 5.39 è ottenuta con la precisione di identificazione del sensore costante. Un altro termine che viene mantenuto costante durante le simulazioni fatte per calcolare il valore delle aree è la velocità del robot.
Facendo aumentare la velocità del robot, il vincolo sulla frequenza del sensore diventerà più stretta.

5.4 Sensor Fusion

L'architettura *SafeNet* presentata in precedenza nel capitolo 3, prevede un nodo che si occupi di identificare la posizione dell'operatore e degli ostacoli, mentre un altro si occupi di determinare la posizione del robot.
Per aumentare l'affidabilità del sistema possono essere usati più dispositivi, anche di natura diversa, che permettono di avere o la stessa informazione ridondata, oppure aggiungere informazioni addizionali.
Ad esempio nel caso del robot potrebbe essere stimata la posizione che raggiungerà nel prossimo ciclo di controllo applicando calcoli di cinematica diretta su un nodo robot diverso da quello che funziona da controllore.
Analogamente, dal lato sensore, si può pensare di avere più sensori che osservano lo stesso fenomeno, in modo da identificare eventuali errori da parte di uno dei due.
Con queste informazioni ridondanti infatti si riesce a garantire la sicurezza architetturale, in quanto si è in grado di identificare più facilmente un errore nel sistema, mentre nel caso di un sensore singolo è molto più difficile.

Nel caso in cui i sensori utilizzati non generino una informazione ridondata ma diano informazioni diverse riguardanti lo stesso fenomeno, allora si può pensare di costruire una informazione aggregata.

Nello specifico un sensore potrebbe essere utilizzato per l'identificazione della posizione dell'operatore, mentre un altro sensore potrebbe essere usato per identificare il tipo di operatore identificato.
In questo modo si riesce a validare comunque l'informazione di un sensore con quella dell'altro, andando a completarla in caso di concordanza.

Nel caso dell'impianto pilota presentato in precedenza, si può pensare di utilizzare quindi il sistema di identificazione della posizione dell'operatore presentato prima e ridondare questa informazione con una rete di telecamere a tempo di volo (TOF).

Figura 5.36 – Time of fligh (TOF) camera

Le telecamere a tempo di volo produrranno un output che, una volta sincronizzato temporalmente con quello dell'altro sensore attivo, potrà essere usato per confrontare i risultati. E' fondamentale assicurarsi quindi che l'output prodotto da i due sistemi di localizzazione sia riferito alla osservazione del medesimo istante del fenomeno osservato. In caso contrario le due informazioni combinate daranno un errata interpretazione di ciò che sta realmente accadendo.

Per quanto riguarda il sensore DFL, è inoltre possibile avere un'informazione ridondata anche senza utilizzare un altro tipo di sensore.
Infatti i nodi che compongono una rete atta a realizzare una localizzazione, possono essere suddivisi in due gruppi, in modo da formare due sotto reti.

Il problema, con questo tipo di soluzione, è quello di avere un numero sufficiente di nodi per ogni sottorete, in modo che questi possano produrre una localizzazione veritiera ed affidabile. E' noto infatti che maggiore è il numero di nodi di una rete, migliore risulta essere la localizzazione.

Disponendo di un numero sufficiente di nodi si riesce quindi a creare due sottoreti, che dovranno però trasmettere i dati a frequenze diverse per non ostacolarsi l'una con l'altra. Ognuna delle due produrrà quindi un output indipendente, che sarà utilizzato nel confronto.

5.5 Sicurezza Funzionale

Un aspetto da considerare per un sistema di cui si vogliono determinare le prestazioni è quello della sicurezza dei dati e della identificazione degli errori e guasti.
In particolare è importante sia identificare un guasto o un errore, e riuscire a trovare il modo di isolarlo.
Una volta isolato possono essere applicate le procedure di recupero della situazione nominale.
Esistono diversi metodi per effettuare queste analisi.
Il primo che viene analizzato è quello che prende il nome di Monitoraggio di Processo (Process Monitoring)[40], ed ha come obiettivi quello di ridurre le inoperosità e aumentare la sicurezza.
Il Process monitoring può essere approcciato in diversi modi: il primo è quello Data-Driven, cioè guidato dai dati raccolti mentre il secondo è quello analitico, in cui i dati raccolti sono confrontati con quelli provenienti da un modello matematico.

Nell'approccio basato sui dati sono percorribili due strade: la prima è quella della ridondanza fisica. In questo caso si dispone di due o più sensori fisici i cui dati vengono incrociati per identificare eventuali errori da parte di uno o più sensori.

La seconda strada è quella della analisi della variabilità dei segnali che si basa su concetti statistici. Il valore misurato sarà comparato con delle soglie predeterminate per capire se si è in presenza di errori/guasti o meno. Questa particolare tecnica prende il nome di Statistical Process Control (SPC) [41]e può essere utilizzata su una singola variabile o su più variabili del sistema da identificare. Questo tipo di metodo è molto utile quando la variabile da monitorare avrà un andamento pressoché costante attorno ad un valore. Questo tipo di verifica può essere rappresentata graficamente come rappresentato nella figura sottostante.

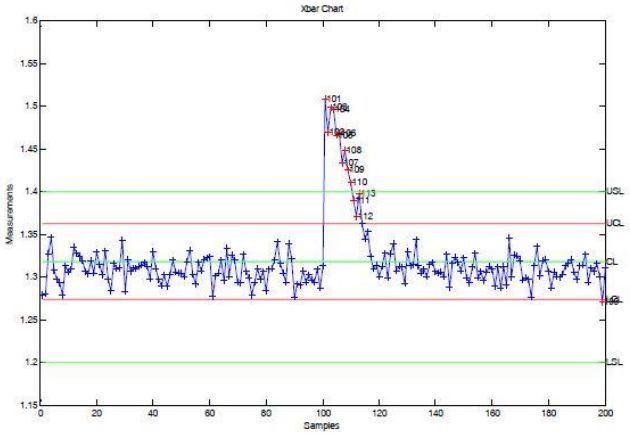


Figure 5.37 – Control Chart

Un metodo più raffinato che è possibile utilizzare è quello che prende il nome di EWMA *(Exponential Weigth Moving Average)[*42].

In questo caso si andrà a calcolare una media mobile del valore da osservare, attribuendo dei pesi esponenziali ai termini che vengono utilizzati nella media. Questo comporta notevoli vantaggi in quanto i termini più vecchi perdono man mano influenza sulla media attuale, inoltre possono essere identificati effetti di deriva in quanto la media crescerebbe anch'essa.

Le soglie avranno in questo caso un andamento esponenziale.

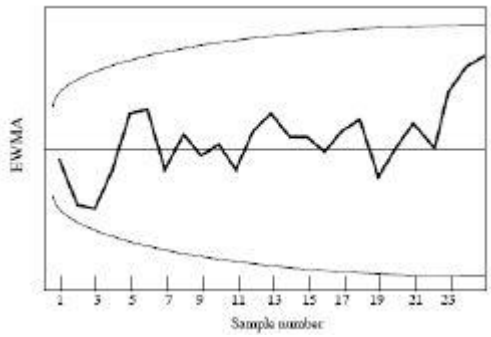


Figure 5.38 – Esempio di EWMA

Per quanto riguarda la sicurezza architetturale, questa è raggiungibile solo con una ridondanza fisica degli elementi. In questo modo si può costruire una sorta di tabella della verità con la quale è possibile identificare eventuali guasti.
Il concetto è moto semplice. Se uno dei due segnala un dato molto diverso dall'altro allora uno

dei due sensori non è affidabile. Si può procedere con uno dei metodi di prima dell'approccio statistico per verificare quale dei due sensori rileva una misura corretta.

Più aumenta il numero di nodi ridondati più ovviamente la probabilità di non identificare un guasto diminuisce mentre aumenta la probabilità di isolare correttamente l'errore, mentre con solo due nodi ridondati è impossibile.

Capitolo 6
Conclusioni e sviluppi futuri

Il lavoro svolto ha avuto l'obiettivo di determinare lo spazio minimo di sicurezza in modo dinamico e a tempo reale.
Per fare ciò è stato necessario eseguire un'analisi sui sistemi in cui applicare queste soluzioni, cercando di mantenere la maggior generalità possibile, andando ad analizzare tutte le fonti di incertezza presenti.
Non esistendo ad oggi, sensori in grado di garantire una corretta localizzazione dell'operatore in modo sicuro, è stato analizzato e studiato come trasformare uno o più sensori che nativamente non sono sicuri, in un sensore generalizzato che sia in grado di soddisfare le performances necessarie al calcolo degli spazi minimi di sicurezza. Nello specifico è stato valutato il comportamento di un sensore distribuito per la radiolocalizzazione.

E' stato analizzato poi come il calcolo delle sicurezza si basi sulle caratteristiche che ha il sensore: la frequenza di aggiornamento della localizzazione infatti si traduce in una incertezza di posizionamento con un conseguente aumento della distanza di sicurezza da mantenere.

L'algoritmo utilizzato per effettuare il calcolo delle zone di sicurezza permette di capire quindi se con una particolare configurazione di sistema (sensore e robot) si riescono a rappresentare in modo soddisfacente le zone di sicurezza attorno al robot.
Se questi requisiti non sono rispettati infatti, la forma della zona di sicurezza non sarebbe simile alla forma del robot ma avrebbe una forma completamente indipendente.
Attraverso le analisi e le simulazioni svolte si è in grado di attribuire questo comportamento alle performance del sensore generalizzato e di individuare quali sono le cause.
Nel caso invece in cui il sensore soddisfi i requisiti, si è in grado di calcolare dinamicamente una zona di sicurezza attorno al robot consistente.

La nozione analitica del sistema a livello architetturale consente in principio anche di utilizzare la tecnica esposta in modalità inversa, ovvero di determinare le specifiche di installazione o del tasso di miglioramento richiesto alle installazioni già predisposte, grazie all'analisi sulle componenti (e.g. latenze, frequenze di aggiornamento, accuratezze spaziali,) che influenzano le prestazioni del sensore equivalente.

In termini ancora più generali, le tecniche esposte si prestano a valutare puntualmente non solo i componenti fisici del sistema (*sensor node, robot node*) ma anche l'esecuzione dei calcoli delle funzioni dedicate alla sicurezza. Qualora infatti fossero disponibili componenti hardware sicuri (di classe adeguata ai requisiti introdotti in sede di *risk assessment* o specificati dalle norma di prodotto), l'esecuzione di algoritmi avanzati in sicurezza necessita comunque di procedure di

calcolo complesse dal punto di vista della sicurezza funzionale (e.g. di non semplice certificazione). La generalizzazione di nodi e algoritmi nell'architettura proposta consente invece di effettuare tali calcoli alla condizione di individuare (e documentare) i livelli di incertezza legati all'acquisizione dati e processamento delle funzioni. Nel caso di calcolo/controllo distribuito, tali grandezze sono state introdotte ed analizzate sia in casi sperimentali controllati (e.g. con sensore molto accurato) al fine di evidenziare gli elementi di incertezza temporale, sia in casi sperimentali reali (e.g. con sensore meno accurato e lento) in modo da evidenziare anche le componenti più critiche ai fini della sicurezza di tipo distanze minime / velocità massime (SSM).

Gli sviluppi futuri di questo lavoro riguardano i seguenti aspetti:

- l'analisi e la realizzazione dell fusione dei dati da parte di sensori diversi (telecamere a tempo di volo e Sistema di radiolocalizzazione);
- l'inserimento di un filtro di Kalman per stimare la posizione dell'operatore a partire dai dati raccolti dal sistema di radiolocalizzazione; in questo modo è possibile aumentare la frequenza con cui il Check Node riceve i dati provenienti dal sensore, riuscendo ad effettuare il calcolo della zona di sicurezza piu frequentemente.
- l'analisi dettagliata degli spazi di frenata dei robot per costruire una mappa che data una configurazione e una velocità riesca a stimare lo spazio necessario per l'arresto, componente importante per il calcolo della minima distanza di sicurezza da mantenere.

Bibliografia

[1] J. Krüger, T. Lien, and A. Verl, “Cooperation of human and machines in assembly lines,” CIRP Annals -Manufacturing Technology, vol. 58, no. 2, pp. 628 –646, 2009.

[2] L. Wang, “Collaborative robot monitoring and control for enhanced sustainability,” The International Journal of Advanced Manufacturing Technology, pp.1–13, 2013.

[3] ISO, “ISO 10218-1:2011, Robots for Industrial Environment: Safety Requirements. Part 1: Robot”, 2011.

[4] IEC/EN 62061,”Safety of machinery: Functional safety of electrical, electronic and programmable electronic control systems”.

[5] F. Vicentini, N. Pedrocchi, and L. M. Tosatti, “Safenet of unsafe devices - extending the robot safety in collaborative workspaces.” in ICINCO (2) Sasiadek, Eds. SciTePress, 2013, pp. 276–283.

[6] Safety of machinery, “Safety-related parts of control systems -- Part 1: General principles for design”, ISO 13489:2006.

[7] SO RFC 768, “UDP datagram definition”, 8/1980.

[8] J.Fryman, M.Björn, “Safety of Industrial Robots: From Conventional to Collaborative Applications”, Robotik 2012.

[9] T. Salmi,O. Väätäinen,T. Malm,J. Montonen,I. Marstio, “Meeting New Challenges and Possibilities with Modern Robot Safety Technologies”, Enabling Manufacturing Competitiveness and Economic Sustainability, 2014, pp 183-188.

[10] UNI EN ISO 13855:2010,” Sicurezza del macchinario - Posizionamento dei mezzi di protezione in funzione delle velocità di avvicinamento di parti del corpo umano”.

[11] “Robots and Robotic Devices: Safety Requirements. Part2: Industrial Robot Systems and Integration”, ISO/FDIS 10218-2:2011, 2011.

[12] “Robots and Robotic Devices: Industrial Safety Requirements, Collaborative Industrial Robots”, ISO/TS 15066:2013, 2013.

[13] Safety Eye Plitz, https://www.pilz.com/eshop/b2b/publicinit.do?category=00014000337042&language=en#

[14] ABB safemove https://www.abb.com/product/seitp327/ad4ce37123328400c12578e00046d439.aspx?productLanguage=it&country=IT&tabKey=2

[15] Virtual Curtain, http://www.sick.com/group/EN/home/products/product_portfolio/optoelectronic_protective_devices/Pages/safety_camera_systems.aspx

[16] SICK, http://www.sick.com/us/en-us/home/products/product_portfolio/distance_sensors/Pages/distance_sensors.aspx

[17] Alberto León-Garcia, Indra Widjaja.” Communication Networks”, 2003, McGraw-Hill, New York.

[18] ISO 7498, "Data Processing, Open System Interconnection, Basic Reference”.

[19] “OpenSafety”, IXXAT Automation.

[20] “RFC 3174 - US Secure Hash Algorithm 1 (SHA1)”, september 2001.

[21] “Standard for a Precision Clock Synchronization Protocol for Networked Mesaurement and Control System”. IEEE 1588-2:2008, 2008.

[22] P.Rocco, B.Lacevic, “Kinetostatic Danger Field - a Novel Safety Assessment for Human-Robot Interaction ”, The 2010 IEEE/RSJ International Conference on Intelligent Robots and Systems, October 2010.

[23] P.Rocco, B.Lacevic, “Safety-Oriented Control of Robotic Manipulators - a Kinematic Approach”, 18th IFAC World Congress September, 2011

[24] G. Hirzinger, A. Albu-Schaer, M. Hahnle, I. Schaefer, and N. Sporer.” On a new generation of torque controlled light-weight robots”. In Robotics and Automation, 2001. Proceedings 2001 ICRA. IEEE International Conference, volume 4, pages 3356-3363, 2001.

[25] A. Kugi, C. Ott, A. Albu-Schaer, and G. Hirzinger.” On the passivity-based impedance control of flexible joint robots”. Robotics, IEEE Transactions on, 24(2):416-429, 2008.

[26] Gunter Schreiber, Andreas Stemmer, and Rainer Bischoff, Icra 2010 workshop on innovative robot control architectures for demanding (research) applications. “In Innovative Robot Control Architectures for Demanding (Research) Applications”, volume 1, pages 15-21, 2010.

[27] Du Val, Patrick, "*Homographies, quaternions, and rotations*". Oxford, Clarendon Press.

[28] F. Vicentini, N.Pedrocchi, M.Giussani, L, Molinari Tosatti, “Dynamic safety in collaborative robot workspaces through a network of devices fulfilling functional safety requirements”, ISR/Robotik 2014;

[29] N. Patwari, et al, “RF sensor networks for device-free localization: measurements, models, and algorithms”, IEEE Signal Processing Mag.,Vol. 22, N. 4, pp. 54-69, Jul. 2005.

[30] K. J. O’Hara, et al, “Pervasive Sensor-less Networks for Cooperative Multi-robot Tasks ”, Springer Journal on Distributed Autonomous Robotic Systems, no. 6, pp. 305-314, 2007.

[31] "ARTICLE 1 - Terms and Definitions". International Telecommunication Union. 19 October 2009. 1.15. "industrial, scientific and medical (ISM) applications (of radio frequency energy).

[32] S. Petersen, et al, “WirelessHART versus ISA100.11a: the format war hits the factory floor,” IEEE Ind. Electronics Mag., Vol. 5, N. 4, pp.23-34, Dec. 2011.

[33] Shirahama, J., “RSS-Based Localization in Environments with Different Path Loss Exponent for Each Link”.

[34] Li, B., Kam, J., Lui, J., & Dempster, A. “Use of Directional Information in Wireless LAN based indoor positioning”.Proceedings of IGNS, Taipei,2007.

[35] J. Wilson, et al, “Radio tomographic imaging with wireless networks”, IEEE Trans. on Mobile Comp., Vol. 9, N. 5, pp. 621-632, 2010.

[36] Bonazzi R., Catena R., Collina S., Formica L., Munna A., Tesini D. “*Telecomunicazioni per l'ingegneria. Codifica di sorgente. Mezzi di trasmissione. Collegamenti*.”, 2004.

[37] V. Rampa, F. Vicentini, S. Savazzi, L. Pedrocchi, M. Ioppolo, M, Giussani “*Safe Human-Robot Cooperation through Sensor-less Radio Localization,*” Proc. of IEEE International Conference on Industrial Informatics (INDIN 2014), Porto Alegre, Brazil, July 2014.

[38] S. Savazzi, et al, “A Bayesian approach to Device-Free Localization:modeling and experimental assessment”, IEEE Journal on Sel. Topics in Signal Proc., Vol. 8, N. 1, pp. 16-29, Feb. 2014.

[39] XL Meng, DB Rubin ,“Maximum likelihood estimation”, Biometrika,1993.

[40] Rossi, P. H., Lipsey, M. W. & Freeman, H. E. “Evaluation: A Systematic Approach” London,2004.

[41] GB Wetherill, DW Brown, “statistical process control”. 1991 – Springer.

[42] Cynthia A. Lowrya, William H. Woodallb, Charles W. Champc & Steven E. Rigdonc “A Multivariate Exponentially Weighted Moving Average Control Chart”, pages 46-53.

[43] Shapiro, S. S.; Wilk, M. B. "An analysis of variance test for normality (complete samples)". *Biometrika* 591–611,1965.

Appendice

Appendice A: Danger Field.

Partiamo definendo un punto materiale *T* avente coordinate $\boldsymbol{r_t} = (x_t, y_t, z_t)^T$ e velocità $\boldsymbol{v_t} = \left(x_{tx}, y_{ty}, v_{tz}\right)^T$.

Siano poi $\rho_t = \|r - r_t\|$ e $v_{\boldsymbol{t}} = \|\boldsymbol{v_t}\|$, in cui ***r*** è un generico punto nelle coordinate mondo.

Si definisce poi l'angolo $\varphi =$ fase $(\mathbf{r} - \mathbf{r}t\ , \mathbf{v}t\ )$ ossia l'angolo che si forma tra il vettore che congiunge il punto materiale *T* con il generico punto e il vettore velocità.

Si definisce quindi ***static danger field*** ($DF_r = DF_r(\boldsymbol{r}, \boldsymbol{rt})$) una funzione scalare differenziabile che soddisfi le seguenti 2 condizioni:

- $\exists\ f_r\ : R^+ \to R\ \ t.c.\quad DF_r(r, rt) \equiv f_r(\rho_t)$
- $\frac{df_r(\rho_t)}{dt} < 0\ , \forall\ \rho_t > 0$

Ossia una funzione può essere chiamata static danger field se è funzione della distanza tra due punti e se è sempre decrescente in funzione della distanza.

Per essere chiamata ***kinetostatic danger field*** ($DF = DF(\boldsymbol{r}, \boldsymbol{r_t}, \boldsymbol{v_t})$ se soddisfa le seguenti proprietà:

- $\exists\ f : R^3 \to R\quad t.c.\ \boldsymbol{DF}(\boldsymbol{r}, \boldsymbol{rt}, \boldsymbol{vt}) = f(\rho_t, v_t, \varphi).$
- $DF(r, r_t, v_t) = 0$ è uno static danger field.
- $\frac{\partial f(\rho_t, v_t, \varphi)}{\partial \rho_t} \equiv\ -\eta < 0\ ,\ \forall\ \rho_t > 0\ , \forall\ v_t > 0\ , \forall \varphi\ \in [-\pi, \pi].$
- $\frac{\partial f(\rho_t, v_t, \varphi)}{\partial v_t}\ < 0\ ,\ \forall\ \rho_t > 0\ , \forall\ v_t > 0\ , \forall \varphi\ \in \left(-\frac{\pi}{2}.\frac{\pi}{2}\right)$
- $\frac{\partial f(\rho_t, v_t, \varphi)}{\partial \varphi} < 0\ ,\ \forall\ \rho_t > 0\ , \forall\ v_t > 0\ , \forall \varphi\ \in [-\pi, \pi].$

Il ***kinetostatic danger field*** cattura due aspetti fondamentali riguardanti il movimento della sorgente di pericolo: la prima è la norma della velocità e la seconda è l'angolo che formano il vettore velocità con il vettore che rappresenta la distanza tra il punto materiale *T* con un generico punto.
Già in questa forma abbiamo modo di vedere che il danger field avrà delle superfici di livello che non saranno sferiche ma avranno delle forme distorte (figura)

A questo punto è necessario estendere questi concetti all'ambito di nostro interesse.

Si può innanzitutto considerare un manipolatore composto da diversi link, ognuno dei quali è approssimabile come un segmento descritto quindi da un'ascissa curvilinea.
La sorgente di pericolo non sarà quindi più un punto materiale ma sarà una curva in R^3 in movimento.
Sia $\boldsymbol{r_t} : [0, S] \rightarrow R^3$ la mappa dei punti della curva $\boldsymbol{r_t}(s) = \big(x(s), y(s), z(s)\big)^T$ in cui s è il parametro libero mentre S è la lunghezza della curva.

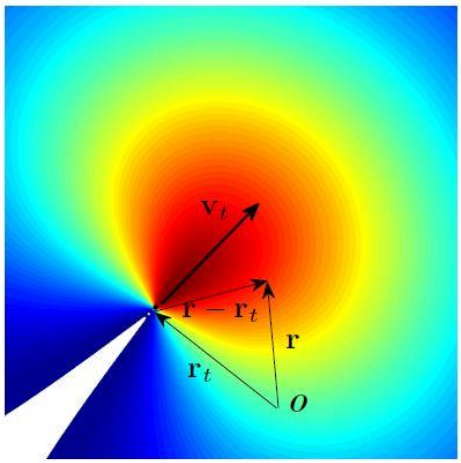


Figura A.1 – Kinetostatic Danger Field

Il difetto dell'ascissa curvilinea è quello che non dà la possibilità di rappresentare la velocità dei punti che descrive.
Per questo motivo, con la definizione dell'ascissa curvilinea, viene definita anche una mappa in cui viene descritta la velocità di ogni punto dell'ascissa stessa.

Sia $\boldsymbol{v_t} : [0,\ S] \rightarrow R^3$ la mappa che assegna ad ogni punto di $\boldsymbol{r_t}(\boldsymbol{s})$ il vettore velocità $\boldsymbol{v_t}(\boldsymbol{s})$ così definito: $\boldsymbol{v_t}(s) = \big(v_x(\boldsymbol{r_t}(s))\ \ v_y(\boldsymbol{r_t}(s)), v_z(\boldsymbol{r\_t}(s))\big)^T$.

Il passo successivo è stato quello di definire il ***cumulative kinetostatic danger field*** (CKSDF) ossia l'integrale di linea lungo l'ascissa curvilinea del Danger Field.

$$CDF(\boldsymbol{r}) = \int_0^S DF(\boldsymbol{r}, \boldsymbol{r_t}, \boldsymbol{v_t}) ds$$

Verrà calcolato un integrale per ogni link del manipolatore. Grazie alla sovrapposizione degli effetti i risultati dei singoli integrali vengono poi sommati.

Siano ora $\boldsymbol{r_i}$, $\boldsymbol{r_{i+1}}$ gli estremi dell' i-esimo link. Ogni punto del link $\boldsymbol{r_s}$ sarà quindi descrivibile come:

$$\boldsymbol{r_s} = \boldsymbol{r_i} + s(\boldsymbol{r_{i+1}} - \boldsymbol{r_i}) \,, \qquad \boldsymbol{s} \in [0, 1]$$

Derivando l'equazione otteniamo le corrispondenti velocità.

$$\boldsymbol{v_s} = \boldsymbol{v_i} + s(\boldsymbol{v_{i+1}} - \boldsymbol{v_i}), \qquad \boldsymbol{s} \in [0,1]$$

Ora verranno introdotte le grandezze necessarie per i calcoli successivi.

$$\boldsymbol{r} - \boldsymbol{rt} = \begin{bmatrix} x \\ y \\ z \end{bmatrix} - \begin{bmatrix} r_{ix} + s\,(r_{i+1\,x} - r_{ix}) \\ r_{iy} + s\,(r_{i+1\,y} - r_{iy}) \\ r_{iz} + s\,(r_{i+1\,z} - r_{iz}) \end{bmatrix} \equiv \begin{bmatrix} \alpha_1 + \alpha_2 s \\ \beta_1 + \beta_2 s \\ \gamma_1 + \gamma_2 s \end{bmatrix}$$

Il modulo del vettore è $\rho_s^2 = \|\boldsymbol{r} - \boldsymbol{r_t}\|^2 = as^2 + bs + c$ in cui $a = \alpha_2^2 + \beta_2^2 + \gamma_2^2, b = 2(\alpha_1\alpha_2 + \beta_1\beta_2 + \gamma_1\gamma_2), e\ c = \alpha_1^2 + \beta_1^2 + \gamma_1^2$.
Il vettore velocità è definito invece come :

$$\boldsymbol{v_s} = \begin{bmatrix} v_{ix} + s(\,v_{i+1\,x} - v_{ix}\,) \\ v_{iy} + s(\,v_{i+1\,y} - v_{iy}) \\ v_{iz} + s(\,v_{i+1\,z} - v_{iz}) \end{bmatrix} = \begin{bmatrix} a_1 + a_2 s \\ b_1 + b_2 s \\ c_1 + c_2 s \end{bmatrix}$$

Il modulo del vettore è $v_s^2 = \|\boldsymbol{v_s}\|^2 = As^2 + Bs + C$ in cui $A = a_2^2 + b_2^2 + c_2^2, B = 2(a_1a_2 + b_1b_2 + c_1c_2), e\ C = a_1^2 + b_1^2 + c_1^2$.

Per determinare l'angolo compreso tra i due vettori appena descritti sfrutteremo il prodotto scalare.

$$\cos\varphi = \frac{\langle \boldsymbol{r} - \boldsymbol{r_s}, \boldsymbol{v_s} \rangle}{\|\boldsymbol{r} - \boldsymbol{r_s}\| \|\boldsymbol{v_s}\|} = \frac{Ms^2 + Ns + P}{\sqrt{as^2 + bs + c}\sqrt{As^s + Bs + C}}$$

In cui $M = \alpha_2 a_2 + \beta_2 b_2 + \gamma_2 c_2, N = \alpha_1 a_2 + \beta_1 b_2 + \gamma_1 c_2 + \alpha_2 a_1 + \beta_2 b_1 + \gamma_2 c_1, P = \alpha_1 a_1 + \beta_1 b_1 + \gamma_1 c_1$.

Il ***cumulative kinetostatic danger field*** dell'i-esimo link è calcolato come l'integrale di linea

$$CDF_i(\boldsymbol{r}) = \int_{\boldsymbol{0}}^{\boldsymbol{1}} DF(\boldsymbol{r}, \boldsymbol{r_s}, \boldsymbol{v_s}) ds = \int_0^1 f(\rho_s, v_s, \varphi) ds$$

Il danger field di un manipolatore composto da n link sarà quindi ottenuto come

$$CDF(\boldsymbol{r}) = \sum_{i=1}^{n} CDF_i(\boldsymbol{r})$$

Ovviamente questo campo è un campo scalare in quanto il risultato dell'integrale definito è un numero.
Per ottenere il campo vettoriale sarà necessario moltiplicarlo per un suo gradiente, dopo averlo normalizzato.

$$\overrightarrow{CDF(\boldsymbol{r})} = CDF(\boldsymbol{r})\ \frac{\nabla CDF(\boldsymbol{r})}{\|\nabla CDF(\boldsymbol{r})\|}$$

$\overrightarrow{CDF(\boldsymbol{r})}$ è un vettore, ancorato in **r**, di intensità CDF(**r**) e diretto come $\nabla CDF(\boldsymbol{r})$**.**

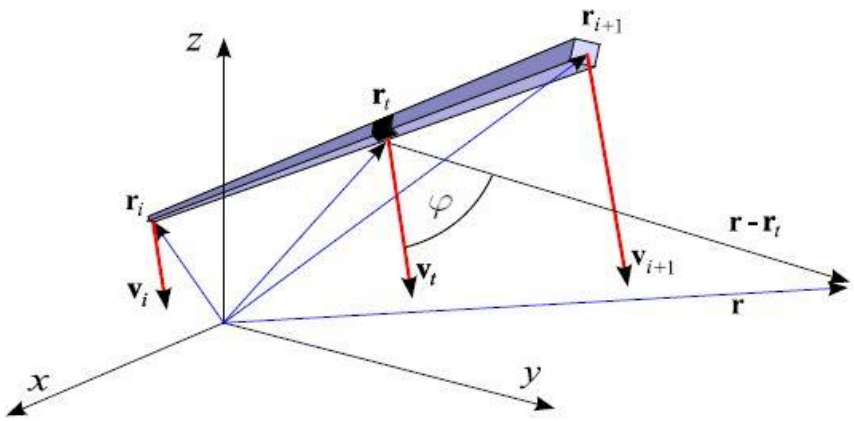


Figura A.2- rappresentazione grandezze introdotte.

Appendice B : NDI Polaris

Il sistema può funzionare con il proprio strumento di controllo, fornito dal costruttore, molto comodo perché permette di vedere immediatamente la posizione dei marker. Questo metodo però non permette di comunicare i dati in modo veloce e continuo.

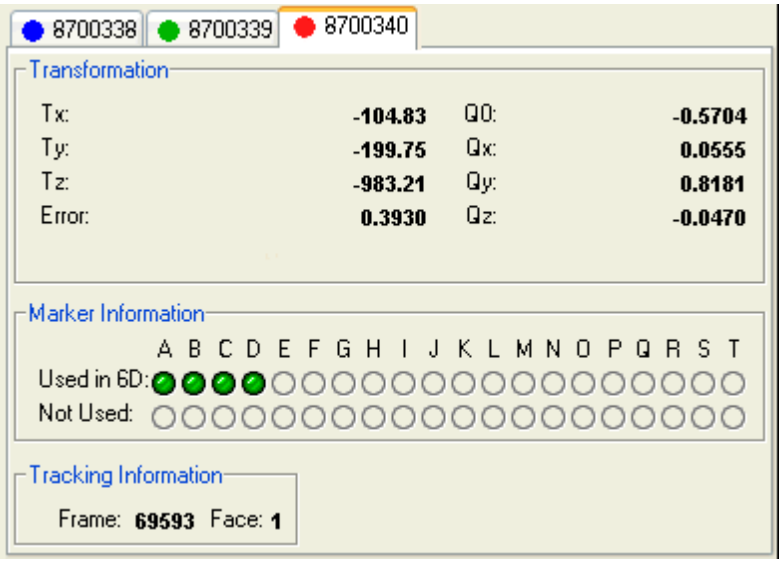


Figura B.1 – Software proprietario

Siccome il sistema è connesso ad un Pc, mediante un cavo USB, è possibile utilizzare dei comandi a basso livello per comunicare con lo strumento.
In questo modo la comunicazione e il flusso di dati avviene in modo continuo e veloce.
Anche in questo caso quindi è stata costruita una struttura di codice in C++ che permettesse di comunicare e ricevere dati dal sensore, riuscendo anche a modificare dei paramentri di misura a run-time.

In particolare, i parametri significativi che si vogliono poter modificare sono:

- Velocità della comunicazione seriale;
- Frequenza di acquisizione del sensore;
- Sistema di riferimento scelto;
- Possibilità di utilizzare il tooltip offset del marker denominato “probe”;

E’ stato quindi creato un “main” C++ che gestisse una interfaccia seriale. Una volta stabilita la comunicazione vengono inviati dei comandi, sotto forma di stringa, che il Polaris è in grado di interpretare. Le risposte del sensore vengono anch’esse lette attraverso la porta seriale e vengono interpretate dal codice.

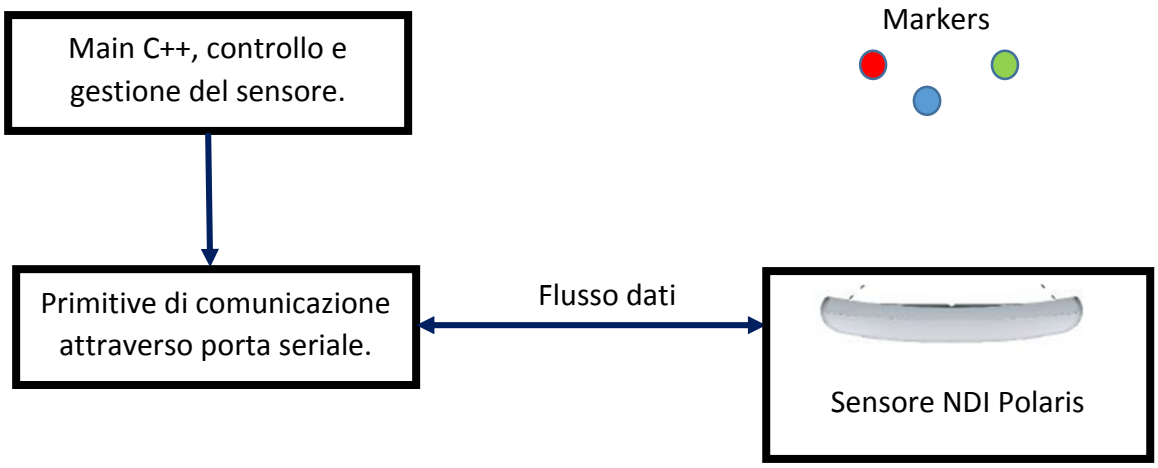


Figura B.2 – Infrastruttura codice

Per quanto riguarda le caratteristiche tecniche del sensore avremo che le velocità a cui può funzionare la comunicazione seriale sono le seguenti:

- 115200 Baude/sec
- 56900 Baude/sec
- 38600 Baude/sec
- 19400 Baude/sec
- 9600 Baude/sec

Le frequenze dichiarate dal costruttore per l’acquisizione sono

- 15 Hz;
- 30 Hz;
- 60 Hz;

In realtà, portando la frequenza a 60 Hz ma inserendo nel campo visivo più markers si ha che la frequenza di campionamento scende automaticamente a 30 Hz.
Nelle varie prove eseguite quindi, è stata sempre utilizzata la frequenza di 30 Hz, mentre è stata modificata la velocità di trasmissione dei dati, creando così una sorta di *rallentamento equivalente*, che permette di analizzare le differenti prestazioni nel caso di campionamenti diversi.

Per quanto riguarda la scelta del sistema di riferimento, il sistema di misura mette a disposizione diverse modalità.
La prima, quella più intuitiva, permette di esprimere le posizioni dei markers rispetto alla terna di riferimento del sensore stesso.

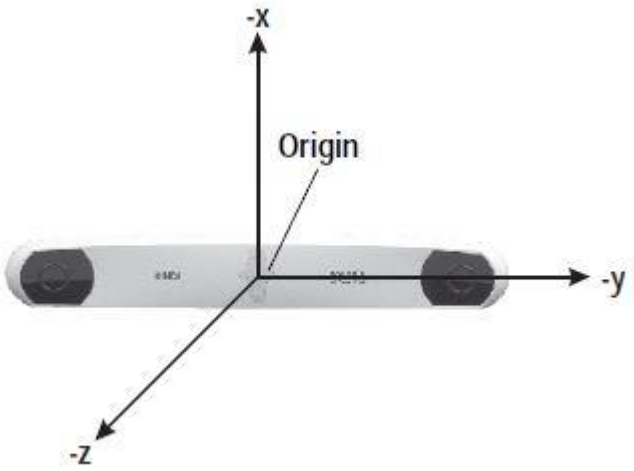


Figura B.3 – Sistema di riferimento globale del sensore

Sfruttare il sistema di riferimento globale del sensore può avere dei vantaggi ma presenta anche degli svantaggi significativi: in questo caso il sensore dovrebbe essere fissato rigidamente in una posizione ben nota nello spazio di lavoro, calibrata. Se dovesse essere spostato i riferimenti verrebbero persi.

Proprio per evitare questi problemi si è scelto di utilizzare un sistema di riferimento associato ad un marker come sistema di riferimento globale. In questo modo tutte le misure sono riferite ad un marker che risulta molto più semplice da vincolare in una posizione, ad esempio nei pressi della base del robot.
E' stata quindi identificata una terna associata ad un marker.

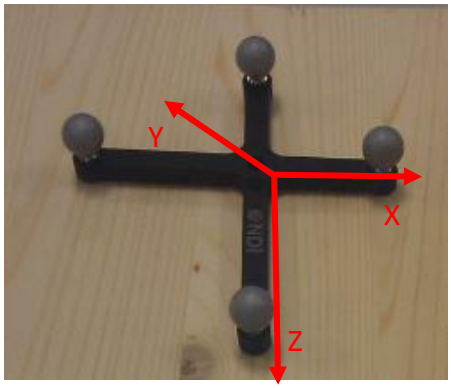


Figura B.4 - Terna associata a marker di riferimento

La scelta del marker che funge da riferimento in realtà è selezionabile all'avvio dell'eseguibile C++, nelle prove eseguite è sempre stato utilizzato quello in figura 5.14.

Una volta fissato il sistema di riferimento, "mobile", è stata quindi necessaria una calibrazione per poter confrontare i valori ottenuti da sistemi differenti in un unico sistema di riferimento. Infatti i tre marker sono stati così utilizzati: un marker come riferimento, un marker agganciato all'ostacolo da identificare e un marker associato all'end effector del robot.

Le coordinate del robot che vengono lette dal robot node sono quelle dell'end effector rispetto alla base del robot, non rispetto al marker di riferimento. Inoltre per verifica, se si volessero confrontare la posizione del robot letto mediante Robot Node e la posizione del robot mediante Sensor Node, quest'ultima non sarà relativa all'end effector, ma al marker montato all'estremità dell'end-effector.

E' quindi necessario utilizzare due matrici di rototraslazione che permettano di conoscere le trasformazioni necessarie per ottenere tutte le coordinate corrette.

Per fare ciò è stata implementata una procedura di hand-eye calibration, (in appendice il codice C++ e spiegazioni ulteriori).

Il risultato sono appunto queste due matrici di rototraslazione che permettono di ricavare le coordinate in un sistema di riferimento unico, permettendo dei confronti immediati.

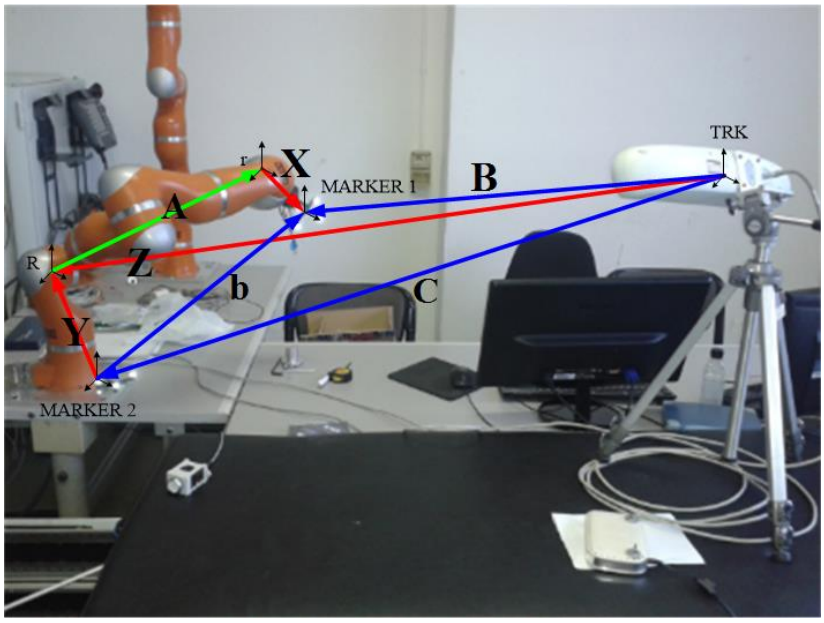


Figura B.5 – Hand- Eye calibration

L'ultimo parametro di impostazione, che è stato utile analizzare, è la possibilità di utilizzare o meno un tooltip offset per il marker denominato probe.

In questo caso si ha infatti che il sensore misura la posizione e l'orientamento della terna sullo strumento in una posizione centrale. Il marker Probe ha, a differneza degli altri, un lungo stelo, che permette di montare il marker in una posizione scomoda. Infatti, si riesce a "delocalizzare" il tracciamento, ma conoscendo la geometria dello strumento si ricostruisce la posizione della punta dello stelo.
Questa procedura è stata effettuata mediante il software proprietario dello strumento in quanto, una volta memorizzate le componenti geometriche la procedura non deve essere più eseguita.

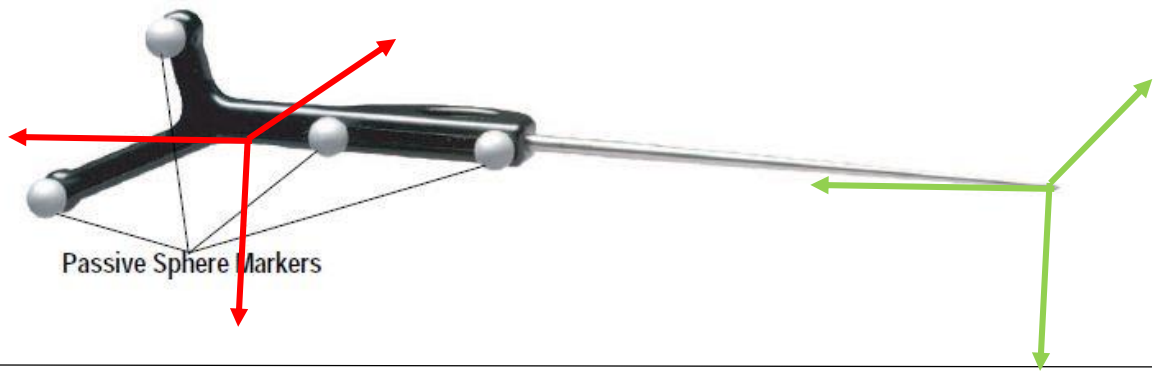


Figura B.6 – tooltip Offset

Viene quindi lasciata la possibilità all'avvio dell'eseguibile C++, di utilizzare o meno questo offset che trasla il sistema di riferimento associato all'oggetto.

Concludendo, questo sensore, collegato con un opportuna interfaccia ad un pc costituisce il Sensor Node e permette quindi la misura di tre oggetti in contemporanea ad ogni istante di tempo.

Anche in questo caso si avrà una struttura multithread sul sensor Node che permette di Acquisire nuove posizioni e inviare dati sicuri al Check Node.
La struttura dati condivisa in questo caso avrà solo i tre campi vettoriali relativi alla posizione dei tre marker.
Anche in questo caso, il sistema era molto flessibile, permettendo di esplicitare le posizioni in diverse notazioni. Per mantenere una continuità è stata scelta la notazione traslazioni cartesiane e quaternione.

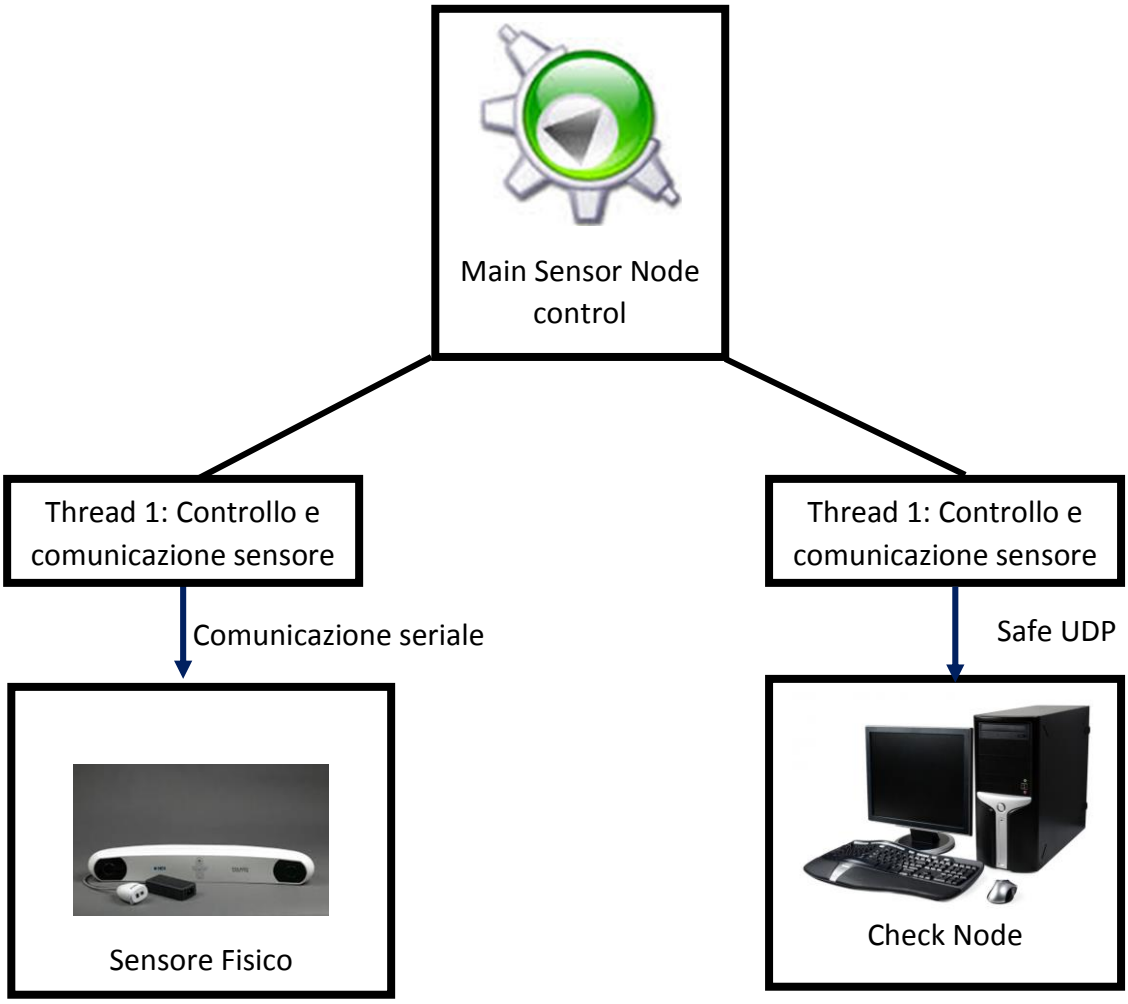


Figura B.7 – infrastruttura codice del programma di controllo e gestione del sistema di tracking ottico.